Grażyna Koba

TERAZBAJTY

Informatyka

dla szkoły podstawowej

Klasa 8

Oznaczenie nieobowiązkowego materiału w związku z usunięciem odpowiednich treści z podstawy programowej zgodnie z rozporządzeniem MEN z 28.06.2024 r.

MiGra

Współpraca autorska:
Katarzyna Koba-Gołaszewska
Paweł Kowalczyk
Michał Łętowski
Marta Skała-Kowalczyk
Paweł Rogoziński

Redakcja i korekta: dr Halina Kubicka

Opracowanie graficzne: Roman Jankowski, Grażyna Koba

Projekt okładki, layout i skład: Studio Grafiki INCOLOR Roman Jankowski

Podręcznik dopuszczony do użytku szkolnego przez ministra właściwego
do spraw oświaty i wychowania i wpisany do wykazu podręczników przeznaczonych
do kształcenia ogólnego do nauczania informatyki, na podstawie opinii rzeczoznawców:
mgr. Andrzeja Dajnowicza, dr. hab. Tomasza Huka, dr Zofii Ireny Krzywickiej-Vauthier.

Etap edukacyjny: II.
Typ szkoły: szkoła podstawowa.
Rok dopuszczenia: 2024.

Numer ewidencyjny w wykazie: 806/5/2024/z2

Zastrzeżonych nazw firm i produktów użyto w tej książce wyłącznie w celu identyfikacji.

ISBN 978-83-964794-5-7

Copyright © by MIGRA Sp. z o.o., Wrocław 2024

Wydanie I

Fotografie: Marcin Mytych – str. 16. (rys. 8., w. 1-3); Grażyna Koba – str. 16. (rys. 8., w. 4); SSPL/Getty Images – str. 140 (honorowe krwiodawstwo); Katarzyna Koba-Gołaszewska – str. 154 (breloczki), str. 157 (rys. 17c); SSPL/Getty Images – str. 161 (rys. 1.); ADOC/EastNews – str. 162 (rys. 2.); IMAGO/EastNews – str. 162 (rys. 3.); AP Images/EastNews – str. 162 (rys. 4.); SPL/EastNews – str. 162 (rys. 5.); DeAgostini/EastNews – str. 163 (rys. 6.); AP Images/EastNews – str. 164 (rys. 7.); AKG Images/EastNews – str. 165 (rys. 8.); Science Source/EastNews – str. 166 (rys. 9.); Marcin Mytych – str. 167 (rys. 10.); Age Fotostock/EastNews – str. 167 (rys. 11.); AP Images/EastNews – str. 168 (rys. 12.); Agencja BE&W – str. 168 (rys. 13.); Science Source/EastNews – str. 168 (rys. 14a, rys. 14b); SPL/EastNews – str. 169 (rys. 15.); Piotr Kamionka/REPORTER – str. 169 (rys. 16.); Michał Koba – str. 208 (rys. 2.); Getty Images –str. 210 (Ateny/Akropol, Warszawa/Plac Zamkowy, Paryż/Wieża Eiffla); Getty Images – str. 219 (rys. 5); Getty Images – str. 222 (grafika z Ziemią widoczną z orbity).

Wydawnictwo: MIGRA Sp. z o.o.
ul. Świeradowska 51/57
50-559 Wrocław
tel.: 71 75 06 230,
faks: 71 75 06 235
e-mail: biuro@migra.pl

Spis treści

Rozdział I Praca z dokumentem tekstowym 7

Temat 1. Więcej o opracowywaniu tekstu 8
1. Stosowanie tabulatorów 8
2. Stosowanie spacji nierozdzielającej 11
3. Listy wypunktowane i numerowane 12
4. Wstawianie tabel do tekstu 14
5. Drukowanie dokumentu tekstowego 16

Temat 2. Praca z dokumentem wielostronicowym 19
1. Wstawianie informacji do stopki i nagłówka 19
2. Szybkie wyszukiwanie słów i znaków 21
3. Tworzenie przypisów 22
4. Podział tekstu na kolumny 23
5. Statystyka dokumentu tekstowego 24

Temat 3. Tworzenie e-gazetki – projekt 27
1. Praca w chmurze 27
2. Przygotowanie projektu grupowego 29
3. Projekt grupowy – e-gazetka 30

Rozdział II Algorytmika i programowanie 33

Temat 4. Wprowadzenie do programowania w języku C++ 34
1. Usystematyzowanie podstawowych pojęć 34
2. Pierwszy program komputerowy w języku C++ 36
3. Stosowanie zmiennych i wykonywanie obliczeń 39
4. Stosowanie instrukcji warunkowej do realizacji algorytmów z warunkami 42
5. Stosowanie instrukcji iteracyjnej do realizacji algorytmów iteracyjnych 44

Temat 5. Wprowadzenie do programowania w języku Python 49
1. Usystematyzowanie podstawowych pojęć 49
2. Programowanie grafiki w języku Python – przypomnienie 51
   2.1. Środowisko programowania IDLE 51
   2.2. Wykorzystanie grafiki żółwia do tworzenia rysunków 52
3. Wyprowadzenie napisów na ekran 53
4. Stosowanie zmiennych i wykonywanie obliczeń 56
5. Stosowanie instrukcji warunkowej do realizacji algorytmów z warunkami 60
6. Stosowanie instrukcji iteracyjnej do realizacji algorytmów iteracyjnych 62

Temat 6. Zapisywanie algorytmów na liczbach naturalnych w wybranych językach programowania 67
1. Realizacja algorytmu Euklidesa w wersji z odejmowaniem 67
2. Badanie podzielności liczb naturalnych 73
3. Realizacja algorytmu Euklidesa w wersji z dzieleniem 76
4. Algorytm wyodrębniania cyfr danej liczby 79

Temat 7. Algorytmy wyszukiwania i porządkowania 85
1. Wyszukiwanie elementu w zbiorze nieuporządkowanym 85
   1.1. Wybieranie większej z dwóch liczb 85
   1.2. Wyszukiwanie największego elementu w zbiorze nieuporządkowanym 88
   1.3. Wyszukiwanie danego elementu w zbiorze nieuporządkowanym 92
2. Wyszukiwanie elementu w zbiorze uporządkowanym metodą połowienia 94
3. Stosowanie metody połowienia do gry w zgadywanie liczby 96
4. Porządkowanie elementów zbioru metodą przez wybieranie 100
5. Porządkowanie elementów zbioru metodą przez zliczanie 102

Temat 8. Wykorzystanie funkcji i tablic do zapisywania w języku C++ algorytmów porządkowania i wyszukiwania 107
1. Stosowanie podprogramów w środowiskach programowania Baltie i Scratch – przypomnienie 107
2. Definiowanie i stosowanie funkcji w języku C++ 110
3. Stosowanie tablic w języku C++ do przechowywania danych 112
4. Zapisywanie w języku C++ wybranych algorytmów wyszukiwania i porządkowania 115

Temat 9. Wykorzystanie funkcji i list do zapisywania w języku Python algorytmów porządkowania i wyszukiwania 123
1. Stosowanie procedur do utworzenia kompozycji w języku Scratch 123
2. Definiowanie i stosowanie funkcji w języku Python 125
3. Stosowanie list do wprowadzania danych 128
4. Zapisywanie w języku Python wybranych algorytmów porządkowania i wyszukiwania 131

Rozdział III Projekty multimedialne 139

Temat 10. Prezentacja multimedialna, modele 3D, druk 3D i obróbka filmów – projekt łączący poznane technologie 140
1. Prezentacja multimedialna z modelami 3D i filmami 140
   1.1. Przygotowanie planu prezentacji i slajdu tytułowego 142
   1.2. Wstawianie obrazów, filmów i modeli 3D 143
   1.3. Wstawianie hiperłączy, wykresów i tabel 146
   1.4. Dodawanie animacji i przejść slajdów 149
2. Podstawowy montaż filmu 151
3. Projektowanie i druk 3D pomocy dydaktycznych 154

Temat 11. Historia i rozwój informatyki – projekt 161
1. Przykłady dawnych przyrządów i maszyn do wykonywania obliczeń 161
2. Pierwsze komputery 163
3. Zadanie projektowe „Historia i rozwój informatyki” 170

Rozdział IV Obliczenia w arkuszu kalkulacyjnym 173

Temat 12. Więcej o pracy w arkuszu kalkulacyjnym 174
1. Adresowanie mieszane 174
2. Linie siatki w arkuszu kalkulacyjnym a obramowania komórek 176
3. Drukowanie tabeli arkusza kalkulacyjnego 177

Temat 13. Przedstawianie danych w postaci wykresu i zadania projektowe 180
1. Podstawowe pojęcia i zasady tworzenia wykresów 180
2. Tworzenie wykresu dla jednej serii danych 181
3. Tworzenie wykresu dla kilku serii danych 183
4. Dostosowywanie typu wykresu do rodzaju danych 184
5. Projekt grupowy – impreza sportowa 186

Temat 14. Zastosowanie arkusza kalkulacyjnego 189
1. Wykorzystanie arkusza kalkulacyjnego do realizacji algorytmów 189
2. Fizyka – badanie ruchu jednostajnego prostoliniowego 191
3. Geografia – przygotowanie referatu o województwach w Polsce 192
4. Modelowanie i symulacja 193
   4.1. Tworzenie modelu na przykładzie rzutu sześcienną kostką do gry 193
   4.2. Symulacja modelu rzutu sześcienną kostką do gry 193
5. Porządkowanie i filtrowanie danych 194

Rozdział V Internet 199

Temat 15. Tworzenie strony internetowej z wykorzystaniem znaczników HTML – projekt 200
1. Narzędzia do tworzenia stron internetowych 200
2. Ogólne zasady projektowania stron internetowych 201
3. Kod źródłowy strony internetowej 202
4. Tworzenie strony internetowej „Podróże po stolicach Europy” 203
   4.1. Formatowanie tekstu 204
   4.2. Wypunktowania i tabele 206
   4.3. Hiperłącza do innych stron 206
   4.4. Wstawianie grafiki 207

Temat 16. Systemy zarządzania treścią – projekt 211
1. Czym są blogi? 211
2. Czym są systemy zarządzania treścią? 212
3. Przykładowy CMS – WordPress 213
   3.1. Tworzenie blogu w systemie WordPress 213
   3.2. Inne możliwości systemu WordPress 216
4. Projekty grupowe 218
   4.1. Trzy atrakcje w mojej miejscowości 218
   4.2. Dookoła świata z Internetem 220

Oznaczenia przyjęte w podręczniku



Warto powtórzyć – wykaz zagadnień do powtórzenia z poprzednich lekcji



Metoda – opis istotnych metod, pojęć i zasad



Sposób postępowania – sposób postępowania prowadzący do osiągnięcia określonego celu



Przykłady – pokazanie przykładowego rozwiązania



Ćwiczenia – praktyczne sprawdzenie i utrwalenie poznawanych zagadnień



Słownik – wyjaśnienie ważnych terminów



Czy wiesz, że – dodatkowe informacje na temat zagadnień dotyczących informatyki

Na końcu tematów znajdują się działy:



Warto zapamiętać – zestawienie najważniejszych informacji podsumowujących temat



Pytania i polecenia – sprawdzające zdobytą wiedzę



Zadania – sprawdzające stopień opanowania wiedzy i zdobytych umiejętności

Dla zainteresowanych – zadania rozwijające zainteresowania i kreatywność



Przykłady innych projektów grupowych – tematy projektów do wykonania w grupach



Przeczytaj, jeśli chcesz wiedzieć więcej… – rozszerzenie wiedzy na temat interesujących zagadnień z dziedziny informatyki

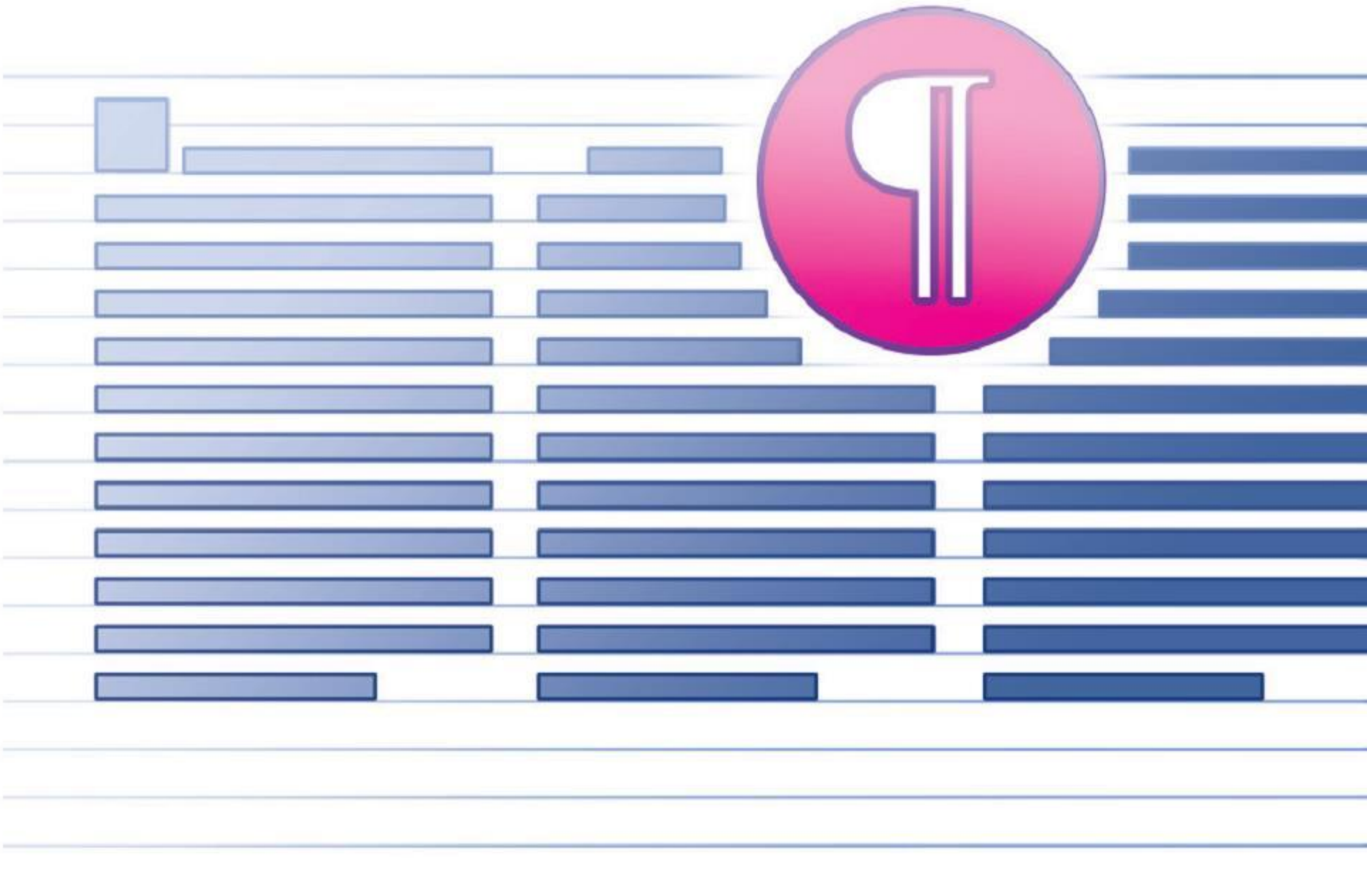
Praca z dokumentem tekstowym

Rozdział 1

Temat 1. Więcej o opracowywaniu tekstu

Temat 2. Praca z dokumentem wielostronicowym

Temat 3. Tworzenie e-gazetki – projekt



Temat 1. Więcej o opracowywaniu tekstu

1. Stosowanie tabulatorów
2. Stosowanie spacji nierozdzielającej
3. Listy wypunktowane i numerowane
4. Wstawianie tabel do tekstu
5. Drukowanie dokumentu tekstowego

Warto powtórzyć

1. W jaki sposób tworzy się akapity w dokumencie tekstowym?
2. Do czego służy wcięcie w tekście i jak je wykonujemy?
3. Na czym polega formatowanie tekstu?
4. Jakie są podstawowe zasady redagowania tekstu?
5. W jaki sposób można szybko skopiować formatowanie fragmentu tekstu?

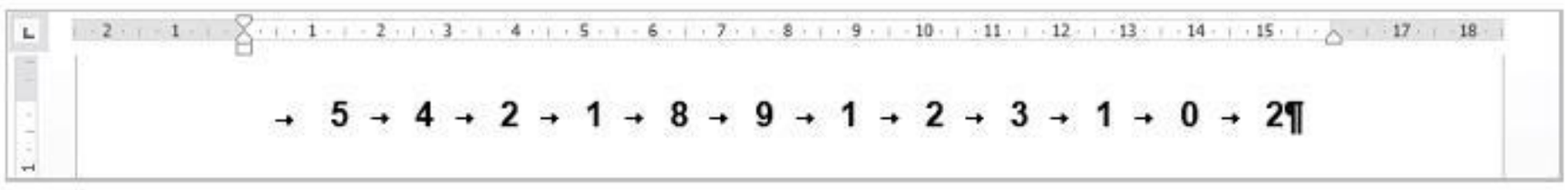
1. Stosowanie tabulatorów

W klasie siódmej omówiliśmy kilka zasad redagowania tekstu, zwracając uwagę na dostosowywanie tekstu do jego przeznaczenia. Edytory tekstu posiadają wiele innych funkcji umożliwiających opracowywanie tekstu.

{ Chcemy przesunąć tekst w ściśle określone miejsce, np. wyrównać tekst w kolumnach bez użycia tabeli. Jak to zrobić bez użycia zwykłych spacji? }

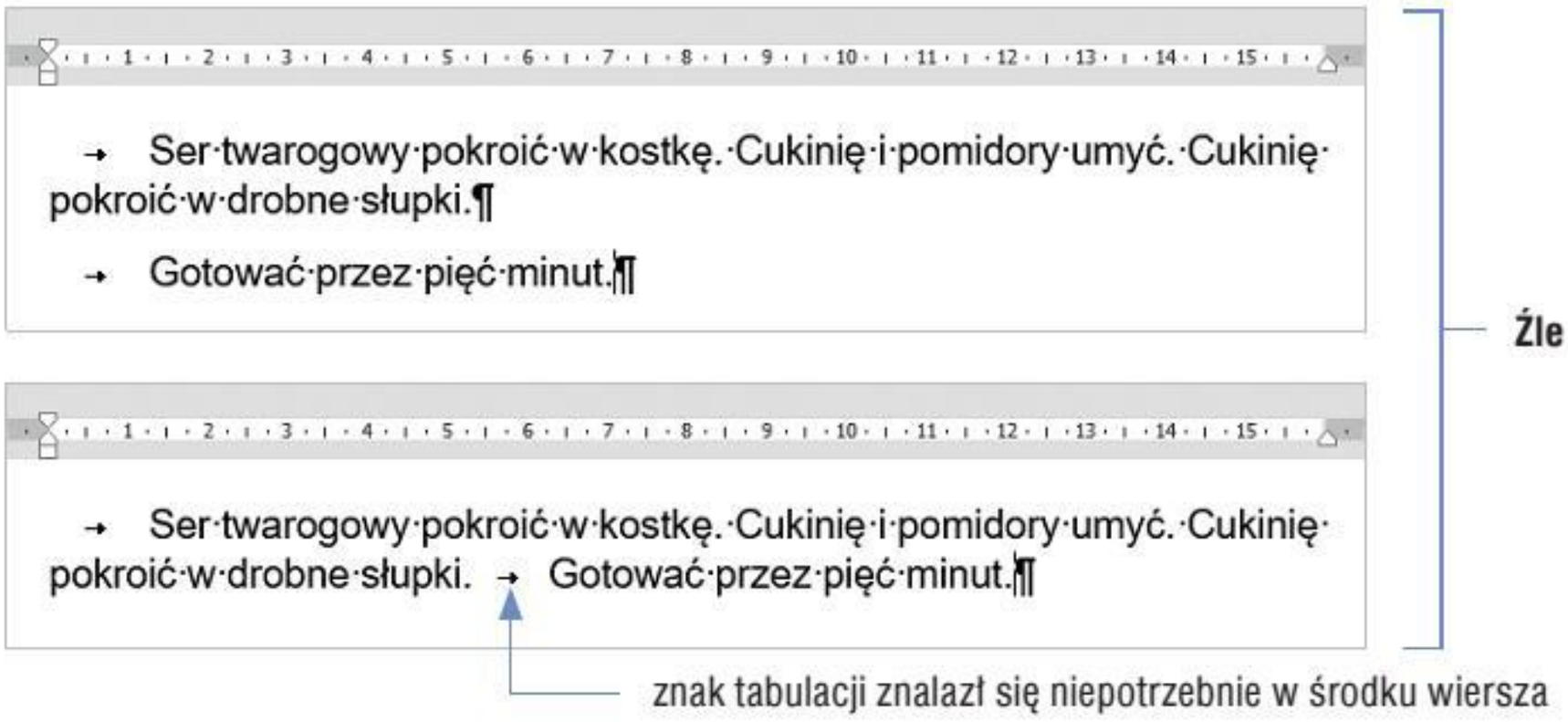
Do tego celu używamy **tabulatora** wyznaczającego w wierszu pozycję, do której przesunie się kursor tekstowy i tekst znajdujący się po prawej stronie tabulatora.

Jeśli w dokumencie tekstowym nie ustawimy własnych pozycji tabulatorów, czasami naciśnięcie klawisza **Tab** przesunie kursor tekstowy w miejsce wyznaczone przez **tabulator domyślny**. Za każdym razem zostanie wprowadzony **znak tabulacji** →, należący do znaków niewidocznych na wydruku. Standardowo znaki tabulacji są rozmieszczone na całej szerokości wiersza w takich samych, domyślnie ustawionych odległościach (rys. 1.).

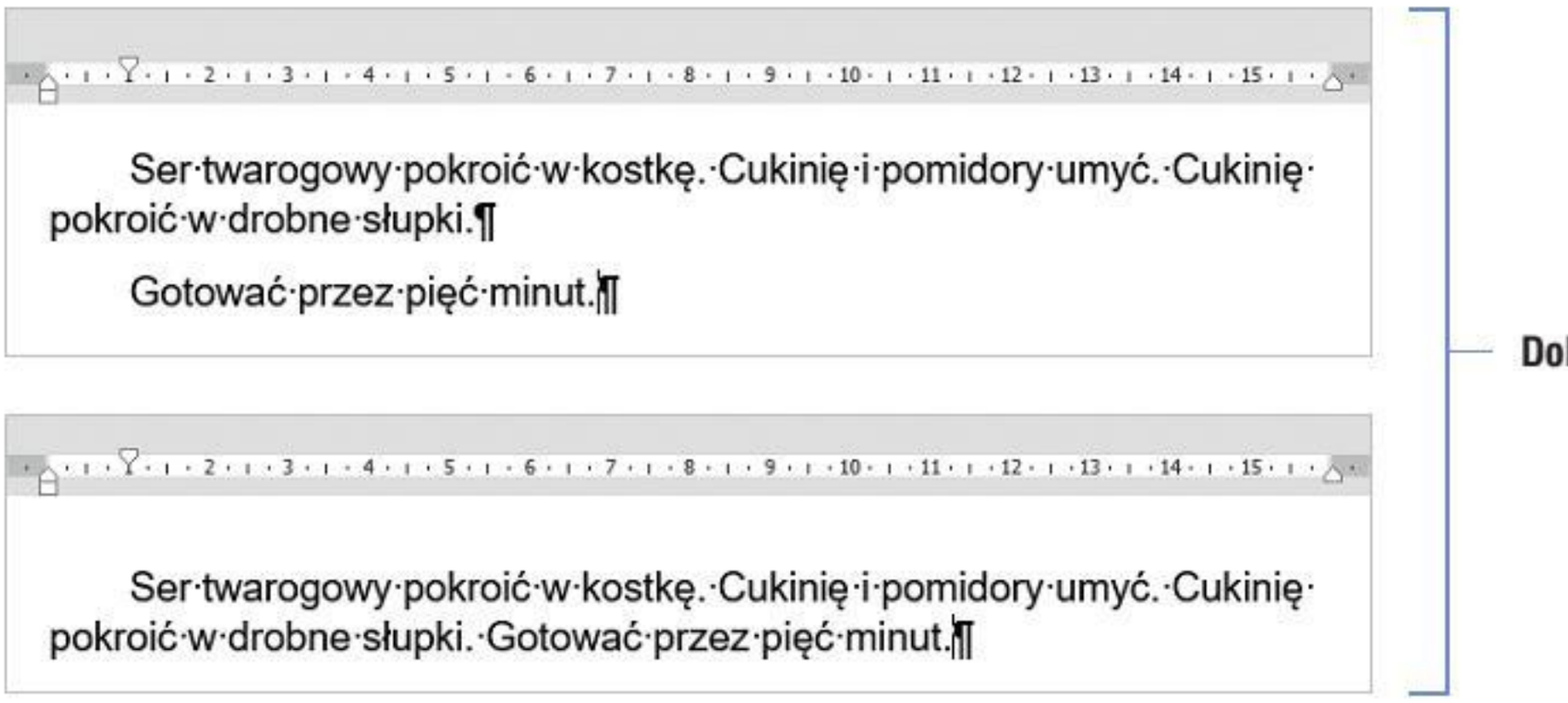


Rys. 1. Przykład stosowania tabulatora domyślnego (pierwszy tabulator ustawiono w odległości 1,25 cm od lewego marginesu, każdy kolejny w odległości 1,25 cm od poprzedniego)

Nie powinno się używać tabulatora ani spacji do wykonywania wcięć pierwszego wiersza akapitu, ponieważ znaki te mogą pozostać w tekście i np. po połączeniu dwóch akapitów tekst drugiego akapitu będzie niepotrzebnie oddalony od pierwszego (rys. 2a).



Rys. 2a. Niepoprawnie wykonane wcięcia – za pomocą tabulatorów



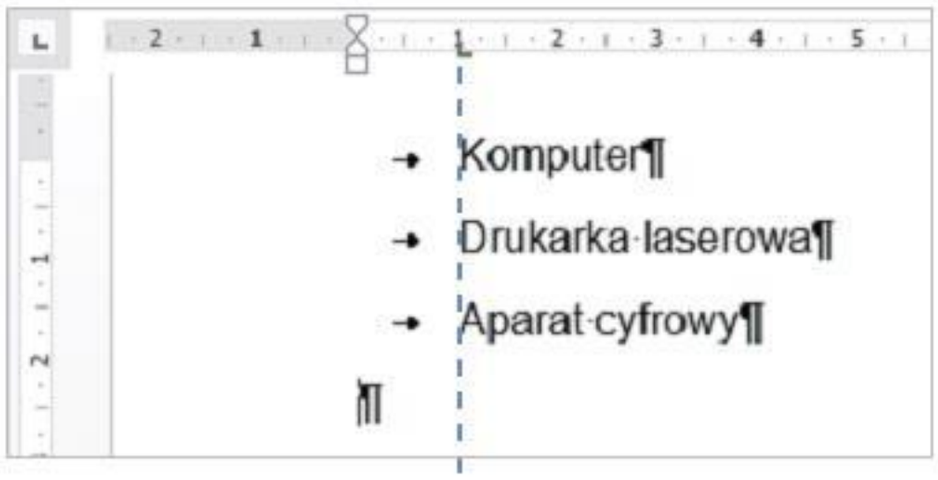
Rys. 2b. Poprawnie wykonane wcięcia – za pomocą suwaka na linijce

Większość edytorów tekstu umożliwia stosowanie własnych tabulatorów (tzw. **tabulatorów specjalnych**).

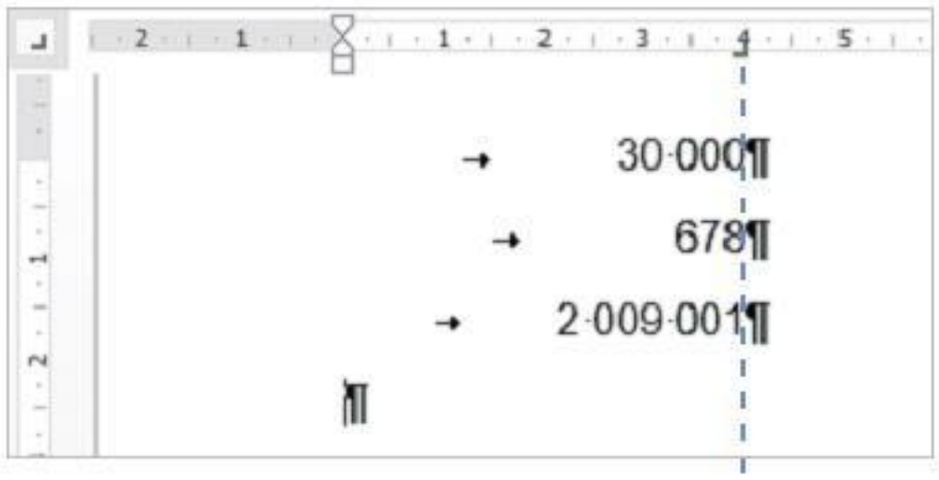
Aby ustawić tabulator specjalny, należy kliknąć odpowiednie miejsce na linijce – pojawi się tam znacznik tabulatora (rys. 3a-3d). Znacznik wstawionego tabulatora można usunąć, przeciągając go poza linijkę. Pozycje tabulatorów, w tym tabulatorów domyślnych, można zmieniać lub usuwać w oknie **Akapit** (przycisk **Tabulatory**).

Rodzaj tabulatora można zmienić po naciśnięciu przycisku umieszczonego z lewej strony linijki – domyślnie ustawiony jest tabulator lewy (rys. 3a). Naciskając ten przycisk, można uzyskać różne rodzaje tabulatorów specjalnych, m.in.: prawy (rys. 3b), środkowy (rys. 3c), dziesiętny (rys. 3d).

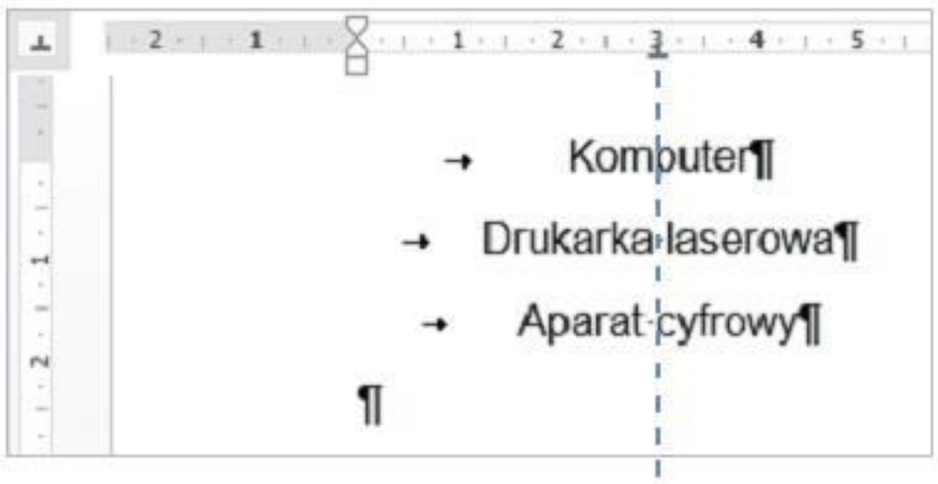
Tabulatory ustalamy dla aktualnie redagowanego akapitu. Przejście do nowego akapitu, czyli naciśnięcie klawisza **Enter**, zachowuje ostatnio ustawione tabulatory.



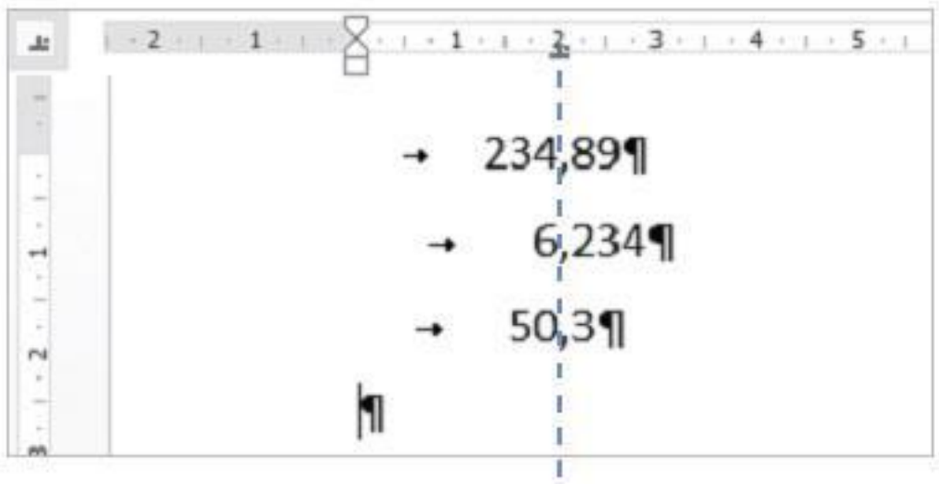
Rys. 3a. Zastosowanie tabulatora lewego



Rys. 3b. Zastosowanie tabulatora prawego

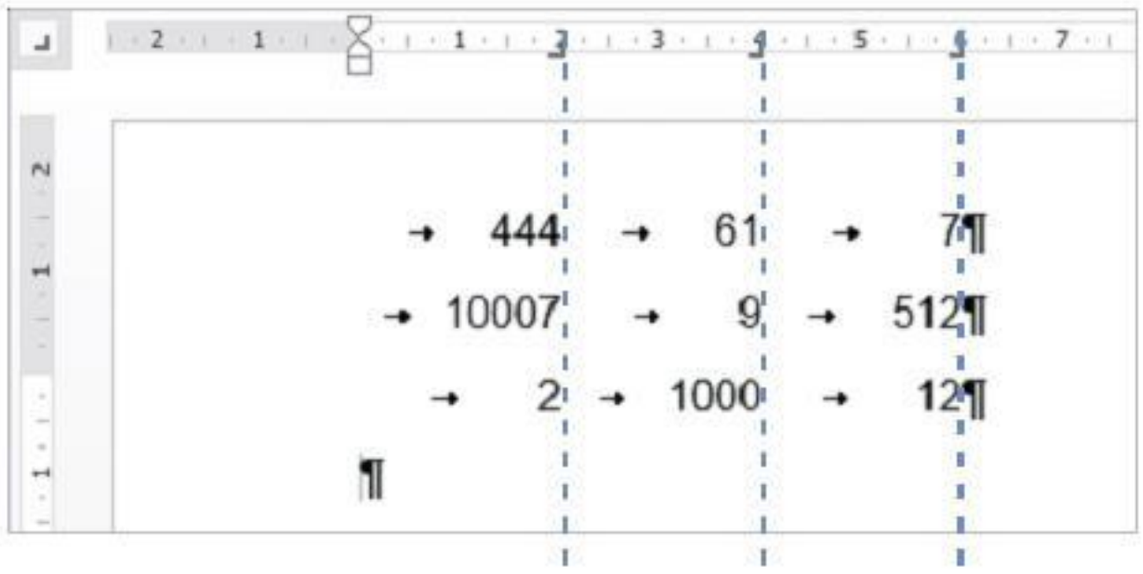


Rys. 3c. Zastosowanie tabulatora środkowego



Rys. 3d. Zastosowanie tabulatora dziesiętnego

Jeśli chcemy zachować takie samo ustawienie tabulatorów w całym tekście lub w dowolnym jego fragmencie, należy przed ustaleniem położenia tabulatorów zaznaczyć tekst i dopiero potem ustawić pozycje tabulatorów.



Rys. 4. Przykład wyrównywania tekstu z użyciem tabulatorów. Na linijce wstawiono trzy tabulatory: w odległości 2 cm, 4 cm i 6 cm od lewego marginesu

Ćwiczenie 1. Ustawiamy tabulatory

1. Naciśnij klawisz **Tab** i sprawdź, jakie jest domyślne ustawienie tabulatorów w dokumencie. Pamiętaj o kliknięciu przycisku pokazywania wszystkich znaków.
2. Ustaw tabulatory specjalne i wpisz w kolumnach przykładowe liczby, jak na rysunkach 3b i 3d.
3. Ustal dla następnego akapitu tabulatory specjalne w innej pozycji, np. w odległości 1 cm, 3 cm, 5 cm, i wpisz w kolumnach te same liczby.
4. Jeśli linijka nie jest widoczna, wyświetl ją. Jeśli nie wiesz, jak to zrobić, skorzystaj z **Pomocy** do programu. Poszukaj tam też więcej informacji na temat ustawiania tabulatorów w używanym przez ciebie programie.

Do wyrównania dowolnego tekstu (napisów lub liczb) w kolumnach należy stosować tabulatory, a nie spacje.

Ćwiczenie 2. Wyrównujemy kolumny tekstu za pomocą spacji

Przygotuj plik z ocenami uzyskanymi w bieżącym półroczu z poszczególnych przedmiotów. Wstaw za nazwą przedmiotu kilka spacji, aby w ten sposób wyrównać kolumny z ocenami.

1. Oceny umieszczaj w kolumnach wyrównanych spacjami.
2. Oceny z klasówek wpisz pogrubioną czcionką koloru czerwonego.
3. Zapisz plik pod nazwą *Oceny1*.
4. Obejrzyj dokument w **Podglądzie wydruku (Plik/Drukuj)** i wydrukuj go.

Ćwiczenie 3. Wyrównujemy kolumny tekstu za pomocą tabulatorów

1. Wykonaj ćwiczenie 2., ale zamiast spacji użyj do wyrównywania kolumn z ocenami tabulatorów.
2. Zapisz plik pod nazwą *Oceny2*.
3. Obejrzyj dokument w **Podglądzie wydruku** i wydrukuj go.

Ćwiczenie 4. Zalety wyrównywania tekstu za pomocą tabulatorów

1. Oceń sposoby wyrównywania kolumn zastosowane w ćwiczeniach 2. i 3. Który sposób przynosi lepsze efekty?
2. Zwiększ rozmiar czcionki zastosowanej w ćwiczeniach 2. i 3. o dwa punkty. Omów zauważone zmiany.

2. Stosowanie spacji nierozdzielającej

{ W napisanym tekście na końcu wiersza pozostał spójnik „z”. Jak przenieść go do następnego wiersza? }

W poprawnie zredagowanym tekście:

- jednoliterowe spójniki i przyimki nie powinny pozostawać na końcu wiersza,
- tytuł naukowy umieszczany przed imieniem i nazwiskiem (mgr, dr, prof.) należy zapisać w tym samym wierszu co imię i nazwisko,
- skrót słowa „rok” (r.), występujący w zapisie daty po liczbie określającej rok, należy zapisać w tym samym wierszu co rok,
- liczby określające miary należy zapisać w tym samym wierszu co jednostki tych miar, na przykład każde z wyrażeń „1000 km”, „20 kg”, „2345 zł” powinno znajdować się w jednym wierszu.

Aby mieć pewność, że w tworzonym przez nas dokumencie zachowamy te zasady, stosujemy **spację nierozdzielającą** (potocznie zwaną twardą spacją).

S **Spacja nierozdzielająca**

Stosowana na potrzeby rozmieszczania tekstu w obrębie jednego wiersza. Wyrazy rozdzielone taką spacją są traktowane jak jeden wyraz.

Jeśli wstawimy spację nierozdzielającą, musimy usunąć zwykłą spację. Spacja nierozdzielająca należy do znaków niewidocznych na wydruku. Po włączeniu opcji pokazywania wszystkich znaków widzimy ją w tekście w postaci małego okręgu (w programie Microsoft Word – rys. 5b) lub zacieniowanego małego prostokąta (w programie LibreOffice Writer).

Uwaga

Aby wstawić spację nierozdzielającą, ustawiamy kursor tekstowy za daną literą (np. spójnikiem „i"), usuwamy zwykłą spację (jeśli występuje za tą literą), naciskamy i przytrzymujemy klawisze **Ctrl + Shift +Spacja**.

Zapraszamy·Panią·dr·hab.·Katarzynę·Nowak·na·uroczystość·z·okazji·15-lecia·działalności·szkoły.¶

Spotkanie·odbędzie·się·15·października·2024·r.·w·budynku·szkolnym.¶

Rys. 5a. Przykład źle zredagowanego tekstu

Zapraszamy·Panią·dr°hab.·Katarzynę·Nowak·na·uroczystość·z·okazji·15-lecia·działalności·szkoły.¶

Spotkanie·odbędzie·się·15°października·2024°r.·w°budynku·szkolnym.¶

Rys. 5b. Przykład poprawnie zredagowanego tekstu

Ćwiczenie 5. Wstawiamy spacje nierozdzielające

1. Utwórz w edytorze tekstu nowy plik.
2. Przepisz tekst widoczny na rysunku 5a. Zastosuj czcionkę Times New Roman, pochyloną, o rozmiarze 12 punktów.
3. Ustaw marginesy i wcięcia tak jak przedstawiono na rysunku 5a.
4. Wstaw spacje nierozdzielające tam, gdzie jest to uzasadnione zasadami redagowania tekstu. Omów różnice w wyglądzie przepisanego tekstu i tekstu pokazanego na rysunkach 5a i 5b.

3. Listy wypunktowane i numerowane

Jeśli konieczne jest ponumerowanie lub wypunktowanie kolejnych pozycji w tekście, należy zastosować **listy numerowane** lub **wypunktowane**. Edytory tekstu oferują różne rodzaje numeracji i wypunktowań. W większości edytorów można też tworzyć listy wielopoziomowe (z podpunktami). Obowiązuje przy tym reguła, że każdy akapit stanowi osobny element listy.

Aby zastosować bądź zmodyfikować listę numerowaną lub wypunktowaną, nacisnąć przycisk numerowania albo wypunktowania na karcie **Narzędzia główne** (Microsoft Word) lub na pasku narzędzi (LibreOffice Writer).

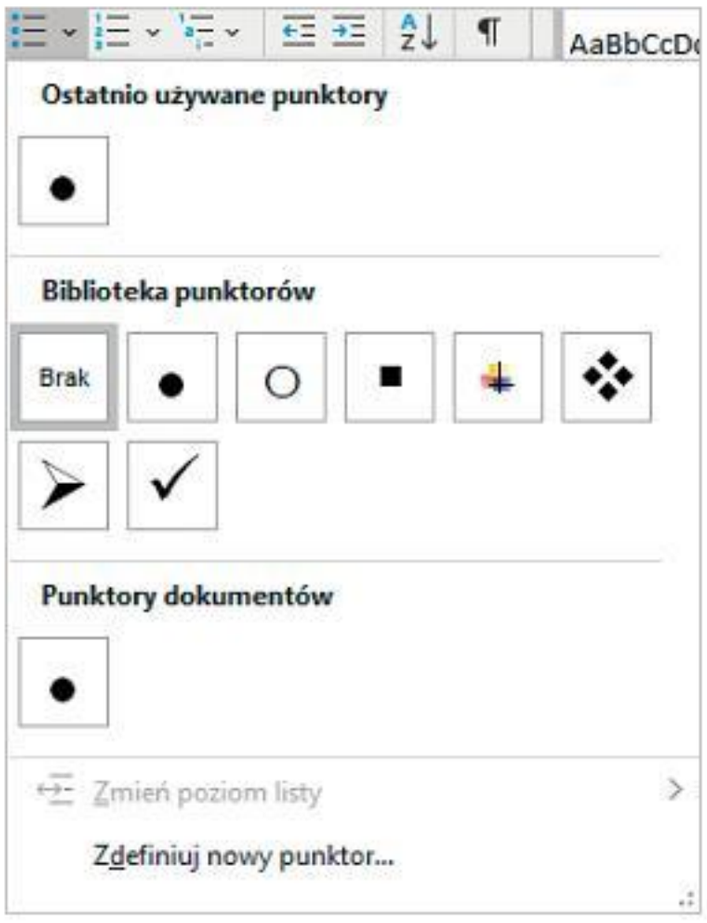
Aby wprowadzić listę numerowaną lub wypunktowaną w już napisanym tekście, należy go najpierw zaznaczyć.

Ćwiczenie 6. Poznajemy możliwości tworzenia listy numerowanej lub wypunktowanej

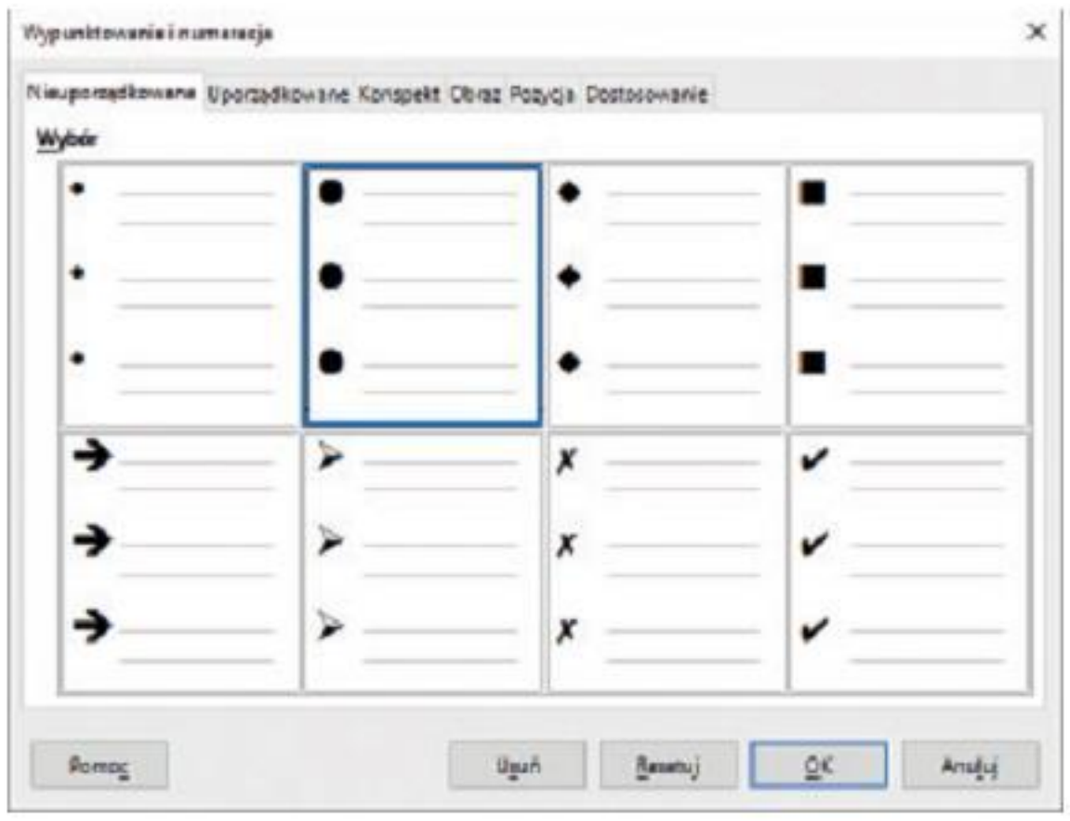
Odszukaj opcję menu, która umożliwia wstawianie listy numerowanej lub wypunktowanej w twoim edytorze tekstu. Sprawdź, jakie rodzaje numeracji i wypunktowania oferuje program, którego używasz. Możesz skorzystać z **Pomocy**.

W odpowiednich oknach można wybrać proponowane przez program rodzaje numerowania lub wypunktowania, a także tworzyć własne style lub modyfikować istniejące.

Sposób tworzenia listy numerowanej lub wypunktowanej ustalamy dla aktualnie redagowanego akapitu. Przejście do nowego akapitu, czyli naciśnięcie klawisza **Enter**, powoduje kontynuację numerowania lub wypunktowywania akapitów.



Rys. 6a. Przykładowe rodzaje wypunktowania (Microsoft Word)



Rys. 6b. Przykładowe rodzaje wypunktowania (LibreOffice Writer)

Aby w danym punkcie rozpocząć pisanie w nowym wierszu bez tworzenia nowego punktu, należy zastosować **ręczny podział wiersza**. W tym celu trzeba nacisnąć kombinację klawiszy **Shift** + **Enter**.

Ćwiczenie 7. Stosujemy listę numerowaną w napisanym tekście

1. Otwórz plik *Oceny2* zapisany w ćwiczeniu 3.
2. Zastosuj numerowanie nazw przedmiotów.
3. Zapisz plik pod tą samą nazwą.

Dobra rada

Nie wpisuj numerów pozycji listy samodzielnie. W pracy nad tekstem stosuj listy numerowane lub wypunktowane.

Ćwiczenie 8. Stosujemy wypunktowanie w nowym tekście

1. W edytorze tekstu utwórz nowy dokument tekstowy. Opisz w nim poznane na lekcjach informatyki programy użytkowe, podając ich nazwę i ogólne zastosowanie. Wymień przynajmniej pięć przykładów. Zastosuj wypunktowanie. Na przykład:
   Przykładowe programy użytkowe:
   – *edytor grafiki Paint – do tworzenia lub modyfikowania obrazów,*
   – …
2. Zapisz plik pod nazwą *Programy użytkowe*.

Zastosowanie list numerowanych lub wypunktowanych powoduje, że każdy wiersz akapitu jest automatycznie wyrównywany (odpowiednio ustawione zostają tabulatory i wcięcia). Jeśli natomiast numery wpisujemy ręcznie, to aby osiągnąć ten sam efekt, trzeba samemu ustawić tabulatory i wcięcia tekstu. Zazwyczaj edytor tekstu automatycznie zmienia ręczne ustawienie numeracji na listę numerowaną (po wpisaniu pierwszego punktu i naciśnięciu klawisza **Enter**).

Wprowadzenie zmian na liście numerowanej (dodanie lub usunięcie punktu) powoduje przenumerowanie pozostałych elementów listy.

Aby umieścić dany punkt listy na niższym poziomie, trzeba nacisnąć klawisz **Tab** lub przycisk **Zwiększ wcięcie** na karcie **Narzędzia główne** (Microsoft Word) lub na pasku narzędzi (LibreOffice Writer).

Aby umieścić dany punkt listy na wyższym poziomie, trzeba nacisnąć klawisze **Shift** + **Tab** lub przycisk **Zmniejsz wcięcie**.

Ćwiczenie 9. Tworzymy listę z podpunktami

1. Utwórz nowy dokument. W kolejnych wierszach wpisz nazwy wszystkich miesięcy. Pod każdą nazwą wpisz w trzech punktach krótką informację o danym miesiącu.
2. Zastosuj listę numerowaną z podpunktami. Nazwy miesięcy powinny być oznaczone liczbami rzymskimi, a informacje – arabskimi.
3. Zapisz plik pod nazwą *Miesiące*.

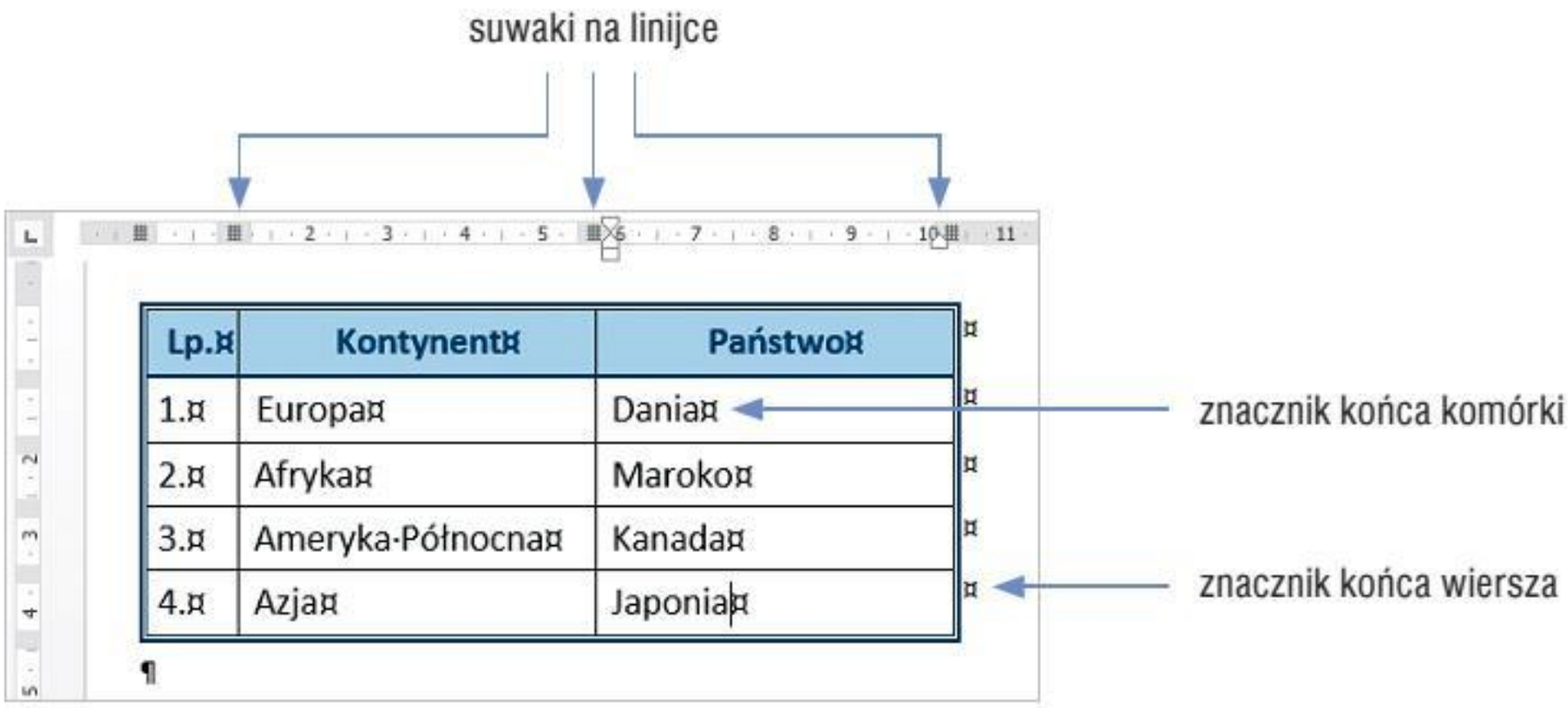
4. Wstawianie tabel do tekstu

Większość edytorów tekstu umożliwia wstawianie lub rysowanie tabel (prostsze edytory tekstu, np. WordPad czy Notatnik, mogą nie posiadać funkcji wstawiania tabel). Do dokumentu tekstowego można również wstawić tabelę wykonaną w innym programie (np. w arkuszu kalkulacyjnym) lub skopiowaną ze strony internetowej. Można wstawić lub narysować tabelę o różnej liczbie kolumn i wierszy. Po wstawieniu tabeli można zmieniać szerokość kolumn i wysokość wierszy, dodawać nowe wiersze i kolumny, usuwać wiersze, kolumny lub komórki, dzielić lub scalać komórki oraz formatować tabelę lub jej wybrane elementy.

Tabelę tworzy się, korzystając z opcji menu **Tabela**. **Aby wykonać dowolną operację na tabeli**, najlepiej skorzystać z menu kontekstowego (kursor tekstowy należy umieścić w odpowiednim miejscu wewnątrz tabeli).

Każdy wiersz tabeli kończy się **znacznikiem końca wiersza**, a każda komórka – **znacznikiem końca komórki**. Są to znaki niewidoczne na wydruku. Tekst wprowadzamy zawsze przed znacznikiem końca komórki (rys. 7.).

Aby zmienić rozmiary komórek, można przesunąć odpowiednie suwaki na linijce poziomej (lub pionowej) lub przeciągać odpowiednie krawędzie siatki tabeli (rys. 7.). Rozmiary komórek można też ustalić bardziej precyzyjnie w oknie **Właściwości tabeli**, wybierając odpowiednie polecenie z menu kontekstowego tabeli.



Rys. 7. Przykład tabeli – ćwiczenie 10.

Ćwiczenie 10. Wstawiamy tabelę do dokumentu tekstowego

1. W edytorze tekstu utwórz nowy dokument i wstaw do niego tabelę składającą się z pięciu wierszy i trzech kolumn.
2. Wprowadź do komórek tabeli informacje podane na rysunku 7. W pierwszej kolumnie zastosuj automatyczną numerację.
3. Dodaj z prawej strony nową kolumnę i wpisz w nagłówku „Stolica", a w komórkach poniżej wpisz odpowiednie nazwy stolic. Dopasuj szerokość komórek do wprowadzanych informacji.
4. Na końcu tabeli wstaw trzy dodatkowe wiersze i wprowadź do nich odpowiednie informacje na temat trzech innych państw.
5. Zapisz plik pod nazwą *Państwa*.

Wskazówka: Jeśli masz problemy z utworzeniem tabeli, odszukaj w **Pomocy** więcej informacji na ten temat.

Komórki tabeli można formatować, np. ustalać parametry czcionki, wyrównanie tekstu w komórce i kierunek tekstu. Można również zmieniać obramowanie komórek i cieniować je (zmieniać kolor tła).

Ćwiczenie 11. Formatujemy komórki tabeli

1. Otwórz plik *Państwa* zapisany w ćwiczeniu 10.
2. Sformatuj komórki tabeli podobnie jak pokazano na rysunku 7.

Formatowanie komórek możemy usprawnić, stosując gotowe style, w których są ustalone m.in. kolory nagłówków tabeli, cieniowania i obramowania komórek.

Do komórek tabeli można wstawić, poza tekstem i liczbami, również rysunki, hiperłącza, wzory matematyczne.

Dobra rada

Edytor Microsoft Word automatycznie zmienia pierwszą literę tekstu wpisanego do komórki tabeli na wielką literę. Aby wyłączyć tę funkcję, należy w opcjach programu odnaleźć funkcję **Autokorekta** i odznaczyć opcję **Początek komórek tabeli wielką literą**.

Przykłady urządzeń peryferyjnych			
Lp.	**Nazwa**	**Wygląd urządzenia**	**Przeznaczenie**
1.	projektor		Przekazywanie obrazu bezpośrednio z karty graficznej komputera na specjalny ekran lub ścianę.
2.	drukarka		Drukowanie tekstu i obrazu. Rozróżniamy drukarki: igłowe, atramentowe i laserowe.
3.	skaner		Zamiana obrazu drukowanego (tekst, rysunek, zdjęcie) na jego elektroniczne odwzorowanie.
4.	kamera internetowa		Prowadzenie wideokonferencji, nadzorowanie i obserwacja pomieszczeń.

Rys. 8. Fragment przykładowej tabeli – ćwiczenie 12.

Ćwiczenie 12. Wstawiamy obrazy do komórek tabeli

1. W edytorze tekstu utwórz nowy dokument i wstaw do niego tabelę taką, jak pokazana na rysunku 8.
2. Wstaw do komórek w trzeciej kolumnie odpowiednie obrazy – ilustracje dostępne w Internecie (opcja **Obrazy online**) lub zdjęcia z własnych zbiorów.
3. Dodaj do tabeli dwa wiersze i umieść w nich informację o dwóch innych urządzeniach peryferyjnych.
4. Do formatowania tabeli zastosuj wybrany styl. Ustal taką samą wysokość wierszy tabeli, w których umieszczono informacje o urządzeniach. Nazwy urządzeń zapisz małymi literami.
5. Zapisz plik pod nazwą *Urządzenia*.

5. Drukowanie dokumentu tekstowego

Dokumenty tekstowe tworzone w edytorze tekstu drukuje się podobnie jak każdy dokument komputerowy. Przed wydrukowaniem dokumentu tekstowego warto obejrzeć wygląd dokumentu na całej stronie (stronach) w **Podglądzie wydruku**.

We właściwościach wydruku można ustalać różne opcje drukowania, m.in.: orientację strony (poziomą, pionową), rodzaj wydruku (kolorowy, odcienie szarości), drukowanie stron parzystych czy nieparzystych, druk obustronny lub jednostronny.

Wielkość strony do drukowania jest ustalona standardowo, najczęściej jest to format A4. Program sam dzieli dłuższy tekst na strony. Format strony można również zmienić we właściwościach wydruku lub korzystając z opcji zamieszczonych na karcie **Układ** w grupie **Ustawienia strony** (Microsoft Word) lub opcji **Styl strony** z menu **Format** (LibreOffice Writer).

Ćwiczenie 13. Drukujemy dokument tekstowy

1. Sprawdź, jakie opcje wydruku można ustalić w edytorze tekstu, z którego korzystasz.
2. Otwórz plik *Urządzenia* utworzony w ćwiczeniu 12.
3. Wydrukuj ten dokument.

Warto zapamiętać

- Tabulatory stosuje się do przesuwania tekstu w określone miejsce wiersza. Nie należy używać do tego celu spacji.
- Spację nierozdzielającą stosujemy do łączenia trwale dwóch znaków lub dowolnych ciągów znaków, np. spójnika i wyrazu, który znajduje się za nim, aby spójnik nie pozostawał na końcu wiersza.
- Do umieszczenia informacji w punktach i podpunktach należy stosować listy numerowane, wypunktowane lub wielopoziomowe.
- Do tekstu można wstawić tabelę. W komórkach tabeli można umieszczać m.in. tekst, liczby, obrazy, wzory matematyczne.

Pytania i polecenia

1. W jaki sposób można szybko wyrównać liczby w kolumnach, nie posługując się tabelą?
2. Do czego służy zwykła spacja, a do czego spacja nierozdzielająca?
3. Dlaczego warto stosować listy numerowane lub wypunktowane?
4. Omów na konkretnym przykładzie, jak utworzyć tabelę i ją sformatować.
5. W jaki sposób wydrukować dokument tekstowy? Jakie przykładowe opcje drukowania można ustawić?

Zadania

1. Korzystając z edytora tekstu, zapisz swój plan lekcji. Nazwę każdego dnia tygodnia umieść w oddzielnej kolumnie. Pogrub nazwy dni. Pod każdą nazwą dnia umieść kolumnę z nazwami przedmiotów nauczania. Obok nazw przedmiotów umieść numery sal lekcyjnych (w osobnej kolumnie). W pierwszej kolumnie wpisz godzinę rozpoczęcia i zakończenia lekcji – według wzoru (rys. 9.). Pamiętaj o możliwości wprowadzenia indeksu górnego. Wybierz poziomy układ strony. Zapisz plik pod nazwą *Plan1*.

	Poniedziałek		Wtorek	
$8^{00} - 8^{45}$	matematyka	12	chemia	10
$8^{55} - 9^{40}$	język polski	35		
itd.				

Rys. 9. Wzór – zadanie 1.

2. Napisz sprawozdanie z zawodów sportowych w pływaniu. Wpisz nazwiska i imiona zawodników, styl, dystans i uzyskane czasy według wzoru (rys. 10.). Dopisz jeszcze dane kilku osób. Zapisz dokument w pliku pod nazwą *Pływanie*.
Wskazówka: W kolumnie *Styl pływania* zastosuj tabulator lewy, w kolumnie *Dystans* – prawy, a do zapisu czasu – dziesiętny.

Nazwisko i imię	Styl pływania	Dystans [m]	Czas [s]
Bukowska Kasia	dowolny	50	35,58
Jankowska Basia	motylkowy	50	39,23
Nowak Jola	klasyczny	50	43,74
itd.			

Rys. 10. Wzór – zadanie 2.

3. Opisz w kilku akapitach, czym będziesz się zajmować po lekcjach. Potraktuj akapity jako kolejne punkty. Zapisz dokument w pliku pod nazwą *Czas wolny*.
4. Otwórz plik *Czas wolny* zapisany w zadaniu 3. i wstaw tam, gdzie to niezbędne, spacje nierozdzielające. Zapisz plik pod tą samą nazwą.
5. Otwórz plik *Czas wolny* zapisany w zadaniu 4. Dodaj do punktów podpunkty, uszczegóławiając opis spędzania czasu wolnego.
6. Przygotuj tabelę z nazwami szkolnych przedmiotów i swoimi bieżącymi ocenami. Zapisz dokument w pliku pod nazwą *Oceny3*.
7. Przedstaw w tabeli poznane wzory na pola i obwody figur płaskich: kwadratu, trójkąta, trapezu, równoległoboku, rombu, koła. Wstaw do tabeli odpowiednie rysunki z opisem. Wyrównaj teksty w komórkach do środka w poziomie. Na rysunku 11. pokazano wzór tabeli. Zapisz dokument w pliku pod nazwą *Figury*.

Lp.	Nazwa	Pole	Obwód	Rysunek
1.	Kwadrat	a^2	$4a$	

Rys. 11. Wzór – zadanie 7.

8. Utwórz tabelę z nazwami państw należących do Unii Europejskiej i ich stolic. W tabeli umieść dodatkowe informacje o danym państwie: powierzchnia całkowita, liczba ludności, język urzędowy. Postaraj się sformatować tabelę tak, aby była jak najbardziej przejrzysta i czytelna (patrz wzory tabel na rysunkach 7., 8. i 11.).

Dla zainteresowanych

9. Przygotuj tabelę podobną do tej z zadania 7. i umieść w niej wzory na objętość brył. W tabeli umieść również odpowiednie rysunki z opisami. Zapisz dokument w pliku pod nazwą *Bryły*.
10. Twoja klasa wyjeżdża na wycieczkę do Trójmiasta. Zaprojektuj tabelę, w której umieścisz podstawowe informacje o trzech miastach: Gdańsku, Gdyni i Sopocie. Wybierz po trzy najciekawsze miejsca do zwiedzenia w każdym z miast i umieść krótką informację na ich temat w jednej z kolumn tabeli. Potrzebne informacje wyszukaj w podręcznikach, Internecie lub innych źródłach.

Temat 2. Praca z dokumentem wielostronicowym

1. Wstawianie informacji do stopki i nagłówka
2. Szybkie wyszukiwanie słów i znaków
3. Tworzenie przypisów
4. Podział tekstu na kolumny
5. Statystyka dokumentu tekstowego

Warto powtórzyć

1. Jakie są podstawowe zasady redagowania tekstu?
2. Na czym polega dostosowanie formy dokumentu do jego treści?
3. W jaki sposób wykonuje się kopiowanie dokumentów z dysku twardego na inny nośnik pamięci?
4. Jakie znaki nie są widoczne na wydruku? Podaj przykłady.

1. Wstawianie informacji do stopki i nagłówka

Jeśli tworzymy dokument składający się z wielu stron, warto ponumerować kolejne strony, a także wpisać dodatkowe informacje do **nagłówka** i **stopki**. Edytory tekstu, poza możliwością redagowana stopki i nagłówka, udostępniają inne funkcje usprawniające pracę z dokumentem wielostronicowym.

Numery stron można umieścić na górze lub na dole strony, wykorzystując opcję **Numer strony** umieszczoną na karcie **Wstawianie** (Microsoft Word) lub w menu **Wstaw** (LibreOffice Writer). Po wstawieniu numery stron pojawią w **nagłówku** (w programie LibreOffice Writer zwanym **główką**) lub w **stopce** dokumentu.

N **Nagłówek i stopka**

Odpowiednio górna i dolna część strony dokumentu, których zawartość pojawia się automatycznie na każdej stronie dokumentu.

W nagłówku (główce) lub stopce oprócz numeru strony można umieścić także inne informacje, np. o autorze tekstu (imię i nazwisko), tytuł tekstu lub tytuł rozdziału i numer strony. W pismach firmowych w nagłówku umieszcza się dane firmy (nazwę, adres, numery telefonu i faksu, adres e-mail).

Aby wpisać treść nagłówka (główki) lub stopki, można kliknąć dwukrotnie w obszarze nagłówka lub stopki lub wybrać odpowiednią opcję na karcie **Wstawianie** (Microsoft Word) lub w menu **Wstaw** (LibreOffice Writer). Żeby zakończyć redakcję nagłówka lub stopki, należy kliknąć dwukrotnie w obszarze tekstu zasadniczego.

Ćwiczenie 1. Przeglądamy dokumenty tekstowe

Sprawdź, jakie informacje umieszczono w nagłówkach i stopkach wybranych dokumentów (np. pobranych z Internetu).

Ćwiczenie 2. Wpisujemy informacje do nagłówka i stopki

1. Sprawdź w **Pomocy** dołączonej do edytora tekstu, w jaki sposób wstawia się nagłówek i stopkę w programie, którego używasz.
2. Otwórz ostatnio tworzony dokument tekstowy i umieść w nagłówku, po lewej stronie, swoje imię i nazwisko, a w stopce – wyrównany do środka numer strony.

Ćwiczenie 3. Wpisujemy informacje do stopki i nagłówka

1. Odszukaj w Internecie treść księgi pierwszej „Pana Tadeusza" Adama Mickiewicza. Skopiuj tekst do dokumentu tekstowego.
2. Odczytaj, ile stron ma tekst (liczba stron dokumentu jest widoczna zwykle na dole okna programu).
3. Wstaw do stopki numery stron wyrównane do prawego marginesu.
4. W nagłówku (główce) umieść informacje o autorze i tytule utworu.
5. Do tekstu umieszczonego w nagłówku i stopce zastosuj czcionkę Arial o rozmiarze 10 punktów. W nagłówku użyj czcionki koloru niebieskiego.
6. Zapisz plik pod nazwą *Pan Tadeusz I*.

W niektórych dokumentach trzeba wpisać inne informacje w stopce i nagłówku na stronach parzystych, a inne na nieparzystych, np. w stopce na stronach parzystych numer strony trzeba wpisać po lewej stronie, a na stronach nieparzystych po prawej, w nagłówku na stronach parzystych trzeba wpisać tytuł rozdziału, a na stronach nieparzystych – podrozdziału.

Aby zróżnicować treść nagłówka i stopki dla stron parzystych i nieparzystych, należy po uaktywnieniu nagłówka (stopki) zaznaczyć taką możliwość w narzędziach nagłówków i stopek (Microsoft Word) lub w programie LibreOffice Writer kliknąć przycisk **Główka** (**Stopka**) i z listy wyboru wybrać polecenie **Formatowanie główki** (**stopki**). Następnie można zredagować nagłówek i stopkę oddzielnie dla strony parzystej i dla nieparzystej.

Ćwiczenie 4. Wpisujemy inną informację do nagłówka i stopki na stronach parzystych, a inną na nieparzystych

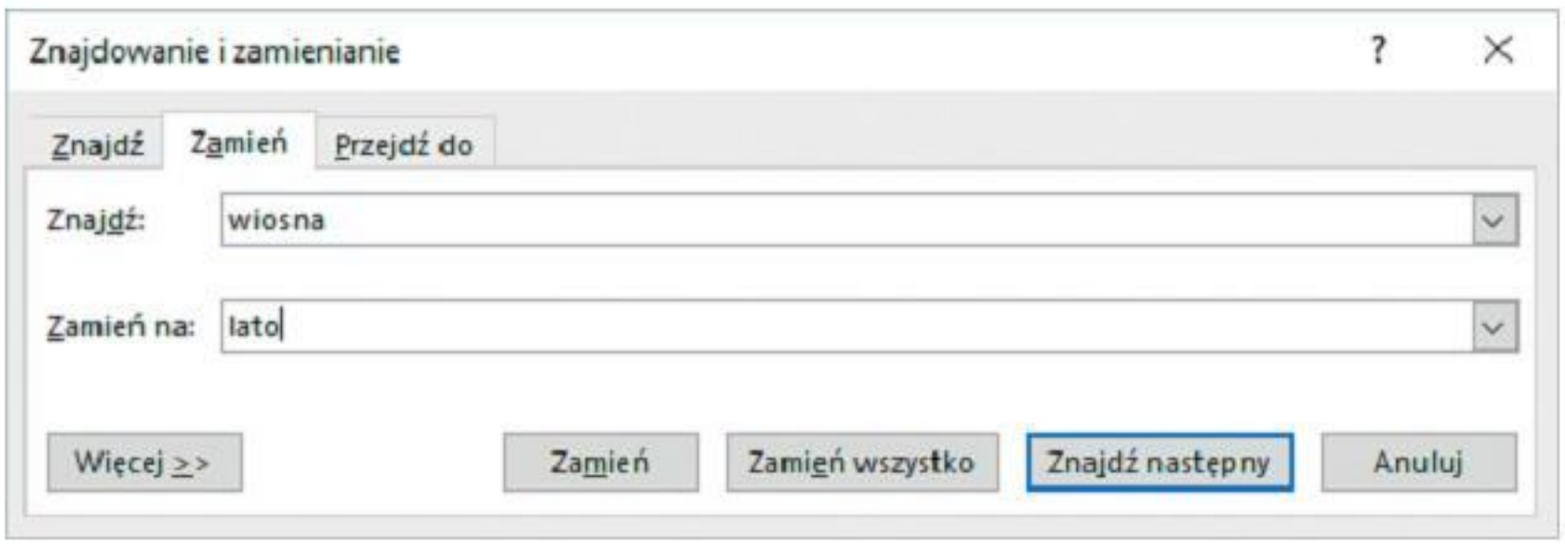
1. Odszukaj w Internecie treść księgi trzeciej „Pana Tadeusza" Adama Mickiewicza.
2. Wstaw w stopce na stronach nieparzystych numer strony po prawej stronie, a na stronach parzystych – po lewej.
3. Wstaw w nagłówku (główce) na stronach nieparzystych, po prawej stronie, imię i nazwisko autora oraz tytuł „Pan Tadeusz", a na stronach parzystych, po lewej stronie – napis „Księga trzecia UMIZGI". Zastosuj czcionkę Arial, pochyloną, pogrubioną, o rozmiarze 10 punktów.
4. Zapisz plik pod nazwą *Pan Tadeusz III*.

2. Szybkie wyszukiwanie słów i znaków

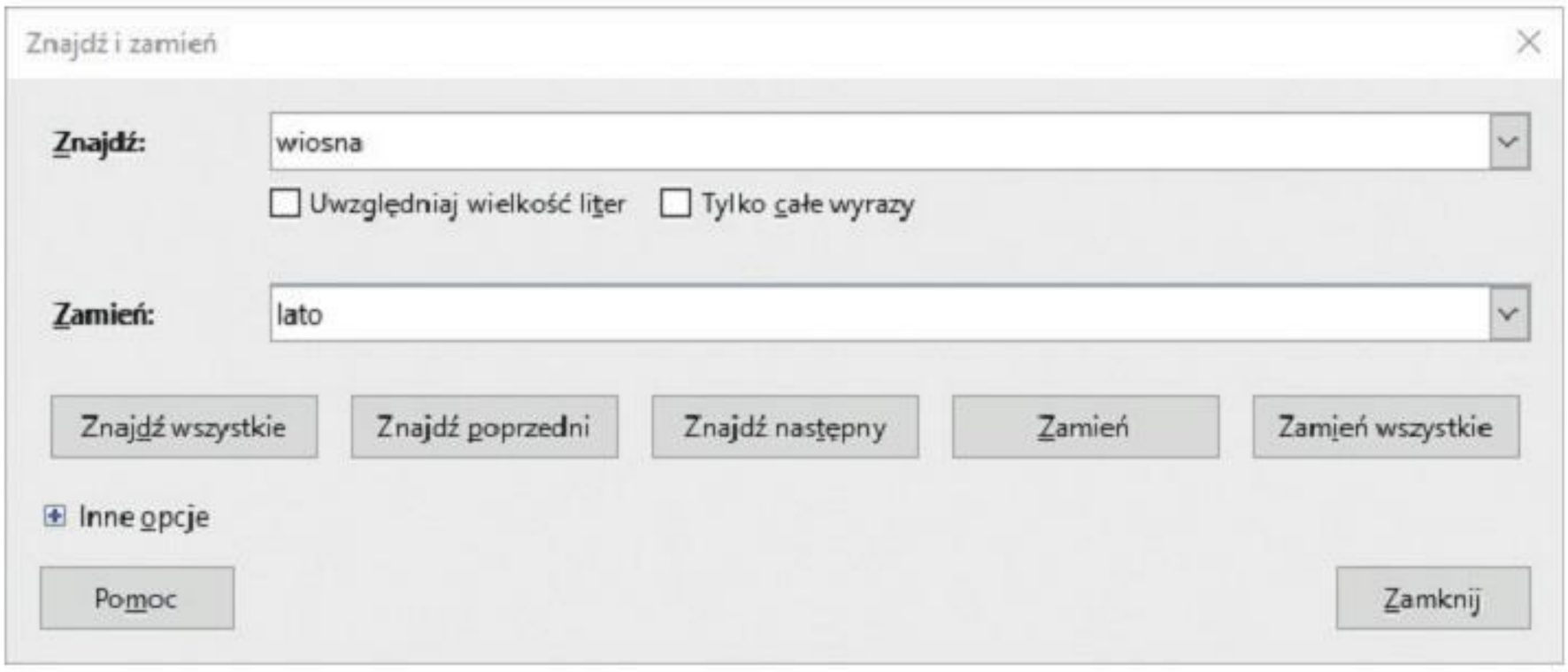
W napisanym w edytorze tekstu referacie z historii nazwę jednej miejscowości kilkakrotnie wpisaliśmy z błędem literowym. Jak można szybko skorygować omyłkę w całym tekście?

Edytory tekstu umożliwiają wyszukiwanie słów, pojedynczych znaków i dowolnych ciągów znaków. Można zamienić np. wszystkie przecinki na średniki. Można również wyszukać i zamienić znaki niewidoczne na wydruku, np. spację na spację nierozdzielającą. Przed zamianą słów lub znaków należy zastanowić się, czy wszystkie zamiany w tekście będą poprawne. Czasem zamianę trzeba wykonać osobno dla każdego wyszukanego słowa lub znaku.

Aby znaleźć wyraz, znak lub ciąg znaków i zamienić je na inne, należy wybrać opcję **Zamień** (lub **Znajdź i zamień**). Opcje te znajdują się zwykle w grupie **Edytowanie** (w **Narzędziach głównych**) lub w menu **Edycja**.



Rys. 1a. Okno **Znajdowanie i zamienianie** (Microsoft Word)



Rys. 1b. Okno **Znajdź i zamień** (LibreOffice Writer)

Ćwiczenie 5. Wyszukujemy i zamieniamy słowa

1. Otwórz plik *Pan Tadeusz III* zapisany w ćwiczeniu 4. Zamień imiona Tadeusz na Adam. Zwróć uwagę, czy zostanie zachowana odpowiednia forma lub odmiana imienia.
2. Nie zapisuj zmian.

Korzystając z możliwości zamiany znaków, można także usuwać niepotrzebne spacje, znaki akapitu lub inne znaki. Wówczas pole **Zamień na** (**Zamień**), do którego wpisujemy tekst zastępujący wyszukiwany tekst (rys. 1a i 1b), należy pozostawić puste.

Więcej możliwości wyszukiwania otrzymamy po wybraniu przycisku **Więcej** (Microsoft Word) lub **Inne opcje** (LibreOffice Writer). W programie Microsoft Word opcję zamiany znaków niewidocznych na wydruku można znaleźć w oknie **Znajdowanie i zamienianie** po naciśnięciu przycisku **Więcej**, a następnie przycisku **Specjalne**.

Ćwiczenie 6. Wyszukujemy i zamieniamy znaki

1. Otwórz plik *Czas wolny* zapisany w zadaniu 5. z tematu 1.
2. Korzystając z możliwości wyszukiwania i zamiany znaków, połącz wszystkie akapity tekstu. Odpowiedz ustnie na pytanie: *Jakiego ciągu znaków będziesz szukać i na co go zamienisz?* Korzystając z możliwości wyszukiwania i zamiany znaków, sprawdź, czy w tekście została zachowana zasada: „stawiaj spację po kropce, a nie przed nią". Popraw wyszukane błędnie wstawione spacje. Odpowiedz ustnie na pytanie: *Jakiego ciągu znaków będziesz szukać i na co go zamienisz?*
3. W podobny sposób sprawdź, czy spacje wstawiono po przecinkach i innych znakach interpunkcyjnych, a nie przed nimi. Popraw błędnie wstawione spacje. Każdorazowo odpowiedz ustnie na pytanie: *Jakiego ciągu znaków będziesz szukać i na co go zamienisz?*
4. Zapisz plik pod tą samą nazwą.

3. Tworzenie przypisów

Przypisy stosuje się w celu podania źródła informacji lub zapisania komentarza bądź wyjaśnienia. Przypis pojawia się u dołu danej strony i w tym miejscu możemy go edytować. Przypis może pojawić się również na końcu całego tekstu (tzw. przypis końcowy). Prostsze edytory tekstu, tj. WordPad i Notatnik, nie mają możliwości wstawiania przypisów.

Po wybraniu opcji wstawiania przypisu dolnego na karcie **Odwołania** (Microsoft Word) lub w menu **Wstaw** (LibreOffice Writer) w miejscu położenia kursora tekstowego pojawia się odnośnik do wstawionego przypisu. Ten sam odnośnik pojawia się u dołu danej strony i w tym miejscu wprowadzamy treść przypisu.

Gdy klikniemy dwukrotnie (Microsoft Word) lub jednokrotnie (LibreOffice Writer) odnośnik przypisu w tekście, zostaniemy przeniesieni do treści przypisu u dołu strony, natomiast wskazanie odnośnika kursorem myszy powoduje wyświetlenie treści przypisu bezpośrednio przy odnośniku.

Ćwiczenie 7. Umieszczamy przypisy dolne w tekście

1. Odszukaj w Internecie informacje na temat tabletów. Skopiuj do dokumentu tekstowego krótkie informacje o trzech wybranych tabletach. Do nazwy każdego tabletu zastosuj przypis dolny, w którym umieść źródło informacji, czyli adres strony internetowej.
2. Zapisz plik pod nazwą *Przypisy*.

4. Podział tekstu na kolumny

Tekst w kolumnach najczęściej przygotowuje się do artykułów gazetowych, folderów, ogłoszeń. Sposób wyświetlania tekstu w kolumnach – ich liczbę, szerokość, odstępy można ustalić, korzystając z opcji **Kolumny** na karcie **Układ** w grupie **Ustawienia strony** (Microsoft Word) lub w menu **Format** (LibreOffice Writer).

Jeśli w kolumnach chcemy umieścić fragment istniejącego tekstu, należy go wcześniej zaznaczyć. Szerokość kolumny i odstęp między kolumnami można ustawić, przesuwając odpowiednie suwaki na linijce (podobnie jak w przypadku zmiany rozmiarów komórek tabel czy szerokości marginesów).

S **Sekcja**

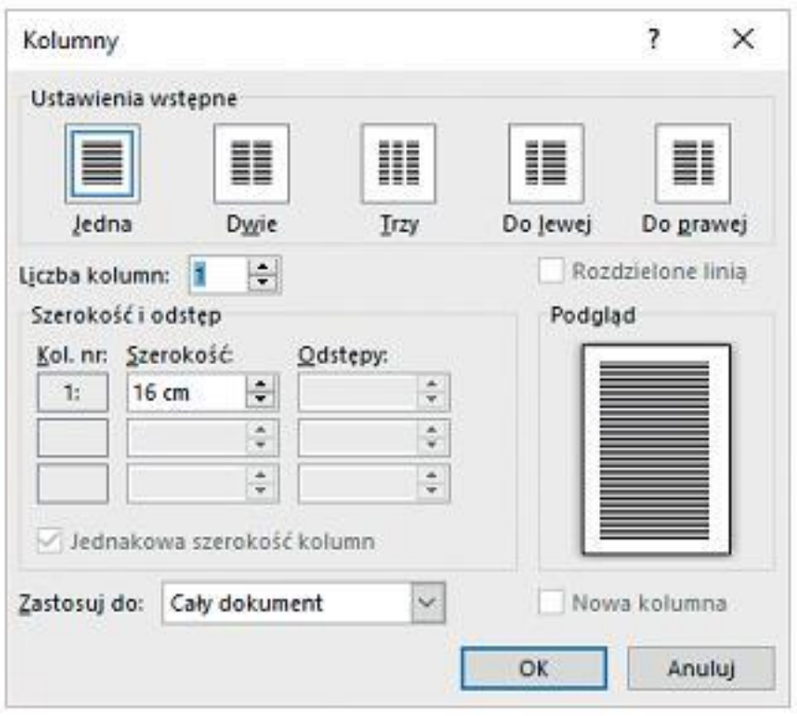
Część dokumentu, dla której możemy ustalać inne od użytych w pozostałych częściach dokumentu parametry formatowania strony, np. układ poziomy.

Dobra rada

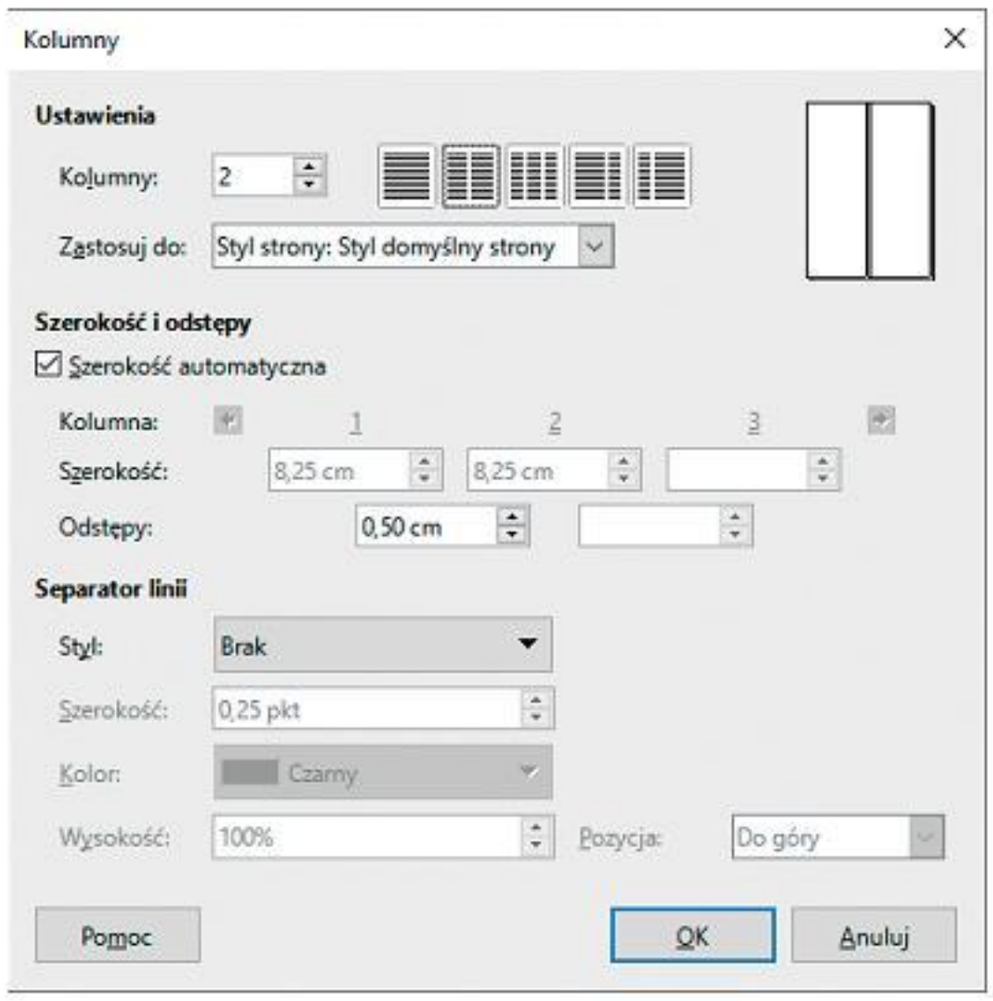
Aby umieścić tekst w kolumnach o tej samej długości, należy wstawić na jego końcu **ciągły znak podziału sekcji**, korzystając z opcji **Znaki podziału** na karcie **Układ** w grupie **Ustawienia strony** (Microsoft Word) lub z opcji **Sekcja** w menu **Wstaw** (LibreOffice Writer).

Ćwiczenie 8. Umieszczamy tekst w kolumnach

1. Otwórz plik *Pan Tadeusz I* zapisany w ćwiczeniu 3.
2. Odszukaj w swoim edytorze tekstu opcję umieszczania tekstu w kolumnach.
3. Umieść tekst w dwóch kolumnach. Dobierz szerokość kolumn, odstęp między nimi oraz rozmiar czcionki tak, aby nie zmieniać układu wierszy oryginalnego tekstu (nie łamać wierszy).
4. Zapisz plik pod tą samą nazwą.



Rys. 2a. Okno, w którym można ustalić sposób formatowania tekstu w kolumnach (Microsoft Word)



Rys. 2b. Okno, w którym można ustalić sposób formatowania tekstu w kolumnach (LibreOffice Writer)

5. Statystyka dokumentu tekstowego

Musimy przygotować w edytorze tekstu krótką notatkę do szkolnej gazetki, która powinna zawierać 500 znaków razem ze znakami spacji. W jaki sposób można szybko policzyć znaki w napisanym tekście?

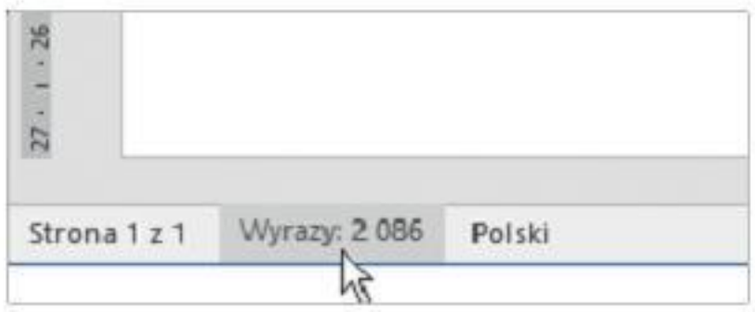
Uwaga

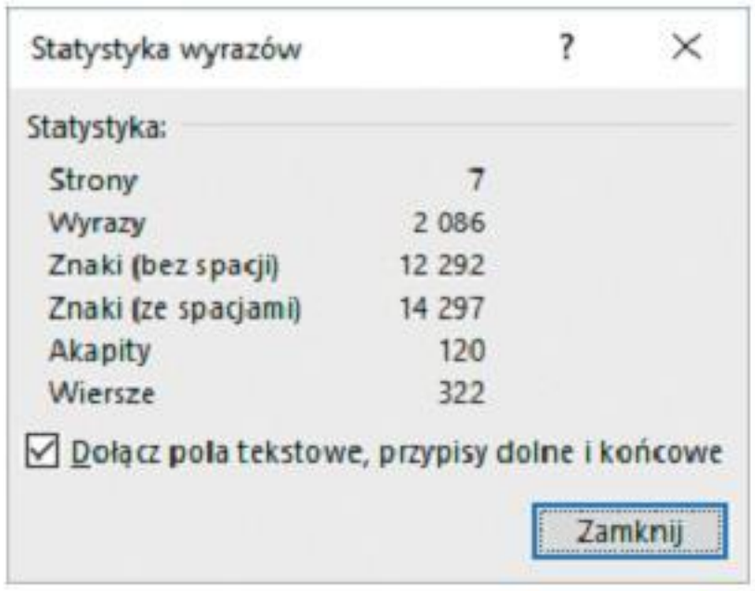
W edytorze tekstu każdy ciąg znaków oddzielony od innego ciągu spacjami lub innym separatorem (m.in. znakiem tabulatora, akapitu, podziału wiersza) jest traktowany jako wyraz.

Edytory tekstu umożliwiają zliczanie wyrazów w dokumencie. Zależnie od wersji edytora można sprawdzić, ile stron, akapitów, wierszy, znaków (w tym spacji) zawiera dokument.

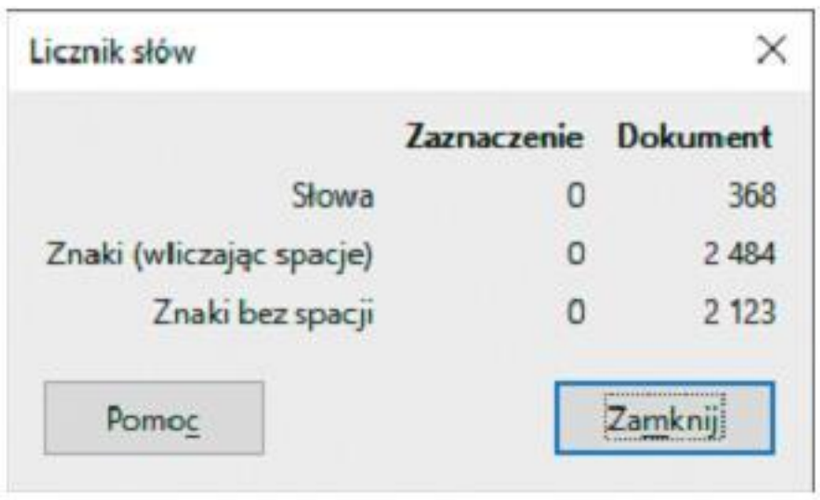
Aby otworzyć okno **Statystyka wyrazów**, można:

- w programie Microsoft Word kliknąć przycisk z informacją o liczbie wyrazów wyświetlaną u dołu obszaru roboczego okna (rys. 3a),
- w programie LibreOffice Writer wybrać opcję **Narzędzia/Licznik słów** (rys. 3b).





Rys. 3a. Okno **Statystyka wyrazów** (Microsoft Word)



Rys. 3b. Okno **Licznik słów** (LibreOffice Writer)

Ćwiczenie 9. Korzystamy ze statystyki wyrazów

1. Otwórz plik *Pan Tadeusz I* zapisany w ćwiczeniu 8.
2. Sprawdź, ile wyrazów i znaków z wyjątkiem spacji zawiera tekst księgi pierwszej „Pana Tadeusza".

Wskazówka: W programie Microsoft Word opcję **Statystyka wyrazów** można również znaleźć na karcie **Recenzja**.

Warto zapamiętać

- W dokumencie wielostronicowym:
  - numery stron należy umieścić w stopce lub nagłówku,
  - zależnie od przeznaczenia dokumentu, w nagłówku i stopce można wpisać informacje o autorze, tytuł rozdziału, podrozdziału, dane firmy: nazwę, adres itp.
- O ile to możliwe, należy automatyzować wykonywane czynności, stosując np. wyszukiwanie i zamianę znaków.
- Jeśli chcemy podać źródło informacji lub zapisać komentarz od autora, stosujemy przypisy.
- Tekst można umieszczać w kolumnach. Najpierw trzeba ustalić ich liczbę, szerokość i odstęp między nimi.
- Edytory tekstu umożliwiają zliczanie znaków, wyrazów i akapitów w dokumencie tekstowym.

Pytania i polecenia

1. Jaką funkcję pełnią w dokumencie stopka i nagłówek?
2. W jaki sposób wpisuje się informacje do stopki i nagłówka? Omów na przykładzie.
3. Omów na przykładzie, w jaki sposób można zróżnicować treść nagłówka i stopki dla stron parzystych i nieparzystych.
4. W jaki sposób można wyszukać w dokumencie tekst lub znaki specjalne i zamienić na inne?
5. Do czego służą przypisy? Omów na przykładzie sposób wstawiania przypisów dolnych do dokumentu tekstowego.
6. W jaki sposób należy umieścić fragment tekstu w trzech kolumnach o różnej szerokości?
7. W jaki sposób można sprawdzić, z ilu wyrazów składa się tekst?

Zadania

1. Utwórz nowy dokument tekstowy. Przygotuj wzór pisma firmowego twojej szkoły. W nagłówku umieść nazwę oraz logo szkoły, a w stopce – adres, numer telefonu i faksu, adres WWW, adres e-mail. Nazwę szkoły umieść pośrodku nagłówka, odpowiednio sformatuj. Nazwę szkoły możesz zapisać, korzystając np. z galerii WordArt. Jeśli szkoła nie ma swojego logo, zaproponuj własny rysunek lub skorzystaj z ilustracji dostępnych w Internecie (opcja **Obrazy online** w Microsoft Word). Odpowiednio sformatuj wprowadzone teksty.
2. Utwórz nowy dokument tekstowy. Opisz na jednej stronie ostatnio przeczytaną książkę (lub obejrzany film). Zapisz plik pod nazwą określającą tytuł książki (filmu). Sprawdź, czy w tekście została zachowana: „między wyrazami należy wstawiać jedną spację". Korzystając z możliwości wyszukiwania i zamiany znaków, popraw tekst. Następnie wyjustuj tekst i przenieś pojedyncze spójniki i przyimki pozostawione na końcu wiersza do następnego wiersza. Zapisz plik pod tą samą nazwą.
3. Otwórz plik *Pan Tadeusz III* zapisany w ćwiczeniu 4.
   a. Jeśli w dokumencie zastosowano ręczne podziały wiersza, zamień je na znaki końca akapitów.
   b. Jeśli w dokumencie zastosowano znaki końca akapitów, zamień je na ręczne podziały wiersza.

4. Otwórz plik *Przypisy* zapisany w ćwiczeniu 7. Odszukaj w Internecie informacje na temat laptopów. Skopiuj do dokumentu tekstowego krótkie informacje o trzech wybranych laptopach. Do nazwy każdego laptopa zastosuj przypis dolny i umieść w nim źródło informacji, czyli adres strony internetowej. Zapisz plik pod tą samą nazwą.
5. Otwórz plik zapisany w zadaniu 2. Umieść pierwszą część tekstu w dwóch kolumnach o tej samej długości, a drugą część w trzech kolumnach o tej samej długości. Pamiętaj o przeniesieniu pojedynczych spójników i przyimków pozostających na końcu wiersza do następnego wiersza. Zapisz plik pod tą samą nazwą.
   Wskazówka: Postaraj się tak podzielić tekst, aby podział był uzasadniony merytorycznie.
6. Napisz w edytorze tekstu krótką notatkę z lekcji informatyki (temat 2.), zawierającą nie więcej niż 600 znaków (nie licząc spacji).

Dla zainteresowanych

7. Utwórz nowy dokument tekstowy. Opisz w nim zasady redagowania tekstu. W dokumencie zastosuj poznane zasady formatowania i redagowania tekstu.
8. Utwórz nowy dokument tekstowy. Opisz korzyści i zagrożenia wynikające z rozwoju informatyki i powszechnego dostępu do informacji oraz wyjaśnij zagrożenia wynikające z uzależnienia się od komputera. W dokumencie zastosuj poznane zasady formatowania i redagowania tekstu. Tekst umieść w dwóch kolumnach, tak aby w jednej znalazł się opis korzyści, a w drugiej – zagrożeń.

Temat 3. Tworzenie e-gazetki – projekt

1. Praca w chmurze
2. Przygotowanie projektu grupowego
3. Projekt grupowy – e-gazetka

Warto powtórzyć

1. W jaki sposób można przenieść wydrukowany obraz do komputera?
2. Jakie są podstawowe zasady redagowania tekstu?
3. Jakie są najważniejsze zasady ustalania parametrów formatowania tekstu?
4. W jaki sposób można wstawić obraz do dokumentu tekstowego?
5. Jakie są możliwości formatowania obrazu wstawionego do tekstu?
6. W jaki sposób umieszcza się tekst w kolumnach?
7. W jaki sposób można sprawdzić, z ilu wyrazów i znaków składa się tekst?

1. Praca w chmurze

Internet umożliwia przechowywanie danych w tzw. chmurze, a dokładnie – na dyskach serwerów internetowych firm oferujących taką usługę.

Operacje na danych (tj. umieszczanie plików w chmurze, przeglądanie ich i pobieranie z chmury) można wykonywać za pomocą przeglądarki internetowej lub specjalnej aplikacji, którą należy zainstalować na komputerze.

Dostęp do danych w chmurze można uzyskać z dowolnego komputera lub urządzenia mobilnego podłączonego do Internetu w dowolnym miejscu na świecie i w dowolnym momencie. Oprogramowanie dba o to, aby dane na wszystkich komputerach były identyczne (proces ten nazywa się **synchronizacją danych**).

Uzyskanie przestrzeni dyskowej w chmurze umożliwiają rozwiązania takie, jak: Microsoft OneDrive, Dropbox, Dysk Google.

Sposób korzystania z chmury wyjaśnimy na przykładzie usługi Dysk Google.

Po zalogowaniu się na konto Google, możemy:

- umieszczać w chmurze pliki (zdjęcia, obrazy i inne dokumenty komputerowe),
- udostępniać dane innym użytkownikom, m.in.:
  - udostępnić plik,
  - udostępnić wybrany folder,
  - umożliwić współdzielenie danego folderu (inny użytkownik może nie tylko przeglądać folder, pobierać pliki, lecz także umieszczać swoje pliki w tym folderze),
- tworzyć dokumenty komputerowe z wykorzystaniem aplikacji umieszczonych w chmurze,
- pracować grupowo nad jednym dokumentem.

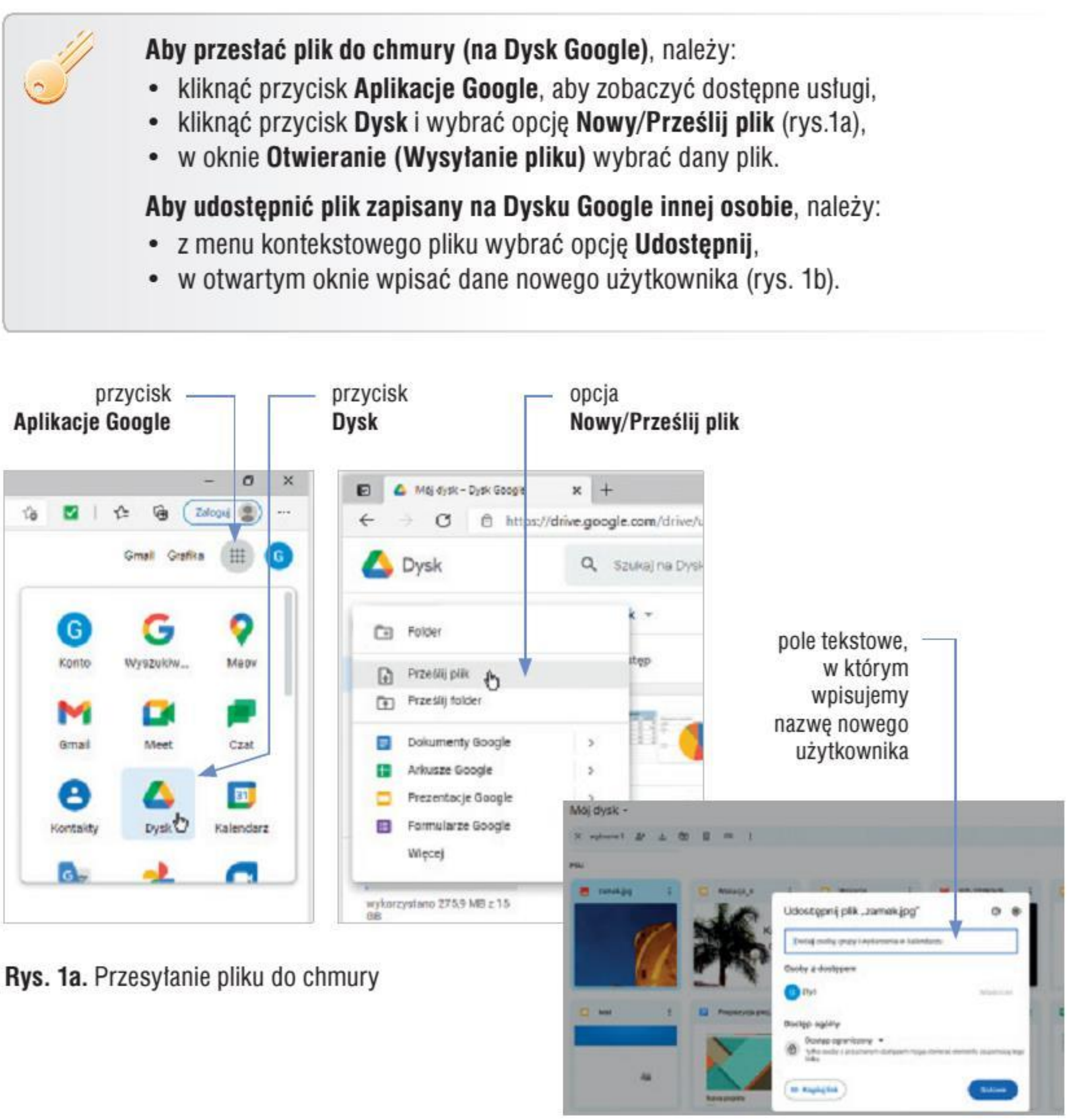
Uwaga: Jeśli mamy założone konto Gmail, wystarczy się na nie zalogować.

Aby przesłać plik do chmury (na Dysk Google), należy:

- kliknąć przycisk **Aplikacje Google**, aby zobaczyć dostępne usługi,
- kliknąć przycisk **Dysk** i wybrać opcję **Nowy/Prześlij plik** (rys.1a),
- w oknie **Otwieranie (Wysyłanie pliku)** wybrać dany plik.

Aby udostępnić plik zapisany na Dysku Google innej osobie, należy:

- z menu kontekstowego pliku wybrać opcję **Udostępnij**,
- w otwartym oknie wpisać dane nowego użytkownika (rys. 1b).



Rys. 1a. Przesyłanie pliku do chmury

Rys. 1b. Udostępnianie innej osobie pliku zapisanego na Dysku Google – ćwiczenie 1.

Ćwiczenie 1. Umieszczamy plik w chmurze i udostępniamy go innemu użytkownikowi

1. Zaloguj się na konto Google.
2. Umieść wybrany plik w chmurze, a następnie udostępnij go koledze lub koleżance.

Wskazówka: Sprawdź, jakie masz możliwości udostępniania pliku.

Ćwiczenie 2. Udostępniamy link do folderu innemu użytkownikowi

1. Sprawdź w usłudze Dysk Google, w jaki sposób umieścić folder w chmurze.
2. Umieść wybrany folder (np. z tekstami do projektu) w chmurze i prześlij link do folderu wybranej osobie z grupy.

Wskazówka: Z menu kontekstowego pliku wybierz polecenie **Pobierz link**.

W usłudze Dysk Google możemy tworzyć dokumenty, wykorzystując aplikacje umieszczone w chmurze, tj.: Dokumenty Google, Arkusze Google, Prezentacje Google i inne. Można je wybrać z listy po kliknięciu przycisku **Nowy**.

Ćwiczenie 3. Tworzymy dokument tekstowy z wykorzystaniem aplikacji Dokumenty Google

1. W aplikacji Dokumenty Google utwórz nowy dokument tekstowy. Napisz w nim jedno zdanie (np. tytuł projektu z punktu 3.).
2. Nazwij dokument *Projekt1* i prześlij go do folderu umieszczonego w chmurze w ćwiczeniu 2.

2. Przygotowanie projektu grupowego

Możliwości pracy w chmurze oraz wyszukiwania informacji w Internecie, a także poznane zasady tworzenia strony internetowej wykorzystamy podczas pracy nad projektami.

Zanim przystąpimy do wykonywania zadań projektowych, powinniśmy się zastanowić, jakich narzędzi użyjemy. Na przykład do tworzenia strony internetowej możemy wykorzystać edytor plików tekstowych Notatnik i język HTML lub utworzyć stronę w postaci bloga. Do zaplanowania podróży możemy wykorzystać wyszukiwarkę internetową, edytor tekstu i arkusz kalkulacyjny.

Na etapie planowania określamy zadania i gromadzimy odpowiednie materiały. Ustalamy sposób przechowywania dokumentów komputerowych i ich udostępniania, np. z wykorzystaniem chmury. Aby zebrać informacje, korzystamy z różnych źródeł, m.in. encyklopedii, w tym encyklopedii multimedialnych i internetowych, słowników (także internetowych), podręczników szkolnych, czasopism, prowadzimy wywiady.

Po zgromadzeniu informacji porządkujemy je i wybieramy najistotniejsze. Należy pamiętać, że nie wszystkie informacje pochodzące z Internetu są rzetelne. Najlepiej korzystać z poleconych przez nauczyciela portali edukacyjnych i stron zaufanych wydawców.



Jeżeli wykorzystujemy cudze materiały, powinniśmy pamiętać o przestrzeganiu przepisów dotyczących korzystania z cudzych materiałów. Pamiętaj, że bez ograniczeń można korzystać z grafiki i innych utworów w tzw. domenie publicznej, ponieważ autorskie prawa majątkowe do tych utworów wygasły lub autorzy zrzekli się ich.

Autorskie prawa majątkowe wygasają po siedemdziesięciu latach od śmierci twórcy, a w przypadku utworu nieznanego autora – po siedemdziesięciu latach od daty pierwszego rozpowszechnienia (publicznego udostępnienia).

W Wikipedii zbiory znajdujące się w domenie publicznej oznaczono symbolem: .

Nad projektem grupowym pracuje zespół uczniów. W pracy zespołowej istotne jest dobre zaplanowanie działań i przydział zadań szczegółowych.

Etapy tworzenia projektu grupowego

1. Określenie zadania.
2. Analiza zadania i ustalenie zadań szczegółowych.
3. Określenie formy dokumentów oraz dobór nazw plików.
4. Wybranie odpowiednich narzędzi – programów komputerowych.
5. Ustalenie założeń dotyczących łączenia dokumentów.
6. Przydzielenie zadań poszczególnym osobom (grupom).
7. Wykonanie zadań szczegółowych.
8. Połączenie dokumentów.
9. Sprawdzenie, czy zadania wykonano według wcześniejszych założeń.
10. Wydrukowanie wybranych dokumentów.

Uwagi:

- W trakcie pracy nad dokumentem komputerowym należy zapisywać go co jakiś czas w wybranym folderze. Ważne jest wcześniejsze ustalenie nazw plików.
- Jeśli nie chcemy, aby pliki zostały nadpisane (w przypadku kopiowania plików do tego samego folderu), należy dopilnować, aby każdy został inaczej nazwany.
- Jeżeli komputery pracują w sieci lokalnej, należy korzystać z możliwości przesyłania plików poprzez sieć w celu skopiowania ich czy przeniesienia na inny komputer.

3. Projekt grupowy – e-gazetka

Gazetka szkolna może być przygotowana w tradycyjnej, papierowej formie. Wiele gazet i czasopism ma jednak także swoje e-wydania, dlatego przygotujemy szkolną e-gazetkę. Gazetkę utworzymy w edytorze tekstu (Microsoft Word lub LibreOffice Writer).

W przypadku e-gazety otrzymany dokument zapiszemy w formacie **PDF**. Dokument PDF (zapisany w pliku z rozszerzeniem *pdf*) możemy wyświetlić na ekranie monitora lub wydrukować (korzystając np. z darmowego programu Adobe Acrobat Reader). Tak przygotowaną gazetkę możemy udostępnić np. na stronie internetowej szkoły. Jeśli będziemy ją chcieli rozpowszechniać w postaci papierowej, będzie ją można również wydrukować.

Niezależnie od formy wydania gazetki, wykonamy kilka zadań szczegółowych.

Zadanie projektowe – tworzenie e-gazetki

Przykładowy podział zadania na zadania szczegółowe

Zadanie 1. Ogólne ustalenie tematyki gazetki.
Zadanie 2. Opracowanie graficznej koncepcji gazetki.
Zadanie 3. Zebranie materiałów (tekstów, zdjęć) i przygotowanie na ich podstawie artykułów.
Zadanie 4. Redakcja tekstów i uporządkowanie materiałów.
Zadanie 5. Ustalenie kolejności artykułów na poszczególnych stronach i rozmieszczenie ilustracji.
Zadanie 6. Formatowanie tekstu i wstawionych obrazów.
Zadanie 7. Końcowa korekta i ocena gazetki.
Zadanie 8. Zapisanie gazetki w formacie PDF; ewentualne wydrukowanie.

Wskazówki do zadań szczegółowych

1. Tematykę gazetki (zadanie 1.) należy omówić wspólnie. Gazetka może składać się ze stałych rubryk, w których można zamieszczać m.in.: informacje bieżące (szkolne i klasowe), osiągnięcia uczniów (naukowe i sportowe), ciekawostki (informatyczne lub inne), kącik grafiki komputerowej (np. w formie porad, prac uczniowskich), kącik literacki (twórczość uczniów).
   W gazetce mogą się również znaleźć rubryki, których treść będzie inna w każdym numerze, np. osiągnięcia technologii informacyjno-komunikacyjnych, cuda świata, zabytki kultury polskiej lub światowej, znane postacie ze świata i z kraju. Liczba i rodzaj rubryk zależą od inwencji autorów.
2. Opracowaniem graficznym gazetki, czyli tzw. **layoutem** (zadanie 2.), może się zająć mniejsza grupa (najwyżej czteroosobowa). Należy ustalić: format gazety (np. A4), liczbę stron (np. 4), liczbę i szerokość kolumn na poszczególnych stronach, krój i wielkość czcionki dla poszczególnych elementów, m.in. tytułów i tekstu artykułów. Warto opracować wzorzec (tzw. **makietę**) każdej strony. Można ją przygotować np. w edytorze tekstu. Na rysunku 2. pokazano przykładową makietę wykonaną w programie Microsoft Word z włączonymi liniami siatki. Można również przejrzeć przykładowe e-gazety dostępne bezpłatnie w Internecie.
3. Materiały (zdjęcia, rysunki i teksty) powinny być przygotowane w postaci elektronicznej. Jeśli materiały są w postaci papierowej, należy je zeskanować. Należy ustalić jednoznaczne nazwy plików, w których będą zapisywane artykuły i zdjęcia (w nazwie pliku można uwzględnić numer lub nazwę rubryki).
4. Jeśli dany artykuł ma zmieścić się w kolumnie o określonych wcześniej wymiarach, można po ustaleniu kroju i wielkości czcionki ustalić, jaką liczbę znaków może zawierać ten artykuł. Sprawniej będzie wówczas przebiegało **łamanie** – nie trzeba będzie usuwać niemieszczących się fragmentów tekstu. Niektóre informacje – zwłaszcza krótkie – można umieszczać w polu tekstowym. Opcję wstawiania pola tekstowego znajdziesz na karcie **Wstawianie** (Microsoft Word) lub w menu **Wstaw** (LibreOffice Writer). Pole tekstowe można formatować, m.in.: ustalać rozmiary, kolory i oblewać tekstem.

P **PDF**
(ang. *Portable Document Format* – przenośny format dokumentu)

W formacie PDF można zapisać dowolny dokument (np. dokument tekstowy utworzony w edytorze tekstu, dokument arkusza kalkulacyjnego, prezentację multimedialną). Dokumenty w formacie PDF są wyświetlane i drukowane tak samo niezależnie od użytego komputera i systemu operacyjnego.

L **Layout**
(z ang. – *układ*)

Graficzny wzorzec publikacji, obejmujący: wielkość strony, szerokość szpalt (kolumn), krój czcionek, rozmieszczenie stałych elementów strony (np. numerów stron), kolorystykę.

M **Makieta**

Wzorzec rozmieszczenia tekstów oraz ilustracji w szpaltach (kolumnach) na stronie.



Rys. 2. Przykładowy wygląd makiety pierwszej strony gazetki

5. Bardzo ważne jest odszukanie i poprawienie wszystkich błędów redakcyjnych (zadanie 4.). Należy też zadbać, aby jednoliterowe spójniki i przyimki nie pozostawały na końcach wierszy.
6. Po odpowiednim rozmieszczeniu wszystkich tekstów i obrazów oraz dokładnej końcowej **korekcie** tekstu (zadanie 7.), w tym sprawdzeniu zgodności z makietą, dokument można zapisać w formacie PDF.

Korekta K

Poprawianie błędów literowych, interpunkcyjnych, ortograficznych.

Łamanie Ł

Wstawienie do odpowiednich kolumn: tekstów, tytułów, zdjęć, rysunków oraz ich rozmieszczenie według makiety.

Jeżeli program, którego używamy, nie posiada funkcji tworzenia plików PDF, można skorzystać z jednego z wielu darmowych programów (np. PDFCreator), które tworzą w systemie wirtualną drukarkę. Wysłanie polecenia drukowania na takiej drukarce powoduje utworzenie dokumentu PDF.



Aby zapisać dokument w formacie PDF, należy:
- w Microsoft Office – w oknie **Zapisz jako** wybrać typ pliku **PDF** lub w menu **Plik** opcję **Eksportuj/Utwórz dokument PDF**,
- w LibreOffice Writer w menu **Plik** wybrać opcję **Eksportuj jako PDF**.



Przykłady innych projektów grupowych

1. Przygotowanie sprawozdania z klasowego wyjazdu na targi komputerowe.
2. Tworzenie klasowej lub szkolnej kroniki (e-kroniki).
3. Pomoc do nauki geografii: opis kraju, do którego chcielibyśmy pojechać.
   Opis powinien zawierać:
   - ogólne dane o kraju (m.in. liczba ludności, powierzchnia),
   - historię kraju (można np. w tabeli umieścić daty, wydarzenia i osoby z nimi związane),
   - opis zwyczajów, informacje o mieszkańcach, o noszonych strojach tradycyjnych,
   - spis miejsc interesujących pod względem turystycznym, ważniejszych zabytków (rysunki, zdjęcia).

   Każdy z punktów opisu może przygotować inna grupa, a potem należy połączyć wszystkie dokumenty. Rysunki i zdjęcia trzeba włączyć do odpowiednich dokumentów tekstowych. Można je przygotować w edytorze graficznym. Informacje o danym kraju można wyszukać w Internecie lub innych źródłach.
4. Wasza klasa przyjaźni się z klasą z innego miasta (kraju). Należy przygotować (podobnie jak w zadaniu 3.) informacje o zaprzyjaźnionej klasie, szkole, o miejscowości, w której mieści się szkoła, o jej atrakcjach turystycznych.

Algorytmika i programowanie

Rozdział II

Temat 4. Wprowadzenie do programowania w języku C++

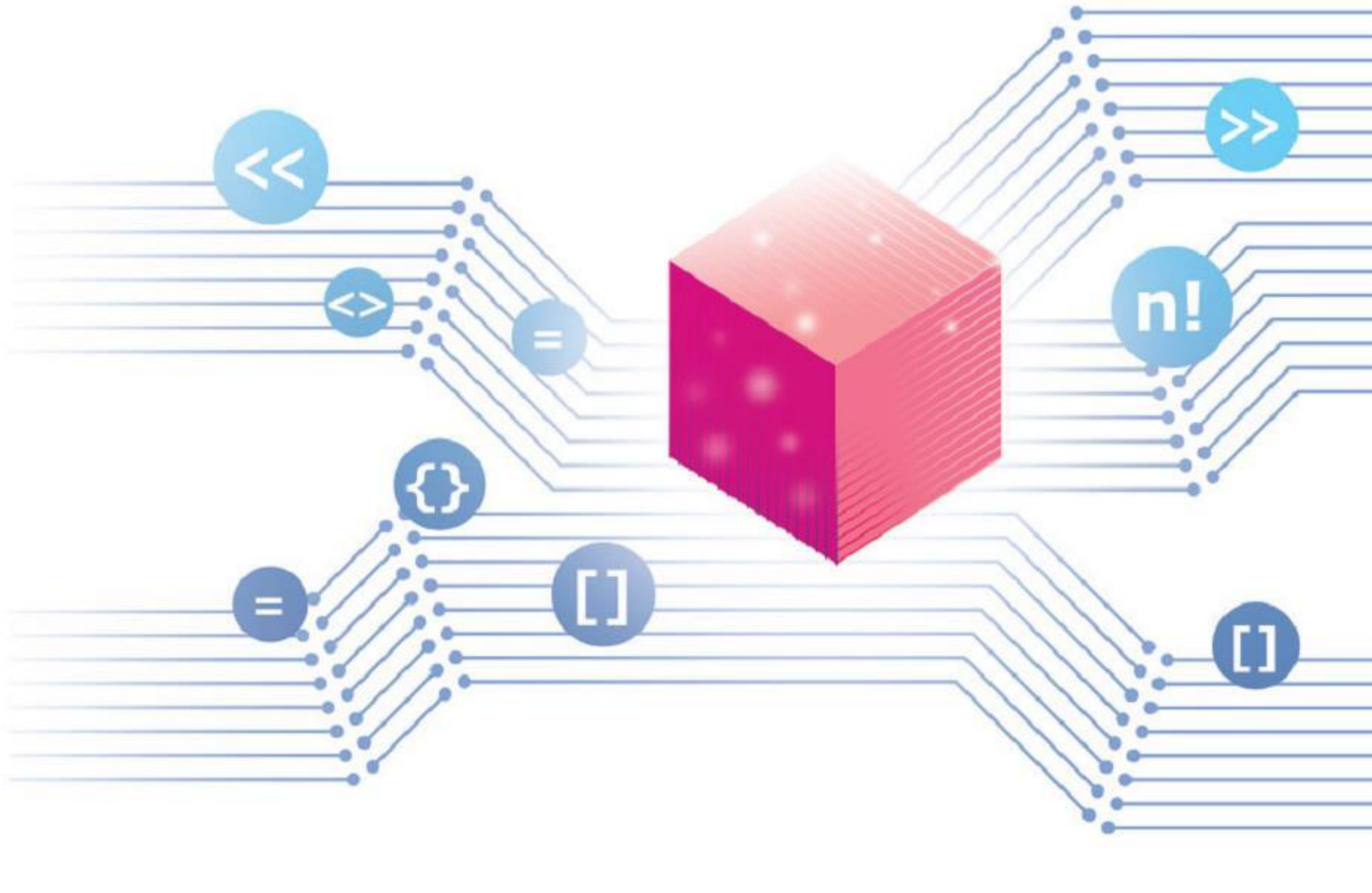
Temat 5. Wprowadzenie do programowania w języku Python

Temat 6. Zapisywanie algorytmów na liczbach naturalnych w wybranych językach programowania

Temat 7. Algorytmy wyszukiwania i porządkowania

Temat 8. Wykorzystanie funkcji i tablic do zapisywania w języku C++ algorytmów porządkowania i wyszukiwania

Temat 9. Wykorzystanie funkcji i list do zapisywania w języku Python algorytmów porządkowania i wyszukiwania



Temat 4.

Wprowadzenie do programowania w języku C++

1. **Usystematyzowanie podstawowych pojęć**
2. **Pierwszy program komputerowy w języku C++**
3. **Stosowanie zmiennych i wykonywanie obliczeń**
4. **Stosowanie instrukcji warunkowej do realizacji algorytmów z warunkami**
5. **Stosowanie instrukcji iteracyjnej do realizacji algorytmów iteracyjnych**



Warto powtórzyć

1. Jak tworzymy zmienną w środowisku programowania Baltie?
2. W jaki sposób możemy nadać wartość zmiennej?
3. Jaką instrukcję stosujemy, aby zaprogramować sytuację warunkową?
4. Jakie instrukcje stosujemy, aby zaprogramować iteracje?

1. Usystematyzowanie podstawowych pojęć

Na lekcjach informatyki w klasie ósmej omówimy wybrane algorytmy, które będziemy zapisywać w języku programowania C++. Niektóre z nich przedstawimy również (dla porównania) w dydaktycznych środowiskach programowania – Scratch i Baltie.

Pojęcia związane z programowaniem:

- **Algorytm** – uporządkowany i uściślony sposób rozwiązania danego **problemu**, zawierający szczegółowy opis wykonywanych czynności w skończonej liczbie kroków.
- **Język programowania** – specjalny język służący do pisania **programów komputerowych**. Jest on zbiorem określonych instrukcji i zasad składni.
- **Program komputerowy** – ciąg instrukcji języka programowania, realizujący **algorytm**.
- **Program (kod) źródłowy** – program napisany w języku programowania (np. języku C++), w postaci ciągu instrukcji tekstowych.
- **Program (kod) wynikowy (maszynowy)** – program w języku wewnętrznym komputera (ciąg instrukcji dla procesora wraz z danymi dla tych instrukcji).

- **Algorytm z warunkami** – algorytm zawierający sytuacje warunkowe. Aby w języku programowania napisać program realizujący algorytm z warunkami, stosujemy **instrukcję warunkową**.
- **Iteracja** – polega na powtarzaniu tej samej operacji (ciągu operacji). Iterację implementujemy (zapisujemy w kodzie źródłowym), stosując tzw. **pętlę**. Z pętlą mamy do czynienia, gdy w pewnym kroku algorytmu wracamy do jednego z wcześniejszych kroków, co powoduje, że kroki te mogą zostać wykonane wiele razy.
- **Algorytm iteracyjny** – algorytm zawierający powtarzanie poleceń. Aby w języku programowania napisać program realizujący algorytm iteracyjny, stosujemy **instrukcje iteracyjne**.
- **Zmienna** – w programie komputerowym to nazwana część pamięci operacyjnej komputera (RAM) o unikatowym adresie. W każdym języku programowania jest określony sposób korzystania ze zmiennych, w tym nadawania im wartości. W niektórych językach programowania musimy określić nie tylko nazwę zmiennej, ale i jej **typ**.
- **Typ zmiennej** – wyznacza zbiór wartości, które może ona przyjmować (np. liczby całkowite, liczby rzeczywiste).



Program komputerowy może występować w dwóch postaciach:
- jako program (kod) źródłowy,
- jako program (kod) wynikowy (maszynowy).

Program napisany w języku programowania (**kod źródłowy**) nie jest zrozumiały dla komputera. Komputer wykonuje tylko program napisany w języku wewnętrznym (**kod wynikowy**). Tłumaczeniem programu źródłowego na kod maszynowy zajmuje się specjalny program zwany **translatorem**, a proces tłumaczenia programu napisanego w języku programowania wysokiego poziomu na język wewnętrzny komputera nazywamy **translacją**. Może ona przebiegać w formie **kompilacji** lub **interpretacji**.

W językach kompilowanych (np. C++) musimy napisać cały program, zapisać go w pliku, skompilować i dopiero po poprawnej kompilacji uruchomić. Jeśli **kompilator** znajdzie w programie błędy, wskaże je. Wtedy program należy poprawić, zapisać, ponownie skompilować, a następnie uruchomić. W środowisku programowania Baltie pod elementami graficznymi ukryto również język kompilowany – język programowania C.

W językach interpretowanych (np. Scratch, Python) możemy napisać program składający się z wielu poleceń (tworząc tzw. skrypt), zapisać go w pliku, a następnie uruchomić. Mówimy wówczas, że pracujemy w **trybie skryptowym**.

K **Kompilacja**

Przetłumaczenie programu w całości na język zrozumiały dla procesora, tak by mógł go wykonać komputer. Jeśli kompilacja przebiegnie poprawnie, można uruchomić program. Raz skompilowany program nie wymaga już powtórnej operacji tłumaczenia.

I **Interpretacja**

Tłumaczenie programu tworzonego w języku programowania instrukcja po instrukcji, tak aby komputer mógł wykonać każdą z nich. Tłumaczenie następuje każdorazowo przy uruchomieniu programu.

W odróżnieniu od kompilatora, **interpreter** próbuje wykonać program instrukcja po instrukcji. Jeżeli w programie wystąpi błąd, interpreter przerwie wykonanie programu i wyświetli komunikat o błędzie. Należy poprawić wskazany błąd, ponownie zapisać program w pliku i uruchomić. Jeśli interpreter nie znajdzie więcej błędów, zostanie wykonany cały program.

W dydaktycznych środowiskach programowania (np. Scratch, Baltie) zwykle upraszcza się sposób wprowadzania poleceń (instrukcji języka programowania). Można je zastąpić elementami graficznymi umieszczanymi w obszarze roboczym – z tego powodu takie środowiska nazywa się „wizualnymi".



Niezależnie od zastosowanego środowiska:

- kolejność występowania instrukcji w programie (także prezentowanych przez elementy graficzne) powinna odpowiadać kolejności operacji realizujących dany algorytm,
- postać instrukcji musi być zgodna z zasadami składni danego języka – nie może zabraknąć nawet jednej spacji, jednego dwukropka, przecinka, średnika czy innego znaku w miejscach, w których są wymagane.

2. Pierwszy program komputerowy w języku C++

Uwaga !

Aby pisać programy w języku C++, należy zainstalować wybraną aplikację zawierającą edytor i kompilator języka C++, np. GCC, Code::Blocks, Dev-C++. Są one dostępne bezpłatnie w Internecie.

Aby tworzyć programy komputerowe, możemy korzystać z wyspecjalizowanych pakietów programistycznych. Pakiety takie zawierają m.in. edytor **kodu źródłowego**, w którym piszemy program w języku programowania (np. w C++), kompilator i inne narzędzia wspomagające programowanie. Taką aplikację (lub zespół aplikacji) określa się też jako **zintegrowane środowisko programistyczne**.

Etapy tworzenia programu w języku C++

1. W edytorze wbudowanym do środowiska programistycznego wybranego języka programowania piszemy kod źródłowy programu (rys. 1.).
2. Zapisujemy kod źródłowy programu w pliku (w języku C++ zwykle w pliku z rozszerzeniem *cpp*).
3. Kompilujemy program, korzystając z opcji **Kompiluj** (**Compile**). Przebieg kompilacji obserwujemy w dolnej części okna programu (rys. 1. i 2.).
4. Jeśli program został skompilowany, uruchamiamy go, zazwyczaj korzystając z opcji **Uruchom** (**Run**). Wynik działania programu pokaże się w osobnym oknie (rys. 3.).
5. Jeżeli w trakcie kompilacji wystąpiły błędy, należy je poprawić (kompilator wskazuje prawdopodobne miejsce popełnienia błędu – rys. 2.) i ponownie przejść do kroku 2.

Każdy język programowania ma swój zbiór instrukcji. Postać instrukcji jest inna dla różnych języków. Języki programowania posiadają również odpowiednie zasady składni oraz właściwe słownictwo – podobnie jak języki naturalne, którymi posługują się ludzie. Instrukcje języka programowania realizują czynności takie, jak: wprowadzanie danych, wyprowadzanie wyników, wykonywanie obliczeń, sprawdzanie warunków czy powtarzanie operacji.

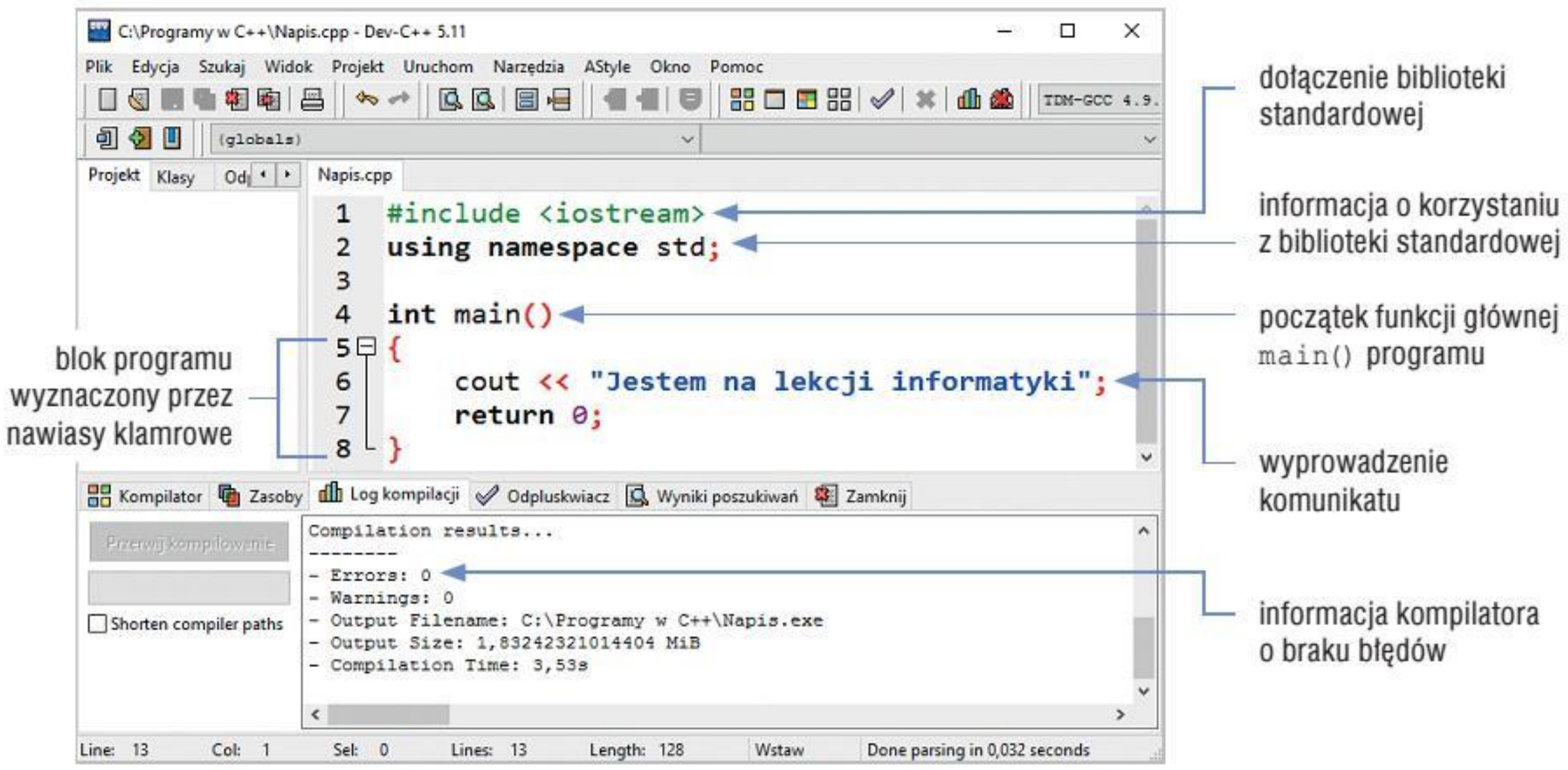
W trakcie wykonywania programu procesor rozpoznaje i wykonuje swoje wewnętrzne instrukcje. Uruchomić i wykonać można tylko program poprawnie skompilowany (rys. 1.).

Funkcja główna `main()` rozpoczyna działanie programu. Przed nazwą funkcji powinno się znajdować **słowo kluczowe** **`int`**, informujące, że funkcja `main()` zwraca wartość liczbową. Instrukcja **`return`** `0` na końcu funkcji powoduje zwrócenie do systemu operacyjnego wartości 0, informującej o prawidłowym wykonaniu programu. Każda instrukcja programu powinna być zakończona średnikiem.

S **Słowo kluczowe**

Słowo mające szczególne znaczenie w danym języku programowania, np. oznaczające określony rozkaz, instrukcję lub polecenie.

C++

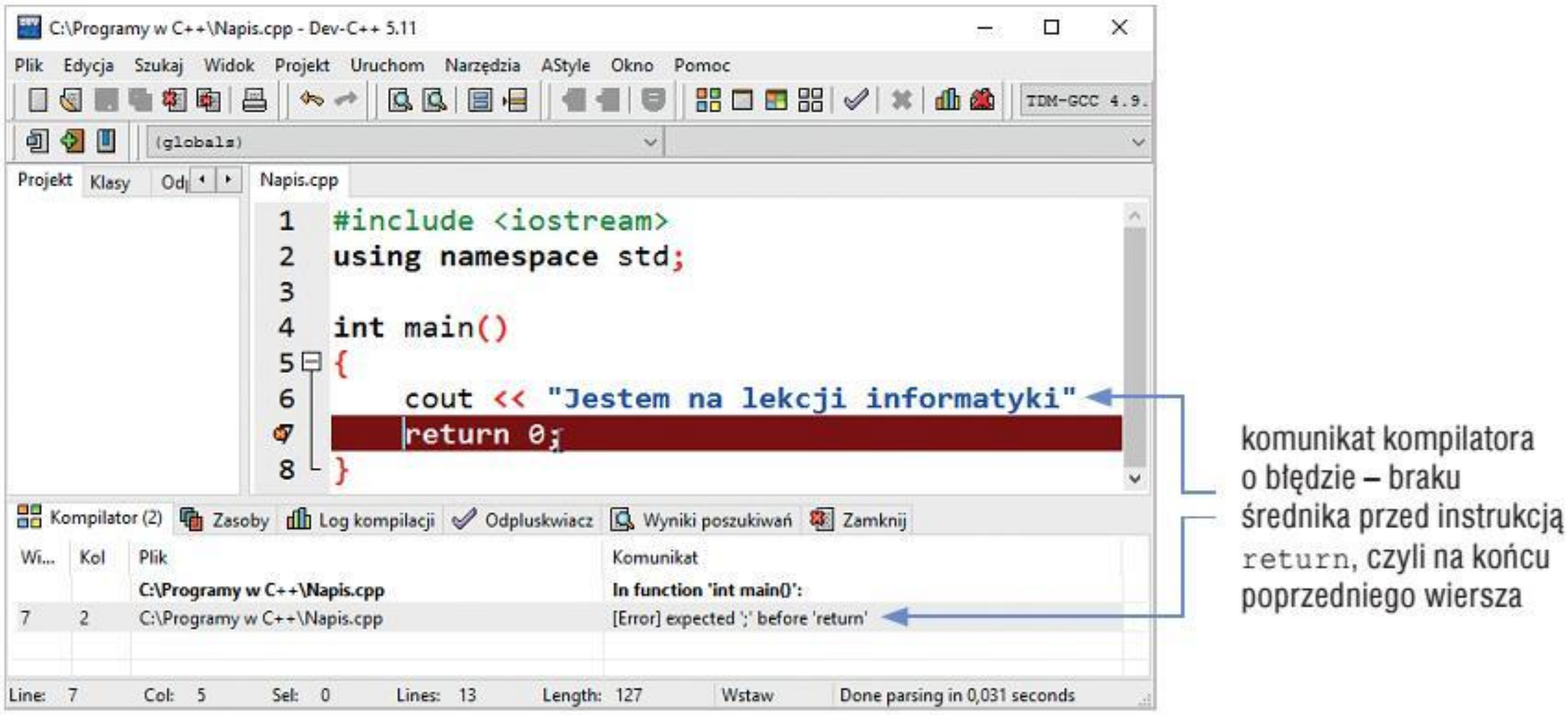


Rys. 1. Okno środowiska programistycznego języka C++ z widocznym programem źródłowym – kompilacja przebiegła prawidłowo

Ćwiczenie 1. Piszemy pierwszy program w języku C++

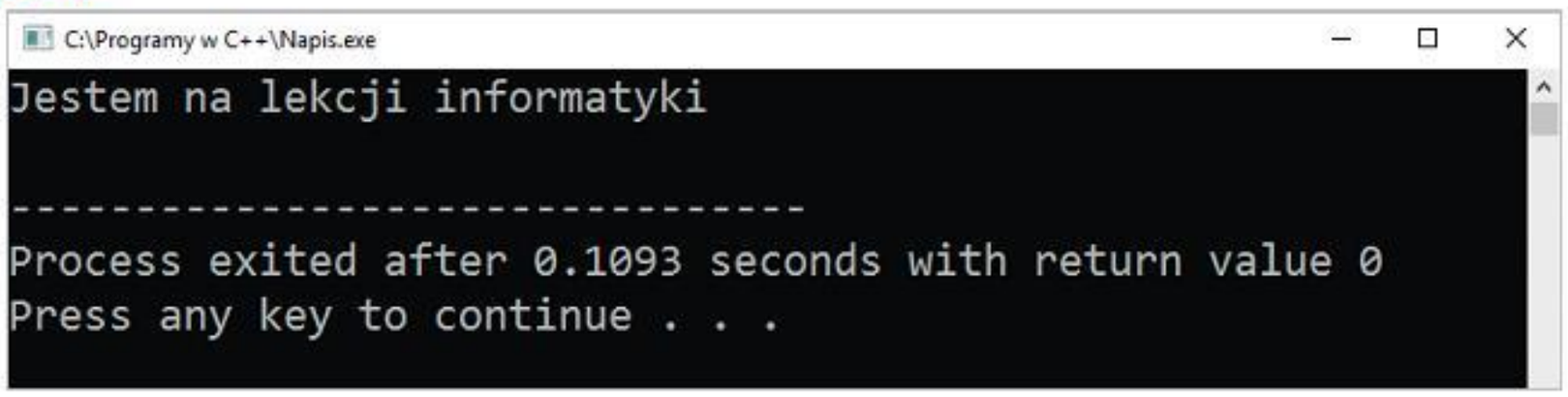
1. Napisz program wyświetlający na ekranie napis „Jestem na lekcji informatyki” (rys. 1.).
2. Zapisz program w pliku pod nazwą *Napis*.
3. Skompiluj i uruchom program.
4. Jeśli kompilator wykrył błędy, popraw je. Zapisz program w pliku pod tą samą nazwą. Skompiluj i uruchom program ponownie.

C++



Rys. 2. Okno środowiska programistycznego języka C++ z widocznym programem źródłowym – kompilator wykrył błąd w programie

C++



Rys. 3. Wynik działania programu – ćwiczenie 1.

Po wykonaniu programu wyniki pojawią się na urządzeniu zewnętrznym, np. na ekranie monitora. Zobaczymy je w osobnym oknie (oknie wykonania programu) – rys. 3.

Aby powiększyć czcionkę w oknie wykonania programu, należy kliknąć przycisk (w lewym górnym rogu okna) i wybrać polecenie **Właściwości/Czcionka.**

Ćwiczenie 2. Modyfikujemy program w języku C++

1. Zmodyfikuj program zapisany w ćwiczeniu 1., tak aby w drugim wierszu wyświetlał twoje imię.
2. Zapisz program w pliku pod tą samą nazwą. Skompiluj i uruchom program.

Wskazówka: Do wyświetlania napisów i wyników stosujemy polecenie `cout <<`. Aby po wyświetleniu napisu kolejny napis umieścić w nowym wierszu, należy dodać sekwencję znaków `\n` po pierwszym napisie. Można to zapisać na kilka sposobów:

```
cout << "Jestem na lekcji informatyki\n";
cout << "Tomek";
```

lub krócej:

```
cout << "Jestem na lekcji informatyki\n" << "Tomek";
```

lub jeszcze krócej:

```
cout << "Jestem na lekcji informatyki\nTomek";
```

Sekwencja znaków `\n` reprezentuje znak przejścia do nowego wiersza. W kodzie ASCII znak ten ma symbol LF i kod 10.

3. Stosowanie zmiennych i wykonywanie obliczeń



W każdym języku programowania musimy określić sposób korzystania ze zmiennych i nadawania im wartości.

Aby utworzyć zmienną w środowiskach programowania Scratch i Baltie, określaliśmy jej nazwę (w programie Baltie również typ). Zmiennej przypisywaliśmy wartość, wprowadzając ją z klawiatury po uruchomieniu programu, lub podstawialiśmy pod zmienną konkretną wartość od razu w programie.
Na przykład (Baltie):



zmiennej *licznik* typu całkowitego przypisujemy wartość 0,



zmiennej *liczba* typu rzeczywistego przypisujemy wartość 159,50,



wartość zmiennej *litera* typu znakowego będzie wprowadzona z klawiatury po uruchomieniu programu.

W języku C++ każdą zmienną używaną w programie należy **zadeklarować** – określić jej nazwę i typ. Typ zmiennej wyznacza zbiór wartości, które zmienna może przyjmować (np. liczby całkowite, liczby rzeczywiste). Zmienne deklarujemy w dowolnym miejscu programu, ale przed ich pierwszym użyciem.
Na przykład:

`int liczba;`	deklaracja zmiennej *liczba* typu całkowitego,
`int a, b;`	deklaracja zmiennych *a* i *b* typu całkowitego (w przypadku zmiennych tego samego typu można je wymienić po przecinku),
`float wzrost, wynik;`	deklaracja zmiennych *wzrost* i *wynik* typu rzeczywistego.

W języku C++ małe i wielkie litery w nazwach instrukcji, zmiennych itp. mają różne znaczenie – np. *liczba* i *Liczba* to dwie różne zmienne. W nazwach zmiennych wolno stosować litery (bez polskich znaków diakrytycznych), cyfry i znak podkreślenia. Nazwa nie może zaczynać się od cyfry.

Uwaga

W większości języków programowania część dziesiętną liczby zapisujemy po kropce, a nie po przecinku jak w matematyce.



Zmiennej można nadać wartość za pomocą **instrukcji przypisania**. Znak (operator) = oznacza przypisanie zmiennej zapisanej po lewej stronie znaku wartości umieszczonej po jego prawej stronie.

Na przykład:

```
liczba = 20;
b = 30;
wzrost = 165.5;
wynik = 12.3 + 14.9;
```

zmiennej *liczba* przypisujemy wartość 20,
zmiennej *b* przypisujemy wartość 30,
zmiennej *wzrost* przypisujemy wartość 165,5,
zmiennej *wynik* przypisujemy wartość wyrażenia 12,3 + 14,9.

W języku C++ można również w jednej instrukcji określić typ zmiennej i nadać jej wartość.

Na przykład:

```
int liczba = 20;
float wzrost = 165.5;
```



Wartość zmiennej można również wprowadzać z klawiatury po uruchomieniu programu. W tym celu stosujemy polecenie `cin >>`. Operator >> oznacza wprowadzenie danych do odpowiedniej zmiennej, podanej po prawej stronie. Jeśli chcemy wprowadzić dane do kilku zmiennych, użyjemy operatora >> wielokrotnie.

Na przykład:

`cin >> wzrost;` — po wprowadzeniu liczby z klawiatury zostaje ona zapamiętana w zmiennej *wzrost*,

`cin >> a >> b;` — po wprowadzeniu dwóch liczb z klawiatury zostaną one zapamiętane kolejno w zmiennych *a* i *b*.

Operator	Działanie	Przykład instrukcji przypisania
+	dodawanie	`suma = a + b;`
-	odejmowanie	`roznica = a - b;`
*	mnożenie	`iloczyn = a * b;`
/	dzielenie w zależności od typu danych: • całkowite (z obcięciem części ułamkowej) – dla zmiennych typu całkowitego • zmiennoprzecinkowe (z zachowaniem części ułamkowej) – jeśli przynajmniej jedna dana jest zmienną typu rzeczywistego	`iloraz = a / b;`
%	obliczenie reszty z dzielenia dwóch zmiennych typu całkowitego	`reszta = a % b;`

Tabela 1. *Podstawowe operatory arytmetyczne w języku C++*

C++

Suma.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int a,b,suma;
7      cout << "Podaj dwie liczby\n";
8      cin >> a >> b;
9      suma = a + b;
10     cout << "Suma wynosi: "<< suma;
11     return 0;
12 }
13
```

Rys. 4. Program obliczający sumę dwóch liczb całkowitych wprowadzanych z klawiatury i wyprowadzający wynik na ekran – ćwiczenie 3.

Ćwiczenie 3. Deklarujemy zmienne w programie i wykonujemy na nich obliczenia

1. Program pokazany na rysunku 4. pozwala na wprowadzenie dwóch liczb całkowitych z klawiatury, obliczenie ich sumy i wyprowadzenie wyniku na ekran. Przepisz kod programu pokazanego na rysunku 4.
2. Zapisz program w pliku pod nazwą *Suma*. Skompiluj i uruchom program.

Ćwiczenie 4. Modyfikujemy program

1. Zmodyfikuj program *Suma* zapisany w ćwiczeniu 3. tak, aby wynik działania programu wyglądał jak na rysunku 5.
2. Zapisz program w pliku pod tą samą nazwą. Skompiluj i uruchom program.

C++

```
C:\Programy w C++\Suma.exe
Podaj dwie liczby
a = 543
b = 98
a + b = 641
--------------------------------
```

Rys. 5. Wynik działania programu – ćwiczenie 4.

Ćwiczenie 5. Uzupełniamy program

1. W programie zapisanym w ćwiczeniu 4. zadeklaruj nową zmienną typu całkowitego o nazwie *iloczyn*. Dodaj do programu obliczanie iloczynu liczb *a* i *b* oraz wyprowadzanie wyniku mnożenia na ekran (podobnie jak wyniku sumowania w ćwiczeniu 4.).
2. Zapisz program w pliku pod nazwą *Działania*. Skompiluj i uruchom program.

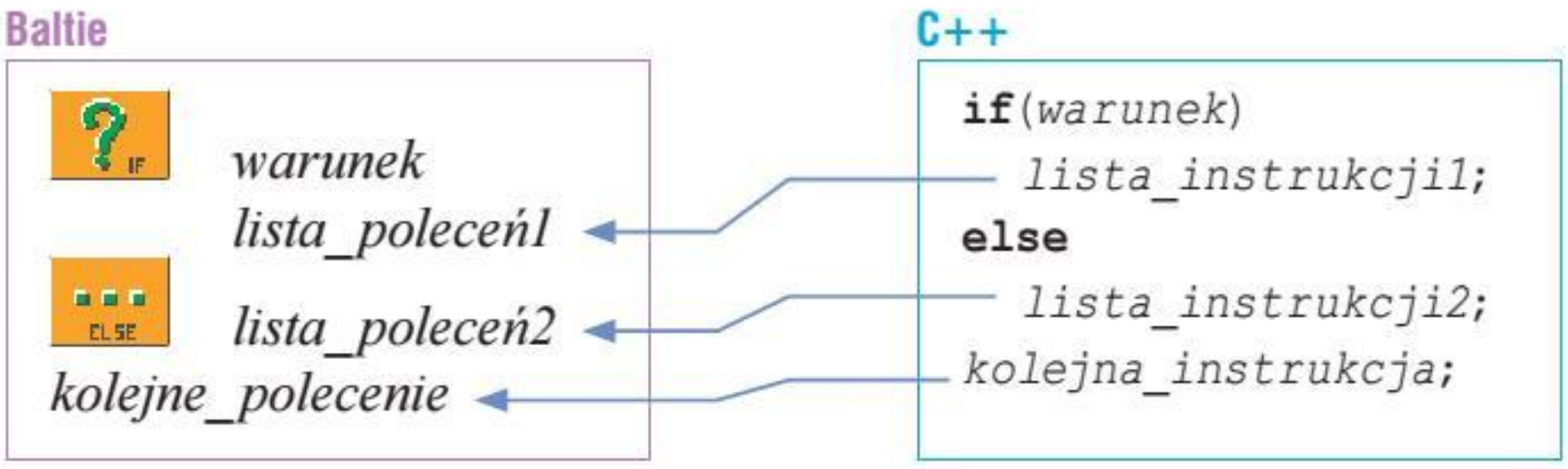
Wskazówka: Aby po wyświetleniu sumy kolejny napis umieścić w nowym wierszu, należy umieścić znaki `\n` na końcu poprzedniej instrukcji, na przykład w następujący sposób:

```
cout << "a + b = " << suma << "\n";
```

4. Stosowanie instrukcji warunkowej do realizacji algorytmów z warunkami

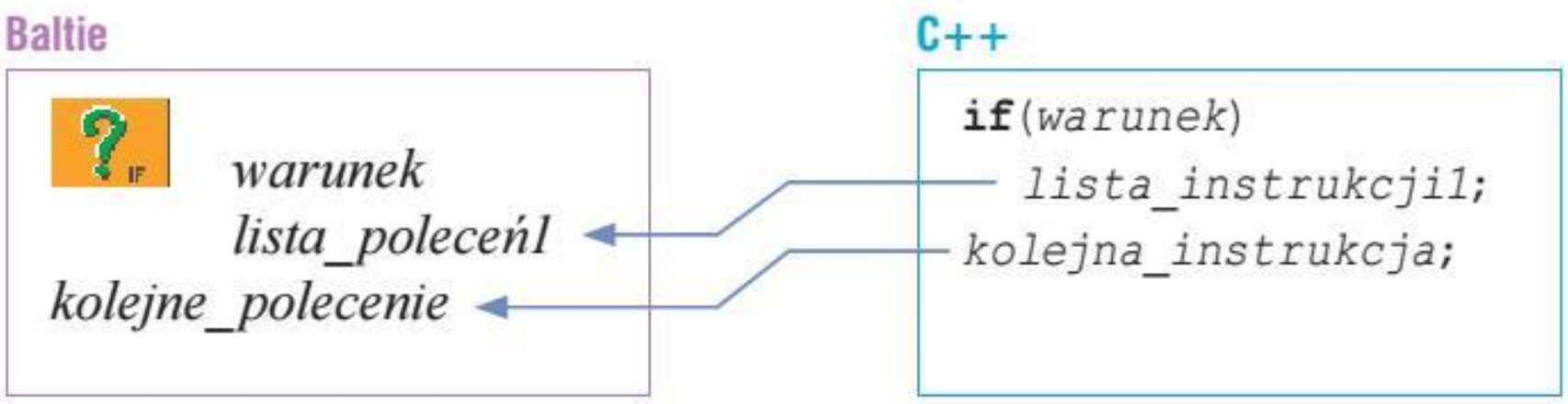
Do programu obliczającego sumę i iloczyn dwóch liczb chcielibyśmy dodać obliczenie ilorazu. W jaki sposób zaprogramować sytuację, w której program nie wykona działania w przypadku dzielnika równego zero?

Należy w tym celu zastosować **instrukcję warunkową**. W języku C++ jest to instrukcja `if`, która działa podobnie do instrukcji warunkowych w języku Python oraz w środowiskach programowania Scratch i Baltie.



Rys. 6a. Instrukcja warunkowa w języku C++ – po lewej odpowiadająca jej instrukcja w Baltiem

Instrukcja warunkowa `if` (rys. 6a) występuje również w wersji uproszczonej (rys. 6b). Jako *lista_instrukcji* może wystąpić jedna instrukcja (w tym instrukcja warunkowa) lub więcej instrukcji ujętych w blok {}. *Warunek* musimy ująć w nawiasy okrągłe.



Rys. 6b. Uproszczona postać instrukcji warunkowej w języku C++ – po lewej odpowiadająca jej instrukcja w Baltiem

Uwaga

W języku C++ wartość równą 0 uważa się za logiczny **fałsz**, a wartość różną od 0 – za logiczną **prawdę**.

Sprawdzany jest warunek logiczny – jeśli *warunek* jest spełniony (jest prawdziwy), wykonywana jest *lista_instrukcji1*. Jeśli *warunek* nie jest spełniony (jest fałszywy) – wykonywana jest *lista_instrukcji2* po słowie `else` (rys. 6a), a jeśli brak tej części instrukcji – wykonywana jest od razu *kolejna_instrukcja* po instrukcji warunkowej (rys. 6b).

Operatory		Określenie	Przykład wyrażenia	Interpretacja wyrażenia
Porównania	==	równy	`a == b`	*a* równe *b*
	!=	różny	`a != b`	*a* różne od *b*
	<	mniejszy	`a < b`	*a* mniejsze od *b*
	>	większy	`a > b`	*a* większe od *b*
	>=	większy lub równy	`a >= b`	*a* większe lub równe *b*
	<=	mniejszy lub równy	`a <= b`	*a* mniejsze lub równe *b*
Logiczne	||	alternatywa logiczna (lub)	`a < -5 || a > 5`	*a* mniejsze od -5 lub *a* większe od 5
	&&	koniunkcja logiczna (i)	`a > 0 && a < 10`	*a* większe od 0 i *a* mniejsze od 10
	!	negacja logiczna	`!(a == 5)`	nieprawda, że *a* jest równe 5

Tabela 2. *Podstawowe operatory porównania i logiczne w języku C++*

C++

```
15      if(b != 0)
16      {
17          iloraz = a / b;
18          cout << "a / b = " << iloraz;
19      }
20      else
21          cout << "Nie dzielimy przez zero";
22      return 0;
23  }
```

Rys. 7. Przykład stosowania instrukcji warunkowej w wersji pełnej – ćwiczenie 6.

Ćwiczenie 6. Stosujemy instrukcję warunkową

1. Otwórz program *Działania* zapisany w ćwiczeniu 5.
2. Uzupełnij program o obliczenie ilorazu liczb *a* i *b*. Iloraz możemy obliczyć tylko dla *b* różnego od zera – jeśli ten warunek nie zostanie spełniony, program powinien wyświetlić odpowiedni napis (rys. 7.). Dodaj do deklaracji zmiennych nową zmienną *iloraz* i zmień typ wszystkich zmiennych na rzeczywisty **`float`**.
3. Zapisz program w pliku pod tą samą nazwą. Skompiluj i uruchom program.

Wskazówka: Zmiana typu zmiennych *a* i *b* na **`float`** jest potrzebna ze względu na operator dzielenia /. W przypadku liczb typu całkowitego w wyniku dzielenia z użyciem operatora / otrzymujemy tylko część całkowitą ilorazu.

C++

```
11      punkty = 100;
12      if(sr_ocen > 4 && sr_ocen <= 6)
13          punkty_o = 20;
14      cout << "Liczba punktow: " << punkty + punkty_o << endl;
```

Rys. 8. Przykład stosowania instrukcji warunkowej w wersji uproszczonej – ćwiczenie 7.

Ćwiczenie 7. Stosujemy instrukcję warunkową w wersji uproszczonej

1. Pewien uczeń zdobył 100 punktów w konkursie szkolnym. Jeśli jego średnia ocen będzie wyższa niż 4, uczeń otrzyma dodatkowe 20 punktów. Napisz program obliczający i wyświetlający na ekranie liczbę punktów ucznia. Na rysunku 8. pokazano fragment programu.
2. Dodaj do programu wprowadzanie wartości zmiennej *sr_ocen* typu rzeczywistego (`float`). Pozostałe zmienne powinny być typu całkowitego (`int`).
3. Zapisz program w pliku pod nazwą *Punkty*. Skompiluj i uruchom program. Sprawdź działanie programu dla kilku różnych danych.

Wskazówka: Zapis `endl` oznacza początek nowego wiersza. Wstawienie `endl` na końcu instrukcji `cout <<` oznacza, że kursor przejdzie do następnego wiersza.

Uwaga

Aby uniknąć problemów z poprawnym wyświetlaniem znaków na ekranie, w tekstach komunikatów nie używaj liter typowych dla języka polskiego, czyli ze znakami diakrytycznymi: ą, ć, ę, ł, ń, ó, ś, ź, ż.

5. Stosowanie instrukcji iteracyjnej do realizacji algorytmów iteracyjnych

Chcemy obliczać, ile metrów bieżących siatki potrzeba na ogrodzenie każdego z pięciu placów zabaw znajdujących się w różnych miejscach. W algorytmie mamy do czynienia z powtarzaniem poleceń obliczania obwodu placu zabaw, czyli z iteracją. W jaki sposób napisać program realizujący algorytm iteracyjny w wybranym języku programowania?

Należy w tym celu zastosować instrukcję iteracyjną. W języku C++ jedną z instrukcji iteracyjnych jest instrukcja `for`. Do określenia liczby kroków iteracji (powtórzeń) stosujemy tzw. **zmienną sterującą**.

C++

```
for (wyrażenie_początkowe; warunek; wyrażenie_pętli)
   lista_instrukcji;
```

Rys. 9. Ogólna postać instrukcji iteracyjnej `for` w języku C++

Najpierw wykonywane jest *wyrażenie_początkowe* (np. $i = 0$), zwykle ustalające wartość początkową zmiennej sterującej. Następnie, dopóki spełniony jest *warunek*,

wykonywana jest *lista_instrukcji*, a potem obliczana wartość *wyrażenia_pętli*, służącego zazwyczaj do zwiększenia (lub zmniejszenia) wartości zmiennej sterującej. Jako *lista_instrukcji* może wystąpić pojedyncza instrukcja (w tym instrukcja pętli) lub więcej instrukcji ujętych w blok {}.

Na przykład instrukcja:

```
for(i=0; i<5; i++)
{
  cout << "Podaj liczbe: ";
  cin >> a;
}
```

oznacza, że polecenia ujęte w blok {} program wykona 5 razy. Zmienna sterująca o nazwie *i* na początku ma wartość 0. Polecenia w bloku {} są wykonywane, gdy zmienna *i* jest mniejsza od 5 (czyli dla zmiennej *i* równej kolejno: 0, 1, 2, 3, 4).

Wyrażenie `i++` oznacza „zmiennej *i* przypisz wartość *i* zwiększoną o 1". Jest to skrót instrukcji przypisania: `i = i + 1`. W języku Scratch wartość licznika zmienialiśmy w podobny sposób, stosując polecenie **zmień**, reprezentowane przez element z napisem „**zmień** *licznik* **o** 1".

Ćwiczenie 8. Stosujemy instrukcję iteracyjną `for` w języku C++

1. Program pokazany na rysunku 10. realizuje algorytm obliczania obwodu pięciu placów zabaw, których wymiary mają być wprowadzane z klawiatury. Przepisz kod programu pokazany na rysunku 10. Wyjaśnij znaczenie poszczególnych wierszy programu.
2. Zapisz program w pliku pod nazwą *Ogrodzenia_5*. Skompiluj i uruchom program. Sprawdź jego działanie dla różnych wartości danych *dlugosc* i *szerokosc*.
3. Odpowiedz na pytania:
   a. Ile razy komputer wykona instrukcję `cin >> szerokosc;`?
   b. Jaka jest wartość zmiennej *i* na koniec programu?

Wskazówka: Aby sprawdzić, jaka jest wartość zmiennej *i* na koniec programu, wyświetl jej wartość na ekranie, umieszczając polecenie: `cout << i` przed poleceniem **`return`** `0`.

C++

Ogrodzenia_5.cpp

```
#include <iostream>
using namespace std;

int main()
{
    int szerokosc, dlugosc, i, obwod;

    for(i = 0; i < 5; i++)
    {
        cout << "Podaj szerokosc placu zabaw: ";
        cin >> szerokosc;
        cout << "Podaj dlugosc placu zabaw: ";
        cin >> dlugosc;
        obwod = 2 * szerokosc + 2 * dlugosc;
        cout << "Obwod placu zabaw o podanych wymiarach wynosi: " << obwod << endl;
    }
    return 0;
}
```

Rys. 10. Program realizujący algorytm iteracyjny (C++) – ćwiczenie 8.

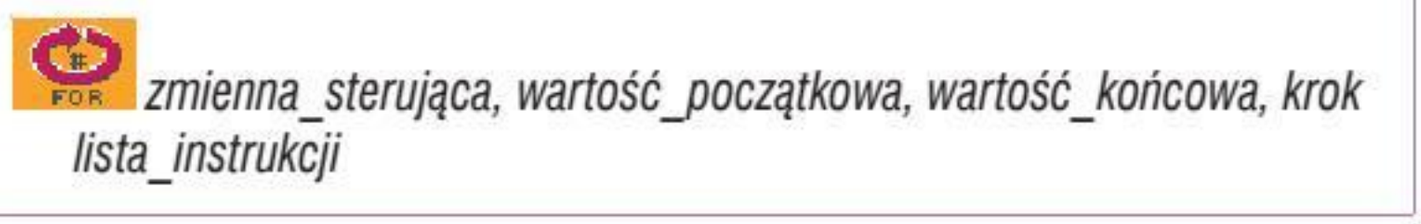
Ćwiczenie 9. Obliczamy obwody *n* placów zabaw

1. Zmodyfikuj program *Ogrodzenia_5* zapisany w ćwiczeniu 8. tak, aby obliczał obwody *n* placów zabaw (wartość *n* wprowadź z klawiatury).
2. Zapisz program w pliku pod nazwą *Ogrodzenia_n*. Skompiluj i uruchom program.
3. Sprawdź działanie programu dla różnych wartości danych *n*, *a* i *b*.

W dydaktycznych środowiskach programowania również zapisywaliśmy powtarzające się polecenia. W języku Scratch stosowaliśmy m.in. polecenie **powtórz**. Liczbę powtórzeń określaliśmy, zmieniając liczbę w polu tekstowym elementu reprezentującego to polecenie. Mogliśmy też umieścić w tym polu element reprezentujący zmienną.

W środowisku programowania Baltie powtarzające się polecenia łączyliśmy w blok i umieszczaliśmy w nawiasach klamrowych {}. Liczbę powtórzeń umieszczaliśmy przed nawiasem otwierającym. Ale możemy również algorytm iteracyjny zrealizować w podobny sposób jak w języku C++. Stosujemy w tym celu instrukcję iteracyjną **for**, którą rozpoczyna element **FOR** . Użyjemy tej instrukcji do zapisania algorytmu obliczania obwodów pięciu placów zabaw.

Baltie



Rys. 11. Ogólna postać instrukcji iteracyjnej ***for*** w środowisku Baltie

Liczba powtórzeń zależy od wartości *zmiennej_sterującej*, która zmienia się od *wartości_początkowej* do *wartości_końcowej* o *krok*. Jeśli *krok* nie zostanie podany, wartość *zmiennej_sterującej* zmienia się o 1.

Baltie



Rys. 12. Stosowanie instrukcji iteracyjnej ***for*** (Baltie) – ćwiczenie 10.

Ćwiczenie 10. Stosujemy instrukcję iteracyjną **for** w środowisku Baltie

1. Utwórz w programie Baltie program, którego treść podano w ćwiczeniu 8.
2. Zapisz program w pliku pod nazwą *Ogrodzenia_5*. Uruchom program i sprawdź jego działanie dla różnych wartości danych *dlugosc* i *szerokosc*.
3. Porównaj utworzony program z realizacją tego samego algorytmu w języku C++. Jakie znajdujesz podobieństwa, a jakie różnice, zwłaszcza w działaniu instrukcji iteracyjnej?

Warto zapamiętać

- Każdy język programowania ma swój zbiór instrukcji, zasady składni i właściwe słownictwo.
- Aby utworzyć program w języku C++, piszemy kod źródłowy w odpowiednim edytorze, zapisujemy program w pliku i kompilujemy. Jeśli kompilacja przebiegła prawidłowo, uruchamiamy program.
- Zmienną używaną w programie należy zadeklarować przed pierwszym użyciem: określić jej nazwę i typ. Zmiennej nadajemy wartość za pomocą instrukcji przypisania. Możemy również wprowadzić wartość zmiennej z klawiatury, stosując polecenie `cin >>`.
- W języku C++ do zapisywania algorytmów, w których występują sytuacje warunkowe, stosujemy instrukcję warunkową **`if`**. Działanie tej instrukcji jest podobne jak w języku Baltie i Scratch.
- Jeśli w algorytmie występują powtórzenia, możemy zastosować instrukcję iteracyjną **`for`**. Liczba powtórzeń (iteracji) jest w niej zwykle z góry określona.

Pytania i polecenia

1. Czym różni się program źródłowy od programu wynikowego?
2. Jakie są etapy tworzenia programu w języku C++?
3. W jaki sposób deklaruje się zmienne w języku C++?
4. Jak nadaje się wartości zmiennym w języku C++?
5. Wyjaśnij zapisy:
   a) `int liczba;`
   b) `liczba = 20;`
   c) `int liczba = 20;`
6. Wyjaśnij działanie instrukcji warunkowej **`if`** w języku C++.
7. Czym się różni operator == od = ? Podaj przykłady użycia każdego z tych operatorów.
8. Kiedy zostanie wykonana instrukcja umieszczona po słowie **`else`** w instrukcji warunkowej?
9. Wyjaśnij działanie instrukcji iteracyjnej **`for`** w języku C++.
10. Dlaczego zmienną *i* w pętli

```
for (i = 0; i < 5; i++)
   cout << i;
```

nazywamy zmienną sterującą?

Zadania

1. Napisz program obliczający sumę i średnią arytmetyczną trzech liczb rzeczywistych wprowadzanych z klawiatury. Wyniki obliczeń i odpowiednie komunikaty wyświetl na ekranie. Zapisz program w pliku pod nazwą *Średnia*.
2. Napisz program obliczający pole trapezu (*pole*) o podstawach *a* i *b* oraz wysokości *h*. Wszystkie zmienne zadeklaruj jako liczby rzeczywiste (`float`). Wartości zmiennych *a*, *b* i *h* wprowadzaj z klawiatury. Wyniki obliczeń i odpowiednie komunikaty wyświetl na ekranie. Zapisz program w pliku pod nazwą *Trapez*.
3. Otwórz program *Trapez* zapisany w zadaniu 2. i dodaj sprawdzanie, czy wysokość *h* jest większa od zera. Dla *h* większego od zera należy wykonać obliczenia pola i wyświetlić wynik, a w przeciwnym wypadku wyświetlić komunikat „niepoprawna dana". Zapisz program w pliku pod tą samą nazwą.
4. Korzystając z programu *Działania* zapisanego w ćwiczeniu 6., sprawdź, czym się różni wynik dzielenia w zależności od typu danych. Nie zapisuj zmian.
5. Otwórz program *Działania* zapisany w ćwiczeniu 6. Dodaj do programu możliwość obliczenia różnicy liczb *a* i *b*. Zapisz program w pliku pod tą samą nazwą.
6. Zmodyfikuj program z zadania 5. tak, aby wykonał obliczenia dla dziesięciu wartości zmiennych a i b wprowadzonych z klawiatury. Zapisz program w pliku pod nazwą *Działania_10*.
7. Napisz program obliczający objętość (*v*) i pole powierzchni (*p*) sześcianu o boku *a*. Wartość zmiennej *a* wprowadzaj z klawiatury, a wyniki *v* i *p* wyprowadzaj na ekran. Zapisz program w pliku pod nazwą *Sześcian*.

Dla zainteresowanych

8. Otwórz program zapisany w zadaniu 7. Zadeklaruj zmienną *wybor* typu całkowitego. Wartość zmiennej *wybor* wprowadzaj z klawiatury. Zmodyfikuj program tak, aby dla wartości zmiennej *wybor* równej 1 obliczał objętość sześcianu, a dla wartości zmiennej *wybor* równej 0 – pole powierzchni sześcianu. Zapisz program w pliku pod nazwą *Sześcian_w*.
9. Zmodyfikuj program z zadania 8. tak, aby wykonał obliczenia dla dziesięciu wartości zmiennej *a* wprowadzonych z klawiatury. Zapisz program w pliku pod nazwą *Sześcian_w_10*.

Przeczytaj, jeśli chcesz wiedzieć więcej…

Język C++ jest rozwinięciem języka C, wymyślonego na początku lat 70. XX wieku. Język C powstał z kolei na bazie języków BCPL i B. Następna litera w alfabecie („D") również jest nazwą języka programowania, wykorzystującego cechy języka C++. Istnieją także języki C# i Objective-C, wykorzystujące niektóre koncepcje języków C i C++. Na świecie istnieje obecnie ponad 1000 różnych języków programowania!

Temat 5.

Wprowadzenie do programowania w języku Python

1. Usystematyzowanie podstawowych pojęć
2. Programowanie grafiki w języku Python – przypomnienie
   2.1. Środowisko programowania IDLE
   2.2. Wykorzystanie grafiki żółwia do tworzenia rysunków
3. Wyprowadzenie napisów na ekran
4. Stosowanie zmiennych i wykonywanie obliczeń
5. Stosowanie instrukcji warunkowej do realizacji algorytmów z warunkami
6. Stosowanie instrukcji iteracyjnej do realizacji algorytmów iteracyjnych



Warto powtórzyć

1. Jak tworzymy program w języku Scratch?
2. W jaki sposób w języku Scratch wyprowadzamy napisy na ekran monitora?
3. Jak tworzymy zmienną w języku Scratch?
4. W jaki sposób w języku Scratch przypisujemy zmiennej konkretną wartość, a jak wprowadzamy wartość zmiennej z klawiatury?
5. Za pomocą jakiego polecenia realizujemy sytuację warunkową w języku Scratch?
6. W jaki sposób zapisujemy iterację w języku Scratch?

1. Usystematyzowanie podstawowych pojęć

Na lekcjach informatyki w klasie ósmej omówimy wybrane algorytmy, które będziemy zapisywać w języku programowania Python. Niektóre z nich przedstawimy również (dla porównania) w dydaktycznych środowiskach programowania – Scratch i Baltie.

Pojęcia związane z programowaniem:

- **Algorytm** – uporządkowany i uściślony sposób rozwiązania danego **problemu**, zawierający szczegółowy opis wykonywanych czynności w skończonej liczbie kroków.
- **Język programowania** – specjalny język służący do pisania **programów komputerowych**. Jest on zbiorem określonych instrukcji i zasad składni.
- **Program komputerowy** – ciąg instrukcji języka programowania realizujący **algorytm**.

- **Program (kod) źródłowy** – program napisany w języku programowania (np. języku Python), w postaci ciągu instrukcji tekstowych.
- **Program (kod) wynikowy (maszynowy)** – program w języku wewnętrznym komputera (ciąg instrukcji dla procesora wraz z danymi dla tych instrukcji).
- **Algorytm z warunkami** – algorytm zawierający sytuacje warunkowe. Aby w języku programowania napisać program realizujący algorytm z warunkami, stosujemy **instrukcję warunkową**.
- **Iteracja** – polega na powtarzaniu tej samej operacji (ciągu operacji). Iterację implementujemy (zapisujemy w kodzie źródłowym), stosując tzw. **pętlę**. Z pętlą mamy do czynienia, gdy w pewnym kroku algorytmu wracamy do jednego z wcześniejszych kroków, co powoduje, że kroki te mogą zostać wykonane wiele razy.
- **Algorytm iteracyjny** – algorytm zawierający powtarzanie poleceń. Aby w języku programowania napisać program realizujący algorytm iteracyjny, stosujemy **instrukcje iteracyjne**.
- **Zmienna** – w programie komputerowym to nazwana część pamięci operacyjnej komputera (RAM) o unikatowym adresie. W każdym języku programowania jest określony sposób korzystania ze zmiennych, w tym nadawania im wartości. W niektórych językach programowania musimy określić nie tylko nazwę zmiennej, ale i jej **typ**.
- **Typ zmiennej** – wyznacza zbiór wartości, które może ona przyjmować (np. liczby całkowite, liczby rzeczywiste).

Kompilacja K

Przetłumaczenie programu w całości na język zrozumiały dla procesora, tak by mógł go wykonać komputer. Jeśli kompilacja przebiegnie poprawnie, można uruchomić program. Raz skompilowany program nie wymaga już powtórnej operacji tłumaczenia.

Interpretacja I

Tłumaczenie programu tworzonego w języku programowania instrukcja po instrukcji, tak aby komputer mógł wykonać każdą z nich. Tłumaczenie następuje każdorazowo przy uruchomieniu programu.



Program komputerowy może występować w dwóch postaciach:

- jako program (kod) źródłowy,
- jako program (kod) wynikowy (maszynowy).

Program napisany w języku programowania (**kod źródłowy**) nie jest zrozumiały dla komputera. Komputer wykonuje tylko program napisany w języku wewnętrznym (**kod wynikowy**). Tłumaczeniem programu źródłowego na kod maszynowy zajmuje się specjalny program zwany **translatorem**, a proces tłumaczenia programu napisanego w języku programowania wysokiego poziomu na język wewnętrzny komputera nazywamy **translacją**. Może ona przebiegać w formie **kompilacji** lub **interpretacji**.

W językach kompilowanych (np. C++) musimy napisać cały program, zapisać go w pliku, skompilować i dopiero po poprawnej kompilacji uruchomić. Jeśli **kompilator** znajdzie w programie błędy, wskaże je. Wtedy program należy poprawić, zapisać, ponownie skompilować, a następnie uruchomić. W środowisku programowania Baltie pod elementami graficznymi jest ukryty również język kompilowany – język programowania C.

W językach interpretowanych (np. Scratch, Python) możemy napisać program składający się z wielu poleceń (tworząc tzw. skrypt), zapisać go w pliku, a następnie uruchomić. Mówimy wówczas, że pracujemy w **trybie skryptowym**.

W odróżnieniu od kompilatora, **interpreter** próbuje wykonać program instrukcja po instrukcji. Jeżeli w programie wystąpi błąd, interpreter przerwie wykonanie programu i wyświetli komunikat o błędzie. Należy poprawić wskazany błąd, ponownie zapisać program w pliku i go uruchomić. Jeśli interpreter nie znajdzie więcej błędów, zostanie wykonany cały program.

W dydaktycznych środowiskach programowania (np. Scratch, Baltie) zwykle upraszcza się sposób wprowadzania poleceń (instrukcji języka programowania). Można je zastąpić elementami graficznymi umieszczanymi w obszarze roboczym – z tego powodu takie środowiska nazywa się „wizualnymi".



Niezależnie od zastosowanego środowiska:

- kolejność występowania instrukcji w programie (także przedstawionych jako elementy graficzne) powinna odpowiadać kolejności operacji realizujących dany algorytm,
- postać instrukcji musi być zgodna z zasadami składni danego języka – nie może zabraknąć nawet jednej spacji, jednego dwukropka, przecinka, średnika czy innego znaku w miejscach, w których są wymagane.

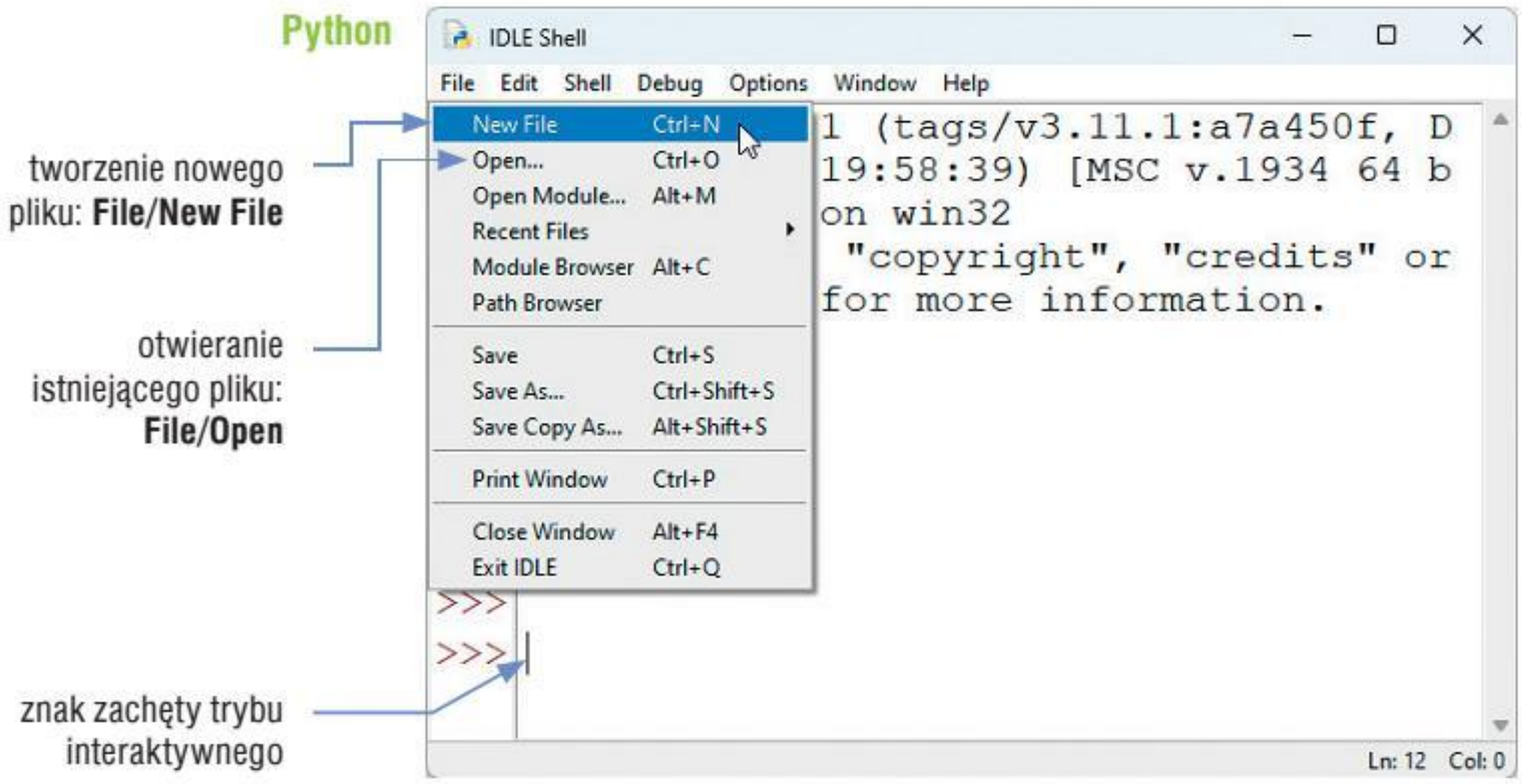
2. Programowanie grafiki w języku Python – przypomnienie

2.1. Środowisko programowania IDLE

Programy w języku Python piszemy w specjalnym edytorze, zawartym w zintegrowanym środowisku programistycznym o nazwie **IDLE**. Pakiet IDLE zawiera także interpreter i inne narzędzia wspomagające programowanie.

Środowisko programistyczne IDLE udostępnia dwa tryby pracy: **interaktywny** i **skryptowy**.
IDLE zawiera narzędzia ułatwiające pisanie programów, m.in.: edytor kodu źródłowego, interpreter i tzw. **powłokę** (narzędzie umożliwiające m.in. pracę w trybie interaktywnym).

Do tworzenia programów w języku Python służy tryb skryptowy, do którego możemy przejść z trybu interaktywnego, z okna powłoki Pythona (Python Shell) – rys. 1. Po wybraniu opcji **File/New File** otworzy się okno edytora kodu źródłowego (rys. 2a), w którym piszemy program. Okno to otwiera się również po wybraniu opcji **File/Open**, gdy otwieramy istniejący plik.



Rys. 1. Powłoka Pythona (Python Shell)

2.2. Wykorzystanie grafiki żółwia do tworzenia rysunków

Pierwsze programy w języku Python pisaliśmy już w klasie 7., tworząc grafikę z wykorzystaniem modułu `turtle`. Zanim zaczniemy poznawać inne możliwości języka Python, przypomnimy sobie, jak pisaliśmy program wykorzystujący grafikę żółwia.

Python

pawie oczko.py - D:/Python/pawie oczko.py

File Edit Format Run Options Window Help

```
import turtle

turtle.color("black","green")
turtle.begin_fill()
turtle.circle(100)
turtle.end_fill()

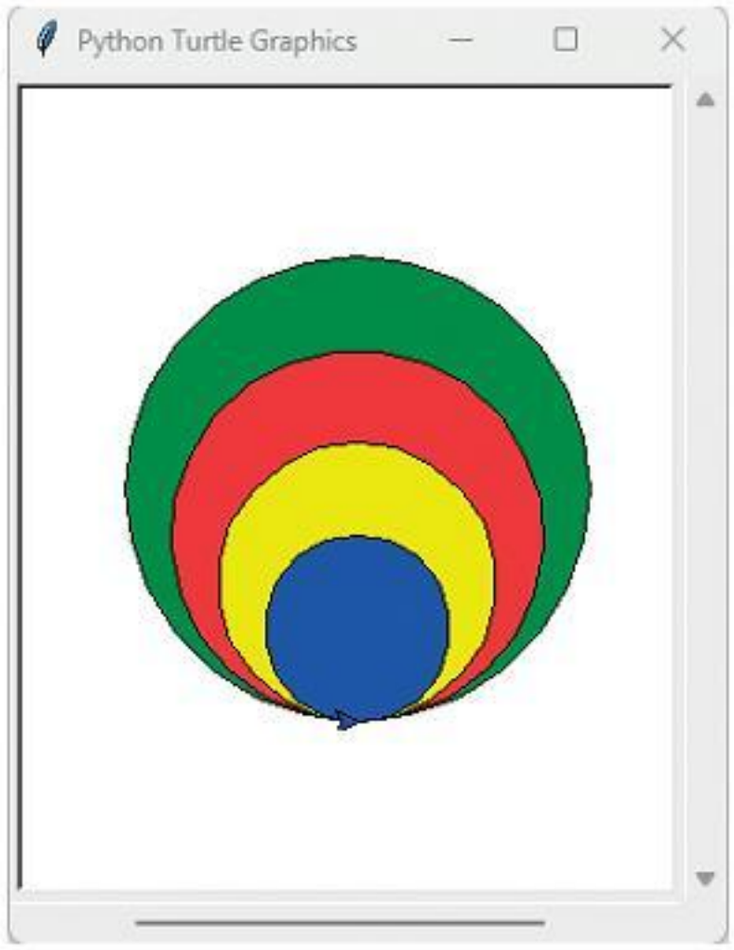
turtle.color("black","red")
turtle.begin_fill()
turtle.circle(80)
turtle.end_fill()

turtle.color("black","yellow")
turtle.begin_fill()
turtle.circle(60)
turtle.end_fill()

turtle.color("black","blue")
turtle.begin_fill()
turtle.circle(40)
turtle.end_fill()
```

Ln: 22 Col: 0

Rys. 2a. Okno edytora kodu źródłowego z napisanym programem



Rys. 2b. Wynik działania programu widoczny w oknie **Python Turtle Graphics**

Program zapisujemy w pliku na dysku, wybierając w oknie edytora opcję **File/Save As**. Następnie uruchamiamy program poprzez wywołanie interpretera Pythona, wybierając opcję **Run/Run Module** (lub naciskając klawisz **F5**).

W oknie edytora kodu źródłowego umieszczono również opcje otwierania plików w menu **File** (**New File** i **Open**).

W przypadku używania modułu `turtle` wynik działania programu pojawi się w oknie **Python Turtle Graphics** (rys. 2b).

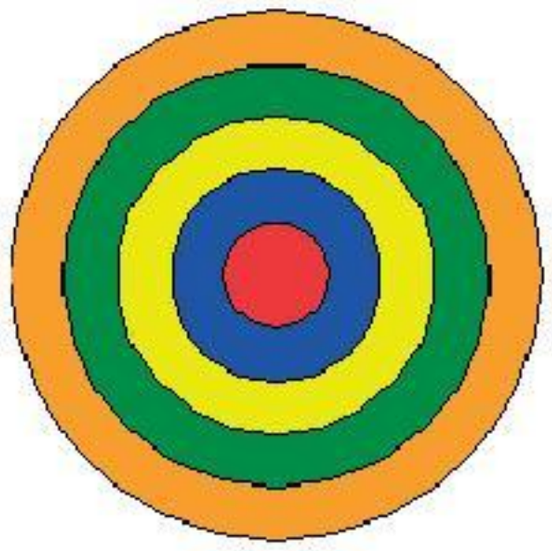
Ćwiczenie 1. Programujemy grafikę w języku Python

1. Uruchom środowisko programistyczne IDLE/Python.
2. Napisz program rysujący pawie oczko, podobne do pokazanego na rysunku 2b. Na rysunku 2a. pokazano kod przykładowego programu. Przeanalizuj go, objaśniając poszczególne instrukcje. Użyj kolorów według własnego pomysłu.
3. Zapisz program w pliku pod nazwą *Pawie oczko*. Uruchom program.

Ćwiczenie 2. Modyfikujemy program

1. Zmodyfikuj program zapisany w ćwiczeniu 1., aby powstała figura złożona z kół współśrodkowych, podobna do pokazanej na rysunku 3. Na koniec schowaj żółwia.
2. Zapisz program w pliku pod nazwą *Koła*. Uruchom program.

Wskazówka: Poza poleceniami użytymi w programie pokazanym na rysunku 2a, przydatne mogą być polecenia: `left()`, `right()`, `forward()`, `penup()`, `pendown()`, `hideturtle()`.



Rys. 3. Wynik działania programu – ćwiczenie 2.

3. Wyprowadzenie napisów na ekran

Język Python, poza instrukcjami umożliwiającymi tworzenie grafiki, posiada instrukcje realizujące czynności, takie jak: wprowadzanie danych, wyprowadzanie wyników, wykonywanie obliczeń, określanie warunków czy powtarzanie operacji. Język Python, tak jak inne języki programowania, posiada odpowiednie zasady składni oraz właściwe słownictwo – podobnie jak języki naturalne, którymi posługujemy się na co dzień.

Wyprowadzanie na ekran komunikatów i wyników umożliwia **instrukcja wyjścia** – funkcja `print()`. Argumentami funkcji (podanymi w nawiasach) mogą być teksty, wyrażenia arytmetyczne lub zmienne oddzielone przecinkami.

Po wyświetleniu komunikatu lub wyniku następuje przejście do nowego wiersza. Jeśli argumentem jest wyrażenie, najpierw zostanie obliczone, a następnie program wyprowadzi jego wartość.

Na przykład:

```
print("Informatyka")
```

– wyprowadza na ekran napis „Informatyka”,

```
print(23 + 17)
```

– wyprowadza na ekran wartość wyrażenia, czyli 40,

```
print(wzrost)
```

– wyprowadza na ekran wartość zmiennej wzrost (o ile zmiennej tej została wcześniej przypisana wartość),

```
print("suma liczb 345 i 36 wynosi:", 345 + 36)
```

– wyprowadza na ekran napis „suma liczb 345 i 36 wynosi: ” i w tym samym wierszu obliczoną wartość wyrażenia (czyli 381).

Python

Napis.py - D:\Python\Napis.py

File Edit Format Run Options Window Help

```
print("Zaczynamy lekcje z programownia")
print("Programujemy w języku Python")
print("Grażyna")
```

Rys. 4a. Okno edytora kodu źródłowego (Python) – ćwiczenie 4.

Python

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36)
[MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more
information.
>>>
= RESTART: D:\Python\Napis.py
Zaczynamy lekcje z programownia
Programujemy w języku Python
Grażyna
>>>
```

Rys. 4b. Okno powłoki Pythona z wynikiem działania programu (wyświetlonymi napisami) – ćwiczenie 4.

Ćwiczenie 3. Wyświetlamy napisy na ekranie

1. Napisz program wyświetlający na ekranie napis: „Jestem na lekcji informatyki.”.
2. Zapisz program w pliku pod nazwą *Napisy*. Uruchom program.

Aby uniknąć popełniania błędów, możesz stosować w edytorze kodu źródłowego operacje kopiowania, wycinania i wklejania fragmentu tekstu z wykorzystaniem **Schowka**, tak jak w każdym innym edytorze tekstu.

Poza poprawnym zapisaniem instrukcji i słów kluczowych musimy również pamiętać o odpowiednim wstawieniu m.in. nawiasów, przecinków, spacji. Jeśli niepoprawnie zapiszemy instrukcję, komputer jej nie wykona, a interpreter wyświetli komunikat o błędzie.

Interpreter tłumaczy program instrukcja po instrukcji. Co to oznacza?

Na rysunku 5a pierwsza instrukcja jest poprawna, ale druga zawiera błąd. W takiej sytuacji (po zapisaniu i uruchomieniu programu) interpreter wyświetli odpowiedni komunikat (rys. 5b). Zostanie wykonana tylko pierwsza instrukcja programu, a druga i trzecia – już nie. Komunikaty interpretera mogą pojawiać się w oddzielnym oknie.

Jeśli podczas pisania programu zwrócimy uwagę na kolory czcionki, możemy uniknąć niepotrzebnych błędów. Warto zauważyć, że czcionka ma różne kolory, np. w nazwie `print()` jest fioletowa, ale w błędnym zapisie tej instrukcji – czarna. Napis umieszczony wewnątrz nawiasów w instrukcji `print()` jest zielony, ale po wyświetleniu na ekranie tekst jest niebieski.

błędnie zapisana instrukcja `print()`

Python

*Napis.py - D:\Python\Napis.py

File Edit Format Run Options Window Help

```
print("Zaczynamy lekcje z programownia")
prnt("Programujemy w języku Python")
print("Grażyna")
```

Rys. 5a. Okno edytora kodu źródłowego z błędnie zapisaną instrukcją programu

Python

wyświetlony pierwszy napis

IDLE Shell

File Edit Shell Debug Options Window Help

```
= RESTART: D:\Python\Napis.py
Zaczynamy lekcje z programownia
Traceback (most recent call last):
  File "D:\Python\Napis.py", line 2, in <module>
    prnt("Programujemy w języku Python")
NameError: name 'prnt' is not defined. Did you mean: 'print'?
>>>
```

komunikat interpretera informujący o nieznanej nazwie (tu: `prnt`)

Rys. 5b. Okno powłoki Pythona z komunikatem interpretera

Ćwiczenie 4. Wyświetlamy napisy na ekranie

1. Zmodyfikuj program zapisany w ćwiczeniu 3., tak aby w drugim wierszu wyświetlało się zdanie „Programujemy w języku Python.", a w trzecim – twoje imię. Zapisz i uruchom program. Jeśli interpreter wykrył błędy, popraw je. Zapisz program i ponownie go uruchom.
2. Dodaj jeszcze dwa inne napisy, w dwóch kolejnych wierszach. Zapisz program i ponownie go uruchom.
3. Zwróć uwagę na różne kolory czcionki kodu pisanego w edytorze kodu źródłowego i napisów wyświetlanych po uruchomieniu programu. Czy jest w tej różnej kolorystyce jakaś prawidłowość? Wyjaśnij na przykładzie wykonywanego ćwiczenia.

4. Stosowanie zmiennych i wykonywanie obliczeń

Aby utworzyć zmienną w środowiskach programowania Scratch i Baltie, określaliśmy jej nazwę (w Baltiem również typ). W języku C++ każdą zmienną trzeba zadeklarować, czyli określić jej nazwę i typ.

W języku Python nie trzeba deklarować zmiennych. Nie musimy wcześniej określać typu zmiennej. Interpreter określa typ zmiennej na podstawie rodzaju danych, jaki jej przypiszemy. Nie ustanawia jednak typu zmiennej na zawsze, ponieważ przy kolejnym przypisaniu może się on zmienić. Na przykład jeśli na początku programu przypiszemy zmiennej *liczba* wartość 34 (która jest liczbą całkowitą), to jej typ jest całkowity. Jeśli jednak w dalszej części programu przypiszemy jej wartość 45,8 (która jest liczbą rzeczywistą), to zmienna *liczba* będzie już typu rzeczywistego.



Zmiennej stosowanej w programie możemy nadać konkretną wartość za pomocą **instrukcji przypisania**.

Znak (operator) = oznacza przypisanie zmiennej zapisanej po lewej stronie znaku wartości umieszczonej po jego prawej stronie.

Na przykład:

```
rok = 2018
```

– zmiennej *rok* przypisujemy wartość 2018, która jest liczbą całkowitą, czyli zmienna *rok* będzie typu całkowitego (`int`),

```
wzrost = 159.5
```

– zmiennej *wzrost* przypisujemy wartość 159,5, która jest liczbą rzeczywistą, czyli zmienna *wzrost* będzie typu rzeczywistego (`float`),

```
nazwisko = "Kowalski"
```

nazwisko będzie typu tekstowego (`str`).

Uwagi:

- W języku Python, tak jak w większości języków programowania, część dziesiętną liczby zapisujemy po kropce, a nie po przecinku jak w matematyce.
- W instrukcji przypisania po prawej stronie znaku równości możemy umieścić wyrażenie, np.: `rok = rok + 1` (co oznacza: zmiennej *rok* przypisz wartość zmiennej *rok* zwiększoną o 1).

W języku Python małe i wielkie litery w nazwach zmiennych (także instrukcji) mają różne znaczenie, np. *Suma* i *suma* to dwie różne zmienne. W nazwach zmiennych powinno się używać liter, znaku podkreślenia i cyfr. Nazwa nie może zaczynać się od cyfry. Przyjęte jest stosowanie małych liter i niestosowanie polskich liter. W przypadku nazw kilkuczłonowych zamiast spacji stosujemy znak podkreślenia. Należy nadawać nazwy, które określają znaczenie danej zmiennej, np. *suma*, *liczba_elementow*.

Ćwiczenie 5. Nadajemy wartości zmiennym

1. W trybie interaktywnym sprawdź, w jaki sposób można przypisać wartość zmiennej. Umieść kolejno następujące polecenia (każde po znaku zachęty):

```
liczba = 23
liczba = liczba + 1
print(liczba)
reszta = liczba % 3
print(reszta)
```

2. Napisz jeszcze pięć instrukcji przypisania wartości (liczb lub wyrażeń) zmiennym według własnego pomysłu. Wykorzystaj operatory arytmetyczne z tabeli 1.

Operator	Działanie	Przykład	Wynik
+	dodawanie	23 + 56	79
-	odejmowanie	987 - 233	754
*	mnożenie	432 * 6	2592
//	dzielenie całkowite (z obcięciem części ułamkowej)	55 // 3	18
/	dzielenie zmiennoprzecinkowe (z zachowaniem części ułamkowej)	55 / 3	18.333333333333332
%	obliczenie reszty z dzielenia dwóch liczb całkowitych	37 % 4	1

Tabela 1. *Podstawowe operatory arytmetyczne w języku Python*

Zmiennej stosowanej w programie możemy nadać również wartość za pomocą instrukcji przypisania, wprowadzając wartość z klawiatury po uruchomieniu programu. W tym celu stosujemy **instrukcję wejścia** – funkcję `input()`.

Argumentem funkcji (podanym w nawiasach) jest tekst (ciąg znaków, łańcuch). Tekst pojawi się na ekranie jako swego rodzaju „zaproszenie" do wpisania ciągu znaków z klawiatury. Jeśli użyjemy funkcji `input()` w instrukcji przypisania, wpisany przez użytkownika ciąg znaków zostanie zapamiętany w zmiennej podanej po lewej stronie:

`nazwa_zmiennej = input("zaproszenie do wprowadzenia danej z klawiatury").`

Na przykład:

```
nazwisko = input("Wprowadź nazwisko: ")
```

– jeśli wpiszemy z klawiatury nazwisko „Kowalski", w zmiennej *nazwisko* zostanie zapamiętany ciąg znaków „Kowalski",

```
a = input("Wprowadź liczbę: ")
```

– jeśli wpiszemy z klawiatury liczbę 346, w zmiennej *a* zostanie zapamiętany ciąg znaków „346", a nie liczba 346.

Aby wprowadzona za pomocą instrukcji `input()` dana została zapamiętana jako liczba, musimy dodać instrukcję, która zamieni ciąg znaków na liczbę:
`int()` – w przypadku liczb całkowitych, `float()` – w przypadku liczb rzeczywistych.

Na przykład:

```
a = int(input("Wprowadź liczbę: "))
```

– jeśli wpiszemy z klawiatury liczbę 346, w zmiennej a zostanie zapamiętana liczba całkowita 346,

```
srednia = float(input("Podaj średnią ocen"))
```

– jeśli wpiszemy z klawiatury liczbę 4.3, w zmiennej *srednia* zostanie zapamiętana liczba rzeczywista 4,3.

Ćwiczenie 6. Piszemy program z użyciem zmiennych wprowadzanych z klawiatury

1. Program pokazany na rysunku 6. pozwala na wprowadzenie z klawiatury dwóch liczb całkowitych *a* i *b*, obliczenie ich sumy i wyprowadzenie wyniku (wartości zmiennej *suma*) na ekran. Przepisz kod tego programu.
2. Zapisz program w pliku pod nazwą *Suma*. Uruchom program.

Wskazówka: Jeśli zamkniemy okno programu i chcemy ponownie uruchomić program, możemy wybrać w oknie powłoki opcję **File/Open**. Program otworzy się w oknie edytora kodu źródłowego (rys. 6.).

Python

Suma.py - C:\Python\Suma.py

File Edit Format Run Options Window Help

```
a = int(input("Wprowadź liczbę: "))
b = int(input("Wprowadź liczbę: "))
suma = a + b
print("Suma wynosi:", suma)
```

Rys. 6. Program obliczający sumę dwóch liczb całkowitych wprowadzanych z klawiatury i wyprowadzający wynik na ekran – ćwiczenie 6.

Po uruchomieniu programu (opcja **Run/Run Module** lub klawisz **F5** w oknie edytora kodu źródłowego) wynik działania programu pokaże się w oknie powłoki Pythona (rys. 7a).

Python

Python Shell

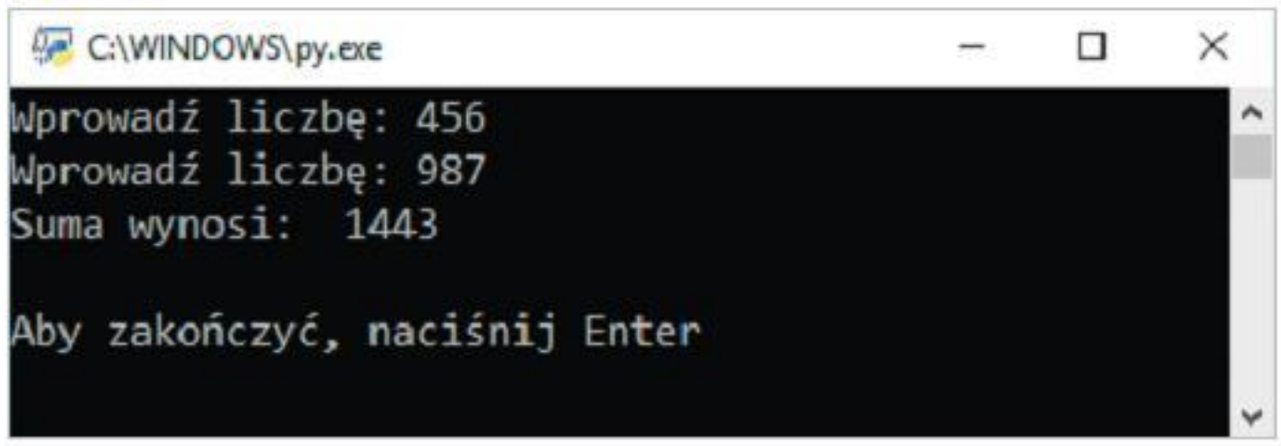
File Edit Shell Debug Options Window Help

```
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct 5 2020, 15:34:40) [MSC v.1927 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Python\Suma.py ==========
Wprowadź liczbę: 456
Wprowadź liczbę: 987
Suma wynosi: 1443
```

Rys. 7a. Wynik działania programu w oknie powłoki Pythona (Python Shell) – ćwiczenie 6.

Program możemy również uruchomić, klikając dwukrotnie w Eksploratorze plików nazwę pliku z zapisanym programem – otworzy się okno, w którym zobaczymy wynik działania programu (rys. 7b). Aby program został wykonany, niezbędny jest interpreter Pythona.

Python



Rys. 7b. Wynik działania programu – ćwiczenie 6.

Aby okno pokazane na rysunku 7b nie zamknęło się automatycznie po wyświetleniu napisów, możemy umieścić na końcu programu instrukcję oczekiwania na naciśnięcie klawisza **Enter**: `input("\n\nAby zakończyć, naciśnij Enter")`.

W tym przypadku wartość zwracana przez funkcję `input()` jest pomijana (nie jest przypisywana żadnej zmiennej). Zapis `\n` odpowiada kodowi znaku przejścia do nowego wiersza. Umieszczenie zapisu `\n\n` oznacza wstawienie dwóch nowych wierszy.

Ćwiczenie 7. Modyfikujemy program

1. Zmodyfikuj program *Suma* zapisany w ćwiczeniu 6., aby wynik działania programu wyglądał jak na rysunku 8.
2. Zapisz program w pliku pod tą samą nazwą. Uruchom program tak.

Wskazówka: Jeśli przy uruchomieniu programu wprowadzisz liczbę rzeczywistą, to interpreter wyświetli błąd.

Python



Rys. 8. Wynik działania programu – ćwiczenie 7.

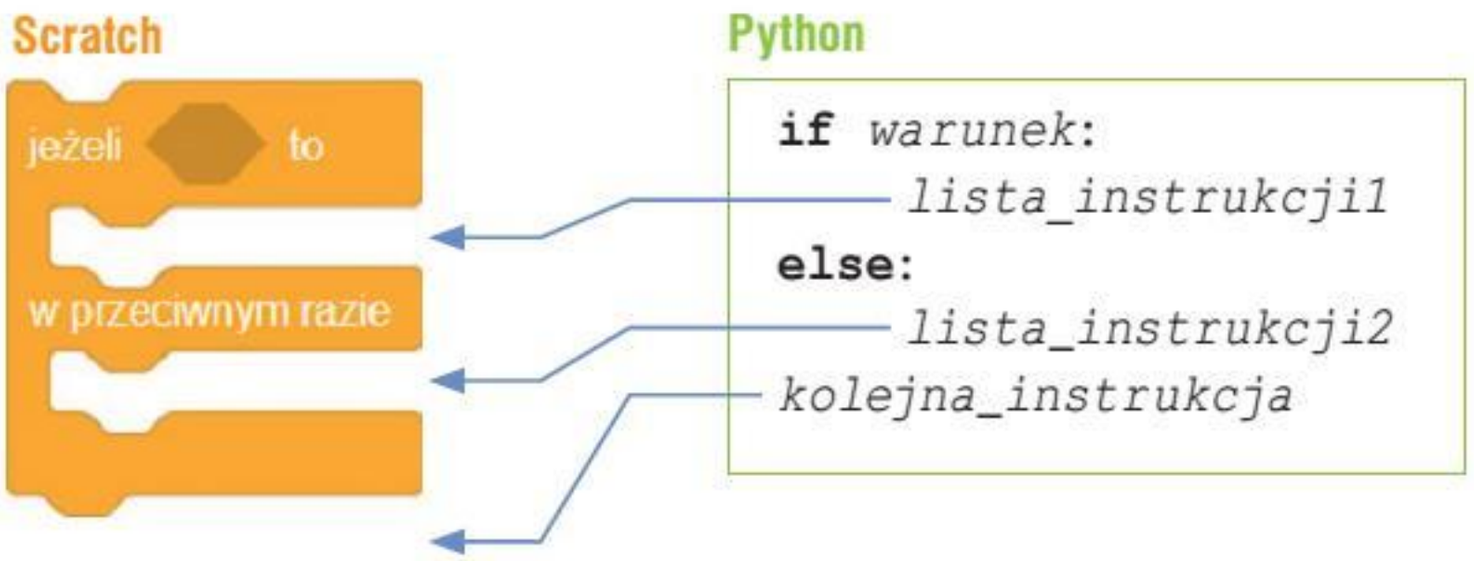
Ćwiczenie 8. Uzupełniamy program

1. Do programu zapisanego w ćwiczeniu 7. dodaj obliczanie iloczynu liczb *a* i *b* oraz wyprowadzanie wyniku mnożenia na ekran (podobnie jak wyniku sumowania w ćwiczeniu 7.).
2. Zapisz program w pliku pod nazwą *Działania*. Uruchom program.

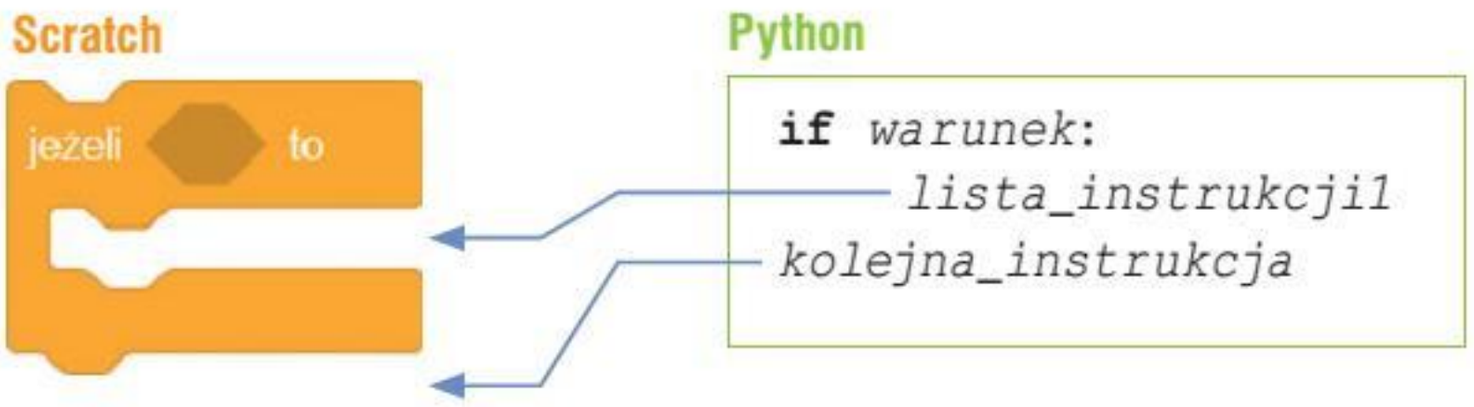
5. Stosowanie instrukcji warunkowej do realizacji algorytmów z warunkami

Zamierzamy napisać w języku Python program obliczający iloraz dwóch liczb pod warunkiem, że druga liczba jest różna od zera. W jaki sposób w języku Python zrealizować sytuację warunkową?

Aby w języku programowania napisać program realizujący algorytm z warunkami, stosujemy **instrukcję warunkową**. Język Python zawiera taką instrukcję – działa ona podobnie jak w środowiskach programowania Baltie i Scratch oraz w języku C++. Instrukcja ta występuje w wersjach pełnej i uproszczonej (rys. 9a, 9b). Jako *lista_instrukcji1* i *lista_instrukcji2* może wystąpić jedna instrukcja (w tym instrukcja warunkowa) lub ciąg instrukcji. Po *warunku* i słowie kluczowym **`else`** umieszczamy dwukropek.



Rys. 9a. Instrukcja warunkowa w języku Python – po lewej odpowiadający jej element w języku Scratch



Rys. 9b. Uproszczona postać instrukcji warunkowej w języku Python – po lewej odpowiadający jej element w języku Scratch

Sprawdzany jest warunek logiczny – jeśli warunek jest spełniony (jest prawdziwy), wykonywana jest *lista_instrukcji1*. Jeśli warunek nie jest spełniony (jest fałszywy) – wykonywana jest *lista_instrukcji2* po słowie **`else`** (rys. 9a), a jeśli brak tej części instrukcji – od razu wykonywana jest *kolejna_instrukcja* po instrukcji warunkowej (rys. 9b).

W języku Python ważne są wcięcia w programie – *listę_instrukcji1* i *listę_instrukcji2* należy przesunąć w prawo przynajmniej o jedną spację (przyjęte jest wcięcie składające się z czterech spacji). Jeśli *lista_instrukcji1* i *lista_instrukcji2* obejmują więcej instrukcji, wszystkie należy tak przesunąć. Jest to sposób wyróżnienia **bloku kodu**.

Jeśli nie zastosujemy wcięcia, komputer potraktuje *listę_instrukcji1* i *listę_instrukcji2* jako *kolejne_instrukcje* i wykona niezależnie od spełnienia warunku.

Operatory		Określenie	Przykład wyrażenia	Interpretacja wyrażenia
Porównania	==	równy	`a == b`	*a* równe *b*
	!=	różny	`a != b`	*a* różne od *b*
	<	mniejszy	`a < b`	*a* mniejsze od *b*
	>	większy	`a > b`	*a* większe od *b*
	>=	większy lub równy	`a >= b`	*a* większe lub równe *b*
	<=	mniejszy lub równy	`a <= b`	*a* mniejsze lub równe *b*
Logiczne	or	alternatywa logiczna (lub)	`a < -5 or a > 5`	*a* mniejsze od –5 lub *a* większe od 5
	and	koniunkcja logiczna (i)	`a > 0 and a < 10`	*a* większe od 0 i *a* mniejsze od 10
	not	negacja logiczna	`not a == 5`	nieprawda, że *a* jest równe 5

Tabela 2. *Podstawowe operatory porównania i logiczne w języku Python*

Python

```
if b != 0:
    iloraz = a / b
    print("a / b = ", iloraz)
else:
    print("Nie dzielimy przez zero")
```

Rys. 10. Przykład stosowania instrukcji warunkowej w języku Python – ćwiczenie 9.



Ćwiczenie 9. Stosujemy instrukcję warunkową

1. Otwórz program *Działania* zapisany w ćwiczeniu 8.
2. Uzupełnij program o obliczenie ilorazu liczb *a* i *b*. Iloraz możemy obliczyć tylko dla *b* różnego od zera – jeśli ten warunek nie będzie spełniony, program powinien wyświetlić odpowiedni napis (rys. 10.).
3. Zapisz program w pliku pod tą samą nazwą. Uruchom program i sprawdź jego działanie dla różnych danych.

Uwaga: Wynik dzielenia liczb całkowitych przypisany zmiennej *iloraz* może nie być liczbą całkowitą. W takiej sytuacji zmienna *iloraz* będzie miała typ rzeczywisty i również otrzymamy poprawny wynik dzielenia.

Python

```
punkty = 100
if sr_ocen > 4 and sr_ocen <= 6:
    punkty_o = 20
print("Liczba punktów:", punkty + punkty_o)
```

Rys. 11. Przykład stosowania instrukcji warunkowej w wersji uproszczonej (Python) – ćwiczenie 10.

Ćwiczenie 10. Stosujemy instrukcję warunkową w wersji uproszczonej

1. Pewien uczeń zdobył 100 punktów w konkursie szkolnym. Jeśli jego średnia ocen będzie wyższa niż 4, uczeń otrzyma dodatkowe 20 punktów. Napisz program obliczający i wyświetlający na ekranie liczbę punktów ucznia. Na rysunku 11. pokazano fragment programu.
2. Dodaj do programu wprowadzanie wartości zmiennej *sr_ocen* typu rzeczywistego (`float`).
3. Zapisz program w pliku pod nazwą *Punkty*. Uruchom program. Sprawdź działanie programu dla kilku różnych danych.

6. Stosowanie instrukcji iteracyjnej do realizacji algorytmów iteracyjnych

Chcemy obliczać, ile metrów bieżących siatki potrzeba na ogrodzenie każdego z pięciu placów zabaw znajdujących się w różnych miejscach. W algorytmie mamy do czynienia z powtarzaniem poleceń obliczania obwodu placu zabaw, czyli z iteracją. W jaki sposób napisać program realizujący algorytm iteracyjny w wybranym języku programowania?

Scratch



Rys. 12a. Element reprezentujący polecenie **powtórz** w języku Scratch

Python

```
import turtle

for i in range(4):
    turtle.left(90)
    turtle.forward(100)
```

Rys. 12b. Rysowanie kwadratu w języku Python

Python

```
for zmienna in lista_wartości:
    lista_instrukcji
```

Rys. 12c. Ogólna postać instrukcji iteracyjnej `for` w języku Python

Aby w języku programowania napisać program realizujący algorytm iteracyjny, korzystamy z **instrukcji iteracyjnych** (instrukcji pętli). W języku Python jedną z takich instrukcji jest instrukcja **`for`**, którą poznaliśmy podczas rysowania kompozycji graficznych z wykorzystaniem modułu `turtle`.

Jako *lista_instrukcji* może wystąpić pojedyncza instrukcja (w tym instrukcja pętli) lub więcej instrukcji (blok instrukcji). *Lista_instrukcji* musi być przesunięta w prawo przynajmniej o jedną spację (przyjęte jest wcięcie składające się z czterech spacji).

Liczbę iteracji określa długość *listy_wartości* po słowie **`in`** (rys. 12b). *Lista_instrukcji* zostanie wykonana dla wszystkich wartości z *listy_wartości*. *Listę_wartości* zapisujemy w różny sposób. Możemy użyć funkcji `range()`, która tworzy sekwencję wartości całkowitych.
Na przykład:

```
for i in range(5):
    print(i)
```

– instrukcja `print(i)` zostanie wykonana pięć razy, a funkcja `range()` wygeneruje kolejne liczby całkowite z przedziału ⟨0, 5), czyli zmienna *i* będzie przyjmować kolejno wartości: 0, 1, 2, 3, 4.

Argumentem funkcji `range()` może być konkretna wartość lub zmienna, np. `range(n)`. Wartość zmiennej *n* możemy wprowadzać z klawiatury. Funkcja `range()` może mieć też dwa lub trzy argumenty.

Ćwiczenie 11. Stosujemy instrukcję **`for`** do rysowania kompozycji graficznych w języku Python z wykorzystaniem modułu `turtle`

1. Przepisz program pokazany na rysunku 13a. Zapisz program w pliku pod nazwą *Kwiat1*. Uruchom program. Ile razy będzie wykonana instrukcja `forward(50)`?
2. Zmodyfikuj program tak, aby najpierw żółw narysował wszystkie okręgi zielone, potem żółte, a na końcu czerwone. Dodatkowo dystans (liczbę) w poleceniu `forward()` wprowadzaj z klawiatury po uruchomieniu programu.
3. Zapisz program w pliku pod nazwą *Kwiat2*. Uruchom program.

Wskazówka: Aby wyświetlić numery wierszy, wybierz z menu **Options** opcję **Show Line Numbers**.

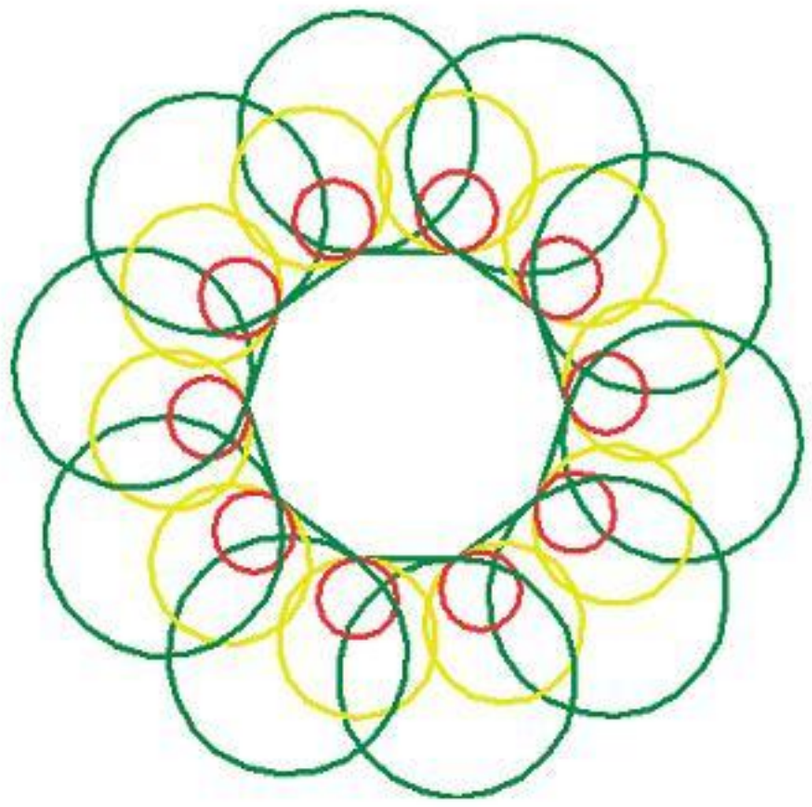
Python

kwiat.py - D:/Python/kwiat.py

File Edit Format Run Options Window Help

```
1  from turtle import *
2
3  pensize(3)
4  for i in range(10):
5      color("green")
6      circle(60)
7      forward(50)
8      color("yellow")
9      circle(40)
10     color("red")
11     circle(20)
12     right(36)
13 hideturtle()
14
15 mainloop()
```

Rys. 13a. Program w języku Python rysujący kompozycję z okręgów – ćwiczenie 11.



Rys. 13b. Wynik działania programu – ćwiczenie 11.

Ćwiczenie 12. Stosujemy instrukcję `for` w języku Python

1. Program pokazany na rysunku 14. realizuje algorytm obliczania obwodu pięciu placów zabaw, których wymiary mają być wprowadzane z klawiatury. Przepisz kod programu pokazany na rysunku 14. Wyjaśnij znaczenie poszczególnych wierszy programu.
2. Zapisz program w pliku pod nazwą *Ogrodzenia_5*. Uruchom program.
3. Odpowiedz na pytania:
   a. Ile razy komputer wykona instrukcję
      `obwod = 2 * szerokosc + 2 * dlugosc`?
   b. Jaka jest wartość zmiennej *i* na koniec programu?

Wskazówka: Aby sprawdzić, jaka jest wartość zmiennej *i* na koniec programu, wyświetl jej wartość na ekranie, umieszczając polecenie: `print(i)` na końcu programu.

Python

*Ogrodzenia_5.py - C:\Python\Ogrodzenia_5.py

File Edit Format Run Options Window Help

```
for i in range(5):
    szerokosc = int(input("Podaj szerokość placu zabaw: "))
    dlugosc = int(input("Podaj długość ogrodu: "))
    obwod = 2 * szerokosc + 2 * dlugosc
    print("Obwód placu zabaw o podanych wymiarach wynosi:", obwod)

input("\n\nNaciśnij Enter, aby zakończyć")
```

Rys. 14. Program realizujący algorytm iteracyjny w języku Python – ćwiczenie 12.

Ćwiczenie 13. Obliczamy obwody *n* placów zabaw

1. Zmodyfikuj program *Ogrodzenia_5* zapisany w ćwiczeniu 12., aby obliczał obwody *n* placów zabaw (wartość *n* wprowadź z klawiatury).
2. Zapisz program w pliku pod nazwą *Ogrodzenia_n*. Uruchom program.
3. Sprawdź działanie programu dla różnych wartości danych *n*, *dlugosc* i *szerokosc*.

Algorytm obliczania obwodów *n* placów zabaw możemy zrealizować w języku Scratch w podobny sposób jak w języku Python. Stosujemy w tym celu polecenie **powtórz**. Musimy również utworzyć zmienne (polecenia dotyczące stosowania zmiennych w języku Scratch umieszczono w grupie **Zmienne**).

Ćwiczenie 14. Stosujemy iterację w języku Scratch

1. Utwórz w programie Scratch program realizujący algorytm obliczania obwodu *n* placów zabaw. Wartość zmiennej *n* i wymiary placów wprowadzaj z klawiatury. Porównaj program z realizacją tego samego algorytmu w języku Python. Jakie widzisz podobieństwa, a jakie różnice?
2. Zapisz program w pliku pod nazwą *Ogrodzenia_n*. Uruchom program i sprawdź jego działanie dla różnych wartości danych *n*, *dlugosc* i *szerokosc*.

Warto zapamiętać

- Kompilator tłumaczy program w całości na język zrozumiały dla procesora, tak aby mógł go wykonać komputer. Natomiast interpreter tłumaczy program instrukcja po instrukcji, tak aby komputer mógł wykonać każdą z nich.
- Środowisko programistyczne Pythona (IDLE) udostępnia dwa tryby pracy: interaktywny i skryptowy. W trybie skryptowym możemy napisać program w edytorze kodu źródłowego, zapisać go w pliku, a następnie uruchomić; jeśli program będzie poprawny, zostanie wykonany.
- Każdy język programowania ma swój zbiór instrukcji, zasady składni i właściwe słownictwo.
- Aby wyświetlić komunikaty i wyniki na ekranie, w języku Python stosujemy instrukcję wyjścia – funkcję `print()`.
- W języku Python wartość przypisana zmiennej określa typ tej zmiennej.
- Zmiennej możemy nadać wartość za pomocą instrukcji przypisania:
  - podając w programie konkretną wartość

  lub
  - wprowadzając tę wartość z klawiatury po uruchomieniu programu (używając funkcji `input()`).
- W języku Python do zapisywania algorytmów, w których występują sytuacje warunkowe, stosujemy instrukcję warunkową **`if`**. Działanie tej instrukcji jest podobne jak w środowiskach programowania Baltie i Scratch oraz w języku C++.
- Jeśli w algorytmie występują powtórzenia, możemy zastosować instrukcję iteracyjną **`for`**. Liczba powtórzeń (iteracji) jest w niej zwykle z góry określona.

Pytania i polecenia

1. Czym różni się kompilacja od interpretacji?
2. W jaki sposób wyświetlamy na ekranie komunikaty i wyniki w języku Python?
3. Jak korzystamy ze zmiennych w języku Python?
4. Jaka będzie wartość zmiennej *b* po wykonaniu programu w języku Python:

```
a = 5
b = 30
a = a + 10
b = a
```

5. Czy po wykonaniu instrukcji przypisania `waga = input("Wprowadź wagę")` i wprowadzeniu z klawiatury liczby 60 komputer zapamięta w zmiennej *waga* liczbę 60? Uzasadnij odpowiedź.
6. W jaki sposób w języku Scratch wprowadzaliśmy dane z klawiatury po uruchomieniu programu (jakich używaliśmy poleceń)? Wskaż na konkretnym przykładzie, na czym polega podobieństwo do wprowadzania danych z klawiatury w języku Python.
7. Czym się różni operator == od = ? Podaj przykłady użycia każdego z tych operatorów.
8. Jak działa instrukcja warunkowa w języku Python? Porównaj jej działanie do działania instrukcji warunkowej w języku Scratch.
9. Kiedy zostaną wykonane instrukcje umieszczone po słowie **`else`** w instrukcji warunkowej?
10. Na czym polega działanie instrukcji **`for`** w języku Python? Podaj przykłady.

Zadania

1. Napisz program obliczający sumę i średnią arytmetyczną trzech liczb rzeczywistych wprowadzanych z klawiatury. Wyniki obliczeń i odpowiednie komunikaty wyświetl na ekranie. Zapisz program w pliku pod nazwą *Średnia*.
2. Napisz program obliczający pole trapezu (*pole*) o podstawach *a* i *b* oraz wysokości *h*. Wartości zmiennych *a*, *b* i *h* wprowadzaj z klawiatury. Wyniki obliczeń i odpowiednie komunikaty wyświetl na ekranie. Zapisz program w pliku pod nazwą *Trapez*.
3. Otwórz program *Trapez* zapisany w zadaniu 2. i dodaj sprawdzanie, czy wysokość *h* jest większa od zera. Dla *h* większego od zera należy wykonać obliczenia pola i wyświetlić wynik, a w przeciwnym wypadku wyświetlić komunikat „błędna dana". Zapisz program w pliku pod tą samą nazwą.
4. Korzystając z programu *Działania* zapisanego w ćwiczeniu 9., sprawdź, czym się różni wynik dzielenia w zależności od typu danych. Nie zapisuj zmian.
5. Otwórz program *Działania* zapisany w ćwiczeniu 9. Dodaj do programu możliwość obliczenia różnicy liczb *a* i *b*. Zapisz program w pliku pod tą samą nazwą.
6. Zmodyfikuj program z zadania 5., aby wykonał obliczenia dla dziesięciu wartości zmiennych *a* i *b* wprowadzonych z klawiatury. Zapisz program w pliku pod nazwą *Działania_10*.
7. Napisz program obliczający objętość (*v*) i pole powierzchni (*p*) sześcianu o boku *a*. Wartość zmiennej *a* wprowadzaj z klawiatury, a wyniki *v* i *p* wyprowadzaj na ekran. Zapisz program w pliku pod nazwą *Sześcian*.
8. Utwórz własną kompozycję graficzną w języku Python wykorzystującą grafikę żółwia. Zastosuj instrukcję `for`.

Dla zainteresowanych

9. Otwórz program zapisany w zadaniu 7. Zmodyfikuj program, aby dla wartości zmiennej *wybor* równej 1 obliczał objętość sześcianu, a dla wartości zmiennej *wybor* równej 0 – pole powierzchni sześcianu (wartość zmiennej *wybor* wprowadzaj z klawiatury). Zapisz program w pliku pod nazwą *Szescian_w*.
10. Zmodyfikuj program z zadania 9., aby wykonał obliczenia dla dziesięciu wartości zmiennej *a* wprowadzonych z klawiatury. Zapisz program w pliku pod nazwą *Sześcian_w_10*.

Przeczytaj, jeśli chcesz wiedzieć więcej…

Język Python zaczął powstawać w roku 1989 jako następca mniej znanego języka ABC. Python odziedziczył po ABC cechę wyróżniającą go wśród innych języków programowania – stosowanie wcięć do wydzielania bloków kodu. W innych językach używa się w tym celu na przykład nawiasów klamrowych `{}` (w C++) lub słów kluczowych `begin` … `end` (w Pascalu). Wcięcia stosuje się, aby zachować przejrzystość programu.

Temat 6.

Zapisywanie algorytmów na liczbach naturalnych w wybranych językach programowania

1. **Realizacja algorytmu Euklidesa w wersji z odejmowaniem**
2. **Badanie podzielności liczb naturalnych**
3. **Realizacja algorytmu Euklidesa w wersji z dzieleniem**
4. **Algorytm wyodrębniania cyfr danej liczby**

Warto powtórzyć

1. Czym jest algorytm i w jaki sposób można go przedstawiać?
2. Czym jest specyfikacja problemu (zadania)?
3. Na czym polega przedstawienie algorytmu w postaci listy kroków?
4. W jaki sposób przedstawiamy w schemacie blokowym sytuację warunkową?
5. W jaki sposób przedstawiamy w schemacie blokowym iterację? Czym jest pętla?

1. Realizacja algorytmu Euklidesa w wersji z odejmowaniem

Jesteś stolarzem i musisz z dwóch desek otrzymać deseczki o takiej samej, ale jak największej, długości. Jak obliczyć długość deseczki, aby podczas cięcia obu desek wykorzystać każdą z nich w całości?

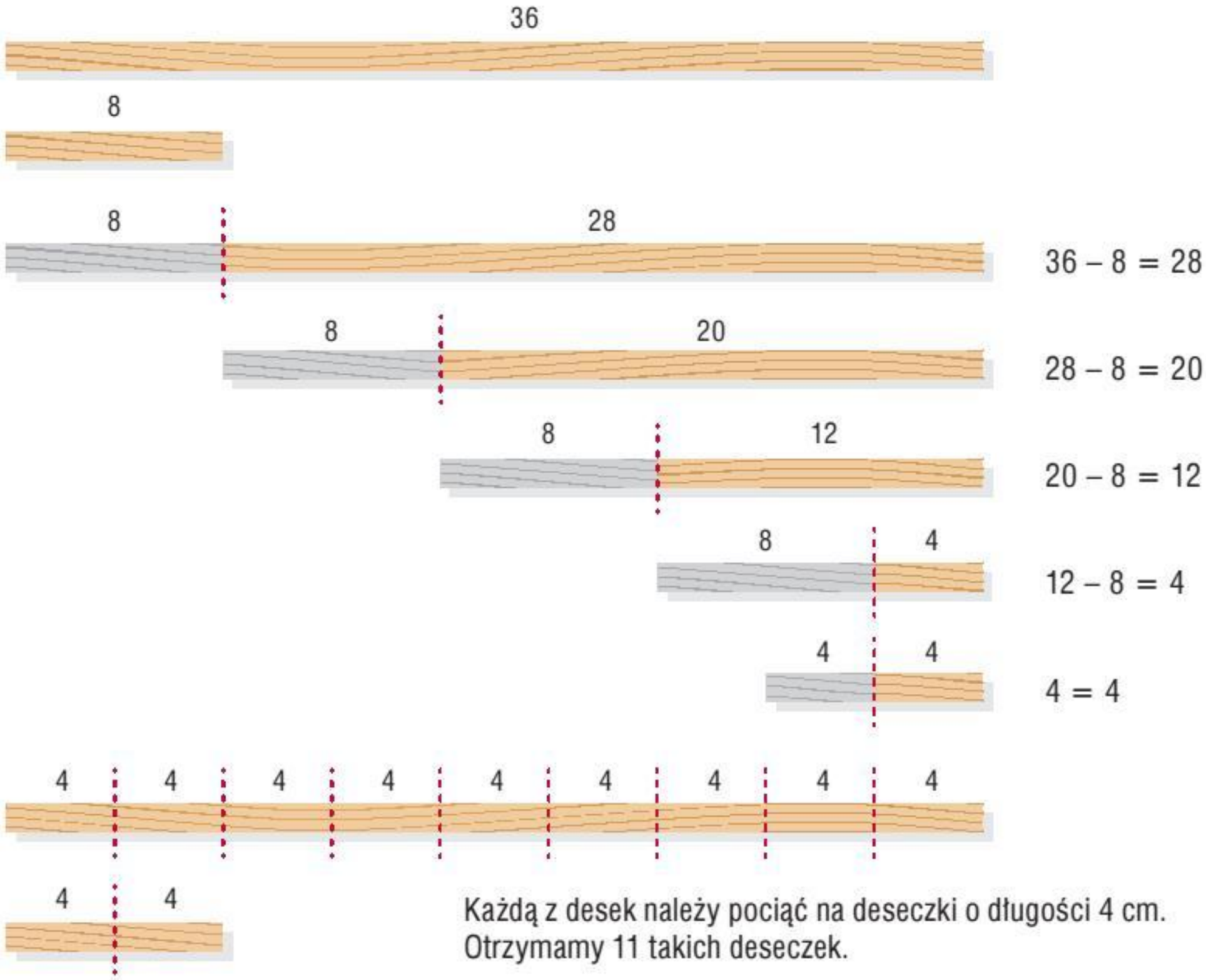
Przyjmijmy, że mamy dwie deski o wymiarach 36 cm i 8 cm. Chcemy otrzymać z obydwu desek możliwie najdłuższe deseczki o takich samych długościach. Szukamy zatem wspólnej miary obydwu desek. Deski będziemy przycinać zgodnie z algorytmem Euklidesa.

Euklides oparł swój algorytm na spostrzeżeniu, że wspólna miara dwóch odcinków odkłada się również całkowitą liczbę razy w różnicy długości tych odcinków. Odejmował od długości dłuższego odcinka długość krótszego aż do otrzymania odcinków o równej długości. Zatem wspólną miarą odcinków okazała się długość dowolnego z końcowych odcinków.

N **NWD**

Największy wspólny dzielnik dwóch lub więcej liczb naturalnych (różnych od zera) to największa liczba naturalna, przez którą dzieli się bez reszty każda z tych liczb.

Do znajdowania największego wspólnego dzielnika (**NWD**) dwóch liczb naturalnych służy **algorytm Euklidesa**.



Rys. 1. Przykład zastosowania algorytmu Euklidesa w praktyce



Algorytm Euklidesa w wersji z odejmowaniem polega na odejmowaniu mniejszej liczby od większej i zastępowaniu większej liczby otrzymaną różnicą, dopóki większa liczba i otrzymana różnica nie będą równe.

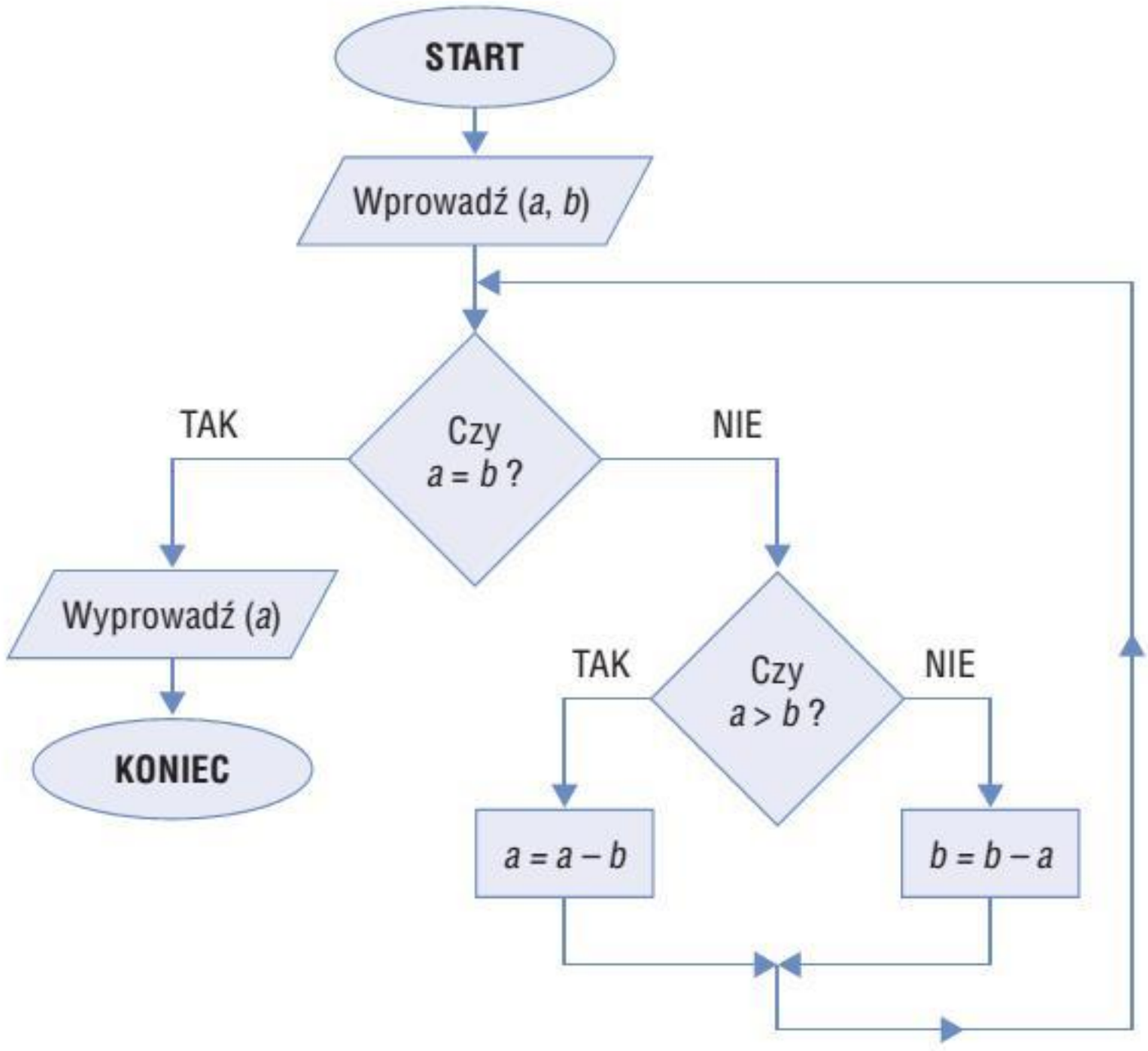
Algorytm Euklidesa – wersja z odejmowaniem

Zadanie: Przedstaw w postaci listy kroków algorytm znajdowania największego wspólnego dzielnika dwóch niezerowych liczb naturalnych.
Dane: Liczby naturalne: a, b ($a \neq 0$, $b \neq 0$).
Wynik: Wartość największego wspólnego dzielnika liczb a i b: NWD.
Lista kroków:

1. Zacznij algorytm.
2. Wprowadź wartość liczby a.
3. Wprowadź wartość liczby b.
4. Sprawdź, czy $a = b$: jeżeli tak, idź do kroku 7.
5. Jeżeli $a > b$, to zmiennej a przypisz wartość wyrażenia $a - b$: $a = a - b$;
   w przeciwnym przypadku zmiennej b przypisz wartość wyrażenia $b - a$: $b = b - a$.
6. Idź do kroku 4.
7. Wyprowadź wynik: NWD jest równe a.
8. Zakończ algorytm.

a	b	Czy a > b?	Działania	Działania
36	**8**	TAK	od większej liczby odejmujemy mniejszą $a - b = 36 - 8 = 28$	zastępujemy większą liczbę otrzymaną różnicą: $a = 28, b = 8$
28	**8**	TAK	od większej liczby odejmujemy mniejszą: $a - b = 28 - 8 = 20$	zastępujemy większą liczbę otrzymaną różnicą: $a = 20, b = 8$
20	**8**	TAK	od większej liczby odejmujemy mniejszą: $a - b = 20 - 8 = 12$	zastępujemy większą liczbę otrzymaną różnicą: $a = 12, b = 8$
12	**8**	TAK	od większej liczby odejmujemy mniejszą: $a - b = 12 - 8 = 4$	zastępujemy większą liczbę otrzymaną różnicą: $a = 4, b = 8$
4	**8**	NIE	od większej liczby odejmujemy mniejszą: $b - a = 8 - 4 = 4$	zastępujemy większą liczbę otrzymaną różnicą: $a = 4, b = 4$
4	**4**	kończymy algorytm, ponieważ $a = b$; przyjmujemy, że NWD jest równe pierwszej z liczb		

Tabela 1. Analiza algorytmu Euklidesa w wersji z odejmowaniem



Rys. 2. Schemat blokowy algorytmu Euklidesa w wersji z odejmowaniem

Ćwiczenie 1. Sprawdzamy działanie algorytmu Euklidesa w wersji z odejmowaniem

Sprawdź działanie algorytmu Euklidesa w wersji z odejmowaniem i wyznacz *NWD* dla liczb: $a = 25$ i $b = 10$ oraz $a = 30$ i $b = 12$.

W algorytmie powtarzamy polecenia, dopóki liczby *a* i *b* są różne, dlatego w programie wykorzystamy odpowiednią instrukcję iteracyjną. Liczba powtórzeń (iteracji) nie jest z góry określona, dlatego w języku Scratch zastosujemy polecenie **powtarzaj aż**, a w środowisku programowania Baltie oraz w językach C++ (rys. 3a) i Python (rys. 3b) – instrukcję `while`. Wewnątrz pętli `while` musimy sprawdzać, która z liczb jest większa, dlatego zastosujemy instrukcję warunkową `if`. Użyjemy także niezbędnych zmiennych.

Algorytm Euklidesa zaprogramujemy w środowiskach programowania Baltie (rys. 4a) i Scratch (rys. 4b), a następnie w języku C++ (rys. 5a) i języku Python (rys. 5b).

C++

```
while(warunek)
   lista_instrukcji;
kolejna_instrukcja;
```

Rys. 3a. Ogólna postać instrukcji iteracyjnej `while` w języku C++

Python

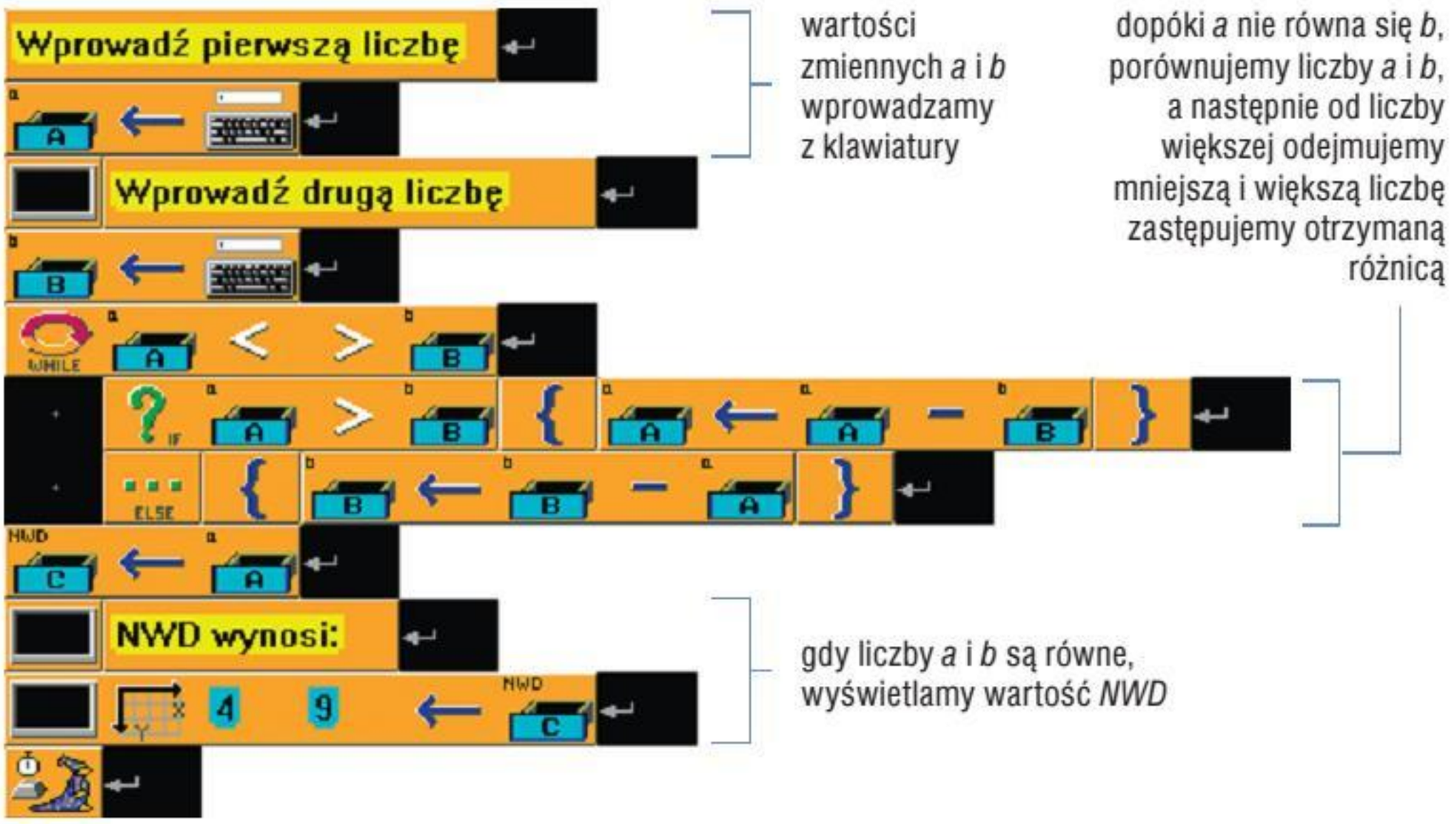
```
while warunek:
   lista_instrukcji
kolejna_instrukcja
```

Rys. 3b. Ogólna postać instrukcji iteracyjnej `while` w języku Python

W środowisku programowania Baltie oraz w językach C++ i Python instrukcja `while` działa tak samo. Instrukcje w pętli są wykonywane (*lista_instrukcji*), dopóki *warunek* jest spełniony. Jeśli *warunek* jest fałszywy, jest wykonywana *kolejna_instrukcja*. W bloku *lista_instrukcji* powinna być instrukcja, która zmienia wartość *warunku* – w przeciwnym wypadku pętla nigdy się nie skończy.

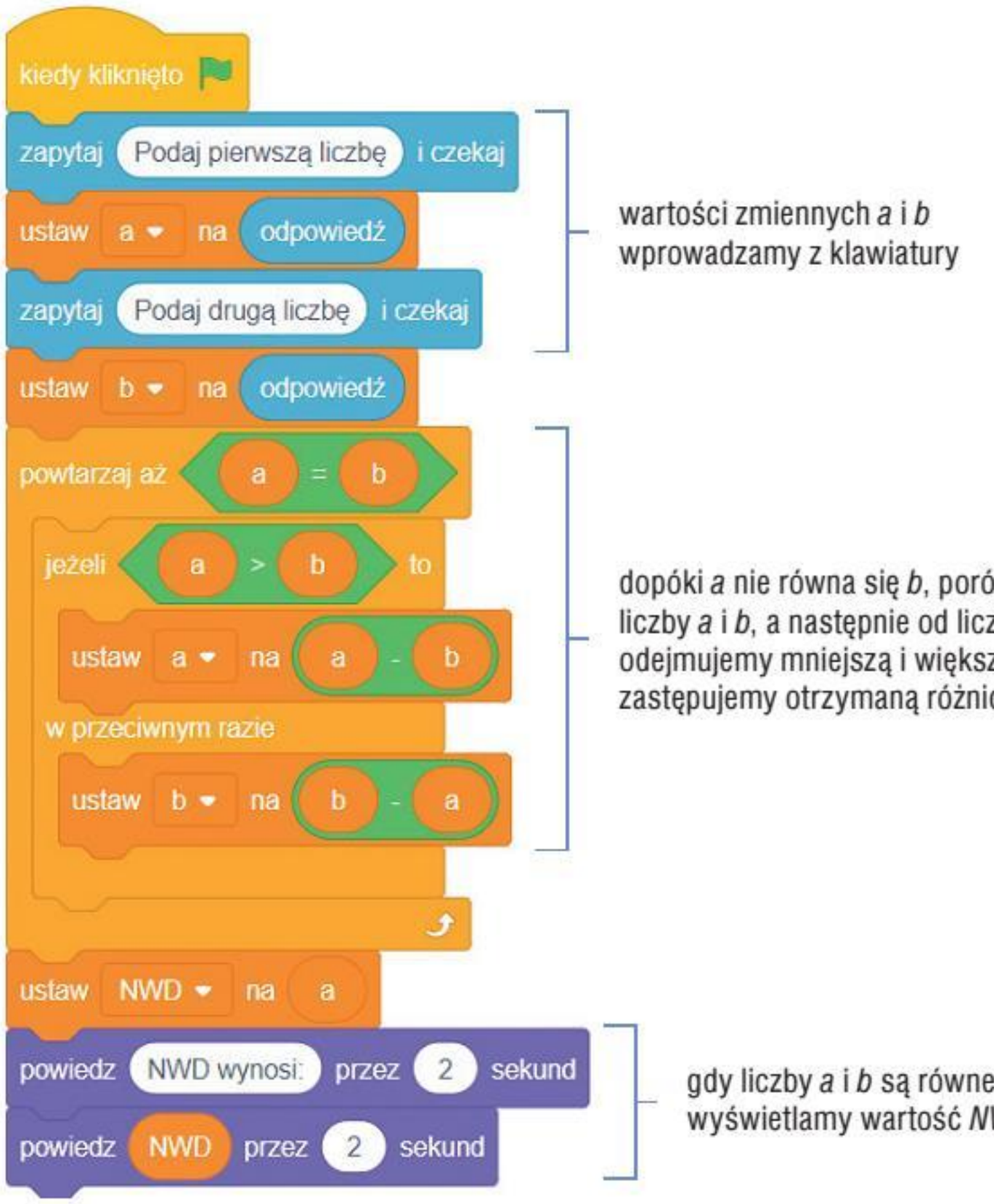
Podobnie działa również instrukcja **powtarzaj aż** w języku Scratch.

Baltie



Rys. 4a. Program realizujący algorytm Euklidesa w wersji z odejmowaniem (Baltie) – ćwiczenie 2.

Scratch



Rys. 4b. Program realizujący algorytm Euklidesa w wersji z odejmowaniem (Scratch) – ćwiczenie 2.

Ćwiczenie 2. W środowisku programowania Baltie lub Scratch tworzymy program realizujący algorytm Euklidesa w wersji z odejmowaniem

1. Umieść w obszarze roboczym polecenia z rysunku 4a lub 4b.
2. Zapisz program w pliku pod nazwą *Euklides_odejmowanie*.
3. Uruchom program i sprawdź jego działanie dla różnych danych.
   Objaśnij działanie programu, m.in. uzasadnij użycie poszczególnych poleceń.
   Kiedy zostanie wykonane polecenie przypisania zmiennej *NWD* wartości zmiennej *a*?

Algorytmy, w których wykorzystujemy technikę iteracji (powtarzania tych samych operacji), to **algorytmy iteracyjne**.

C++

Euklides_odejmowanie.cpp

```cpp
#include <iostream>
using namespace std;

int main()
{
    unsigned int a, b;

    cout << "Podaj dwie liczby: " << endl;
    cin >> a;
    cin >> b;

    while (a != b)
    {
        if (a > b)
            a = a - b;
        else
            b = b - a;
    }

    cout << "NWD = " << a << endl;
    return 0;
}
```

typ danych **unsigned int** określa nieujemne liczby całkowite (liczby naturalne)

wartości zmiennych *a* i *b* wprowadzamy z klawiatury

dopóki *a* nie równa się *b*, porównujemy liczby *a* i *b*, a następnie od liczby większej odejmujemy mniejszą i większą liczbę zastępujemy otrzymaną różnicą

gdy liczby *a* i *b* są równe, wyświetlamy wartość *NWD*

Rys. 5a. Program realizujący algorytm Euklidesa w wersji z odejmowaniem (C++) – ćwiczenie 3.

Python

*Euklides_odejmowanie.py

File Edit Format Run Options Window Help

```python
a = int(input("Podaj pierwszą liczbę: "))
b = int(input("Podaj drugą liczbę: "))

while a != b:
    if a > b:
        a = a - b
    else:
        b = b - a

print("NWD =", a)

input("\n\nAby zakończyć, naciśnij Enter")
```

Ln: 16 Col: 0

wartości zmiennych *a* i *b* wprowadzamy z klawiatury

dopóki *a* nie równa się *b*, porównujemy liczby *a* i *b*, a następnie od liczby większej odejmujemy mniejszą i większą liczbę zastępujemy otrzymaną różnicą

gdy liczby *a* i *b* są równe, wyświetlamy wartość *NWD*

Rys. 5b. Program realizujący algorytm Euklidesa w wersji z odejmowaniem (Python) – ćwiczenie 3.

Ćwiczenie 3. Zapisujemy w języku C++ lub Python algorytm Euklidesa w wersji z odejmowaniem

1. Program pokazany na rysunku 5a lub 5b realizuje algorytm Euklidesa w wersji z odejmowaniem. Przepisz jeden z tych programów i zapisz go w pliku pod nazwą *Euklides_odejmowanie*. Uruchom kilkakrotnie program, by go przetestować dla różnych wartości zmiennych *a* i *b*.
2. Objaśnij działanie programu, m.in. użycie poszczególnych instrukcji zgodnie z listą kroków lub schematem blokowym algorytmu.
3. Porównaj działanie instrukcji iteracyjnej w programach utworzonych w środowisku programowania Baltie i języku C++ lub w języku Scratch i języku Python. Co zauważasz?

Ćwiczenie 4. Modyfikujemy program

1. Zmodyfikuj program zapisany w ćwiczeniu 3., aby szukał NWD dla dziesięciu par liczb *a* i *b* wprowadzanych kolejno z klawiatury i wyświetlał wyniki na ekranie.
2. Zapisz program w pliku pod nazwą *Euklides_odejmowanie_10*.

2. Badanie podzielności liczb naturalnych

W wybranym środowisku programowania chcemy napisać program realizujący algorytm badania podzielności liczb. W jaki sposób w środowiskach programowania Scratch i Baltie oraz językach C++ i Python obliczyć resztę z dzielenia i sprawdzić, czy jest równa zero?

Jeśli reszta z dzielenia dwóch liczb naturalnych jest równa zero, mówimy, że pierwsza z liczb (dzielna) jest podzielna przez drugą (dzielnik).

Na przykład:

- dla $a = 21$ i $b = 4$ reszta z dzielenia a przez b wynosi 1, czyli liczba 21 (dzielna) nie jest podzielna przez 4 (dzielnik),
- dla $a = 42$ i $b = 7$ reszta z dzielenia a przez b wynosi 0, czyli liczba 42 (dzielna) jest podzielna przez 7 (dzielnik),
- dla $a = 245$ i $b = 8$ reszta z dzielenia a przez b wynosi 5, czyli liczba 245 (dzielna) nie jest podzielna przez 8 (dzielnik),
- dla $a = 8$ i $b = 16$ reszta z dzielenia a przez b wynosi 8, czyli liczba 8 (dzielna) nie jest podzielna przez 16 (dzielnik).

W językach programowania resztę z dzielenia możemy wyznaczyć za pomocą **operatora** modulo (np. **%**).

W środowisku Baltie do obliczenia reszty z dzielenia stosujemy operator **%** (rys. 6a).

Na przykład:

- dla $a = 28$ i $b = 5$ reszta z dzielenia a przez b wynosi 3: $reszta = a$ **%** $b = 28$ **%** $5 = 3$,
- dla $a = 100$ i $b = 5$ reszta z dzielenia a przez b wynosi 0: $reszta = a$ **%** $b = 100$ **%** $5 = 0$.
- dla $a = 7$ i $b = 14$ reszta z dzielenia a przez b wynosi 7: $reszta = a$ **%** $b = 7$ **%** $14 = 7$.

O **Operator**

Symbol określający rodzaj operacji wykonywanej przez komputer.

Baltie



Rys. 6a. Obliczanie reszty z dzielenia (Baltie) – ćwiczenie 5.

W języku Scratch do obliczania reszty z dzielenia można wykorzystać polecenie **reszta z dzielenia** z grupy **Wyrażenia** (rys. 6b).

Scratch



Rys. 6b. Obliczanie reszty z dzielenia (Scratch) – ćwiczenie 5.

Ćwiczenie 5. W środowisku programowania Baltie lub Scratch tworzymy program obliczający resztę z dzielenia

1. W wybranym środowisku programowania utwórz program obliczający resztę z dzielenia liczby naturalnej *a* przez liczbę naturalną *b* i wyświetlający wynik (wartość zmiennej *reszta*) na ekranie. Liczby *a* i *b* należy wprowadzać z klawiatury.
2. Zapisz program w pliku pod nazwą *Reszta*.

W językach C++ i Python do obliczenia reszty z dzielenia stosujemy operator **%** (rys. 7a i 7b).

C++

```
reszta = a % b;
```

Rys. 7a. Obliczanie reszty z dzielenia (C++) – ćwiczenie 6.

Python

```
reszta = a % b
```

Rys. 7b. Obliczanie reszty z dzielenia (Python) – ćwiczenie 6.

Ćwiczenie 6. W języku C++ lub Python piszemy program obliczający resztę z dzielenia

1. Napisz program w języku C++ lub Python, obliczający resztę z dzielenia liczby naturalnej *a* przez liczbę naturalną *b* i wyświetlający wynik (wartość zmiennej *reszta*) na ekranie. Liczby *a* i *b* należy wprowadzać z klawiatury.
2. Zapisz program w pliku pod nazwą *Reszta*.
3. Porównaj programy utworzone w środowisku programowania Baltie i języku C++ lub w języku Scratch i języku Python. Co zauważasz?

Algorytm badania podzielności liczb

Zadanie: Przedstaw w postaci listy kroków algorytm badania podzielności liczby naturalnej *a* przez różną od zera liczbę naturalną *b*.
Dane: Liczby naturalne: a, b ($b \neq 0$).
Wynik: Reszta z dzielenia liczby *a* przez liczbę *b* oraz odpowiedni komunikat: „*b* jest dzielnikiem *a*" lub „*b* nie jest dzielnikiem *a*".
Lista kroków:
1. Zacznij algorytm.
2. Wprowadź wartość dzielnej *a*.

3. Wprowadź wartość dzielnika *b*.
4. Zmiennej *reszta* przypisz wartość wyrażenia *a* **%** *b*: *reszta* = *a* **%** *b*.
5. Jeżeli wartość zmiennej *reszta* jest równa 0, wyprowadź na ekran informację: „*b* jest dzielnikiem *a*", w przeciwnym przypadku – „*b* nie jest dzielnikiem *a*".
6. Zakończ algorytm.

Przykłady wykonania algorytmu

Dla *a* = 15 i *b* = 6

reszta = 15 **%** 6 = 3

czy *reszta* = 0? NIE 6 *nie jest dzielnikiem* 15

Dla *a* = 18 i *b* = 6

reszta = 18 **%** 6 = 0

czy *reszta* = 0? TAK 6 *jest dzielnikiem* 18

Ćwiczenie 7. Analizujemy listę kroków algorytmu badania podzielności liczb naturalnych

1. Zapoznaj się z listą kroków algorytmu badania podzielności liczb.
2. Wykonaj algorytm badania podzielności dla kilku par liczb.

Ćwiczenie 8. W środowisku programowania Baltie lub Scratch tworzymy program realizujący algorytm badania podzielności liczb naturalnych

1. Otwórz program *Reszta* zapisany w ćwiczeniu 5. Zmodyfikuj go, aby realizował algorytm badania podzielności liczb zgodnie z podaną listą kroków. Jaką instrukcję zastosujesz, aby zrealizować punkt 5. listy kroków? Uzasadnij odpowiedź.
2. Zapisz program w pliku pod nazwą *Podzielność*.
3. Uruchom program i sprawdź jego działanie dla kilku różnych par liczb. Zakładamy, że wprowadzamy prawidłowe dane, czyli dzielna i dzielnik to liczby naturalne i dzielnik jest różny od zera.

Ćwiczenie 9. W języku C++ lub Python zapisujemy algorytm badania podzielności liczb naturalnych

1. Otwórz program *Reszta* zapisany w ćwiczeniu 6. Zmodyfikuj go, aby realizował algorytm badania podzielności liczb zgodnie z podaną listą kroków. Jaką instrukcję zastosujesz, aby zrealizować punkt 5. listy kroków? Uzasadnij odpowiedź.
2. Zapisz program w pliku pod nazwą *Podzielność*.
3. Uruchom program i sprawdź jego działanie dla kilku różnych par liczb. Zakładamy, że wprowadzamy prawidłowe dane, czyli dzielna i dzielnik to liczby naturalne i dzielnik jest różny od zera.
4. Porównaj programy utworzone w środowisku programowania Baltie i języku C++ lub w języku Scratch i języku Python. Co zauważasz?

Ćwiczenie 10. Modyfikujemy program

1. Zmodyfikuj program zapisany w ćwiczeniu 9., aby sprawdzał podzielność liczb tylko dla dzielnika *b* różnego od zera, a w przypadku gdy dzielnik *b* jest równy zero – wyświetlał komunikat „Nie wolno dzielić przez zero".
2. Zapisz program w pliku pod nazwą *Podzielność_zmodyf*.

3. Realizacja algorytmu Euklidesa w wersji z dzieleniem

W wersji z odejmowaniem algorytmu Euklidesa odejmowanie posłużyło znalezieniu reszty z dzielenia dwóch liczb. W jaki sposób możemy zamiast odejmowania zastosować **dzielenie z resztą**?

Wersja z dzieleniem algorytmu Euklidesa polega na wykonywaniu kolejnych dzieleń liczby a (dzielnej) przez liczbę b (dzielnik), aż reszta z dzielenia osiągnie wartość 0. Na przykład dla liczb $a = 25$ i $b = 10$ NWD obliczamy w następujący sposób:

- dzielimy liczbę a przez b:
  $a / b = 25 / 10 = 2$ reszta 5,
- liczbę a zastępujemy liczbą b, a liczbę b – resztą z poprzedniego dzielenia, czyli $a = 10$, $b = 5$,
- dzielimy *liczbę a* przez b:
  $a / b = 10 / 5 = 2$ reszta 0,
- liczbę a zastępujemy liczbą b, a liczbę b – resztą z poprzedniego dzielenia, czyli $a = 5$, $b = 0$.

Kończymy algorytm, ponieważ reszta z dzielenia wynosi 0. Przyjmujemy, że NWD jest równe ostatniej niezerowej reszcie, czyli $NWD = a = 5$.

Algorytm Euklidesa – wersja z dzieleniem

Zadanie: Przedstaw w postaci listy kroków algorytm znajdowania największego wspólnego dzielnika dwóch niezerowych liczb naturalnych w wersji z dzieleniem.
Dane: Liczby naturalne: a, b $(a \neq 0)$, $(b \neq 0)$.
Wynik: Wartość największego wspólnego dzielnika liczb a i b: *NWD*.

Lista kroków:

1. Zacznij algorytm.
2. Wprowadź wartość liczby a.
3. Wprowadź wartość liczby b.
4. Sprawdź, czy $b = 0$: jeżeli tak, idź do kroku 9.
5. Zmiennej *dzielnik* przypisz wartość zmiennej b:
   $dzielnik = b$.
6. Zmiennej b przypisz wartość reszty z dzielenia a przez b:
   $b = a$ **%** b.
7. Zmiennej a przypisz wartość zmiennej *dzielnik*:
   $a = dzielnik$.
8. Idź do kroku 4.
9. Wyprowadź wynik: *NWD* jest równe a.
10. Zakończ algorytm.

Uwaga: Zmienna *dzielnik* przechowuje dzielnik w trakcie wykonywania obliczeń.

Uwaga

Jeśli pod tę samą zmienną podstawimy inną wartość, poprzednia wartość znika, dlatego w niektórych przypadkach musimy wartość danej zmiennej przechować w tzw. **zmiennej pomocniczej**. W omawianym algorytmie jest to zmienna *dzielnik*, w której pamiętamy aktualny dzielnik.

a	b	Działania	Czy reszta = 0?	Działania
36	**8**	dzielimy (z resztą) liczbę *a* przez *b* $a : b = 36 : 8 = 4$ reszta 4	NIE	liczbę *a* zastępujemy liczbą *b*, a liczbę *b* – resztą z dzielenia
8	**4**	dzielimy (z resztą) liczbę *a* przez *b* $a : b = 8 : 4 = 2$ reszta 0	TAK	kończymy algorytm, ponieważ reszta z dzielenia wynosi 0; przyjmujemy, że NWD jest równe ostatniej niezerowej reszcie, czyli NWD = 4

Tabela 2. Analiza algorytmu Euklidesa w wersji z dzieleniem

Ćwiczenie 11. Sprawdzamy działanie algorytmu Euklidesa w wersji z dzieleniem

Sprawdź działanie algorytmu Euklidesa w wersji z dzieleniem i wyznacz *NWD* dla liczb: $a = 56$ i $b = 6$ oraz $a = 24$ i $b = 16$. Skorzystaj z podanej listy kroków.

W algorytmie powtarzamy polecenia, dopóki liczba *b* jest różna od zera, dlatego w programach wykorzystamy odpowiednią instrukcję iteracyjną. W języku Scratch zastosujemy polecenie **powtarzaj aż**, a w środowisku programowania Baltie oraz w językach C++ i Python – instrukcję `while`.

Ćwiczenie 12. W środowisku programowania Baltie lub Scratch tworzymy program realizujący algorytm Euklidesa w wersji z dzieleniem

1. Umieść w obszarze roboczym polecenia pokazane na rysunku 8a lub 8b. Uzupełnij je o polecenia wprowadzania z klawiatury danych *a* i *b* oraz wyprowadzania wyniku zgodnie z podaną listą kroków algorytmu Euklidesa w wersji z dzieleniem. Utwórz cztery zmienne: *a*, *b*, *dzielnik* i *NWD*.
2. Zapisz program w pliku pod nazwą *Euklides_dzielenie*.

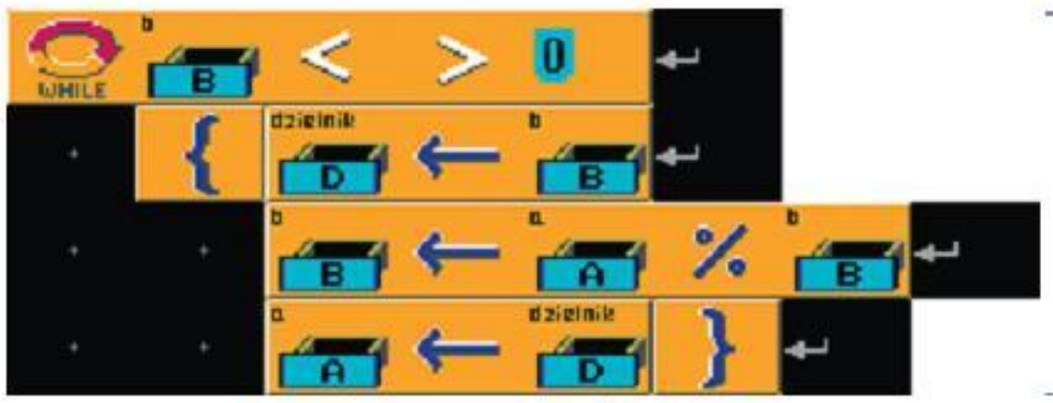
Wskazówka: Wprowadzanie wartości zmiennych, wyświetlanie wyniku i napisów możesz wzorować na programie realizującym algorytm Euklidesa w wersji z odejmowaniem (rys. 4a lub 4b).

Ćwiczenie 13. W języku C++ lub Python zapisujemy algorytm Euklidesa w wersji z dzieleniem

1. Napisz w edytorze kodu źródłowego instrukcje pokazane na rysunku 9a lub 9b. Uzupełnij je o polecenia wprowadzania z klawiatury danych *a* i *b* oraz wyprowadzania wyniku zgodnie z podaną listą kroków algorytmu Euklidesa w wersji z dzieleniem.
2. Zapisz program w pliku pod nazwą *Euklides_dzielenie*.
3. Porównaj programy utworzone w środowisku programowania Baltie i języku C++ lub w języku Scratch i języku Python. Co zauważasz?

Wskazówka: Wprowadzanie wartości zmiennych, wyświetlanie wyniku i napisów możesz wzorować na programie realizującym algorytm Euklidesa w wersji z odejmowaniem (rys. 5a lub 5b).

Baltie



dopóki *b* jest różne od 0, przypisujemy:
zmiennej *dzielnik* – wartość zmiennej *b*,
zmiennej *b* – resztę z dzielenia *a* przez *b*,
zmiennej *a* – wartość zmiennej *dzielnik*

w trakcie wykonywania obliczeń
zmienna *dzielnik* przechowuje
aktualny dzielnik

Rys. 8a. Fragment programu realizującego algorytm Euklidesa w wersji z dzieleniem (Baltie) – ćwiczenie 12.

Scratch



dopóki *b* jest różne od 0, przypisujemy:
zmiennej *dzielnik* – wartość zmiennej *b*,
zmiennej *b* – resztę z dzielenia *a* przez *b*,
zmiennej *a* – wartość zmiennej *dzielnik*

w trakcie wykonywania obliczeń
zmienna *dzielnik* przechowuje
aktualny dzielnik

Rys. 8b. Fragment programu realizującego algorytm Euklidesa w wersji z dzieleniem (Scratch) – ćwiczenie 12.

C++

```
while( b != 0 )
{
    dzielnik = b;
    b = a % b;
    a = dzielnik;
}
```

dopóki *b* jest różne od 0, przypisujemy:
zmiennej *dzielnik* – wartość zmiennej *b*,
zmiennej *b* – resztę z dzielenia *a* przez *b*,
zmiennej *a* – wartość zmiennej *dzielnik*

w trakcie wykonywania obliczeń zmienna *dzielnik*
przechowuje aktualny dzielnik

Rys. 9a. Fragment programu realizującego algorytm Euklidesa w wersji z dzieleniem (C++) – ćwiczenie 13.

Python

```
while b != 0:
    dzielnik = b
    b = a % b
    a = dzielnik
```

dopóki *b* jest różne od 0, przypisujemy:
zmiennej *dzielnik* – wartość zmiennej *b*,
zmiennej *b* – resztę z dzielenia *a* przez *b*,
zmiennej *a* – wartość zmiennej *dzielnik*

w trakcie wykonywania obliczeń zmienna *dzielnik*
przechowuje aktualny dzielnik

Rys. 9b. Fragment programu realizującego algorytm Euklidesa w wersji z dzieleniem (Python) – ćwiczenie 13.

4. Algorytm wyodrębniania cyfr danej liczby

W wybranym środowisku programowania chcemy napisać program realizujący algorytm wyodrębniania cyfr danej liczby. Jak to zrobić w środowiskach programowania Scratch i Baltie oraz językach C++ i Python, wykorzystując obliczanie reszty z dzielenia?

W systemie dziesiętnym używamy potęg liczby 10. Liczbę naturalną w systemie dziesiętnym możemy rozpisać w następujący sposób:

$$647 = 6 \cdot 100 + 4 \cdot 10 + 7 \cdot 1 = 6 \cdot 10^2 + 4 \cdot 10^1 + 7 \cdot 10^0.$$

Cyfry danej liczby naturalnej możemy wyodrębniać od najmniej znaczącej (dla liczby 647 cyfrą tą jest cyfra jedności: 7) lub od najbardziej znaczącej (dla 647 – to cyfra setek: 6).

Do obliczania reszty z dzielenia zastosujemy operator modulo (**%**).



Aby wyodrębnić cyfry danej liczby naturalnej (od najmniej znaczącej), powtarzamy obliczanie reszty z dzielenia tej liczby przez 10 (pierwsza reszta jest cyfrą jedności), następnie od liczby odejmujemy resztę i wynik dzielimy przez 10, aż do otrzymania wyniku odejmowania równego zero.

Przykład wykonania algorytmu

Dla *liczba* = 647

cyfra = *liczba* **%** 10 = 647 **%** 10 = **7**
liczba = (*liczba* – *cyfra*) / 10 = (647 – 7) / 10 = 640 / 10 = 64
czy *liczba* = 0? NIE
cyfra = *liczba* **%** 10 = 64 **%** 10 = **4**
liczba = (*liczba* – *cyfra*) / 10 = (64 – 4) / 10 = 60 / 10 = 6
czy *liczba* = 0? NIE
cyfra = *liczba* **%** 10 = 6 **%** 10 = **6**
liczba = (*liczba* – *cyfra*) / 10 = (6 – 6) / 10 = 0 / 10 = 0
czy *liczba* = 0? TAK

Uwagi: Zmienna *cyfra* służy do pamiętania wartości kolejnych cyfr (tu: reszt z dzielenia kolejnych wartości zmiennej *liczba* przez 10). Wartość zmiennej *liczba*, której cyfry wyodrębniamy, zmienia się w czasie wykonywania algorytmu.



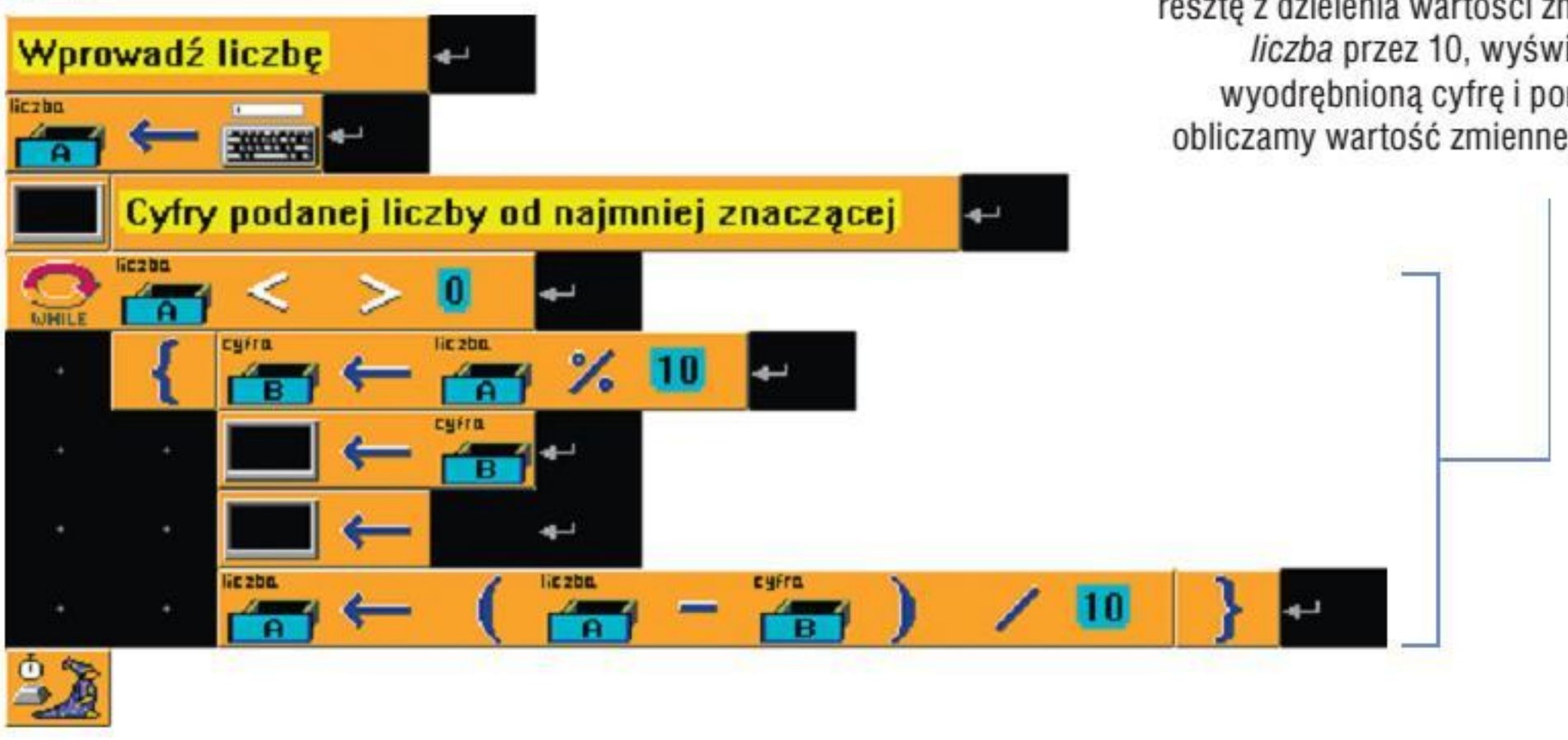
Ćwiczenie 14. Piszemy specyfikację zadania i listę kroków algorytmu wyodrębniania cyfr danej liczby

1. Zapoznaj się z opisem algorytmu wyodrębniania cyfr danej liczby i z podanym przykładem wykonania algorytmu. Wykonaj algorytm dla liczby 8945.
2. Zapisz w zeszycie przedmiotowym specyfikację zadania i listę kroków tego algorytmu.

Wskazówka: Wzoruj się na zapisie specyfikacji zadań i na listach kroków innych algorytmów opisanych w tym temacie.

W algorytmie wyodrębniania cyfr danej liczby powtarzamy polecenia, dopóki wartość zmiennej *liczba* jest różna od zera. W programach wykorzystamy instrukcje iteracyjne, których używa się, gdy liczba powtórzeń (iteracji) nie jest z góry określona: w środowisku programowania Baltie polecenie `while`, a w języku Scratch **– powtarzaj aż**.

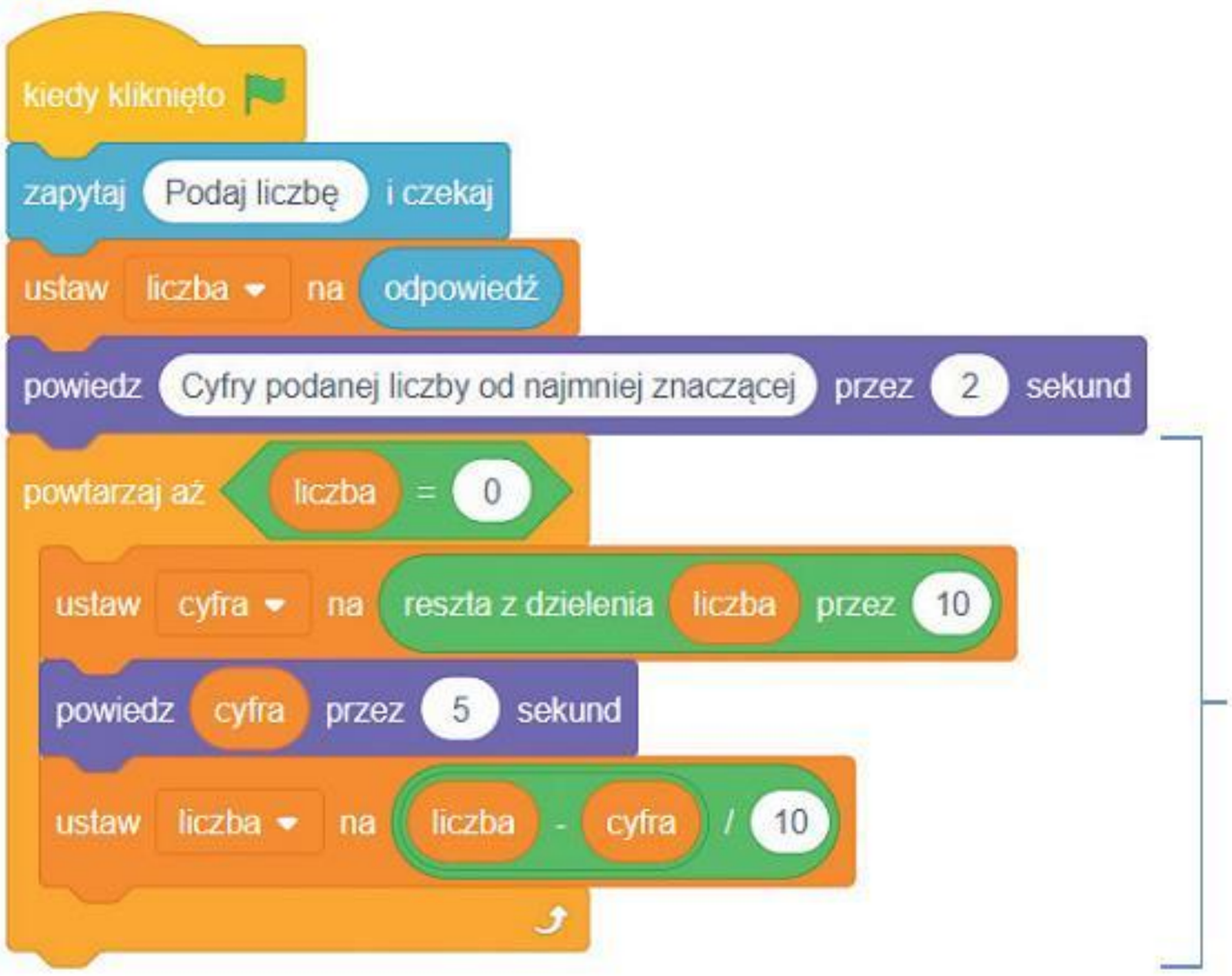
Baltie



dopóki wartość zmiennej *liczba* jest różna od zera, obliczamy resztę z dzielenia wartości zmiennej *liczba* przez 10, wyświetlamy wyodrębnioną cyfrę i ponownie obliczamy wartość zmiennej *liczba*

Rys. 10a. Program realizujący algorytm wyodrębniania cyfr danej liczby (Baltie) – ćwiczenie 15.

Scratch



dopóki wartość zmiennej *liczba* jest różna od zera, obliczamy resztę z dzielenia wartości zmiennej *liczba* przez 10, wyświetlamy wyodrębnioną cyfrę i ponownie obliczamy wartość zmiennej *liczba*

Rys. 10b. Program realizujący algorytm wyodrębniania cyfr danej liczby (Scratch) – ćwiczenie 15.

Ćwiczenie 15. W środowisku programowania Baltie lub Scratch tworzymy program realizujący algorytm wyodrębniania cyfr danej liczby

1. Umieść w obszarze roboczym polecenia pokazane na rysunku 10a lub 10b.
2. Zapisz program w pliku pod nazwą *Cyfry*.
3. Uruchom program i sprawdź jego działanie dla różnych danych.
4. Objaśnij działanie programu, m.in. uzasadnij użycie poszczególnych poleceń.
5. Odpowiedz na pytania:
   Ile razy powtarzają się polecenia umieszczone w pętli dla liczby trzycyfrowej, a ile – dla jednocyfrowej?
   Dla jakiej wartości zmiennej liczba *kończą się obliczenia?*

   Wskazówka: Aby w środowisku Baltie cyfry były wyświetlane z pewnym odstępem, można rozdzielić je czarnym przedmiotem  w instrukcji przypisania:
   
   Czarny przedmiot to pierwszy przedmiot umieszczony w pierwszym wierszu w banku 0.
   Należy odróżniać go od pustego pola  (element z białą kropką pośrodku).

W programach w językach C++ i Python zastosujemy odpowiednie instrukcje iteracyjne, których używamy, gdy liczba powtórzeń (iteracji) nie jest z góry określona. W języku C++ wykorzystamy instrukcję **`do ... while`**, w której warunek (czy *liczba* jest różna od 0?) sprawdzamy na końcu. W języku Python zastosujemy poznaną instrukcję iteracyjną **`while`**.

C++

```
do
   lista_instrukcji;
while(warunek);
kolejna_instrukcja;
```

Rys. 11. Ogólna postać instrukcji **`do ... while`** w języku C++

Instrukcje w pętli (*lista_instrukcji*) są wykonywane, dopóki warunek jest prawdziwy. Jeśli warunek jest fałszywy – wykonywana jest *kolejna_instrukcja*. Niezależnie od wartości początkowej warunku, *lista_instrukcji* zostanie wykonana przynajmniej raz. Jako *lista_instrukcji* może wystąpić pojedyncza instrukcja (w tym instrukcja **`do ... while`**) lub więcej instrukcji ujętych w blok **`{}`**.

Ćwiczenie 16. Zapisujemy w języku C++ lub Python algorytm wyodrębniania cyfr danej liczby

1. Programy pokazane na rysunkach 12a i 12b realizują algorytm wyodrębniania cyfr danej liczby, zaczynając od ostatniej cyfry (cyfry jedności). Przepisz wybrany program i zapisz go w pliku pod nazwą *Cyfry*.
2. Uruchom kilkakrotnie program, by go przetestować dla różnych wartości zmiennej *liczba*.
3. Objaśnij działanie programu, m.in. użycie poszczególnych instrukcji. Porównaj programy utworzone w środowisku programowania Baltie i języku C++ lub w języku Scratch i języku Python. Co zauważasz?

C++

[*] Cyfry.cpp

```cpp
#include <iostream>
using namespace std;

int main()
{
    unsigned int liczba, cyfra;

    cout << "Podaj liczbe: ";
    cin >> liczba;
    cout << "Cyfry liczby " << liczba << " od ostatniej cyfry:" << endl;
    do
    {
        cyfra = liczba % 10;
        cout << cyfra << endl;
        liczba = (liczba - cyfra) / 10;
    }
    while (liczba != 0 );

    return 0;
}
```

dopóki wartość zmiennej *liczba* jest różna od zera, obliczamy resztę z dzielenia wartości zmiennej *liczba* przez 10, wyświetlamy wyodrębnioną cyfrę i ponownie obliczamy wartość zmiennej *liczba*

Rys. 12a. Program realizujący algorytm wyodrębniania cyfr danej liczby (C++) – ćwiczenie 16.

Python

Cyfry.py

File Edit Format Run Options Window Help

```python
liczba = int(input("Podaj liczbę: "))

print("Cyfry liczby", liczba, "od ostatniej:")

if liczba == 0:
    print(liczba)
else:
    while liczba != 0:
        cyfra = liczba % 10
        print(cyfra)
        liczba = (liczba - cyfra) // 10

input("\n\nAby zakończyć naciśnij Enter")
```

dopóki wartość zmiennej *liczba* jest różna od zera, obliczamy resztę z dzielenia wartości zmiennej *liczba* przez 10, wyświetlamy wyodrębnioną cyfrę i ponownie obliczamy wartość zmiennej *liczba*

Ln: 17 Col: 0

Rys. 12b. Program realizujący algorytm wyodrębniania cyfr danej liczby (Python) – ćwiczenie 16.

C++

C:\C++\Cyfry.exe

```
Podaj liczbe: 746
Cyfry liczby 746 od ostatniej:
6
4
7
```

Rys. 13a. Wynik działania programu realizującego algorytm wyodrębniania cyfr danej liczby (C++) – ćwiczenie 16.

Python

C:\WINDOWS\py.exe

```
Podaj liczbę: 746
Cyfry liczby 746 od ostatniej:
6
4
7
```

Rys. 13b. Wynik działania programu realizującego algorytm wyodrębniania cyfr danej liczby (Python) – ćwiczenie 16.

Ćwiczenie 17. Modyfikujemy program

1. Zmodyfikuj program zapisany w ćwiczeniu 16., aby wyodrębniał cyfry dziesięciu liczb wprowadzanych z klawiatury i wyświetlał kolejne wyniki na ekranie. Zapisz program w pliku pod nazwą *Cyfry_10*.
2. Odpowiedz na pytanie: *Ile razy powtarzają się polecenia w pętli* **`do`** *...* **`while`** *lub* **`while`** *dla liczby trzycyfrowej, a ile razy – dla jednocyfrowej?* Uzasadnij odpowiedź.
3. **C++** Zamień instrukcję: `liczba = (liczba - cyfra) / 10;` na instrukcję: `liczba = liczba / 10;`
   Python Zamień instrukcję: `liczba = (liczba - cyfra) // 10` na instrukcję: `liczba = liczba // 10`.
   Czy zmieniło się działanie programu? Uzasadnij odpowiedź.

Warto zapamiętać

- Największy wspólny dzielnik (NWD) dwóch liczb naturalnych możemy znaleźć, stosując algorytm Euklidesa. Algorytm ten możemy zrealizować w dwóch wersjach: z odejmowaniem i z dzieleniem.
- Liczba *b* jest dzielnikiem liczby *a*, jeśli reszta z dzielenia *a* przez *b* jest równa zero. W programie badania podzielności liczb możemy wykorzystać operator modulo (np. **%**) do obliczania reszty z dzielenia.
- W algorytmie wyodrębniania cyfr danej liczby naturalnej wykorzystujemy dzielenie z resztą danej liczby przez dziesięć. Reszty z dzielenia są kolejnymi cyframi liczby.

Pytania i polecenia

1. Do czego służy algorytm Euklidesa?
2. Na czym polega algorytm Euklidesa w wersji z odejmowaniem?
3. Wyjaśnij na przykładzie liczb 36 i 8, dlaczego warunkiem zakończenia algorytmu Euklidesa w wersji z odejmowaniem jest $a = b$.
4. Na czym polega algorytm badania podzielności liczb naturalnych?
5. W jaki sposób można obliczyć w wybranym środowisku programowania resztę z dzielenia dwóch liczb naturalnych? Podaj przykład. Pamiętaj, że dzielnik nie może być równy zero.
6. Jak sprawdzić, czy liczba *y* jest dzielnikiem liczby *x*? Przedstaw na przykładzie.
7. Na czym polega algorytm Euklidesa w wersji z dzieleniem?
8. Czym różni się wersja algorytmu Euklidesa z odejmowaniem od wersji z dzieleniem? Wyjaśnij na przykładzie liczb 36 i 8.
9. Na czym opiera się idea algorytmu wyodrębniania cyfr danej liczby?

Zadania

1. Napisz listę kroków algorytmu sprawdzania, czy dana liczba naturalna jest parzysta. Jeśli liczba jest parzysta, wyświetlaj napis „Podana liczba jest parzysta", a gdy nieparzysta – „Podana liczba jest nieparzysta".
   Wskazówka: Liczba parzysta jest podzielna przez 2.

2. Narysuj schemat blokowy algorytmu sprawdzania, czy dana liczba naturalna jest parzysta.
3. Napisz program sprawdzający, czy wprowadzona z klawiatury liczba jest parzysta. Skorzystaj z listy kroków utworzonej w zadaniu 1. i schematu blokowego narysowanego w zadaniu 2. Zapisz program w pliku pod nazwą *Parzyste*.
4. Narysuj schemat blokowy algorytmu badania podzielności liczb. Skorzystaj z listy kroków podanej w punkcie 2. tematu.
5. Napisz program sprawdzający, czy wprowadzona z klawiatury liczba jest podzielna przez 3. Jeśli tak, wyprowadź komunikat: „Podana liczba jest podzielna przez 3”. Jeśli nie – „Podana liczba nie jest podzielna przez 3”. Dodatkowo w komunikacie możesz wyświetlać wartość tej liczby. Zapisz program w pliku pod nazwą *Podzielność 3*.
6. Dysponujesz pewną kwotą (*kwota*) i planujesz zakupy. Ceny kolejnych wydatków (*cena*) wprowadzasz z klawiatury. Zakupy planujesz dotąd, aż wartość zmiennej *kwota* będzie równa lub mniejsza od zera. Napisz program realizujący to zadanie. Wartość zmiennych *kwota* i *cena* mają być wprowadzane z klawiatury po uruchomieniu programu. Zapisz program w pliku pod nazwą *Zakupy*.
   Wskazówka: Zastanów się, którą instrukcję iteracyjną najlepiej zastosować.

Dla zainteresowanych

7. Na podstawie listy kroków podanej w punkcie 3. tematu narysuj schemat blokowy algorytmu Euklidesa w wersji z dzieleniem.
8. Napisz program zliczający oddzielnie liczby parzyste i nieparzyste spośród *n* liczb naturalnych wprowadzanych z klawiatury. Wartości liczników *licznik_p* (liczby parzyste) i *licznik_np* (liczby nieparzyste) wyświetl na ekranie z odpowiednim komunikatem. Zapisz program w pliku pod nazwą *Liczby*.
   Wskazówka: Aby zliczać wprowadzane liczby (np. parzyste), można zastosować instrukcję: `licznik_p = licznik_p + 1`.
9. Zmodyfikuj program *Zakupy* zapisany w zadaniu 6. w następujący sposób: jeśli spróbujemy przekroczyć dostępną kwotę, czyli wydać więcej, program na to nie pozwoli, tylko zapyta o kolejną wartość zmiennej *cena*, więc musimy zaplanować zakup czegoś tańszego. Program zakończy działanie, gdy zmienna *kwota* będzie równa zero. Zapisz plik pod tą samą nazwą.
10. Zastanów się, jak sprawdzić podzielność jednej liczby przez inną bez konieczności obliczania reszty z dzielenia. Zapisz algorytm w postaci listy kroków, schematu blokowego lub programu.
11. Narysuj schemat blokowy algorytmu wyodrębniania cyfr danej liczby.
12. Napisz program wyodrębniania cyfr danej liczby od najbardziej znaczącej do najmniej znaczącej. Zapisz program w pliku pod nazwą *Cyfry1*.

Przeczytaj, jeśli chcesz wiedzieć więcej…

Euklides z Aleksandrii był greckim matematykiem, który ok. 300 lat p.n.e. stworzył swoje główne dzieło pt. „Elementy”. W dziele tym opisał m.in. sposób obliczania największej wspólnej miary dwóch odcinków (czyli największego wspólnego dzielnika dwóch liczb naturalnych). W tamtym czasie nie znano pojęcia *algorytmu*, jednak opisana przez Euklidesa metoda dzisiaj mogłaby zostać uznana za algorytm. „Elementy” były przez wiele wieków jedynym rzetelnym podręcznikiem do nauki geometrii.

Temat 7. Algorytmy wyszukiwania i porządkowania

1. Wyszukiwanie elementu w zbiorze nieuporządkowanym
   1.1. Wybieranie większej z dwóch liczb
   1.2. Wyszukiwanie największego elementu w zbiorze nieuporządkowanym
   1.3. Wyszukiwanie danego elementu w zbiorze nieuporządkowanym
2. Wyszukiwanie elementu w zbiorze uporządkowanym metodą połowienia
3. Stosowanie metody połowienia do gry w zgadywanie liczby
4. Porządkowanie elementów zbioru metodą przez wybieranie
5. Porządkowanie elementów zbioru metodą przez zliczanie

Warto powtórzyć

1. Na czym polega iteracja?
2. W jaki sposób zapisujemy iterację w wybranym środowisku programowania?
3. W jaki sposób zapisujemy sytuację warunkową w wybranym środowisku programowania?
4. W jaki sposób przypisujemy zmiennej konkretną wartość, a jak wprowadzamy wartość zmiennej z klawiatury w wybranym środowisku programowania?

1. Wyszukiwanie elementu w zbiorze nieuporządkowanym

Algorytm wyszukiwania elementu w zbiorze nieuporządkowanym pokażemy na przykładzie wyszukiwania największej liczby spośród n liczb, zaczynając od prostego algorytmu dla zbioru dwuelementowego – wyboru większej z dwóch liczb. Następnie pokażemy, jak wyszukać dany element w zbiorze nieuporządkowanym.

1.1. Wybieranie większej z dwóch liczb

Zaczniemy od sformułowania zadania, zapisania jego specyfikacji oraz podania listy kroków algorytmu wyboru większej z dwóch liczb naturalnych.

Algorytm wyboru większej z dwóch liczb

Zadanie: Przedstaw w postaci listy kroków algorytm wyboru większej z dwóch liczb.
Dane: Dwie liczby naturalne: a, b ($a \neq b$).
Wynik: Wartość większej z dwóch liczb.

Lista kroków:

1. Zacznij algorytm.
2. Wprowadź wartość liczby *a*.
3. Wprowadź wartość liczby *b*.
4. Sprawdź, czy liczba *a* jest większa od liczby *b*. Jeśli tak, wyprowadź liczbę *a*, w przeciwnym przypadku wyprowadź liczbę *b*.
5. Zakończ algorytm.

Ćwiczenie 1. Analizujemy algorytm wyboru większej z dwóch liczb

1. Korzystając z podanej listy kroków, wykonaj kilkukrotnie algorytm dla różnych wartości zmiennych.
2. Zmodyfikuj listę kroków, aby zrealizować algorytm wyboru mniejszej z dwóch liczb.

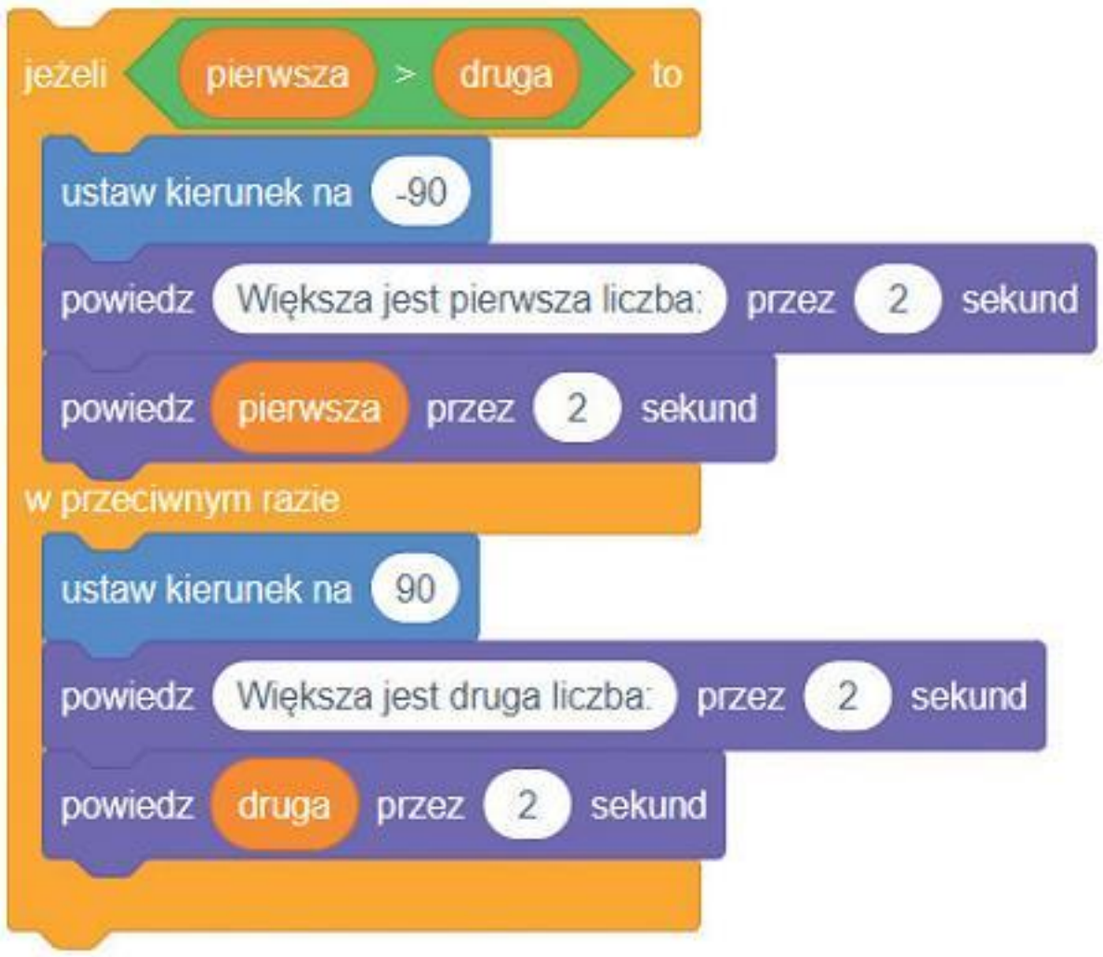
Algorytm wyboru większej z dwóch liczb zaprogramujemy w środowiskach programowania Baltie (rys. 1a) i Scratch (rys. 1b) oraz w językach C++ (rys. 3a) i Python (rys. 3b). W algorytmie występuje sytuacja warunkowa , dlatego zastosujemy również instrukcję warunkową.

Baltie



Rys. 1a. Fragment programu realizującego algorytm wyboru większej z dwóch liczb (Baltie) – ćwiczenie 2.

Scratch



Rys. 1b. Fragment programu realizującego algorytm wyboru większej z dwóch liczb (Scratch) – ćwiczenie 2.

Ćwiczenie 2. Tworzymy program w środowiskach programowania Baltie lub Scratch wybierający większą z dwóch liczb

1. W wybranym środowisku programowania utwórz program wybierający większą z dwóch różnych liczb wprowadzanych z klawiatury, wyprowadzający na ekran odpowiedni komunikat i wynik (większą liczbę). Skorzystaj z podanej listy kroków.
2. Zapisz program w pliku pod nazwą *Większa z dwóch*.

Uwaga (Scratch): Dodatkowo zmień tło i postać duszka. Duszek powinien odwracać się w lewo lub w prawo, zależnie od tego, którą liczbę wprowadza lub wyprowadza (rys. 2.).

Scratch



Rys. 2. Przykładowy ekran programu (Scratch) – ćwiczenie 2.

C++

```
if (a > b)
    cout << "Wieksza jest pierwsza liczba " << a;
else
    cout << "Wieksza jest druga liczba " << b;
```

Rys. 3a. Fragment programu realizującego algorytm wyboru większej z dwóch liczb (C++) – ćwiczenie 3.

Python

```
if a > b:
    print("Większa jest pierwsza liczba", a)
else:
    print("Większa jest druga liczba", b)
```

Rys. 3b Fragment programu realizującego algorytm wyboru większej z dwóch liczb (Python) – ćwiczenie 3.

Ćwiczenie 3. Piszemy program w języku C++ lub Python wybierający większą z dwóch liczb

1. W wybranym języku programowania utwórz program wybierający większą z dwóch różnych liczb wprowadzanych z klawiatury, wyprowadzający na ekran odpowiedni komunikat i wynik (większą liczbę). Skorzystaj z podanej listy kroków.
2. Zapisz program w pliku pod nazwą *Większa z dwóch*.
3. Porównaj realizację algorytmu w programach utworzonych w środowisku Baltie i języku C++ lub w języku Scratch i języku Python. Co zauważasz? Uzasadnij odpowiedź.

1.2. Wyszukiwanie największego elementu w zbiorze nieuporządkowanym

Podczas rozwiązywania niektórych problemów (z różnych dziedzin nauki i życia codziennego) szukamy w danym zbiorze najmniejszego lub największego elementu, np. najmniejszej liczby, najwyższej osoby.

{ Chcemy znaleźć najwyższego ucznia wśród uczniów całej klasy.
W jaki sposób można to zrobić, czyli jaki zastosować algorytm wyszukiwania? }



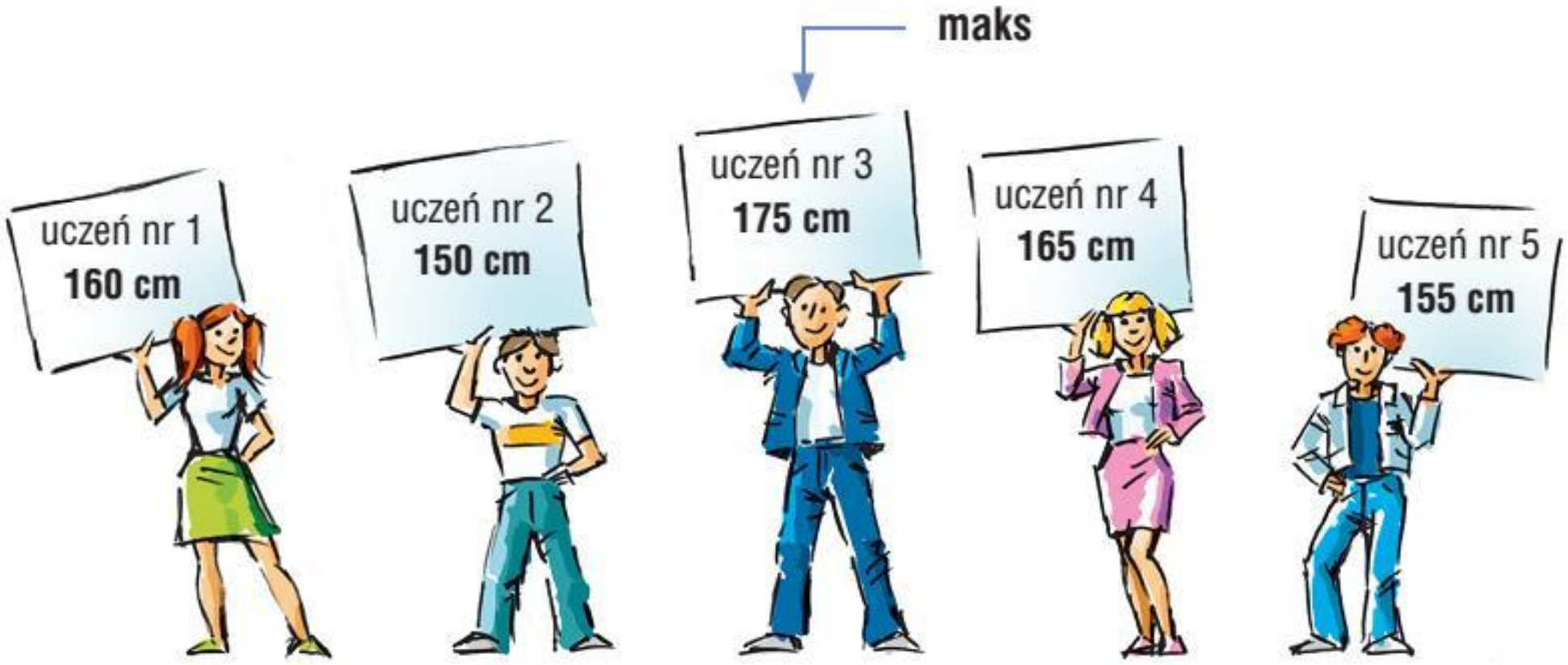
Aby wyszukać największą spośród *n* liczb, porównujemy kolejne liczby ze zbioru z największą znalezioną do tej pory liczbą (zapamiętaną osobno). Na początku algorytmu przyjmujemy, że pierwsza liczba ze zbioru jest największa. Porównania liczb powtarzamy, aż sprawdzimy wszystkie liczby ze zbioru.

Algorytm wyszukiwania największej liczby spośród *n* liczb

Zadanie: Znajdź największą liczbę spośród *n* liczb.
Dane: Liczba naturalna *n* oznaczająca liczbę elementów zbioru, *n* liczb naturalnych wprowadzanych kolejno i zapamiętywanych w zmiennej *a*.
Wynik: Wartość elementu największego: *maks*.
Lista kroków:

1. Zacznij algorytm.
2. Wprowadź liczbę elementów zbioru: *n*.
3. Wprowadź wartość pierwszej liczby: *a*.
4. Przyjmij, że liczbą największą jest pierwsza liczba ze zbioru – zmiennej *maks* przypisz wartość pierwszej liczby: *maks* = *a*.
5. Wprowadź wartość kolejnej liczby: *a*.
6. Porównaj kolejną liczbę ze zbioru z *maks*: czy *a* > *maks*?
7. Jeśli kolejna liczba jest większa od *maks*, to zmiennej *maks* przypisz wartość tej liczby: *maks* = *a*.
8. Powtarzaj *n* – 1 razy kroki 5., 6. i 7.
9. Wyprowadź wynik: *maks*.
10. Zakończ algorytm.

Algorytm wyszukiwania największej liczby spośród *n* liczb zastosujemy do znajdowania wzrostu najwyższego z *n* uczniów klasy (gdzie *n* to liczba uczniów klasy). Sprawdzimy działanie algorytmu dla *n* = 5 (tabela 1.). Na rysunku 4. podaliśmy wzrost uczniów w centymetrach, czyli kolejne wartości *a*.



Rys. 4. Przykład zastosowania algorytmu wyszukiwania największej spośród n liczb

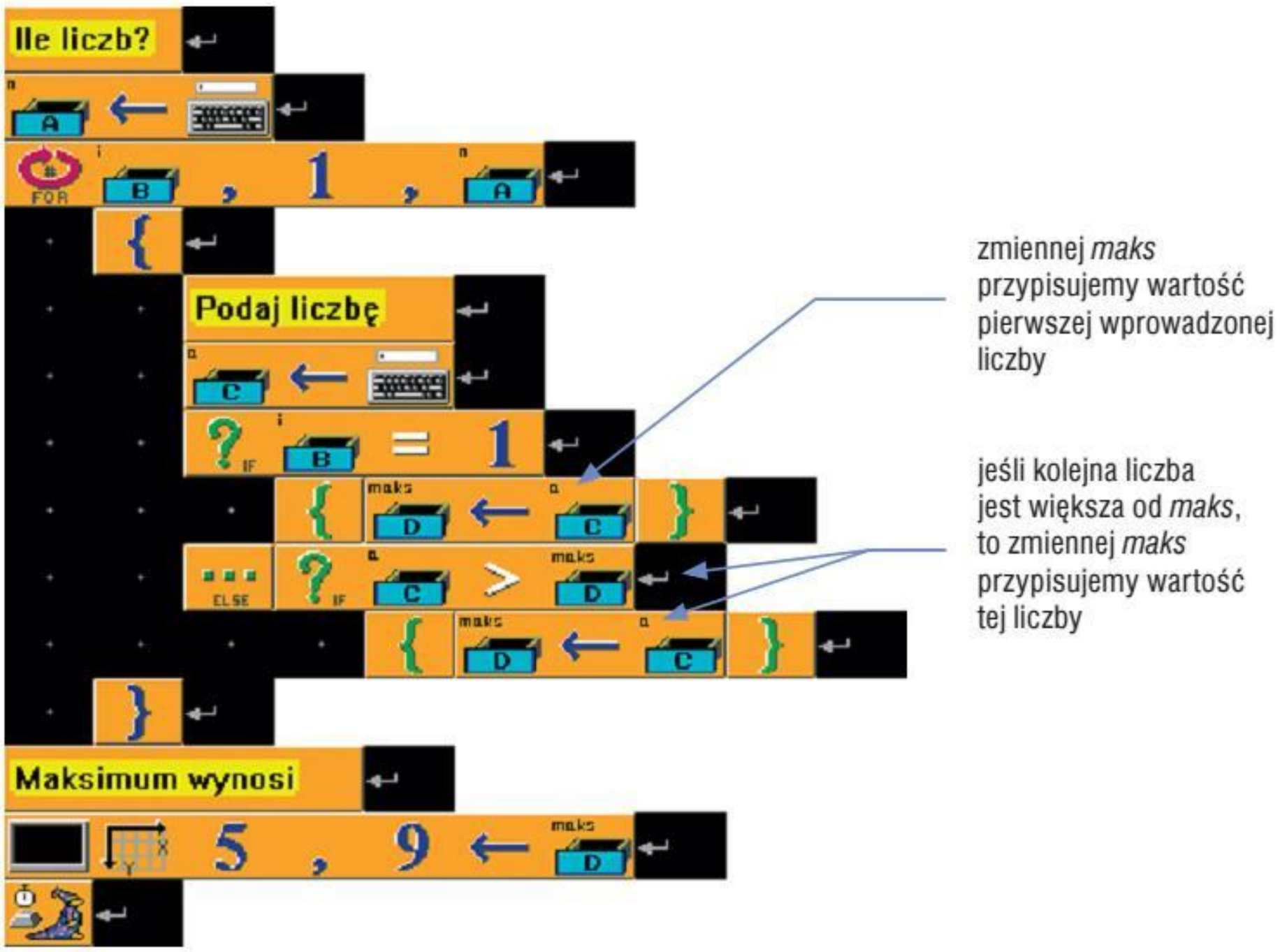
a	czy *a* > *maks*?	rezultat	działanie
wprowadź wzrost pierwszego ucznia: *a* = 160	–	–	zmiennej *maks* przypisz wzrost pierwszego ucznia: *maks* = 160
wprowadź wzrost drugiego ucznia: *a* = 150	porównaj wzrost drugiego ucznia z wartością *maks:* czy 150 > 160?	NIE	pozostaw *maks* bez zmian: *maks* = 160
wprowadź wzrost trzeciego ucznia: *a* = 175	porównaj wzrost trzeciego ucznia z wartością *maks*: czy 175 > 160?	TAK	zmiennej *maks* przypisz wzrost trzeciego ucznia: *maks* = 175
wprowadź wzrost czwartego ucznia: *a* = 165	porównaj wzrost czwartego ucznia z wartością *maks:* czy 165 > 175?	NIE	pozostaw *maks* bez zmian: *maks* = 175
wprowadź wzrost piątego ucznia: *a* = 155	porównaj wzrost piątego ucznia z wartością *maks*: czy 155 > 175?	NIE	pozostaw *maks* bez zmian: *maks* = 175
–	–	–	wzrost najwyższego ucznia to 175 cm.

Tabela 1. *Przykład wykonania algorytmu dla n = 5*

Ćwiczenie 4. Sprawdzamy działanie algorytmu wyszukiwania największej liczby spośród *n* liczb

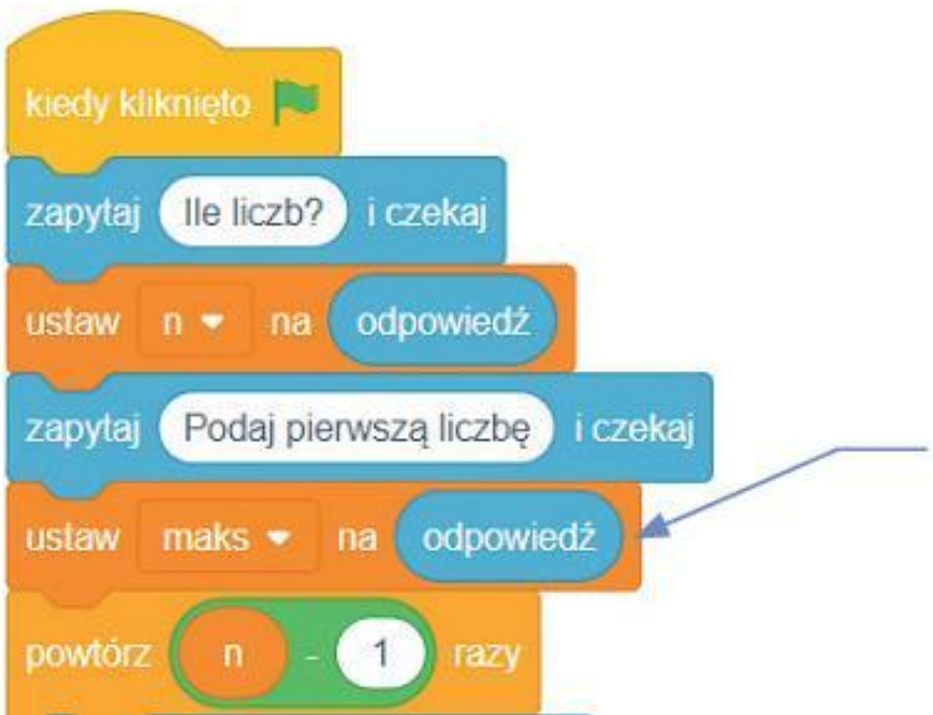
1. Przygotuj odpowiednie pomoce dydaktyczne (rys. 4.) i wykonaj wspólnie z koleżankami i kolegami z klasy algorytm wyszukiwania najwyższego spośród pięciu uczniów.
2. Sprawdź działanie algorytmu wyszukiwania największej liczby:
   a. dla zbioru liczb {70, 160, 155, 180, 185};
   b. dla zbioru liczb {8, -6, 23, 0, -50, 34, -23};
   c. dla dziesięciu wybranych liczb.

Baltie



Rys. 5a. Program realizujący algorytm wyszukiwania największej liczby w zbiorze *n*-elementowym (Baltie) – ćwiczenie 5.

Scratch



Rys. 5b. Program realizujący algorytm wyszukiwania największej liczby w zbiorze *n*-elementowym (Scratch) – ćwiczenie 5.

Algorytm wyszukiwania największej liczby spośród *n* liczb zaprogramujemy w środowiskach programowania Baltie (rys. 5a) i Scratch (rys. 5b), a następnie w językach C++ (rys. 6a) i Python (rys. 6b). W algorytmie powtarzamy polecenia *n* razy, dlatego w każdym z programów zastosujemy odpowiednią instrukcję iteracyjną. Zastosujemy również instrukcję warunkową, ponieważ w algorytmie występują sytuacje warunkowe. Użyjemy również niezbędnych zmiennych.

C++

[*] Maksimum_n.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int main ()
5  {
6      int i, a, n, maks;
7      cout << "Ile liczb? ";
8      cin >> n;
9      for(i = 0; i < n ; i++)
10     {
11         cout << "Podaj liczbe " << i + 1 << ": ";
12         cin >> a;
13         if(i == 0)
14             maks = a;
15         else
16             if(a > maks)
17                 maks = a;
18     }
19     cout << "Maksimum wynosi: " << maks;
20     return 0;
21 }
```

zmiennej *maks* przypisujemy wartość pierwszej wprowadzonej liczby

jeśli kolejna liczba jest większa od *maks*, to zmiennej *maks* przypisujemy wartość tej liczby

Rys. 6a. Program realizujący algorytm wyszukiwania największej liczby w zbiorze *n*-elementowym (C++) – ćwiczenie 6.

Python

*Maksimum_n.py - C:\Python\Maksimum_n.py

File Edit Format Run Options Window Help

```
n = int(input("Ile liczb? "))

for i in range(n):
    a = int(input("Podaj liczbę: "))
    if i == 0:
        maks = a
    else:
        if a > maks:
            maks = a

print("Maksimum wynosi:", maks)

input("\n\nAby zakończyć, naciśnij Enter")
```

zmiennej *maks* przypisujemy wartość pierwszej wprowadzonej liczby

jeśli kolejna liczba jest większa od *maks*, to zmiennej *maks* przypisujemy wartość tej liczby

Rys. 6b. Program realizujący algorytm wyszukiwania największej liczby w zbiorze *n*-elementowym (Python) – ćwiczenie 6.

Ćwiczenie 5. Tworzymy program w środowisku programowania Baltie lub Scratch realizujący algorytm wyszukiwania największej liczby w zbiorze *n*-elementowym

1. Umieść w obszarze roboczym polecenia pokazane na rysunku 5a lub 5b. Zapisz program w pliku pod nazwą *Maksimum_n*.
2. Uruchom program i sprawdź jego działanie dla różnych danych, w tym dla danych pokazanych na rysunku 4. Objaśnij działanie programu, m.in. uzasadnij użycie poszczególnych poleceń zgodnie z podaną listą kroków.

Ćwiczenie 6. Zapisujemy w języku C++ lub Python algorytm wyszukiwania największej liczby w zbiorze *n*-elementowym

1. Programy pokazane na rysunkach 6a i 6b realizują algorytm wyszukiwania największego elementu w zbiorze *n*-elementowym. Przepisz wybrany program i zapisz go w pliku pod nazwą *Maksimum_n*.
2. Uruchom program. Sprawdź jego działanie dla różnych danych, w tym dla danych pokazanych na rysunku 4. Objaśnij działanie programu, m.in. uzasadnij użycie poszczególnych poleceń zgodnie z podaną listą kroków.
3. Porównaj realizację algorytmu w programach utworzonych w środowisku Baltie i języku C++ lub w języku Scratch i języku Python. Co zauważasz? Uzasadnij odpowiedź.

1.3. Wyszukiwanie danego elementu w zbiorze nieuporządkowanym

Szukamy na półce sklepowej butów o rozmiarze 37.
Ktoś jednak poprzestawiał buty i nie są one ustawione według rozmiarów.
Jak sprawdzić, czy buty o rozmiarze 37 stoją na półce i na którym miejscu?



Aby stwierdzić, czy dany element istnieje w nieuporządkowanym zbiorze *n*-elementowym, musimy sprawdzić każdy z elementów zbioru, ponieważ może on znajdować się w dowolnym miejscu (nawet być ostatnim elementem). W szczególnym przypadku szukanego elementu może w zbiorze w ogóle nie być.

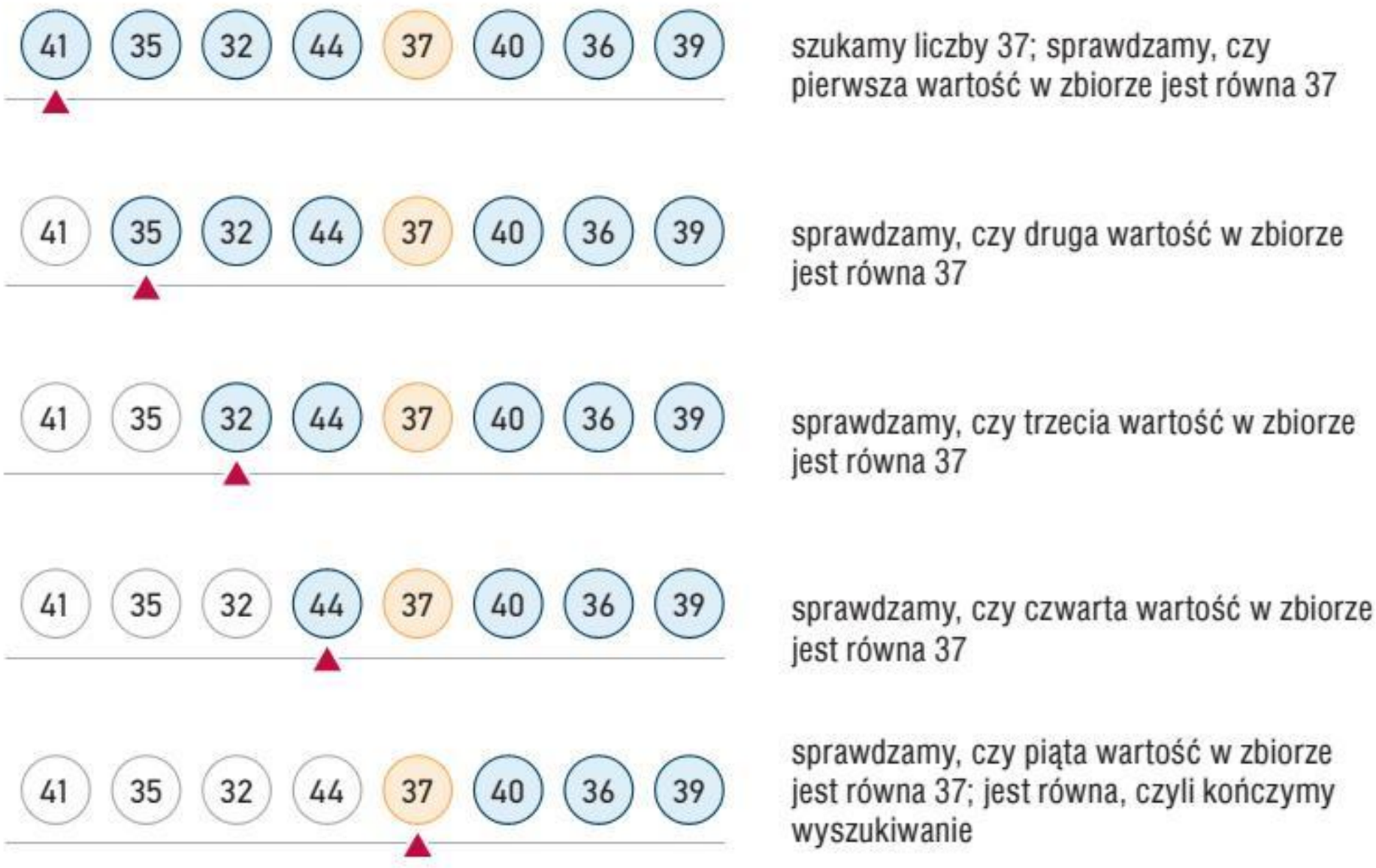
W przypadku poszukiwania na półce pary butów wśród par, które nie są uporządkowane według rozmiarów, będziemy po kolei sprawdzać każdą parę, aby znaleźć buty o rozmiarze 37 (zakładamy, że na półce jest jedna para butów o tym rozmiarze).

Opis algorytmu wyszukiwania elementu w zbiorze nieuporządkowanym (na przykładzie ośmiu liczb)

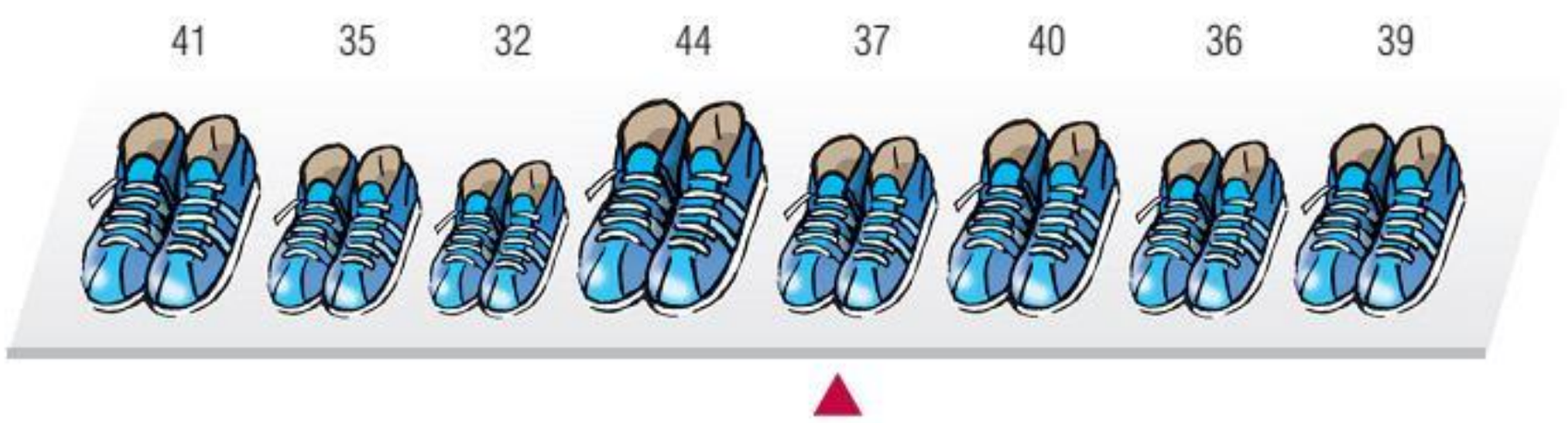
Dany jest nieuporządkowany zbiór ośmiu liczb {41, 35, 32, 44, 37, 40, 36, 39}. Poszukiwana liczba to 37. Zakładamy, że w zbiorze jest tylko jedna taka liczba. Przeglądamy wszystkie kolejne elementy zbioru, stosując przeszukiwanie liniowe.

Zaczynamy od sprawdzenia, czy pierwsza liczba (na rysunku 7. zaznaczona strzałką) jest szukaną liczbą. Jeśli tak, kończymy poszukiwanie. Jeżeli nie – sprawdzamy kolejną liczbę. Algorytm kończy się, gdy znajdziemy liczbę równą szukanej, a jeśli jej nie ma – gdy dojdziemy do końca zbioru.

> **Uwaga**
>
> W sytuacji, kiedy szukany element występuje w zbiorze więcej niż jeden raz, algorytm kończy działanie po natknięciu się na niego po raz pierwszy.



Rys. 7. Przykład wyszukiwania liczby w zbiorze nieuporządkowanym



Rys. 8. Zastosowanie algorytmu wyszukiwania elementu w zbiorze nieuporządkowanym – szukanie na półce butów o danym rozmiarze

W algorytmie wyszukiwania elementu w zbiorze nieuporządkowanym szukamy danego elementu i równocześnie jego położenia w tym zbiorze.

Ćwiczenie 7. Sprawdzamy działanie algorytmu wyszukiwania liczby w zbiorze nieuporządkowanym

Przygotuj odpowiednie pomoce dydaktyczne (rys. 7.) i przedstaw wspólnie z koleżankami i kolegami z klasy algorytm wyszukiwania liczby w zbiorze nieuporządkowanym.

Wskazówka: Możesz zapisać liczby na kartkach i układać je na stole lub podłodze.

2. Wyszukiwanie elementu w zbiorze uporządkowanym metodą połowienia

Szukamy na półce sklepowej butów o rozmiarze 37. Buty są uporządkowane według rozmiarów (od najmniejszego do największego), zaczynając od lewej strony. Jak sprawdzić, czy buty o rozmiarze 37 stoją na półce i na którym miejscu?



Wiemy, że buty zostały ułożone rosnąco według rozmiarów, zaczynając od lewej strony (rys. 10.). Szukamy butów o rozmiarze 37. Czy przeglądamy półkę od samego początku? Raczej nie. Zwykle odruchowo sprawdzamy, czy szukanych butów nie ma pośrodku. Jeśli trafimy na buty o innym rozmiarze, np. 39, to nie przeszukujemy prawej połowy półki, tylko będziemy szukać po lewej stronie – postępujemy zgodnie z **algorytmem wyszukiwania przez połowienie**.

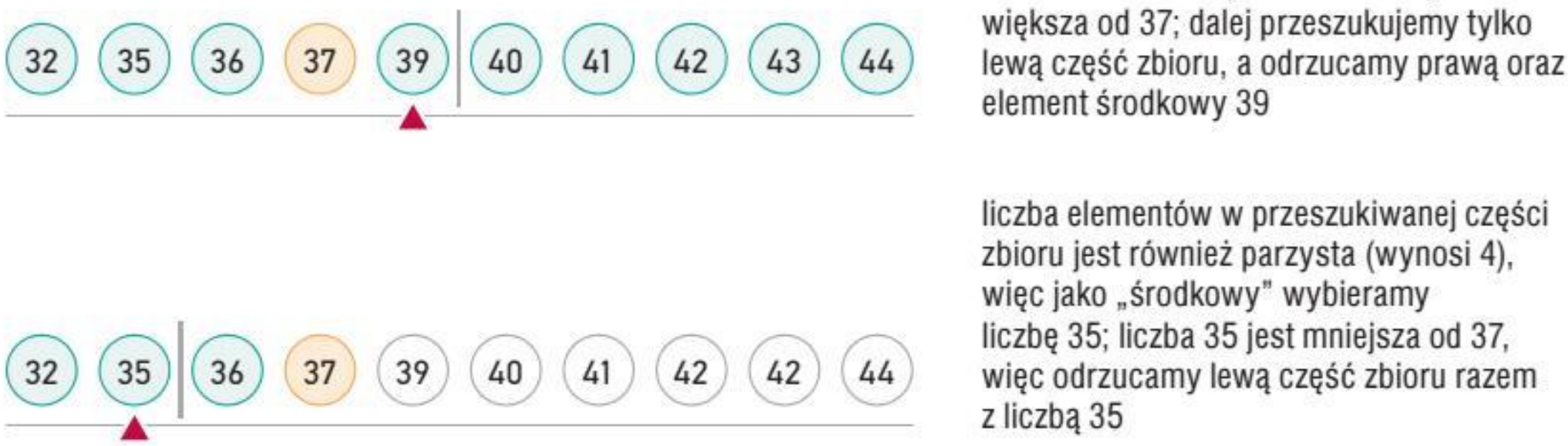
Algorytm wyszukiwania przez połowienie jest przykładem metody „dziel i zwyciężaj". Polega ona na dzieleniu przeszukiwanego zbioru na dwie części i zawężeniu przeszukiwania do jednej z tych części. Metoda podziału pomaga szybko znaleźć poszukiwany element, czyli zwyciężyć.

Opis algorytmu wyszukiwania przez połowienie (na przykładzie dziesięciu liczb)

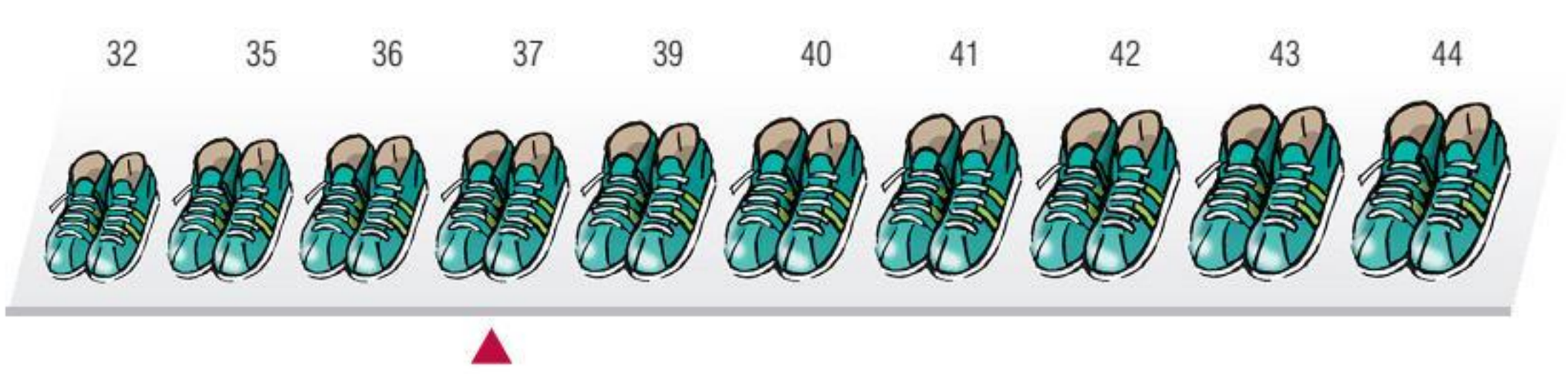
Dany jest uporządkowany w kolejności od najmniejszej do największej zbiór dziesięciu liczb {32, 35, 36, 37, 39, 40, 41, 42, 43, 44}. Poszukiwana liczba to 37. Jeżeli w zbiorze jest kilka takich samych wartości, algorytm znajdzie pozycję jednego z tych elementów (nie jest określone którego).

Dzielimy zbiór liczb na dwie części, rozdzielone środkowym elementem. Zaczynamy od sprawdzenia tego elementu (liczby zaznaczonej strzałką). W przypadku, gdy liczba elementów zbioru jest parzysta, spośród dwóch środkowych elementów (w naszym przypadku liczb 39 i 40) jako „środkowy" wybieramy ten, który jest pierwszy w kolejności (czyli 39).

Jeśli środkowy element jest szukaną liczbą, kończymy wyszukiwanie. Jeśli środkowy element jest większy od poszukiwanej liczby, przeszukujemy podzbiór na lewo od elementu środkowego. Jeśli środkowy element jest mniejszy od poszukiwanej liczby, przeszukujemy podzbiór na prawo od elementu środkowego.



Rys. 9. Przykład wyszukiwania liczby w zbiorze uporządkowanym metodą połowienia



Rys. 10. Zastosowanie algorytmu wyszukiwania elementu w zbiorze uporządkowanym metodą połowienia – szukanie na półce butów o danym rozmiarze

W algorytmie wyszukiwania elementu w zbiorze uporządkowanym szukamy danego elementu i równocześnie jego położenia w tym zbiorze.

Ćwiczenie 8. Sprawdzamy działanie algorytmu wyszukiwania liczby w zbiorze uporządkowanym metodą połowienia

Przygotuj odpowiednie pomoce dydaktyczne (rys. 9.) i wykonaj wspólnie z koleżankami i kolegami z klasy algorytm wyszukiwania liczby w zbiorze uporządkowanym metodą połowienia.

Wskazówka: Możesz zapisać liczby na kartkach i układać je na stole lub podłodze.

3. Stosowanie metody połowienia do gry w zgadywanie liczby

Algorytm wyszukiwania przez połowienie wykorzystuje się w zabawie w zgadywanie liczby. Zabawa polega na odgadywaniu liczby z określonego zakresu. Jedna osoba wybiera liczbę, a druga osoba ją odgaduje. Osoba zgadująca podaje liczbę, a druga osoba udziela odpowiedzi: „za mała" lub „za duża". W ten sposób osoba zgadująca dzieli zbiór liczb, odrzucając część, w której są liczby większe lub mniejsze, aż odgadnie właściwą liczbę.

Ćwiczenie 9. Gramy w zgadywanie liczby

Zagraj z kolegą lub z koleżanką z klasy w zgadywanie liczby. Przyjmijcie zbiór liczb naturalnych z zakresu od 1 do 100.

Algorytm gry w zgadywanie liczby

Zadanie: Odgadnij liczbę w zbiorze liczb naturalnych z zakresu od 1 do 100, stosując algorytm gry w zgadywanie liczby.
Dane: Szukana liczba pamiętana w zmiennej *szukana*, liczby naturalne z zakresu od 1 do 100 podawane przez gracza i zapamiętywane kolejno w zmiennej *liczba*.
Wynik: Odgadnięta liczba: *szukana*.
Lista kroków:

1. Zacznij algorytm.
2. Zmiennej *szukana* przypisz liczbę, którą ma odgadnąć gracz.
3. Wprowadź liczbę podawaną przez gracza do zmiennej *liczba*.
4. Jeżeli *liczba* = *szukana*, idź do kroku 7.
5. Jeżeli *liczba* > *szukana*, wyświetl na ekranie napis „Za duża"; w przeciwnym przypadku wyświetl na ekranie napis „Za mała".
6. Idź do kroku 3.
7. Wyprowadź komunikat „Gratulacje! Odgadnięta liczba to: " i wartość zmiennej *szukana*.
8. Zakończ algorytm.

Algorytm gry w zgadywanie liczby zaprogramujemy w środowiskach programowania Baltie i Scratch, a następnie w języku C++ i języku Python.

Na początku programu należy zmiennej *szukana* przypisać wartość, którą gracz ma odgadnąć, oraz wprowadzać pierwszą liczbę podawaną przez gracza (kolejne będą podawane w pętli).

W algorytmie powtarzamy polecenia, dopóki wartość zmiennej *liczba* jest różna od wartości zmiennej *szukana*, czyli dopóki nie odgadniemy liczby równej szukanej, dlatego w programie wykorzystamy odpowiednią instrukcję iteracyjną. W języku Scratch zastosujemy polecenie **powtarzaj aż** (rys. 11b), a w środowisku programowania Baltie (rys. 11a) oraz w językach C++ (rys. 12a) i Python (rys. 12b) – instrukcję `while`. Zastosujemy również instrukcję warunkową i użyjemy niezbędnych zmiennych.

Ćwiczenie 10. Tworzymy w środowisku programowania Baltie lub Scratch grę w zgadywanie liczby

1. Zaprogramuj grę w zgadywanie liczby zgodnie z podaną listą kroków. W programie zastosuj polecenie przypisania wartości zmiennej *szukana* wartości zmiennej *liczba* wprowadzanej z klawiatury po uruchomieniu programu. Objaśnij działanie instrukcji `while` (rys. 11a) lub **powtarzaj aż** (rys. 11b).
2. Zapisz program w pliku pod nazwą *Zgadnij liczbę*.
3. Uruchom grę i sprawdź jej działanie dla różnych danych (wartości zmiennej *szukana*).
4. Zagraj z kolegą lub koleżanką w zgadywanie liczby. Wpisuj szukaną liczbę tak, aby przeciwnik jej nie widział.

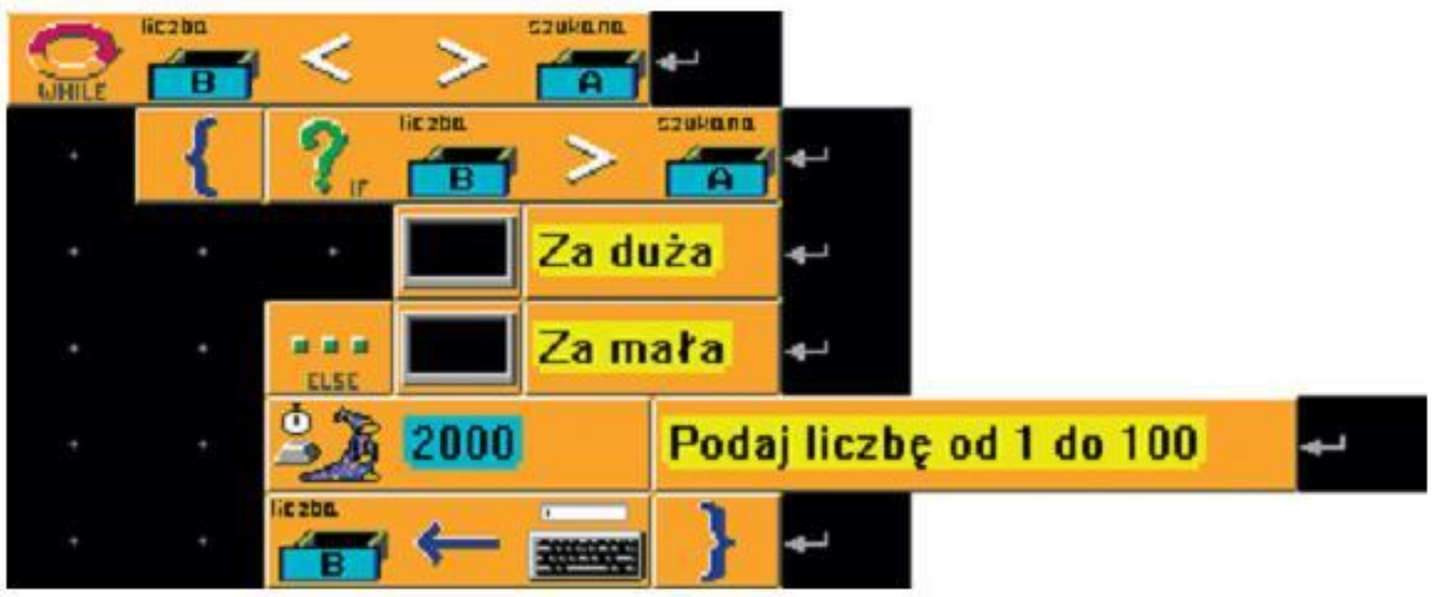
Wskazówki:

Scratch
- Nie pokazuj wartości zmiennej *szukana* na ekranie.
- Zmień tło i postać duszka (rys. 13.).
- Dodatkowo po wyświetleniu napisu „Gratulacje! Odgadnięta liczba to" i wartości zmiennej *szukana* duszek ma zagrać w koszykówkę. W tym celu należy powtarzać wielokrotnie zmianę kostiumu.

Ćwiczenie 11. Zapisujemy w języku C++ lub Python algorytm gry w zgadywanie liczby

1. Zaprogramuj grę w zgadywanie liczby zgodnie z podaną listą kroków. W programie zastosuj polecenie przypisania wartości zmiennej *szukana* wartości zmiennej *liczba* wprowadzanej z klawiatury po uruchomieniu programu. Objaśnij działanie instrukcji `while` (rys. 12a lub 12b).
2. Zapisz program w pliku pod nazwą *Zgadnij liczbę*.
3. Uruchom grę i sprawdź jej działanie dla różnych danych (wartości zmiennej *szukana*).
4. Zagraj z kolegą lub koleżanką w zgadywanie liczby. Wpisuj szukaną liczbę tak, aby przeciwnik jej nie widział.
5. Porównaj realizację algorytmu w programach utworzonych w środowisku Baltie i języku C++ lub w języku Scratch i języku Python (w tym działanie instrukcji iteracyjnej). Co zauważasz? Uzasadnij odpowiedź.

Baltie



Rys. 11a. Fragment programu realizującego algorytm gry w zgadywanie liczby (Baltie) – ćwiczenie 10.

Scratch



Rys. 11b. Fragment programu realizującego algorytm gry w zgadywanie liczby (Scratch) – ćwiczenie 10.

C++

```
while (liczba != szukana)
{
    if (liczba > szukana)
        cout << "Za duza" << endl;
    else
        cout << "Za mala" << endl;
    cout << "Podaj liczbe od 1 do 100: ";
    cin >> liczba;
}
```

Rys. 12a. Fragment programu realizującego algorytm gry w zgadywanie liczby (C++) – ćwiczenie 11.

Python

```
while liczba != szukana:
    if liczba > szukana:
        print("Za duża")
    else:
        print("Za mała")
    liczba = int(input("Podaj liczbę od 1 do 100: "))
```

Rys. 12b. Fragment programu realizującego algorytm gry w zgadywanie liczby (Python) – ćwiczenie 11.

Scratch



Rys. 13. Przykładowy ekran programu realizującego algorytm gry w zgadywanie liczby (Scratch) – ćwiczenia 10. i 12.

Gra będzie ciekawsza, jeśli gracze nie będą znali wcześniej szukanej liczby. Prawie w każdym języku programowania występuje polecenie umożliwiające generowanie liczb losowych.

Ćwiczenie 12. Modyfikujemy grę w zgadywanie liczby w środowisku programowania Baltie lub Scratch

1. Otwórz program *Zgadnij liczbę* zapisany w ćwiczeniu 10.
2. Zmodyfikuj grę, tak aby komputer wybierał losowo szukaną liczbę z przedziału od 1 do 100. Dodaj zliczanie wykonanych prób: utwórz nową zmienną *licznik*. Na początku programu ustaw jej wartość na 1, a w pętli zwiększaj o 1. Po napisie „Gratulacje! Odgadnięta liczba to" i wartości zmiennej *szukana* wyświetl napis „Liczba wykonanych prób to" i wartość zmiennej *licznik*.
3. Zapisz program w pliku pod nazwą *Zgadnij liczbę_los*.

Wskazówki:

Baltie Aby w środowisku programowania Baltie wygenerować liczby losowe z danego przedziału, należy zastosować element **Liczba losowa** .

Na przykład  generuje liczby całkowite z przedziału ⟨0, 100), czyli liczby: 0, 1, 2, …, 99.

W poleceniu  zmiennej *szukana* przypisujemy wartość losową od 1 do 100.

Scratch Aby w języku Scratch wygenerować liczby losowe z danego przedziału, należy zastosować element **losuj liczbę** z grupy **Wyrażenia**.

Na przykład  wygeneruje losowo liczbę naturalną z przedziału ⟨1, 100⟩.

W poleceniu  zmiennej *szukana* przypisujemy wartość losową od 1 do 100.

Ćwiczenie 13. Modyfikujemy grę w zgadywanie liczby w języku C++ lub Python

1. Otwórz program *Zgadnij liczbę* zapisany w ćwiczeniu 11.
2. Zmodyfikuj grę, tak aby komputer wybierał losowo szukaną liczbę z przedziału od 1 do 100.
3. Zapisz program w pliku pod tą samą nazwą.

Wskazówki:

C++ W języku C++ liczby losowe generuje funkcja `rand()`.
Na przykład `rand() % 100 + 1` – wygeneruje losowo liczbę naturalną z przedziału ⟨1, 100⟩.
Na początku programu musimy dołączyć bibliotekę, w której zdefiniowano funkcję `rand()`.

```
#include <stdlib.h>
...
szukana = rand() % 100 + 1;
```

Python W języku Python liczby losowe generuje funkcja `randint()`.
Na przykład `randint(1, 100)` – wygeneruje losowo liczbę naturalną z przedziału ⟨1, 100⟩.
Na początku programu musimy dołączyć bibliotekę, w której zdefiniowano funkcję `randint()`.

```
from random import randint
szukana = randint(1, 100)
```

4. Porządkowanie elementów zbioru metodą przez wybieranie

Czasem porządkujemy (w terminologii informatycznej – **sortujemy**) zbiory przedmiotów, np. ustawiamy książki na półce – od największej do najmniejszej lub alfabetycznie według nazwisk autorów, ustalamy kolejność wyświetlania plików według nazw lub dat utworzenia. Na danych uporządkowanych (posortowanych) szybciej wykonuje się różne operacje (np. wyszukiwanie).

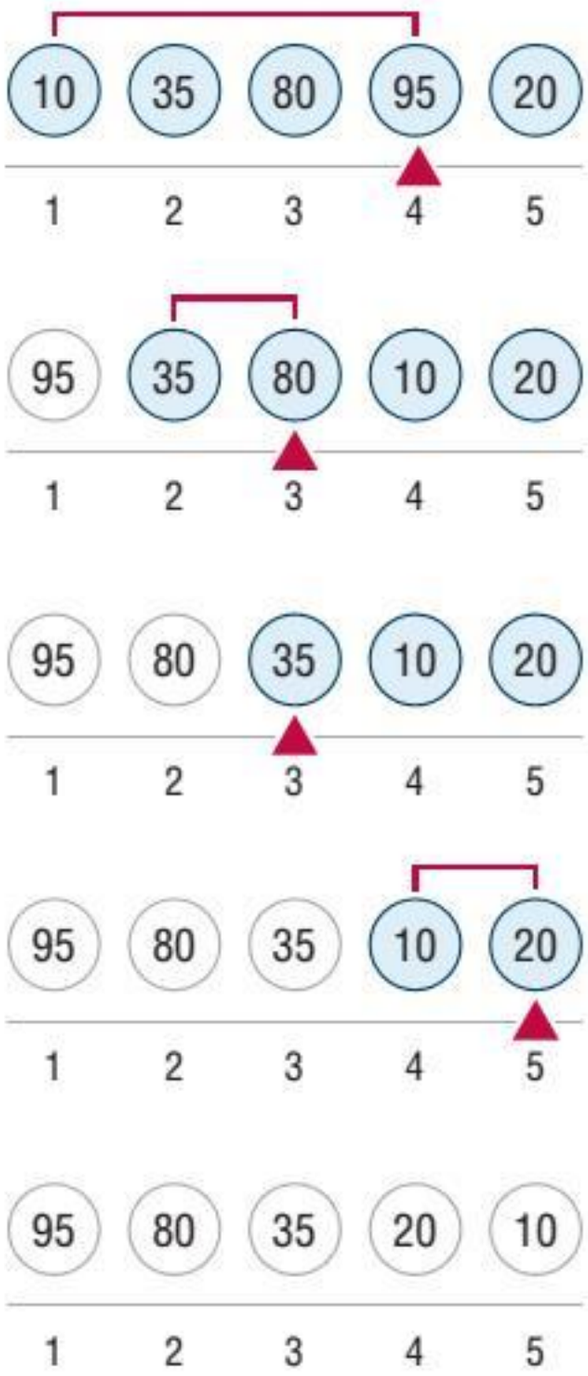
{ Chcemy uporządkować wszystkich uczniów od najwyższego do najniższego.
Czy można do tego celu wykorzystać algorytm wyszukiwania najwyższego ucznia? }

Istnieją różne algorytmy wyszukiwania i porządkowania. Do porządkowania elementów często stosujemy metodę **porządkowania przez wybieranie**.

Aby uporządkować elementy od największego do najmniejszego, stosujemy algorytm wyboru elementu największego. Jeśli chcemy uporządkować elementy od najmniejszego do największego, stosujemy algorytm wyboru elementu najmniejszego.



Porządkowanie zbioru liczb metodą przez wybieranie polega na wyszukiwaniu w nim liczby największej (dla porządku malejącego) lub najmniejszej (dla porządku rosnącego), zamienieniu jej miejscami z liczbą, która jest aktualnie na początku zbioru, a następnie powtarzaniu tych czynności z pominięciem liczb już uporządkowanych (rys. 14.).



liczba 95 jest największa, dlatego zamieniamy ją miejscami z pierwszą liczbą w zbiorze, czyli liczbą 10

liczba 95 jest już umieszczona na właściwym miejscu; teraz szukamy największej spośród pozostałych czterech liczb; jest nią liczba 80, którą zamieniamy miejscami z drugą liczbą w zbiorze, czyli liczbą 35

liczba 35 jest większa od 10 i 20, dlatego nie zmieniamy jej miejsca w zbiorze

na koniec zamieniamy miejscami liczby 10 i 20, ponieważ 20 jest większe od 10

wszystkie liczby zostały uporządkowane

Rys. 14. Przykład porządkowania liczb metodą przez wybieranie – od największej do najmniejszej

Algorytm porządkowania metodą przez wybieranie wykorzystamy do ustawienia uczniów od najwyższego do najniższego.

Algorytm porządkowania metodą przez wybieranie

Zadanie: Uporządkuj dziesięciu uczniów według wzrostu (od ucznia najwyższego do najniższego), stosując sortowanie przez wybieranie.
Dane: Dziesięciu uczniów (ustawionych w przypadkowej kolejności).
Wynik: Dziesięciu uczniów ustawionych w kolejności od najwyższego do najniższego.
Lista kroków:

1. Zacznij algorytm.
2. Znajdź najwyższego ucznia. Zapamiętaj, w którym miejscu stoi.
3. Zamień miejscami ucznia najwyższego z uczniem stojącym na pierwszym miejscu.
4. Powtarzaj kroki 2. i 3. (z pominięciem uczniów już ustawionych na miejscach początkowych), aż uporządkujesz wszystkich uczniów.
5. Zakończ algorytm.

Ćwiczenie 14. Porządkujemy uczniów według wzrostu metodą przez wybieranie

Wybierz z klasy dziesięciu uczniów. Ustaw wybranych uczniów według wzrostu (od najwyższego do najniższego). Postępuj zgodnie z podaną listą kroków.

Ćwiczenie 15. Porządkujemy książki metodą przez wybieranie

Napisz listę kroków algorytmu porządkowania książek od najgrubszej do najcieńszej.

5. Porządkowanie elementów zbioru metodą przez zliczanie

Algorytm porządkowania przez zliczanie pokażemy na przykładzie porządkowania liczb całkowitych. Przed rozpoczęciem porządkowania musimy znać najmniejszą i największą wartość w zbiorze.



Porządkowanie metodą przez zliczanie polega na zliczaniu wystąpień poszczególnych wartości w zbiorze, np. ile mamy jedynek, dwójek, trójek, czwórek, piątek, …, dwudziestek – w przypadku zbioru liczb zawierających wartości od 1 do 20. Następnie wypisujemy daną liczbę tyle razy, ile wynosi liczba jej wystąpień w zbiorze. W ten sposób otrzymujemy uporządkowany zbiór.

Opis algorytmu porządkowania elementów zbioru metodą przez zliczanie (na przykładzie siedmiu liczb)

Dany jest zbiór liczb {5, 6, 5, 2, 3, 5, 3}, który zawiera liczby całkowite z przedziału od 2 do 6 (czyli najmniejszą liczbą jest liczba 2, a największą 6).

Do zliczania wystąpień wszystkich liczb z tego zakresu musimy przygotować pięć liczników do zliczania wystąpień liczb: 2, 3, 4, 5 i 6. Na rysunku 15. liczniki mają formę pudełek. Na początku wszystkie pudełka liczników są puste (mają wartość 0), ale gdy zaczniemy przeglądać zbiór, wartość liczników będzie się zmieniać.

Liczniki przygotowujemy dla wszystkich liczb z przedziału od 2 do 6, nawet dla liczb z tego przedziału, których nie ma w naszym zbiorze. Algorytm porządkowania liczb metodą przez zliczanie zaprogramujemy w tematach 8. (C++) i 9. (Python).



Ćwiczenie 16. Porządkujemy liczby od najmniejszej do największej metodą przez zliczanie

Przygotuj odpowiednie pomoce dydaktyczne (pudełka lub kubeczki i patyczki lub ołówki, a następnie wykonaj wspólnie z koleżankami i kolegami z klasy algorytm porządkowania liczb metodą przez zliczanie dla zbioru składającego się z dziesięciu elementów:

{–2, 5, 2, 0, 4, –2, 4, –1, –2, 5}.

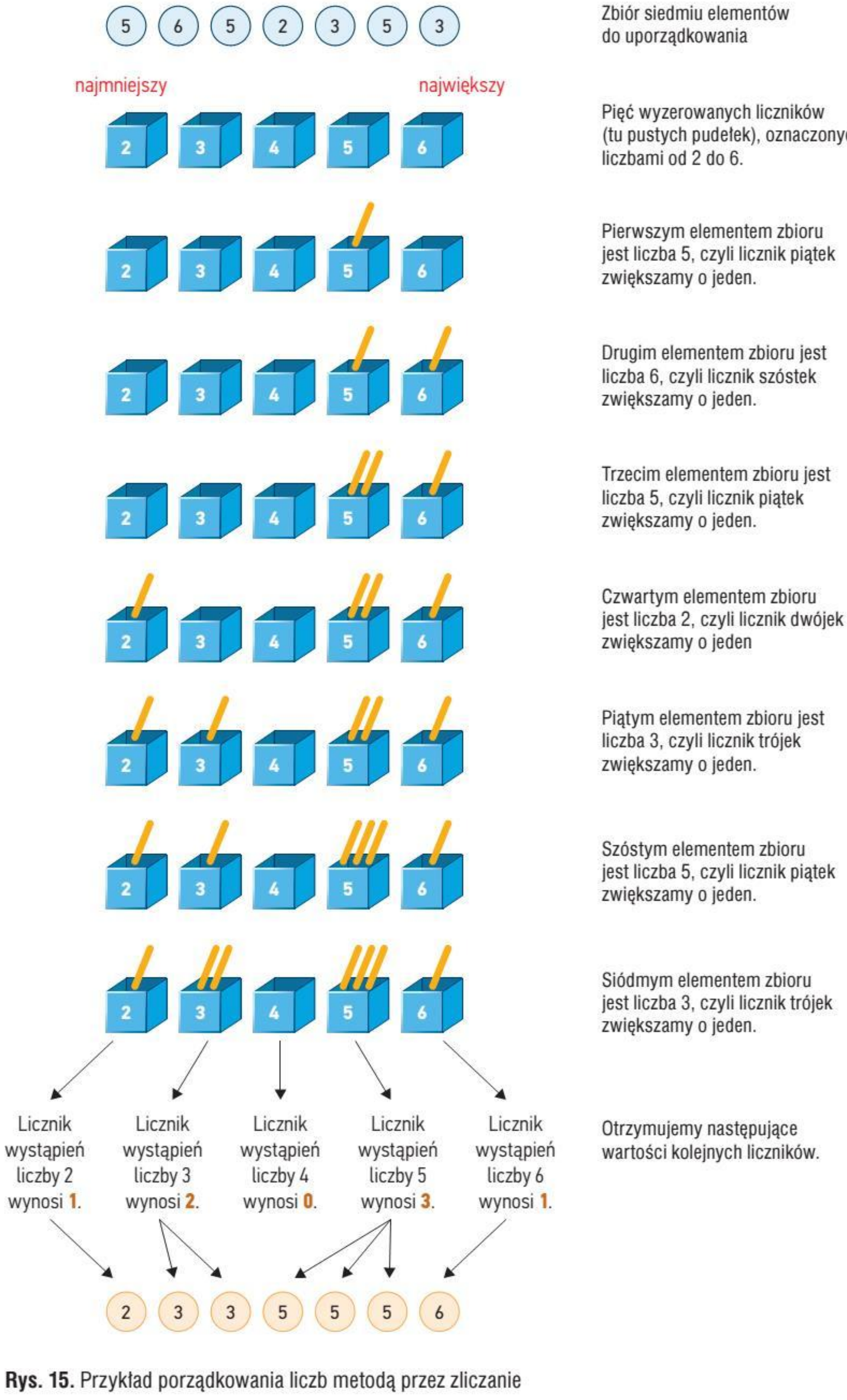
Liczby uporządkuj od najmniejszej do największej.

Wskazówki:

- Pudełek (kubeczków) ma być tyle, ile jest liczb całkowitych z zakresu ⟨*najmniejszy*, *największy*⟩, gdzie najmniejszy to najmniejsza liczba w zbiorze, a *największy* – *największa*.
- Zbiór zaczynamy przeglądać od pierwszego elementu i kolejno wrzucamy patyczki (ołówki) do kubeczków oznaczonych daną liczbą.



Aby zbiór liczb był uporządkowany od największej do najmniejszej, wystarczy zbiór liczników przejrzeć odwrotnie – od prawej strony do lewej.



Zbiór siedmiu elementów do uporządkowania

Pięć wyzerowanych liczników (tu pustych pudełek), oznaczonych liczbami od 2 do 6.

Pierwszym elementem zbioru jest liczba 5, czyli licznik piątek zwiększamy o jeden.

Drugim elementem zbioru jest liczba 6, czyli licznik szóstek zwiększamy o jeden.

Trzecim elementem zbioru jest liczba 5, czyli licznik piątek zwiększamy o jeden.

Czwartym elementem zbioru jest liczba 2, czyli licznik dwójek zwiększamy o jeden

Piątym elementem zbioru jest liczba 3, czyli licznik trójek zwiększamy o jeden.

Szóstym elementem zbioru jest liczba 5, czyli licznik piątek zwiększamy o jeden.

Siódmym elementem zbioru jest liczba 3, czyli licznik trójek zwiększamy o jeden.

Otrzymujemy następujące wartości kolejnych liczników.

Rys. 15. Przykład porządkowania liczb metodą przez zliczanie

Warto zapamiętać

- Wybrany element w zbiorze nieuporządkowanym możemy znaleźć, korzystając z algorytmu przeszukiwania liniowego.
- Wybrany element w zbiorze uporządkowanym możemy znaleźć, korzystając z algorytmu wyszukiwania przez połowienie, który jest przykładem realizacji metody **dziel i zwyciężaj**.
- Zbiór elementów możemy uporządkować, stosując algorytmy porządkowania, np. metodą przez wybieranie lub metodą przez zliczanie.
- W porządkowaniu metodą przez wybieranie wykorzystujemy algorytm wyszukiwania elementu (najmniejszego lub największego) w zbiorze nieuporządkowanym.

Pytania i polecenia

1. Wyjaśnij na przykładzie, w jaki sposób znajduje się największą spośród siedmiu liczb.
2. Dlaczego w programach na rysunkach 5a i 5b oraz na rysunkach 6a i 6b wprowadzaną wartość przypisujemy najpierw zmiennej *maks*?
3. Dlaczego w programie wyszukującym liczbę największą stosujemy instrukcję iteracyjną i warunkową?
4. Na czym polega algorytm wyszukiwania przez połowienie?
5. Jakie może być zastosowanie algorytmu wyszukiwania przez połowienie? Pokaż na przykładzie.
6. Dysponujemy listą dwustu uczestników maratonu. Lista jest uporządkowana według numerów startowych, ale numery nie są kolejnymi liczbami naturalnymi. Jak szybko znaleźć na tej liście nazwisko zawodnika o konkretnym numerze?
7. Jaką liczbę najlepiej podać jako pierwszą w grze w zgadywanie liczby w przypadku zbioru liczb od 1 do 100? Uzasadnij odpowiedź.
8. Na czym polega algorytm porządkowania metodą przez wybieranie?
9. Dlaczego do ustawienia uczniów według wzrostu zastosowaliśmy metodę porządkowania przez wybieranie? Uzasadnij odpowiedź.
10. W jaki sposób porządkuje się liczby metodą przez zliczanie? Wyjaśnij na przykładzie.
11. Dlaczego w porządkowaniu metodą przez zliczanie należy przygotować liczniki dla wszystkich liczb z danego zakresu, nawet jeśli niektóre liczniki pozostaną zerowe? Uzasadnij odpowiedź.
12. Wskaż przynajmniej dwie różnice między porządkowaniem metodą przez wybieranie a metodą przez zliczanie.

Zadania

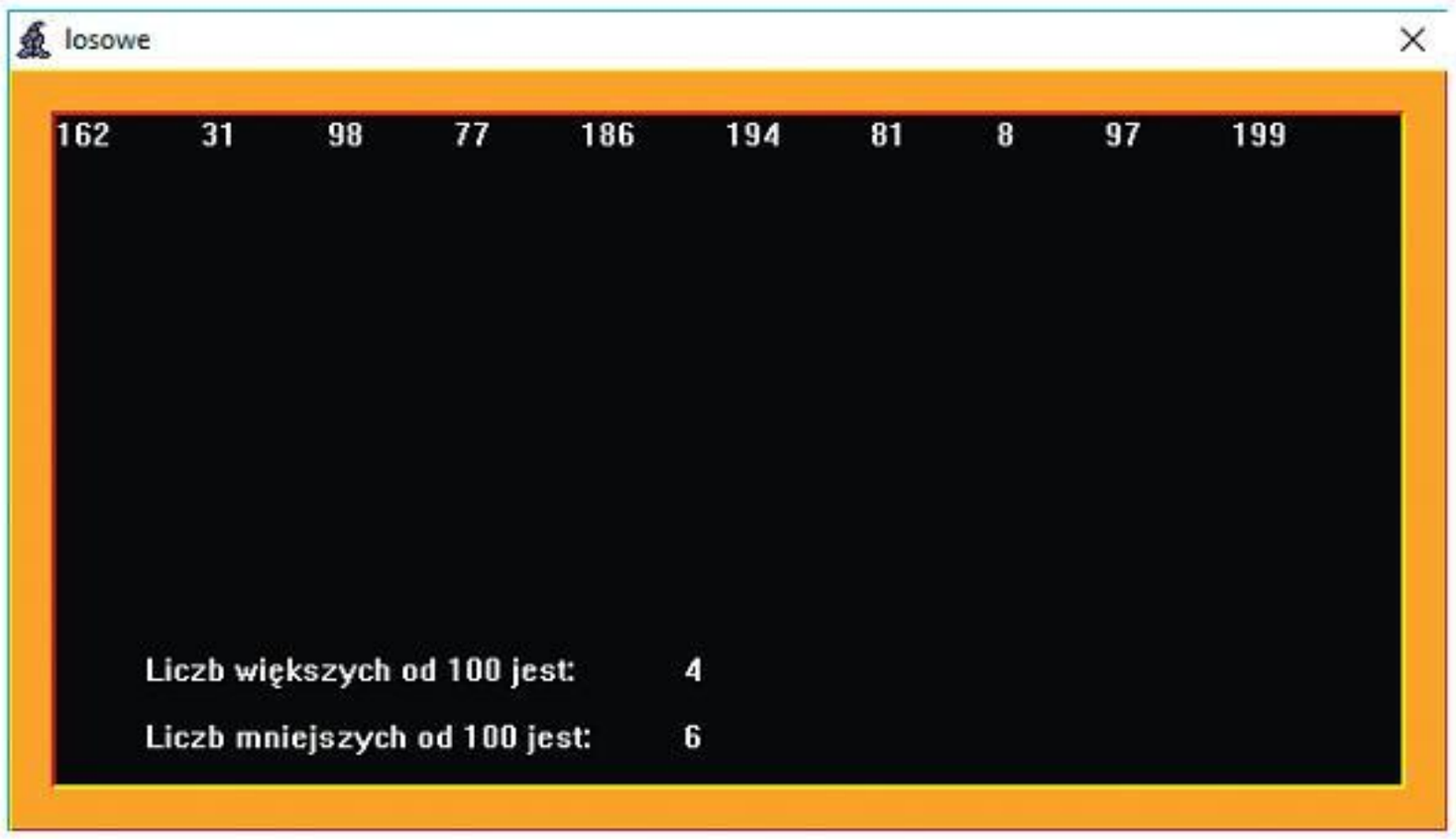
Uwagi:

- Listy kroków możesz zapisać w zeszycie przedmiotowym lub w edytorze tekstu.
- Schematy blokowe możesz rysować w zeszycie przedmiotowym, w edytorze tekstu z wykorzystaniem **Kształtów**, w edytorze grafiki **Paint** lub w wybranym darmowym programie pobranym z Internetu.

1. Narysuj schemat blokowy algorytmu znajdowania większej z dwóch liczb.
2. Zmodyfikuj program zapisany w ćwiczeniu 3., uwzględniając przypadek, gdy wprowadzone z klawiatury liczby są równe. Zapisz program w pliku pod tą samą nazwą.

3. Utwórz program do znajdowania największej z trzech liczb. Zapisz program w pliku pod nazwą *Największa z trzech*.
   Wskazówka: Pamiętaj, aby po sprawdzeniu, czy pierwsza liczba jest większa od drugiej, sprawdzić, czy wybrana z pierwszego porównania liczba jest większa od trzeciej.
4. Na podstawie listy kroków pokazanej w punkcie 1. tego tematu napisz listę kroków algorytmu znajdowania najmniejszej liczby spośród dziesięciu liczb. Sprawdź działanie algorytmu dla danych: 5, 78, 15, 0, 1, 100, 155, 50, 22, 30.
5. Zmodyfikuj program *Maksimum_n* zapisany w ćwiczeniu 5. lub 6. tak, aby wyszukiwał najmniejszy element ze zbioru *n*-elementowego. Zapisz program w pliku pod nazwą *Minimum_n*.
6. Dany jest uporządkowany zbiór liczb: 5, 8, 15, 20, 25, 30, 35, 70, 88, 90, 100, 120. Poszukiwana liczba to 70. Przedstaw na rysunku (podobnym do rys. 9.) sposób znalezienia liczby 70.
7. Dany jest zbiór liczb: 12, 3, 8, 2, 12, 5, 8, 4, 10, 4. Przedstaw na rysunku (podobnym do rys. 15.) uporządkowanie tego zbioru metodą przez zliczanie. Ile liczników musisz przygotować? Ile liczników pozostało pustych?
8. Utwórz w środowisku programowania Baltie lub Scratch program, który będzie generował losowo dziesięć liczb z zakresu 1-200 i zliczał oddzielnie liczby większe od 100 i mniejsze 100. Zastosuj trzy zmienne: *liczba*, *licznik1* i *licznik2*. Liczby powinny być wyświetlane na ekranie w trakcie ich generowania. Na koniec powinny zostać wyświetlone wylosowane liczby, odpowiednie komunikaty i wyniki (rys. 16a i 16b). Zapisz program w pliku pod nazwą *Losowe*.

Baltie



Rys. 16a. Ekran z danymi i wynikami (Baltie) – zadanie 8.

Scratch

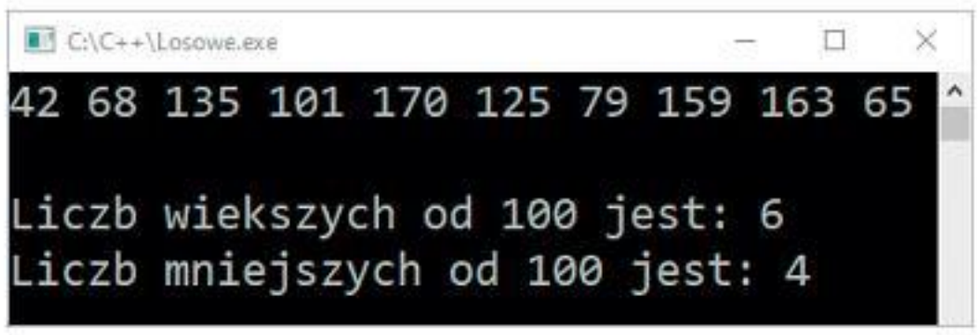


Rys. 16b. Ekran wyświetlania danych i wyników (Scratch) – zadanie 8.

Dla zainteresowanych

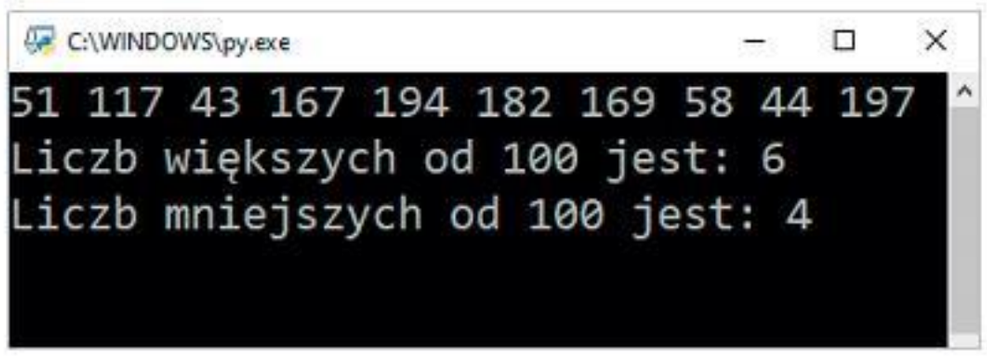
9. Napisz listę kroków i zbuduj schemat blokowy algorytmu znajdowania najniższego z trzech uczniów.
10. Napisz listę kroków algorytmu wyszukiwania przez połowienie. Skorzystaj z opisu metody podanego w punkcie 2. tego tematu.
11. Napisz w języku programowania C++ lub Python program, który będzie generował losowo dziesięć liczb z zakresu 1-200 i zliczał oddzielnie liczby większe od 100 i mniejsze od 100. Zastosuj trzy zmienne: *liczba*, *licznik1* i *licznik2*. Liczby powinny być wyświetlane na ekranie w trakcie ich generowania. Na koniec powinny zostać wyświetlone odpowiednie komunikaty i wyniki. Zapisz program w pliku pod nazwą *Losowe*.

C++



Rys. 17a. Ekran z danymi i wynikami (C++) – zadanie 11.

Python



Rys. 17b. Ekran z danymi i wynikami (Python) – zadanie 11.

Temat 8. Wykorzystanie funkcji i tablic do zapisywania w języku C++ algorytmów porządkowania i wyszukiwania

1. **Stosowanie podprogramów w środowiskach programowania Baltie i Scratch – przypomnienie**
2. **Definiowanie i stosowanie funkcji w języku C++**
3. **Stosowanie tablic w języku C++ do przechowywania danych**
4. **Zapisywanie w języku C++ wybranych algorytmów wyszukiwania i porządkowania**



Warto powtórzyć

1. Jakie są etapy tworzenia programu w języku C++?
2. W jaki sposób deklarujemy zmienne w języku C++? Jak nadajemy im wartości?
3. W jaki sposób wyprowadzamy napisy na ekran monitora w języku C++?
4. Za pomocą jakiej instrukcji realizujemy sytuację warunkową w języku C++?
5. W jaki sposób zapisujemy iterację w języku C++?
6. Do czego służy zmienna sterująca używana w instrukcji `for`?

1. Stosowanie podprogramów w środowiskach programowania Baltie i Scratch – przypomnienie

Podprogramy (procedury i funkcje) służą do opracowywania oddzielnie problemów cząstkowych, które są częściami zadania rozwiązywanego przez dany program. Podprogram realizujący określone zadanie można wykorzystywać w programie wielokrotnie. Można też wykorzystać go w innych programach. Stosowanie podprogramów zwiększa przejrzystość programu.

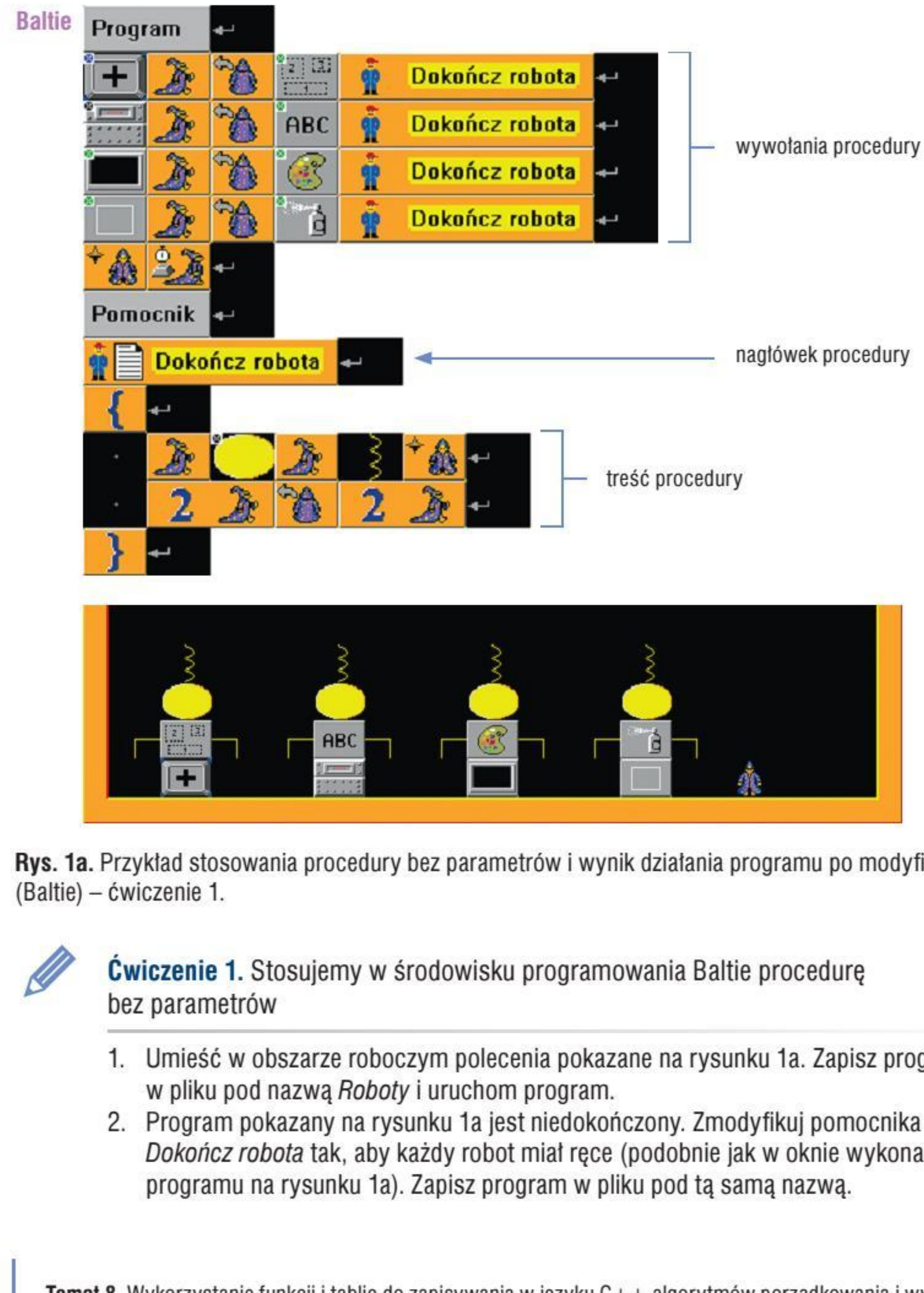
Aby zastosować podprogramy (procedury i funkcje), trzeba je **zdefiniować**, a następnie **wywołać** w programie.

W każdym z poznanych środowisk programowania (Baltie, Scratch) można definiować zarówno procedury z parametrami jak i bez nich. Definiowanie i wywoływanie procedur bez parametrów pokażemy w środowisku programowania Baltie, a z parametrami – w środowisku Scratch. W środowisku Baltie procedury noszą nazwę **pomocników**, a w środowisku Scratch – **bloków**.

Aby zdefiniować procedurę (pomocnika) w środowisku Baltie, należy:

- kliknąć element **Tablica pomocników** na pasku narzędzi okna programu Baltie,
- w oknie **Pomocnicy** kliknąć **Nowego pomocnika** (pomocnik pojawi się w obszarze roboczym pod utworzonymi poleceniami),
- wprowadzić polecenia, które ma wykonywać pomocnik.

Aby wywołać procedurę (pomocnika) w środowisku Baltie, należy kliknąć zdefiniowanego pomocnika widocznego w **Tablicy pomocników** i umieścić go w odpowiednim miejscu programu (rys. 1a).



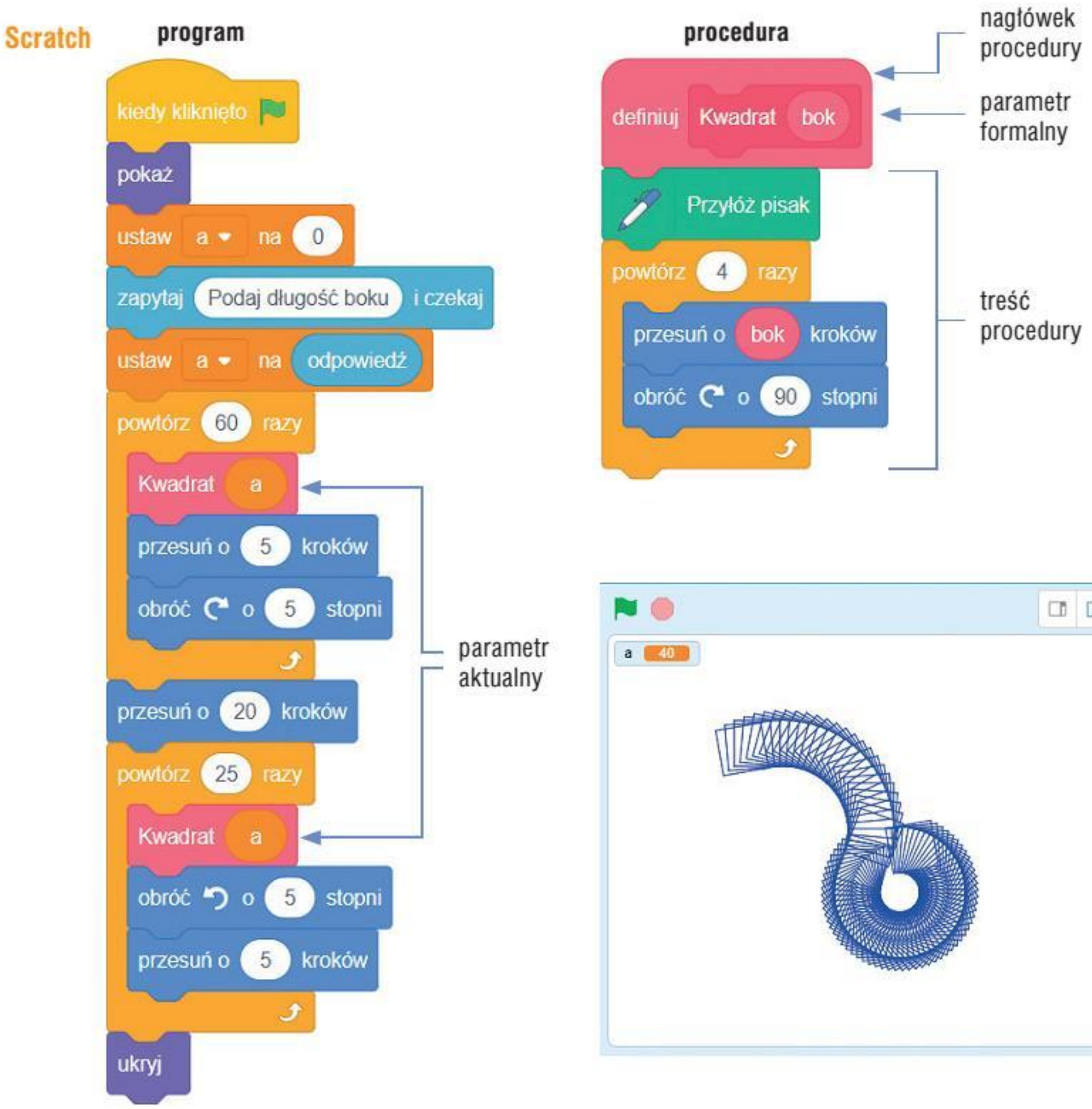
Rys. 1a. Przykład stosowania procedury bez parametrów i wynik działania programu po modyfikacji (Baltie) – ćwiczenie 1.

Ćwiczenie 1. Stosujemy w środowisku programowania Baltie procedurę bez parametrów

1. Umieść w obszarze roboczym polecenia pokazane na rysunku 1a. Zapisz program w pliku pod nazwą *Roboty* i uruchom program.
2. Program pokazany na rysunku 1a jest niedokończony. Zmodyfikuj pomocnika *Dokończ robota* tak, aby każdy robot miał ręce (podobnie jak w oknie wykonania programu na rysunku 1a). Zapisz program w pliku pod tą samą nazwą.

W języku Scratch definiowaliśmy własne procedury rysujące figury geometryczne (np. kwadrat, sześciokąt). Zdefiniowane procedury wykorzystywaliśmy potem do tworzenia ciekawych kompozycji graficznych. W programie pokazanym na rysunku 1b problemem cząstkowym jest narysowanie kwadratu o dowolnym boku. Rozwiązanie problemu zapisujemy w postaci procedury *Kwadrat* z **parametrem formalnym** *bok*, określającym długość boku kwadratu. Następnie dwukrotnie wywołujemy procedurę *Kwadrat* w programie głównym, umieszczając element z jej nazwą i **parametrem aktualnym** *a* wewnątrz poleceń **powtórz**. Wartość zmiennej *a* wprowadzamy z klawiatury.

W przypadku procedury z parametrami program główny komunikuje się z procedurą lub funkcją poprzez **przekazywanie parametrów**.



Rys. 1b. Przykład stosowania procedury z parametrami i wynik działania programu (Scratch) – ćwiczenie 2.

Aby w języku Scratch zdefiniować własną procedurę (blok), należy z grupy **Moje bloki** wybrać element **Utwórz blok**. W otwartym oknie procedurę możemy nazwać i dodać do niej parametry.

Ćwiczenie 2. Stosujemy w języku Scratch procedurę z parametrami

1. Umieść w obszarze roboczym polecenia pokazane na rysunku 1b. Zapisz program w pliku pod nazwą *Kwadraty* i uruchom program dla różnych wartości parametru *a*.
2. Rozbuduj program (i tym samym tworzoną kompozycję) poprzez dodanie dwóch kolejnych wywołań procedury *Kwadrat*. Zapisz program w pliku pod tą samą nazwą.

2. Definiowanie i stosowanie funkcji w języku C++

W języku C++ wszystkie podprogramy nazywamy **funkcjami**. Dzielimy je na dwa rodzaje: funkcje niezwracające wartości (rys. 2a) i funkcje zwracające wartość (rys. 2b).

W definicji funkcji określamy **parametry formalne** (zwane też **parametrami**), które w momencie wywołania funkcji komputer zastąpi **parametrami aktualnymi** (zwanymi też **argumentami**).

C++

```
void nazwa_funkcji(lista_parametrów_formalnych)
{
   lista_instrukcji;
}
```

Rys. 2a. Ogólna postać definicji funkcji niezwracającej wartości (C++)

C++

```
opis_typu nazwa_funkcji(lista_parametrów_formalnych)
{
   lista_instrukcji;
   return wartość;
}
```

Rys. 2b. Ogólna postać definicji funkcji zwracającej wartość (C++)

Funkcję niezwracającą wartości stosujemy, gdy mamy wykonać pewne czynności, np. wprowadzić dane, wyprowadzić dane lub obliczyć wiele wartości. Funkcja taka nie ma typu, dlatego w części nagłówkowej przed nazwą funkcji należy umieścić słowo kluczowe **void**. Ponieważ funkcja nie zwraca wartości, instrukcja **return** nie musi wystąpić (a jeśli występuje, to nie może określać *wartości*). Na rysunku 3. pokazano przykład funkcji wyświetlającej na ekranie wprowadzoną liczbę.

Ćwiczenie 3. Definiujemy w języku C++ funkcję niezwracającą wartości

1. Przepisz program pokazany na rysunku 3.
2. Zapisz program w pliku pod nazwą *Liczba*. Skompiluj i uruchom program.
3. Dodaj w funkcji polecenie wyświetlania napisu „Wprowadzona liczba to: ". Zapisz program w pliku pod tą samą nazwą. Skompiluj go i uruchom.

Wskazówka: Aby przejść do nowego wiersza, zamiast sekwencji znaków \n można użyć słowa kluczowego `endl`, które należy poprzedzić operatorem << (rys. 3.).

C++

Liczba.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  void WypiszLiczbe(int a)
5  {
6      cout << a << endl;
7  }
8
9  int main()
10 {
11     int liczba;
12     cout << "liczba = ";
13     cin >> liczba;
14     WypiszLiczbe(liczba);
15     return 0;
16 }
17
```

nagłówek funkcji

treść funkcji

wywołanie funkcji

informacja o braku typu funkcji i nazwa funkcji

typ i nazwa parametru formalnego (parametr)

parametr aktualny (argument)

Rys. 3. Przykład definicji funkcji niezwracającej wartości z jednym parametrem (C++) – ćwiczenie 3.

W temacie 7. napisaliśmy program realizujący algorytm wyszukiwania największego elementu w zbiorze *n*-elementowym. Chcemy zmodyfikować program, tak aby otrzymać funkcję realizującą ten algorytm. Funkcja będzie potrzebna do zapisania algorytmu porządkowania metodą przez wybieranie. Jak to zrobić w języku C++?

Należy w tym celu zdefiniować funkcję zwracającą wartość z jednym parametrem. Funkcję zwracającą wartość stosujemy, gdy celem funkcji jest obliczenie i zwrócenie pewnej wartości do funkcji wywołującej. Sposób definiowania takiej funkcji pokażemy na przykładzie funkcji z jednym parametrem wyszukującej największy element w zbiorze *n*-elementowym. Parametrem przekazywanym funkcji jest liczba elementów zbioru.

C++

Max_n_funkcja.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int Maksimum(int n)
5  {
6      int i, a, maks;
7
8      for(i=0; i<n; i++)
9      {
10         cout << "Podaj liczbe: ";
11         cin >> a;
12         if(i==0)
13             maks = a;
14         else
15             if(a > maks)
16                 maks = a;
17     }
18     return maks;
19 }
20
```

nagłówek funkcji

treść funkcji

typ zwracanej wartości (tu: `int`)

typ i nazwa parametru formalnego

funkcja zwraca wartość największego elementu zapisaną w zmiennej *maks*

Rys. 4. Przykład definicji funkcji zwracającej wartość z jednym parametrem (C++) – ćwiczenie 4.

Treść funkcji powinna się znaleźć w bloku `{}`. Funkcja zwracająca wartość musi zawierać instrukcję **`return`** z wartością zwracaną do funkcji wywołującej.

Ćwiczenie 4. Definiujemy w języku C++ funkcję zwracającą wartość

1. Zdefiniuj funkcję *Maksimum* z parametrem *n* (gdzie *n* to liczba elementów w zbiorze) wyszukującą największy element w zbiorze *n*-elementowym, gdzie $n > 0$ (rys. 4.).
2. Zapisz program w pliku pod nazwą *Maksimum_n_funkcja*.

C++

```
20
21  int main()
22  {
23      int liczba_elementow, najwiekszy;
24
25      cout << "Podaj liczbe elementow: ";
26      cin >> liczba_elementow;
27      najwiekszy = Maksimum(liczba_elementow);
28      cout << "Maksimum to: " << najwiekszy;
29
30      return 0;
31  }
```

deklaracja zmiennych

wprowadzenie liczby elementów zbioru

wywołanie funkcji *Maksimum* z parametrem aktualnym *liczba_elementow* – wynik przypisujemy zmiennej *najwiekszy*

Rys. 5. Wywołanie funkcji Maksimum w funkcji głównej `main()` – ćwiczenie 5.

Wywołanie funkcji *Maksimum* powoduje wykonanie instrukcji będących treścią funkcji oraz zwrócenie wyniku jej działania. Przypiszemy go zmiennej *najwiekszy* (rys. 5.): `najwiekszy = Maksimum(liczba_elementow);`

Ćwiczenie 5. Wywołujemy funkcję w programie głównym

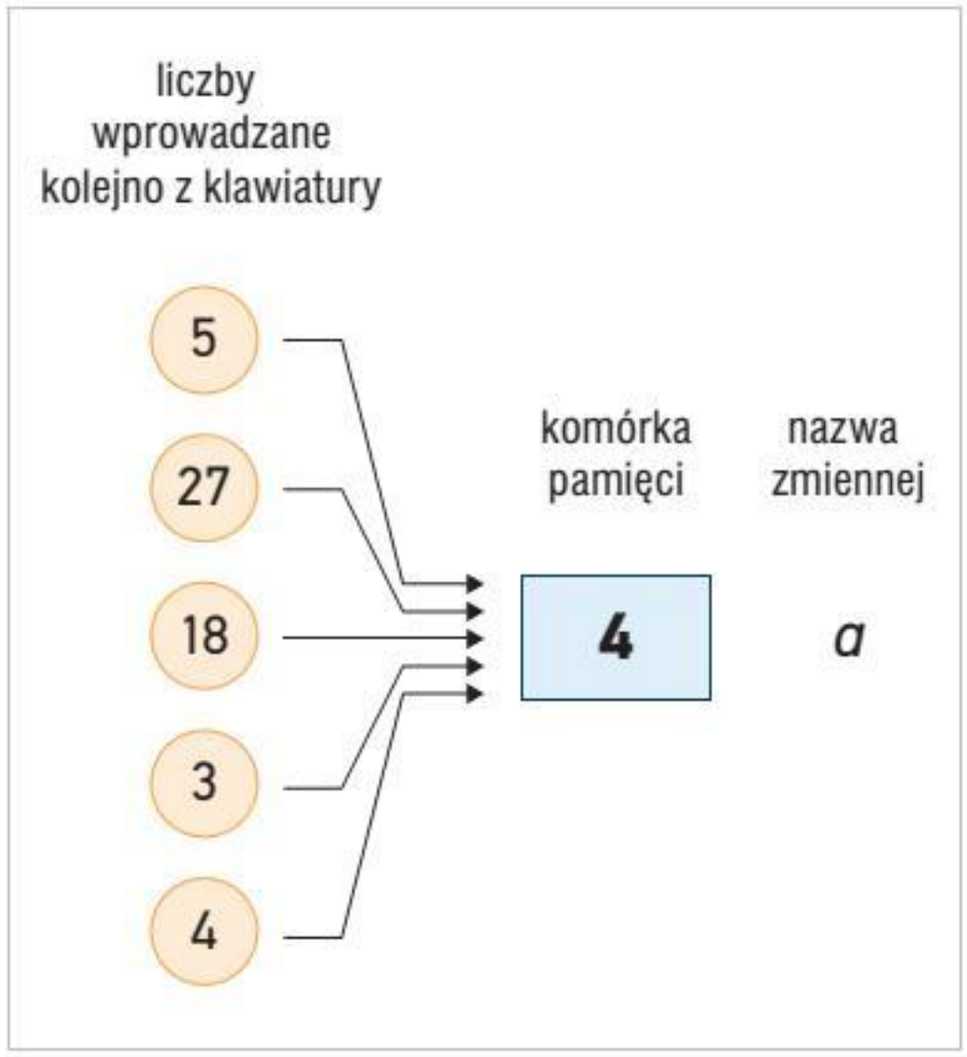
1. Otwórz program *Maksimum_n_funkcja* zapisany w ćwiczeniu 4.
2. Dodaj do programu funkcję główną `main()`, której treść widnieje na rysunku 5.
3. Zapisz program w pliku pod tą samą nazwą. Skompiluj i uruchom program. Sprawdź działanie programu dla kilku różnych danych.

3. Stosowanie tablic w języku C++ do przechowywania danych

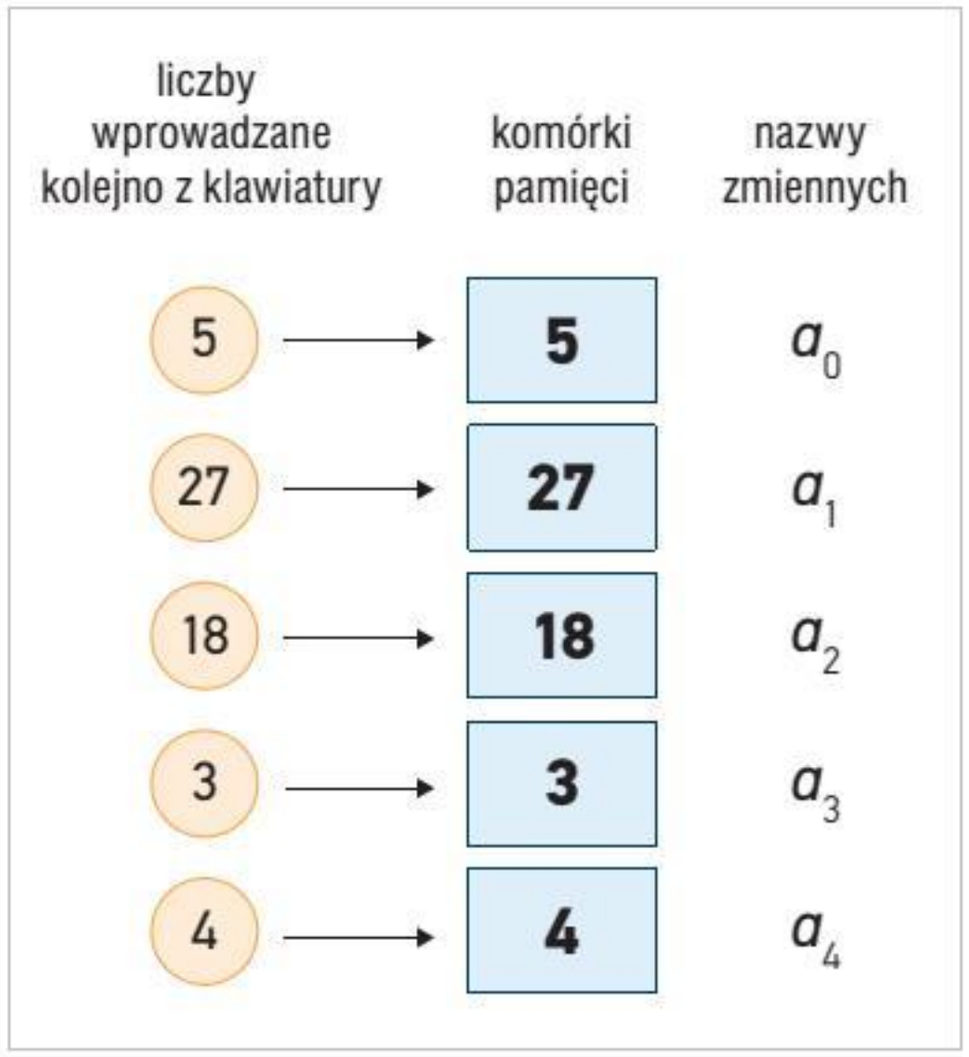
W funkcji *Maksimum* utworzonej w ćwiczeniu 4. wszystkie wprowadzane liczby komputer pamiętał w jednej zmiennej (o nazwie *a*). Gdy wprowadzamy z klawiatury liczby, na przykład ze zbioru {5, 27, 18, 3, 4}, każda kolejno wprowadzona liczba zastępuje poprzednią (rys. 6a). W jaki sposób zapamiętać wszystkie wprowadzone dane?

Niektóre algorytmy wymagają pamiętania wszystkich elementów zbioru w trakcie działania programu. Przykładem jest algorytm porządkowania, ponieważ w efekcie jego działania musimy otrzymać wszystkie elementy w określonym porządku.

Aby zapamiętać wszystkie dane, musimy je różnie nazwać. Moglibyśmy użyć liter alfabetu, ale w przypadku dużej liczby danych zabrakłoby liter. Ponadto odwoływanie się do wielu zmiennych byłoby niewygodne. Dlatego wymyślono tzw. **zmienne indeksowane**, w których do nazwy zmiennej (np. a) dodaje się kolejny numer (**indeks**), np.: $a_0, a_1, \ldots, a_n$. Wówczas dla każdej zmiennej komputer rezerwuje oddzielną komórkę pamięci (rys. 6b).



Rys. 6a. Zapisywanie w pamięci komputera wartości danych wprowadzanych z klawiatury z użyciem jednej zmiennej



Rys. 6b. Zapisywanie w pamięci komputera wartości danych wprowadzanych z klawiatury z użyciem pięciu zmiennych indeksowanych



Aby utworzyć zmienne indeksowane w języku C++, musimy zadeklarować specjalną strukturę danych – **tablicę** (rys. 7.).

C++
```
opis_typu nazwa_tablicy [liczba_elementów_tablicy];
```

Rys. 7. Ogólna postać deklarowania tablicy (C++)

Do elementów tablicy odwołujemy się, podając nazwę tablicy i indeks elementu umieszczony w nawiasach kwadratowych, np.: $a[0]$, $a[1]$, …, $a[n - 1]$ – dla tablicy n-elementowej o nazwie a. W języku C++ pierwszy indeks jest zawsze równy 0. Liczbę elementów tablicy musimy określić wcześniej, np. wpisać w deklaracji tablicy jako konkretną wartość.

Na przykład:
```
int dane[50];
```
oznacza zadeklarowanie tablicy o nazwie *dane* składającej się z pięćdziesięciu elementów typu całkowitego o indeksach od 0 do 49. Do elementów tablicy odwołujemy się przez zmienne: *dane*[0], *dane*[1], …, *dane*[49].

Liczbę elementów tablicy możemy podać również za pomocą wcześniej zdefiniowanej stałej (**const**).
Na przykład:

```
const int N = 10;
int a[N];
```

– oznacza zadeklarowanie tablicy o nazwie *a* składającej się z dziesięciu elementów typu całkowitego (**int**). Do elementów tablicy odwołujemy się przez zmienne: *a*[0], *a*[1], …, *a*[9].

```
const int LICZBA_EL = 100;
float tab[LICZBA_EL];
```

– oznacza utworzenie tablicy o nazwie *tab* składającej się ze stu elementów typu rzeczywistego (**float**). Do elementów tablicy odwołujemy się przez zmienne: *tab*[0], *tab*[1], …, *tab*[99].

Do wprowadzania danych do tablicy zastosujemy funkcję bez parametrów i niezwracającą wartości.



W przypadku definiowania funkcji bez parametrów nawiasy po nazwie funkcji pozostawiamy puste, tak jak w funkcji głównej `main()`.
Aby wywołać funkcję bez parametrów, należy podać jej nazwę i puste nawiasy, np. `WprowadzDane()`.

C++

Tablica.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  const int N = 5;
5  int a[N];
6
7  void WprowadzDane()
8  {
9      for(int i = 0; i < N; i++)
10     {
11         cout << "Podaj dana nr " << i << ": ";
12         cin >> a[i];
13     }
14 }
15 int main()
16 {
17     WprowadzDane();
18     return 0;
19 }
```

określenie liczby elementów tablicy

deklaracja tablicy o nazwie *a*

nagłówek funkcji bez parametrów i niezwracającej wartości

typ zmiennej sterującej *i* można określić w wyrażeniu początkowym pętli

wywołanie funkcji *WprowadzDane*

Rys. 8. Definicja funkcji umożliwiającej wprowadzenie danych z klawiatury do tablicy *a* – ćwiczenie 6.



Ćwiczenie 6. Wprowadzamy dane do tablicy

1. Napisz funkcję o nazwie *WprowadzDane* wprowadzającą dane do tablicy *a,* składającej się z pięciu liczb całkowitych (rys. 8.).
2. Zapisz program w pliku pod nazwą *Tablica*. Skompiluj i uruchom program.

Ćwiczenie 7. Wyprowadzamy dane z tablicy na ekran

1. Otwórz program *Tablica* zapisany w ćwiczeniu 6.
2. Zdefiniuj funkcję o nazwie *WyprowadzDane* wyprowadzającą w kolejnych wierszach dane z tablicy na ekran (rys. 9.).
3. W funkcji głównej `main()` po poleceniu wywołania funkcji *WprowadzDane* dodaj wywołanie funkcji *WyprowadzDane*.
4. Zapisz program w pliku pod tą samą nazwą. Skompiluj i uruchom program.

C++

```
16  void WyprowadzDane()
17  {
18      for(int i = 0; i < N; i++)
19          cout << "a[" << i << "] = " << a[i] << endl;
20  }
```

Rys. 9. Definicja funkcji wyprowadzającej dane z tablicy na ekran – ćwiczenie 7.

4. Zapisywanie w języku C++ wybranych algorytmów wyszukiwania i porządkowania

Zamierzamy wykorzystać możliwość zapamiętania danych w tablicy do zapisania w języku programowania algorytmów porządkowania (metodą przez wybieranie i metodą przez zliczanie) oraz algorytmu wyszukiwania elementu w zbiorze uporządkowanym. Jak to zrobić w języku C++?

Algorytm porządkowania metodą przez wybieranie

Zapiszemy w języku C++ algorytm porządkowania liczb od największej do najmniejszej metodą przez wybieranie. W programie wykorzystamy utworzone wcześniej dwie funkcje *WprowadzDane* (do wprowadzenia danych do tablicy) i *WyprowadzDane* (do wyprowadzenia danych na ekran po uporządkowaniu).

W programie wyszukujemy liczbę największą *maks* w tablicy danych *a* i zapamiętujemy jej pozycję (indeks) w zmiennej *maksi*. Następnie zamieniamy miejscami wyszukaną liczbę z liczbą, która jest aktualnie na początku zbioru. Do zamiany stosujemy zmienną pomocniczą *t*. Wymienione czynności powtarzamy z pominięciem liczb już uporządkowanych.

Zdefiniujemy dwie funkcje:

- *MaksWybor* (z jednym parametrem *starti*) wyszukującą element największy (*maks*) w zbiorze, począwszy od elementu o indeksie *starti*, i zapamiętującą jego pozycję w zmiennej *maksi* (zwracaną na końcu funkcji) – rys. 10.,
- *SortWybor* (niezwracającą wartości i bez parametrów), w której wywołujemy w pętli funkcję *MaksWybor* z parametrem aktualnym *i* oraz zamieniamy miejscami znaleziony element największy *maks* z aktualnie pierwszym elementem zbioru – rys. 11. Po każdym przejściu pętli liczba nieuporządkowanych elementów zbioru jest o 1 mniejsza, ponieważ nie sprawdzamy już elementów ustawionych na właściwych miejscach (uporządkowanych).

W funkcji głównej `main()` wywołamy trzy funkcje: wprowadzania danych (*WprowadzDane*), porządkowania (*SortWybor*) i wyprowadzania danych na ekran (*WyprowadzDane*) – rys. 12.

C++

```
22  int MaksWybor(int starti)
23  {
24      int maksi = starti;
25      int maks = a[maksi];
26      for(int i = starti + 1; i < N; i++)
27          if(a[i] > maks))
28          {
29              maksi = i;
30              maks = a[maksi];
31          }
32      return maksi;
33  }
```

deklarujemy zmienną *maksi* jako liczbę całkowitą i równocześnie przypisujemy jej numer pierwszego elementu nieuporządkowanego zbioru

do początkowej wartości zmiennej sterującej *starti* dodajemy 1, ponieważ wartość pierwszego elementu nieuporządkowanego zbioru już wprowadziliśmy do zmiennej *maks*

w pętli szukamy kolejnych liczb większych od aktualnie największej i zapamiętujemy pozycję największej liczby w tablicy `maksi = i`; polecenia w pętli powtarzamy *n* – (*starti* + 1) razy

funkcja zwraca indeks największego elementu *maksi*

Rys. 10. Definicja funkcji wyszukującej element największy i zwracającej jego pozycję w tablicy – ćwiczenie 8.

C++

```
35  void SortWybor()
36  {
37      for(int i = 0; i < N - 1; i++)
38      {
39          int maksi = MaksWybor(i);
40          int t = a[i];
41          a[i] = a[maksi];
42          a[maksi] = t;
43      }
44  }
```

wywołujemy funkcję *MaksWybor* z parametrem aktualnym *i*, wynik przypisując zmiennej *maksi*, którą równocześnie deklarujemy jako `int`

zmienną pomocniczą *t* stosujemy do zamiany miejscami znalezionego największego elementu z aktualnie pierwszym

Rys. 11. Definicja funkcji porządkującej elementy zbioru metodą przez wybieranie – ćwiczenie 8.

C++

```
46  int main()
47  {
48      WprowadzDane();
49      SortWybor();
50      WyprowadzDane();
51      return 0;
52  }
```

Rys. 12. Wywołanie w programie głównym funkcji wprowadzania danych, porządkowania *metodą* przez wybieranie i wyprowadzania danych na ekran – ćwiczenie 8.

Ćwiczenie 8. Zapisujemy w języku C++ algorytm porządkowania metodą przez wybieranie

1. Przypomnij sobie działanie algorytmu porządkowania metodą przez wybieranie.
2. Otwórz program *Tablica* zapisany w ćwiczeniu 7. Uzupełnij program, dodając definicje funkcji *MaksWybor* i *SortWybor* oraz wywołanie funkcji *SortWybor* w funkcji `main()`.
3. Zapisz program w pliku pod nazwą *Wybieranie*. Skompiluj i uruchom program. Sprawdź działanie programu dla różnych danych.

Algorytm porządkowania metodą przez zliczanie

Zapiszemy w języku C++ algorytm porządkowania liczb metodą przez zliczanie od najmniejszej do największej. W programie wykorzystamy utworzone wcześniej dwie funkcje: *WprowdzDane* do wprowadzania danych do tablicy i *WyprowadzDane* – do wyprowadzania danych na ekran po uporządkowaniu. Funkcję *WprowadzDane* musimy zmodyfikować, dodając sprawdzanie, czy liczba należy do określonego zakresu. Ustalimy zakres <0, *MAX_DANA*>. Zmiennej *MAX_DANA* przypiszemy stałą wartość, np. 99.

C++

Zliczanie.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  const int N = 5;
5  const int MAX_DANA = 99;
6  int a[N];
7
8  void WprowadzDane()
9  {
10     int i;
11     for(i = 0; i < N; i++)
12     {
13         cout << "Podaj dana nr " << i << " (z przedzialu <0," << MAX_DANA << ">): ";
14         cin >> a[i];
15         if(a[i] < 0)
16             a[i] = 0;
17         if(a[i] > MAX_DANA)
18             a[i] = MAX_DANA;
19     }
20 }
```

Rys. 13. Definicja funkcji wprowadzającej dane do tablicy – ćwiczenie 9.

Zdefiniujemy funkcję *SortZlicz* (niezwracającą wartości i bez parametrów), w której utworzymy tablicę *liczniki* [*MAX_DANA* + 1], służącą do zliczania wystąpień elementów zbioru. Liczba liczników, zgodnie z algorytmem, musi umożliwić zliczenie każdej liczby z danego zakresu, więc przygotowujemy 100 liczników (dla liczb od 0 do 99). W pierwszej pętli **`for`** (rys. 15., wiersze 34-35) dla każdej liczby wprowadzonej przez użytkownika zwiększamy odpowiadający jej licznik o 1. Zapis `liczniki[a[i]]++` odpowiada instrukcji przypisania: `liczniki[a[i]] = liczniki[a[i]] + 1`. Na rysunku 14. pokazano przykładową tablicę *liczniki* dla liczb {1, 1, 2, 4, 5}.

0	2	1	0	1	1
liczniki [0]	liczniki [1]	liczniki [2]	liczniki [3]	liczniki [4]	liczniki [5]

Rys. 14. Wartości elementów tablicy liczniki dla liczb {1, 1, 2, 4, 5}

W pętli **`for`** (rys. 15., wiersze 38-39) przeglądamy tablicę *liczniki* i wpisujemy do tablicy z danymi (*a*) daną wartość tyle razy, ile razy pojawiła się w zbiorze (liczbie wystąpień odpowiada element tablicy *liczniki*).

Pętla z wierszy 38-39 jest wykonywana dla każdej możliwej wartości danych wejściowych (pętla **`for`** – wiersz 37.), mamy tu zatem do czynienia z **zagnieżdżeniem pętli**. Zmienna *poz* określa aktualną pozycję w tablicy *a*. Zapis `a[poz++] = i` (wiersz 39.) odpowiada sekwencji instrukcji:

```
a[poz] = i;
poz++;
```

C++

```
29  void SortZlicz()
30  {
31      int liczniki[MAX_DANA + 1] = {};
32      int i, j;
33
34      for(i = 0; i < N; i++)
35          liczniki[a[i]]++;
36      int poz = 0;
37      for(i = 0; i <= MAX_DANA; i++)
38          for(j = 0; j < liczniki[i]; j++)
39              a[poz++] = i;
40  }
```

zerujemy tablicę *liczniki*

dla każdej wprowadzonej liczby zwiększamy odpowiadający jej licznik o 1

układamy elementy tablicy *a* od najmniejszego do największego

Rys. 15. Definicja funkcji porządkującej elementy zbioru metodą przez zliczanie – ćwiczenie 9.

W funkcji głównej `main()` wywołamy trzy funkcje: wprowadzania danych (*WprowadzDane*), porządkowania (*SortZlicz*) i wyprowadzania danych na ekran (*WyprowadzDane*) – rys. 16.

C++

```
43  int main()
44  {
45      WprowadzDane();
46      cout << endl;
47      cout << "Wprowadzone dane:" << endl;
48      WyprowadzDane();
49      cout << "Dane posortowane rosnaco przez zliczanie:" << endl;
50      SortZlicz();
51      WyprowadzDane();
52      return 0;
53  }
```

Rys. 16. Wywołanie w programie głównym funkcji wprowadzania danych, porządkowania *metodą* przez zliczanie i wyprowadzania *danych* na ekran – ćwiczenie 9.

Ćwiczenie 9. Zapisujemy w języku C++ algorytm porządkowania metodą przez zliczanie

1. Przypomnij sobie działanie algorytmu porządkowania metodą przez zliczanie.
2. Otwórz program *Tablica* zapisany w ćwiczeniu 7. Zmodyfikuj funkcję *WprowadzDane* zgodnie z definicją pokazaną na rysunku 13.
3. Uzupełnij program, dodając definicję funkcji *SortZlicz* (rys. 15.) oraz wywołanie funkcji *SortZlicz* w funkcji `main()` – rys. 16.
4. Zapisz program w pliku pod nazwą *Zliczanie*. Skompiluj i uruchom program. Sprawdź działanie programu dla różnych danych.

Algorytm wyszukiwania elementu w zbiorze uporządkowanym

Utworzymy program, który w zbiorze uporządkowanym ma wyszukiwać daną podaną przez użytkownika i podawać jej pozycję w tablicy. Zakładamy, że wprowadzamy dane od najmniejszej do największej (inaczej program nie będzie działał poprawnie). Bez takiego założenia należałoby dodać również funkcję sortowania danych w określonym porządku. W programie wykorzystamy utworzoną wcześniej funkcję *WprowadzDane* do wprowadzenia danych do tablicy.

Zdefiniujemy funkcję *ZnajdzDana* z parametrem *wartosc* (rys. 17.). W funkcji wyznaczamy pozycję elementu środkowego (zmienna *srodek*). Jeśli element środkowy (*a*[*srodek*]) jest szukaną daną (zmienna *wartosc*), to funkcja zwraca zmienną *srodek*, czyli pozycję danego elementu. Jeśli szukana dana jest mniejsza od danej środkowej, odrzucamy prawą część zbioru (liczby większe od elementu środkowego) – ustalamy nowy koniec (`koniec = srodek - 1`). W przeciwnym wypadku odrzucamy lewą część zbioru (liczby mniejsze od elementu środkowego) i ustalamy nowy początek (`poczatek = srodek + 1`). Połowienie powtarzamy dopóki początek zbioru jest mniejszy lub równy końcowi zbioru (`poczatek <= koniec`) – o ile nie znajdziemy szukanego elementu (wtedy komputer wykona instrukcję **`return`** `srodek`).

W algorytmie liczba iteracji zależy od spełnienia (lub niespełnienia) warunku, dlatego użyjemy instrukcji iteracyjnej **`do`** ... **`while`**, którą omówiliśmy w temacie 6.

W funkcji *ZnajdzDana* (rys. 17.) instrukcja return występuje dwukrotnie. Niezależnie od miejsca, w którym występuje, zawsze kończy wykonanie funkcji. Jeśli element środkowy jest szukaną liczbą, funkcja zwraca wartość zmiennej *srodek* (**`return`** `srodek`).

C++

```
16  int ZnajdzDana(int wartosc)
17  {
18      int poczatek, koniec, srodek;
19      poczatek = 0;
20      koniec = N - 1;
21      do
22      {
23          srodek = (poczatek + koniec) / 2;
24          if(a[srodek] == wartosc)
25              return srodek;
26          else
27              if(wartosc < a[srodek])
28                  koniec = srodek-1;
29              else
30                  poczatek = srodek+1;
31      } while(poczatek <= koniec);
32
33      return -1;
34  }
```

wyznaczamy pozycję elementu środkowego

element środkowy jest szukaną liczbą

wyznaczamy nowy koniec lub początek

Rys. 17. Definicja funkcji wyszukującej daną w zbiorze uporządkowanym – ćwiczenie 10.

Jeśli w przeszukiwanym zbiorze nie ma szukanej liczby, funkcja *ZnajdzDana* (rys. 17.) zwraca wartość -1. Programiści często używają wartości niepoprawnych (w tym przypadku indeks tablicy nie może być liczbą ujemną) do zasygnalizowania niepowodzenia operacji lub błędu.

W funkcji głównej `main()` wywołamy funkcję wprowadzania danych (*WprowadzDane*), wprowadzimy z klawiatury daną do wyszukania (zmienna *wartosc*), wywołamy funkcję wyszukiwania danej (*ZnajdzDana*) i wypiszemy wynik. Uwzględnimy również sytuację, gdy podanej wartości nie ma w zbiorze (rys. 18.).

C++

```
35  int main ()
36  {
37      int wartosc, pozycja;
38      WprowadzDane();
39      cout << "Podaj dana do wyszukania: ";
40      cin >> wartosc;
41      pozycja = ZnajdzDana(wartosc);
42      if(pozycja >= 0)
43          cout << "Znaleziono wartosc " << wartosc << " na pozycji "
44          << pozycja << " (pozycje liczone sa od 0)" << endl;
45      else
46          cout << "Nie znaleziono wartosci " << wartosc << endl;
47
48      return 0;
49  }
```

Rys. 18. Wywołanie w programie głównym funkcji wprowadzania danych, wyszukiwania określonej danej i wyprowadzania wyniku na ekran – ćwiczenie 10.

Ćwiczenie 10. Zapisujemy w języku C++ algorytm wyszukiwania elementu w zbiorze uporządkowanym

1. Przypomnij sobie działanie algorytmu wyszukiwania elementu w zbiorze uporządkowanym.
2. Otwórz program *Tablica* zapisany w ćwiczeniu 7. Usuń funkcję *WyprowadzDane*.
3. Uzupełnij program, dodając definicję funkcji *ZnajdzDana* (rys. 17.). W funkcji `main()` wywołaj funkcję *ZnajdzDana* oraz dodaj instrukcje wprowadzania szukanej danej i wyświetlania wyniku (rys. 18.).
4. Zapisz program w pliku pod nazwą *Wyszukiwanie*. Skompiluj i uruchom program. Sprawdź działanie programu dla różnych danych (pamiętaj o wprowadzaniu danych uporządkowanych – od najmniejszej do największej).

Warto zapamiętać

- Aby pisać przejrzyste programy, należy opracowywać oddzielnie problemy cząstkowe, stosując podprogramy (procedury i funkcje). Podprogram należy najpierw zdefiniować, a potem wywołać w programie głównym.
- W języku C++ definiujemy funkcje:
  - niezwracające wartości, gdy chcemy np. wyprowadzić dane,
  - zwracające wartość, gdy celem funkcji jest obliczenie i zwrócenie pewnej wartości, np. wyniku sumowania.
- Parametr to wartość przekazywana do funkcji, a wartość zwracana to wartość przekazywana z funkcji do programu głównego.

- Możemy definiować funkcje z parametrami i bez parametrów. W momencie wywołania funkcji parametr formalny przyjmie wartość parametru aktualnego.
- Aby wywołać funkcję z parametrami, należy podać jej nazwę i parametry aktualne w nawiasach okrągłych. Gdy parametry nie występują, nawiasy pozostawiamy puste.
- Program główny komunikuje się z funkcją poprzez przekazywanie parametrów.
- Aby zaprogramować algorytm, w którym w trakcie działania programu musimy pamiętać wszystkie elementy zbioru, możemy zapisywać te dane w tablicy.
- W języku C++ liczba elementów (rozmiar) tablicy musi być z góry określona, np. jako zdefiniowana wcześniej stała.

Pytania i polecenia

1. Dlaczego definiujemy podprogramy?
2. Czym jest parametr formalny, a czym aktualny w podprogramie (procedurze, funkcji)?
3. Kiedy definiujemy funkcję niezwracającą wartości, a kiedy zwracającą wartość w języku C++?
4. Jak wywołujemy funkcję bez parametrów, a jak z parametrami w języku C++?
5. W jakim celu definiujemy tablicę? Podaj przykład użycia tablicy.
6. Wyjaśnij na przykładach sposób deklarowania tablicy w języku C++.
7. Wyjaśnij sposób zapisania w języku C++ algorytmu porządkowania metodą przez wybieranie? Do czego służą funkcje zdefiniowane w tym programie.
8. Dlaczego w programie realizującym algorytm porządkowania metodą przez wybieranie musimy zapamiętywać pozycję znalezionego elementu największego w oddzielnej zmiennej?
9. Wyjaśnij sposób zapisania w języku C++ algorytmu porządkowania metodą przez zliczanie. Do czego służą funkcje zdefiniowane w tym programie?
10. Do czego potrzebna jest tablica *liczniki* w programie realizującym algorytm porządkowania metodą przez zliczanie?
11. Wyjaśnij sposób zapisania w języku C++ algorytmu wyszukiwania elementu w zbiorze uporządkowanym. Do czego służą funkcje zdefiniowane w tym programie?
12. Dlaczego program realizujący algorytm wyszukiwania elementu w zbiorze uporządkowanym zaczyna się od sprawdzenia, czy środkowy element jest szukaną liczbą?

Zadania

1. Otwórz program *Roboty* zapisany w ćwiczeniu 1. Korzystając z tego programu i zdefiniowanego pomocnika, wyczaruj w górnej części sceny inne cztery obiekty (np. obiekty latające) według własnego pomysłu. Zapisz program w pliku pod nazwą *Moje obiekty*.
2. Otwórz program *Kwadraty* zapisany w ćwiczeniu 2. Usuń skrypt główny, pozostaw definicję procedury *Kwadrat*. Korzystając z tej procedury, utwórz kompozycję z kwadratów według własnego pomysłu. Zapisz program w pliku pod nazwą *Moje kwadraty*.
3. Napisz program w języku C++ umożliwiający wyprowadzenie na ekran monitora napisu „Gwiazdkowe pozdrowienia", a pod nim piętnastu gwiazdek (znaków *). Zdefiniuj funkcję *Gwiazdki* jako niezwracającą wartości, czyli typu `void` i bez parametrów. Zadaniem funkcji jest wyprowadzenie w jednym wierszu piętnastu gwiazdek. Funkcję wywołaj w programie głównym. Zapisz program w pliku pod nazwą *Życzenia*.

4. Zmodyfikuj program *Życzenia* zapisany w zadaniu 3. tak, aby wyświetlił w kolejnym wierszu napis „Wszystkiego najlepszego”, a pod nim ponownie piętnaście gwiazdek. Zapisz program w pliku pod tą samą nazwą.
5. Napisz program w języku C++ umożliwiający obliczenie objętości sześcianu. Zdefiniuj funkcję *Szescian* z jednym parametrem typu całkowitego o nazwie *b*, obliczającą sześcian liczby *liczba* i zwracającą do programu głównego wynik obliczenia. Wywołaj funkcję w programie głównym z parametrem aktualnym *bok*, którego wartość wprowadzaj z klawiatury. Zapisz program w pliku pod nazwą *Objętość sześcianu*.
6. Korzystając ze zdefiniowanej w ćwiczeniu 4. funkcji *Maksimum*, zdefiniuj funkcję *Minimum* i napisz program wyszukujący najmniejszy element w zbiorze *n*-elementowym. Zapisz program w pliku pod nazwą *Minimum_n_funkcja*.
7. Napisz w języku C++ program realizujący algorytm wyszukiwania danego elementu w zbiorze nieuporządkowanym, korzystając z opisu podanego w temacie 7., w punkcie 1.3. Do wprowadzenia danych wykorzystaj funkcję *WprowadzDane()*, a do wyprowadzenia – funkcję *WyprowadzDane()*. Do wyszukania danej zastosuj funkcję *WyszukajDana()*, której wzór pokazano na rysunku 19. Zapisz program w pliku pod nazwą *Wyszukiwanie_nieuporządkowany*.

C++

```
24  int WyszukajDana(int wartosc)
25  {
26      for(int i = 0; i < N; i++)
27          if(a[i]  ==  wartosc)
28              return i;
29      return -1;
30  }
```

Rys. 19. Definicja funkcji wyszukującej dany element w zbiorze nieuporządkowanym i zwracającej pozycję elementu w tablicy

Dla zainteresowanych

8. Zmodyfikuj program zapisany w zadaniu 4. tak, aby liczba gwiazdek była parametrem *n* typu całkowitego funkcji *Gwiazdki*. Wartość parametru aktualnego *liczba_gwiazdek* wprowadzaj w momencie wywołania funkcji w programie głównym. Zapisz program w pliku pod nazwą *Gwiazdki_n*.
9. Do programu zapisanego w zadaniu 5. dodaj funkcję *Kwadrat* z jednym parametrem o nazwie *b* typu całkowitego, obliczającą kwadrat liczby *b* i zwracającą do programu głównego wynik obliczenia. Wywołaj funkcję w programie głównym z parametrem aktualnym *bok*. Zapisz program w pliku pod nazwą *Objętość sześcianu i pole kwadratu*.
10. Otwórz program zapisany w ćwiczeniu 5. Dodaj wprowadzanie danych z wykorzystaniem tablicy. Zmodyfikuj funkcję *Maksimum* tak, aby stosowane były zmienne indeksowane. Zapisz program w pliku pod nazwą *Maksimum_tablica*.
11. Otwórz plik *Wybieranie* zapisany w ćwiczeniu 8. Zmodyfikuj program tak, aby porządkował wprowadzone dane od największej do najmniejszej.
12. Otwórz plik *Zliczanie* zapisany w ćwiczeniu 9. Zmodyfikuj program tak, aby porządkował wprowadzone dane od największej do najmniejszej.
13. Otwórz plik *Wyszukiwanie* zapisany w ćwiczeniu 10. Chcielibyśmy wprowadzać również dane nieuporządkowane. Dodaj do programu funkcję porządkowania danych od najmniejszej do największej metodą przez wybieranie. Wywołaj ją w programie głównym. Zapisz program w pliku pod tą samą nazwą.

Temat 9. Wykorzystanie funkcji i list do zapisywania w języku Python algorytmów porządkowania i wyszukiwania

1. **Stosowanie procedur do utworzenia kompozycji w języku Scratch**
2. **Definiowanie i stosowanie funkcji w języku Python**
3. **Stosowanie list do wprowadzania danych**
4. **Zapisywanie w języku Python wybranych algorytmów porządkowania i wyszukiwania**

Warto powtórzyć

1. Do czego służą podprogramy (procedury i funkcje)?
2. Jak możemy nadać wartość zmiennej w języku Python?
3. W jaki sposób w języku Python wyprowadzamy napisy na ekran monitora?
4. Za pomocą jakiego polecenia realizujemy sytuację warunkową w języku Python?
5. W jaki sposób w języku Python zapisujemy iterację?

1. Stosowanie procedur do utworzenia kompozycji w języku Scratch

W językach Scratch i Python definiowaliśmy własne podprogramy (procedury i funkcje) rysujące figury geometryczne (np. kwadrat, sześciokąt). Podprogramy te wykorzystywaliśmy następnie do tworzenia kompozycji graficznych. W środowisku Baltie definiowaliśmy własne procedury, które wykorzystywaliśmy do tworzenia ciekawych scen, np. budowania domów czy… robotów.

Jak już wiemy, podprogramy (procedury i funkcje) służą do opracowywania oddzielnie problemów cząstkowych, które są częściami zadania rozwiązywanego przez dany program. Aby zastosować podprogram, czyli procedurę lub funkcję, należy ją **zdefiniować**, a następnie **wywołać**.

W programie pokazanym na rysunku 1a problemem cząstkowym jest narysowanie kwadratowego okna o krawędzi dowolnej długości. Problem zapisujemy w postaci procedury *Okno* z parametrem formalnym *krawędź*, określającym długość krawędzi ramy okiennej. Następnie w programie głównym trzykrotnie wywołujemy procedurę *Okno* z parametrem aktualnym *k*.

W chwili wywołania procedury parametr formalny *krawędź* zostanie zastąpiony parametrem aktualnym *k* (wartość zmiennej *k* wprowadzamy z klawiatury).

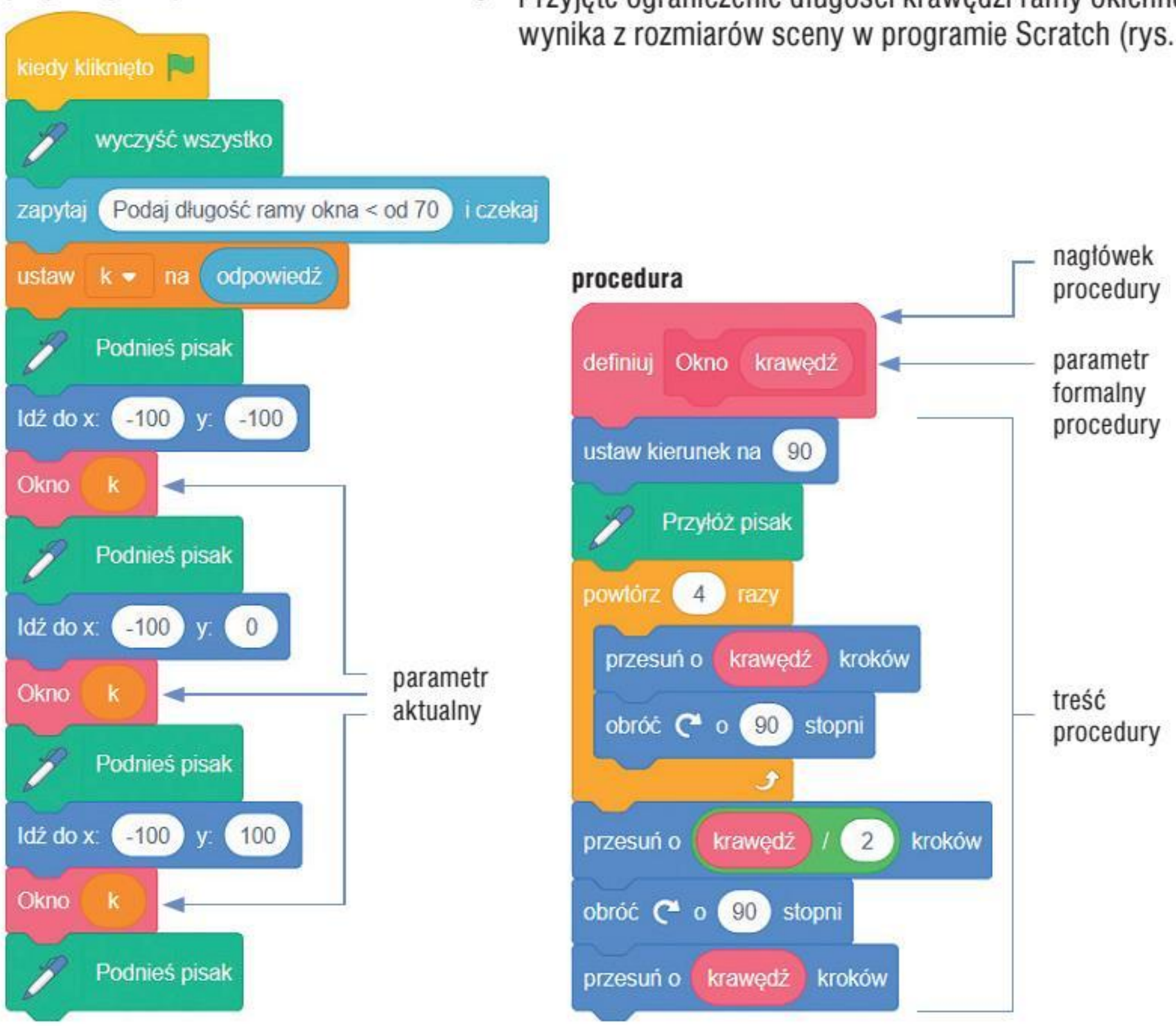
Procedurę (blok) definiujemy, wybierając element **Utwórz blok** z grupy **Moje bloki**. Parametr dodajemy w oknie **Utwórz blok**. Element reprezentujący parametr pojawi się w nagłówku procedury (rys. 1a).

Ćwiczenie 1. Definiujemy procedurę z parametrem w języku Scratch i wywołujemy ją w programie

1. Zdefiniuj procedurę *Okno* z jednym parametrem. Utwórz program rysujący trzy okna, wywołując trzykrotnie procedurę *Okno* (rys. 1a).
2. Zapisz program w pliku pod nazwą *Okna* i uruchom program dla różnych wartości parametru *k*.
3. Rozbuduj program tak, aby duszek-kot narysował kolejne trzy okna. Następnie ma narysować kontur domu, a na koniec stanąć po lewej stronie domu (rys. 1b). Zapisz program w pliku pod nazwą *Dom z oknami*. Sprawdź działanie programu dla różnych danych.

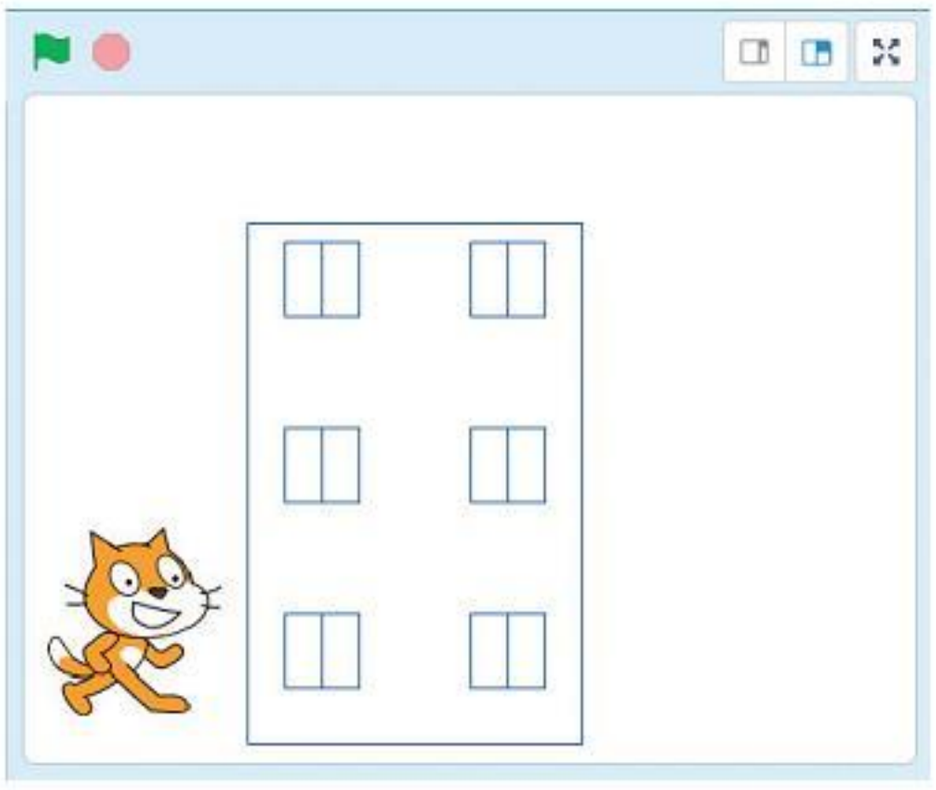
Wskazówki:

- W treści procedury używamy elementu reprezentującego parametr, przeciągając go z nagłówka procedury do danego pola, np. w elemencie **przesuń** (rys. 1a).
- Przyjęte ograniczenie długości krawędzi ramy okiennej wynika z rozmiarów sceny w programie Scratch (rys. 1a).



Rys. 1a. Zastosowanie procedury *Okno* w programie *Okna* (Scratch) – ćwiczenie 1. (pkt. 1.)

Scratch



Rys. 1b. Efekt wykonania programu *Dom z oknami* (Scratch) – ćwiczenie 1. (pkt. 3.)

2. Definiowanie i stosowanie funkcji w języku Python

W języku Python podprogramy nazywamy **funkcjami**. Poznaliśmy kilka wbudowanych funkcji Pythona, m.in.: `print()`, `input()`, `forward()`, `circle()`. W języku Python można definiować własne funkcje.

Funkcje mogą zwracać wartość do programu głównego (rys. 2a) lub nic nie zwracać (rys. 2b).

W definicji funkcji określamy **parametry formalne** (zwane też **parametrami**), które w momencie wywołania funkcji komputer zastąpi **parametrami aktualnymi** (zwanymi też **argumentami**).

Python

```
def nazwa_funkcji(lista_parametrów_formalnych):
    lista_instrukcji
    return wartość
```

Rys. 2a. Ogólna postać definicji funkcji zwracającej wartość (Python)

Python

```
def nazwa_funkcji(lista_parametrów_formalnych):
    lista_instrukcji
```

Rys. 2b. Ogólna postać definicji funkcji niezwracającej wartości (Python)

Lista_instrukcji i słowo **`return`** muszą być przesunięte w prawo przynajmniej o jedną spację (przyjęte jest wcięcie składające się z czterech spacji).

W języku Python nie określamy typu parametru formalnego ani typu zwracanej wartości. Podobnie jak w przypadku zmiennych interpreter automatycznie rozpoznaje typ parametru oraz zwracanej wartości.

W nazwach funkcji można używać tylko liter, znaków podkreślenia i cyfr. Nazwa nie może zaczynać się od cyfry. Przyjęte jest niestosowanie polskich liter. Należy stosować nazwy opisujące działanie danej funkcji. Ważne jest, aby przyjąć jednolity sposób nazewnictwa.

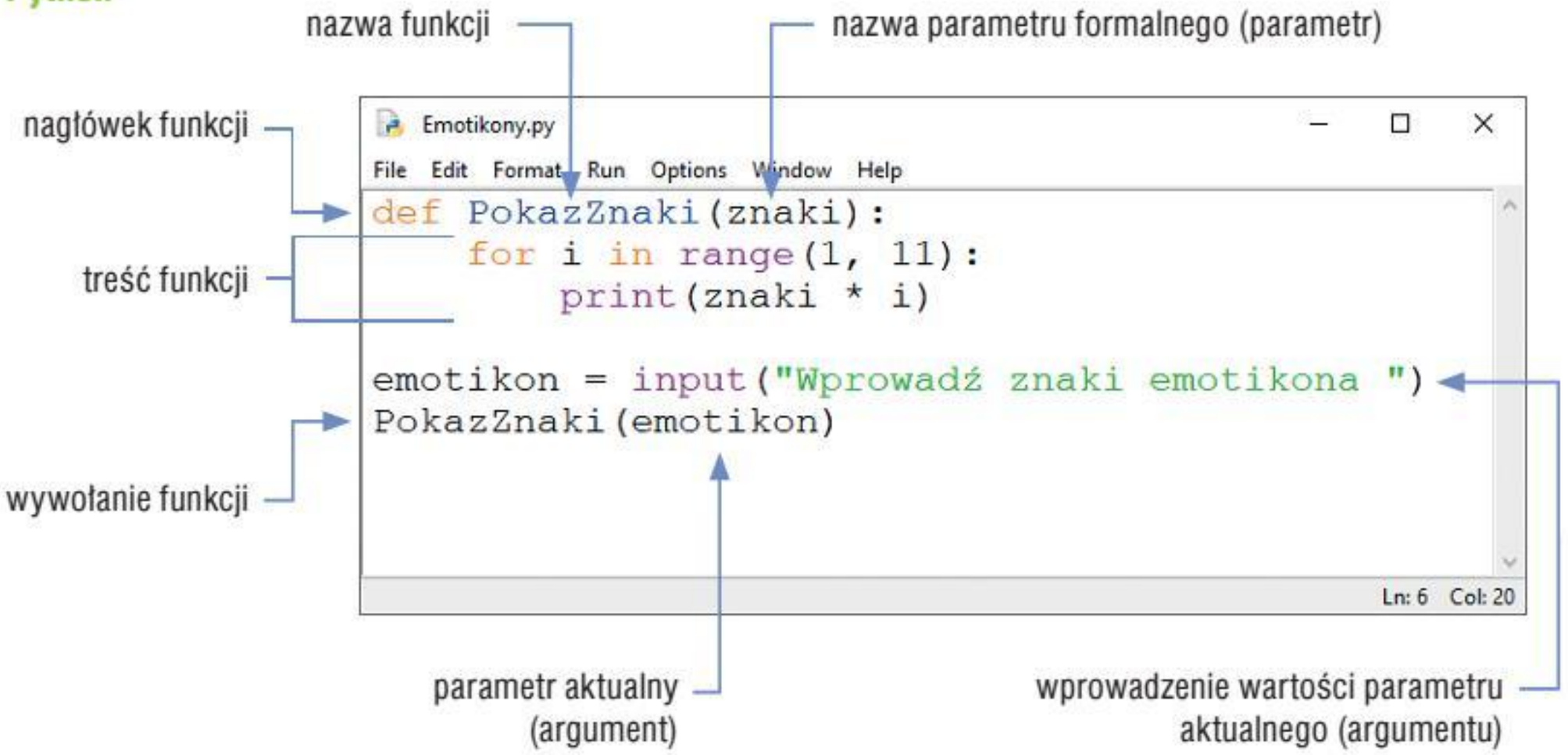
Według popularnego sposobu nazewnictwa w nazwach funkcji i zmiennych używa się wyłącznie małych liter, a w nazwach wieloczłonowych stosuje się znak podkreślenia, np. *wprowadz_dane*, *wyprowadz_dane*, *najwiekszy*.

W podręczniku przyjęliśmy jednak zapis nazw funkcji wielką literą, np. *Maksimum*, *Suma*. Jeśli nazwa jest kilkuczłonowa, następny człon również zaczynamy od wielkiej litery i nie rozdzielamy członów żadnym znakiem, np. *WprowadzDane*, *WyprowadzDane*. W nazwach będziemy stosować skróty, np. *SortWybor*. W ten sposób nazwy funkcji w tym temacie będą odpowiadały nazwom użytym w temacie 8. podręcznika.

Funkcję niezwracającą wartości (rys. 2b) stosujemy na przykład, gdy mamy wyprowadzić dane na ekran lub wyświetlić grafikę. Na rysunkach 3. i 4a. pokazano przykłady funkcji niezwracających wartości. W programie na rys. 3. wprowadzone znaki emotikona zostaną wyświetlone na ekranie 10 razy, a na rys. 4a wyświetli się kompozycja z dziesięciokątów. W treści funkcji nie umieszczamy instrukcji **`return`**.

Aby wywołać funkcję niezwracającą wartości, należy umieścić nazwę funkcji z **parametrami aktualnymi** w odpowiednim miejscu programu głównego (rys. 3.). W przypadku braku parametrów nawiasy po nazwie funkcji pozostawiamy puste.

Python



Rys. 3. Przykład definicji funkcji niezwracającej wartości z jednym parametrem (Python) – ćwiczenie 2.

Ćwiczenie 2. Definiujemy w języku Python funkcję niezwracającą wartości

1. Przepisz program pokazany na rysunku 3.
2. Zapisz program w pliku pod nazwą *Emotikony*. Uruchom program. Objaśnij działanie programu.
3. Dodaj do funkcji drugi parametr (*liczba_em*), który będzie określał długość listy wartości w funkcji `range()`.
4. Zapisz program pod tą samą nazwą i uruchom.

Ćwiczenie 3. Stosujemy funkcję z parametrem do tworzenia grafiki

1. Przepisz program pokazany na rysunku 4a. Zapisz program w pliku pod nazwą *Dziesięciokąty1*.
2. Uruchom program. Objaśnij znaczenie poszczególnych wierszy programu. Które instrukcje są wykonywane w pętli? Ile razy jest wywołana funkcja *Dziesieciokat()* i z jakimi wartościami?
3. Zmodyfikuj program, aby żółw dorysował kompozycję z niebieskich dziesięciokątów tak, jak pokazano na rysunku 4b. Zapisz program w pliku pod nazwą *Dziesięciokąty2*. Uruchom program.

Python

Dziesieciokat_R.py - D:/Python/Dziesieciokat_R.py

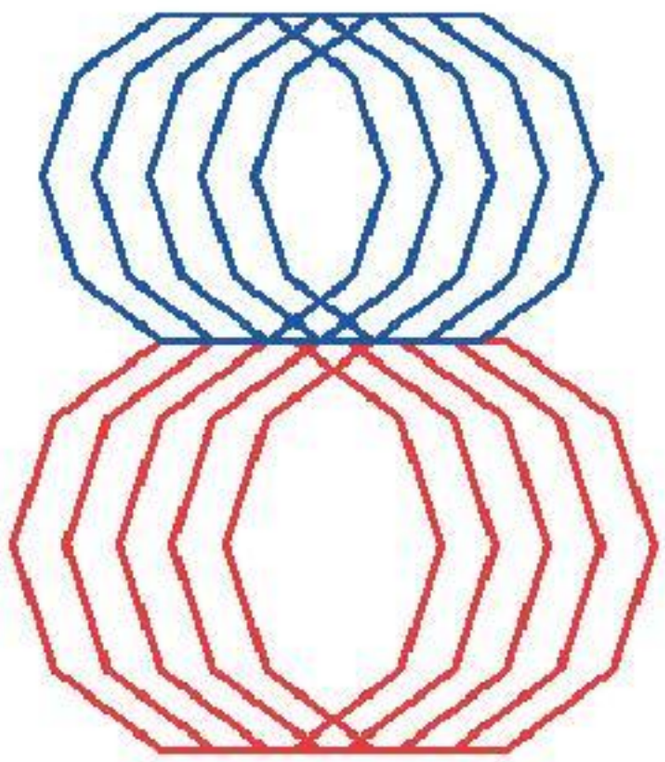
File Edit Format Run Options Window Help

```
from turtle import *

def Dziesieciokat(bok):
    for i in range(10):
        forward(bok)
        right(36)

pensize(3)
pencolor("red")
for i in range(5):
    Dziesieciokat(50)
    forward(20)
hideturtle()
```

Rys. 4a. Program w języku Python z definicją funkcji *Dziesieciokat()* z jednym parametrem – ćwiczenie 3.

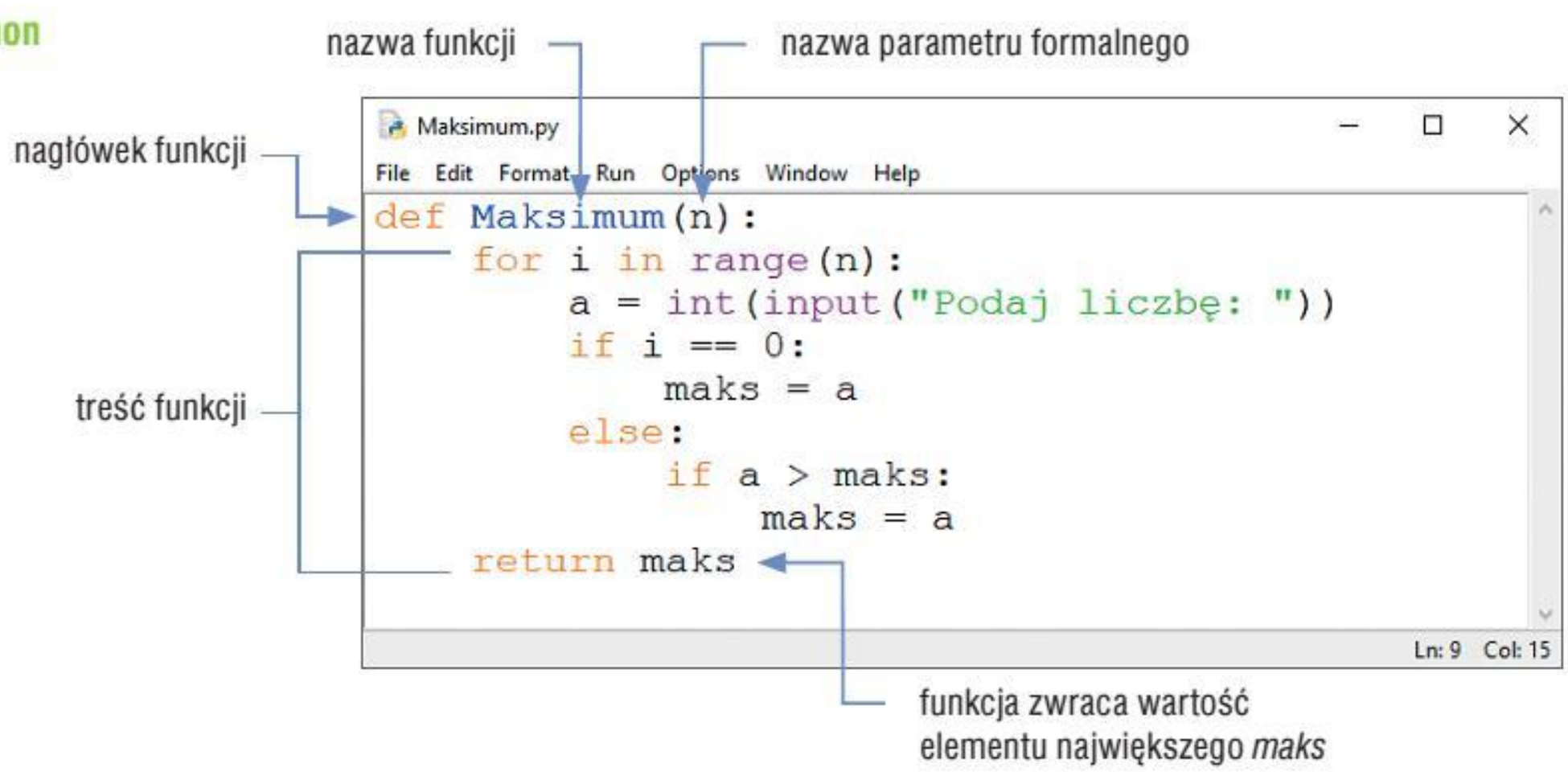


Rys. 4b. Kompozycja z dziesięciokątów o różnych długościach boków – efekt wykonania programu z ćwiczenia 3. (punkt 3.)

Jeśli celem funkcji jest obliczenie i zwrócenie pewnej wartości do programu, stosujemy funkcję zwracającą wartość. Funkcja musi zawierać instrukcję **`return`** z wartością zwracaną do programu głównego (rys. 2a).

Sposób definiowania takiej funkcji pokażemy na przykładzie funkcji z jednym parametrem wyszukującej największy element w zbiorze n liczb ($n > 0$) wprowadzonych z klawiatury (rys. 5.). Parametrem przekazywanym funkcji jest liczba elementów zbioru.

Python



Rys. 5. Przykład definicji funkcji zwracającej wartość z jednym parametrem (Python) – ćwiczenie 4.

Ćwiczenie 4. Definiujemy w języku Python funkcję zwracającą wartość

1. Zdefiniuj funkcję *Maksimum* z parametrem *n* (gdzie *n* to liczba elementów w zbiorze) wyszukującą największy element w zbiorze (rys. 5.).
2. Zapisz program w pliku pod nazwą *Maksimum_n_funkcja*.

Wartość zwróconą przez funkcję możemy przypisać zmiennej w programie głównym.

Wywołanie funkcji *Maksimum* powoduje zwrócenie wyniku jej działania. Przypiszmy go zmiennej *najwiekszy* (rys. 6.):
`najwiekszy = Maksimum(liczba_elementow).`

wywołanie funkcji *Maksimum* z parametrem aktualnym *liczba_elementow* – wynik przypisujemy zmiennej *najwiekszy*

wprowadzenie liczby elementów zbioru

Python

```
liczba_elementow = int(input("Podaj liczbę elementów: "))
najwiekszy = Maksimum(liczba_elementow)
print("Maksimum wynosi:", najwiekszy)
```

Rys. 6. Wywołanie funkcji *Maksimum* – ćwiczenie 5.

Ćwiczenie 5. Wywołujemy funkcję w programie głównym

1. Otwórz program *Maksimum_n_funkcja* zapisany w ćwiczeniu 4.
2. Pod definicją funkcji umieść instrukcje z rysunku 6.
3. Zapisz program w pliku pod tą samą nazwą i ponownie go uruchom. Sprawdź działanie programu dla kilku różnych danych.

3. Stosowanie list do wprowadzania danych

W funkcji *Maksimum* utworzonej w ćwiczeniu 4. wszystkie wprowadzane liczby były pamiętane w jednej zmiennej (np. *a*). Jeśli wprowadzamy z klawiatury liczby ze zbioru {5, 27, 18, 3, 4}, każda kolejno wprowadzona liczba zastępuje poprzednią (temat 8., rys. 6a). W jaki sposób zapamiętać wszystkie wprowadzone dane?

Niektóre algorytmy wymagają pamiętania wszystkich elementów zbioru w trakcie działania programu. Przykładem jest algorytm porządkowania, ponieważ w efekcie jego działania musimy otrzymać wszystkie elementy w określonym porządku.

Aby zapamiętać wszystkie elementy zbioru, stosujemy zmienne indeksowane, w których do nazwy zmiennej (np. *a*) dodaje się kolejny numer (indeks), np.: $a_0, a_1, \ldots, a_n$. Wówczas dla każdej zmiennej jest rezerwowana oddzielna komórka pamięci (temat 8., rys. 6b).

Aby utworzyć zmienne indeksowane w języku Python, możemy zdefiniować specjalną strukturę danych – **listę** (rys. 7.).

Python

```
nazwa_listy = [element1, element2, …, elementn]
```

Rys. 7. Ogólna postać definiowania listy (Python)

Do elementów listy odwołujemy się, podając nazwę listy i indeks elementu umieszczony w nawiasach kwadratowych, np. *a*[0], *a*[1], …, *a*[*n* - 1] – dla listy *n*-elementowej o nazwie *a*. W języku Python pierwszy indeks jest zawsze równy 0. Lista (w odróżnieniu od tablicy) może zawierać elementy różnego typu, dlatego w języku Python nie określamy typu elementów listy. Zarówno typ zmiennych przechowywanych w liście, jak i liczba elementów listy mogą się zmienić w trakcie działania programu.

Wartości elementów listy możemy podać od razu w definicji listy.

Na przykład:

```
L_DNI_W_MIESIACU = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
```

– oznacza zdefiniowanie listy o nazwie *L_DNI_W_MIESIACU* składającej się z dwunastu elementów o indeksach od 0 do 11. Do elementów listy odwołujemy się przez zmienne: *L_DNI_W_MIESIACU*[0], *L_DNI_W_MIESIACU*[1], …, *L_DNI_W_MIESIACU*[11].

Możemy również przypisać od razu taką samą wartość wszystkim elementom listy.

Na przykład:

```
moja_lista = [0] * 10
```

– oznacza zdefiniowanie listy o nazwie *moja_lista* składającej się z dziesięciu elementów, z których każdy jest równy zero (wyzerowanie wszystkich elementów listy).
Do elementów listy odwołujemy się przez zmienne:
moja_lista[0], *moja_lista*[1], …, *moja_lista*[9].

Liczbę elementów możemy też przypisać zmiennej.

Na przykład:

```
N = 100
a = [0] * N
```

– oznacza zdefiniowanie listy o nazwie *a* składającej się ze stu elementów o wartości początkowej zero. Do elementów listy odwołujemy się przez zmienne: *a*[0], *a*[1], …, *a*[99].

Uwaga: Liczbę przykładów definiowania list w języku Python ograniczyliśmy tylko do kilku, przydatnych do dalszych ćwiczeń.

Do wprowadzania danych do listy zastosujemy funkcję bez parametrów i niezwracającą wartości.

Python

```
N = 5
a = [0] * N

def WprowadzDane():
    for i in range(N):
        a[i] = int(input("Podaj liczbę: "))

WprowadzDane()
```

określenie liczby elementów listy

zdefiniowanie listy o nazwie *a* i wyzerowanie jej elementów

nagłówek funkcji bez parametrów

wywołanie funkcji *WprowadzDane*

Rys. 8. Definicja funkcji wprowadzającej do listy dane z klawiatury – ćwiczenie 6.

Ćwiczenie 6. Wprowadzamy dane do listy

1. Napisz funkcję o nazwie *WprowadzDane* wprowadzającą dane do listy *a,* składającej się z pięciu liczb całkowitych (rys. 8.).
2. Zapisz program w pliku pod nazwą *Lista*. Uruchom program.

Wskazówka: Funkcja *WprowadzDane* zmodyfikowała wartości istniejących zmiennych (które na początku były równe zero).

Zapisany w języku Python program można również uruchomić, gdy zamkniemy okno z programem. Należy w tym celu z menu kontekstowego ikony programu (widocznej w Eksploratorze plików) wybrać polecenie:

- **Otwórz** – w otwartym oknie zobaczymy uruchomiony program,
- **Edit with IDLE** – program otworzy się w oknie powłoki Pythona (Python Shell) i będzie można go uruchomić.

Python

```
def WyprowadzDane():
    for i in range(N):
        print(a[i])
```

Rys. 9. Definicja funkcji wyprowadzającej dane z listy na ekran – ćwiczenie 7.

Ćwiczenie 7. Wyprowadzamy dane z listy na ekran

1. Otwórz program *Lista* zapisany w ćwiczeniu 6.
2. Zdefiniuj funkcję o nazwie *WyprowadzDane* wyprowadzającą w kolejnych wierszach dane z listy na ekran (rys. 9.).
3. Po poleceniu wywołania funkcji *WprowadzDane* dodaj wywołanie funkcji *WyprowadzDane*.
4. Zapisz program w pliku pod tą samą nazwą. Uruchom program i sprawdź jego działanie dla różnych danych.

4. Zapisywanie w języku Python wybranych algorytmów porządkowania i wyszukiwania

Zamierzamy wykorzystać możliwość zapamiętania danych na liście do zapisania w języku programowania algorytmów porządkowania (metodą przez wybieranie i metodą przez zliczanie) oraz algorytmu wyszukiwania elementu w zbiorze uporządkowanym. Jak to zrobić w języku Python?

Algorytm porządkowania metodą przez wybieranie

Zapiszemy w języku Python algorytm porządkowania liczb od największej do najmniejszej metodą przez wybieranie. W programie wykorzystamy utworzone wcześniej dwie funkcje *WprowadzDane* do wprowadzenia danych do listy i *WyprowadzDane* – do wyprowadzenia danych na ekran po uporządkowaniu.

W programie wyszukujemy na liście danych *a* największą liczbę *maks* i zapamiętujemy jej pozycję (indeks) w zmiennej *maksi*. Następnie zamieniamy miejscami wyszukaną liczbę z liczbą, która jest aktualnie na początku zbioru. Do zamiany stosujemy zmienną pomocniczą *t*. Wymienione czynności powtarzamy z pominięciem liczb już uporządkowanych.

Zdefiniujemy dwie funkcje:

- *MaksWybor* (z jednym parametrem *starti*) wyszukującą największy element (*maks*) w zbiorze, począwszy od elementu o indeksie *starti*, i zapamiętującą jego pozycję w zmiennej *maksi* (zwracaną na końcu funkcji) – rys. 10.,
- *SortWybor* (niezwracającą wartości i bez parametrów), w której wywołujemy w pętli funkcję *MaksWybor* z parametrem aktualnym *i* oraz zamieniamy miejscami znaleziony element największy *maks* z aktualnie pierwszym elementem zbioru – rys. 11. Po każdym przejściu pętli liczba elementów zbioru jest o 1 mniejsza, ponieważ nie sprawdzamy już elementów ustawionych na właściwych miejscach (uporządkowanych).

Python

zmiennej *maksi* przypisujemy numer aktualnie pierwszego elementu nieuporządkowanego zbioru

do początkowej wartości zmiennej sterującej *starti* dodajemy 1, ponieważ wartość pierwszego elementu zbioru już wprowadziliśmy do zmiennej *maks*

w pętli szukamy kolejnych liczb większych od aktualnie największej i zapamiętujemy pozycję największej liczby na liście: `maksi = i`; polecenia w pętli powtarzamy *n* – (*starti* + 1) razy

```
def MaksWybor(starti):
    maksi = starti
    maks = a[maksi]
    for i in range(starti + 1, N):
        if a[i] > maks:
            maksi = i
            maks = a[maksi]
    return maksi
```

funkcja zwraca indeks największego elementu

Rys. 10. Definicja funkcji wyszukującej największy element i zwracającej jego pozycję na liście – ćwiczenie 8.

Python

```
def SortWybor():
    for i in range(N-1):
        maksi = MaksWybor(i)
        t = a[i]
        a[i] = a[maksi]
        a[maksi] = t
```

wywołujemy funkcję *MaksWybor* z parametrem aktualnym *i*, a wynik przypisujemy zmiennej *maksi*

zmienną pomocniczą *t* stosujemy do zamiany miejscami znalezionego największego elementu z aktualnie pierwszym

Rys. 11. Definicja funkcji porządkującej elementy zbioru metodą przez wybieranie – ćwiczenie 8.

Python

```
WprowadzDane()
SortWybor()
print("Dane posortowane metodą przez wybieranie")
WyprowadzDane()
```

Rys. 12. Wywołanie w programie głównym funkcji wprowadzania danych, porządkowania i wyprowadzania danych na ekran – ćwiczenie 8.

W programie głównym wywołamy trzy funkcje: wprowadzania danych (*WprowadzDane*), porządkowania (*SortWybor*) i wyprowadzania danych na ekran (*WyprowadzDane*) – rys. 12.

Ćwiczenie 8. Zapisujemy w języku Python algorytm porządkowania metodą przez wybieranie

1. Przypomnij sobie działanie algorytmu porządkowania metodą przez wybieranie.
2. Otwórz program *Lista* zapisany w ćwiczeniu 7. Uzupełnij program, dodając definicję funkcji *MaksWybor* i *SortWybor* oraz wywołanie funkcji *SortWybor* w programie głównym.
3. Zapisz program w pliku pod nazwą *Wybieranie*. Uruchom program. Sprawdź działanie programu dla różnych danych (elementów listy).

Algorytm porządkowania metodą przez zliczanie

Zapiszemy w języku Python algorytm porządkowania liczb od najmniejszej do największej metodą przez zliczanie. W programie wykorzystamy utworzone wcześniej dwie funkcje: *WprowdzDane* do wprowadzenia danych do listy i *WyprowadzDane* – do wyprowadzenia danych na ekran po uporządkowaniu. Funkcję *WprowadzDane* musimy zmodyfikować, dodając sprawdzanie, czy liczba należy do określonego zakresu. Ustalimy zakres <0, *MAX_DANA*>. Zmiennej *MAX_DANA* przypiszemy stałą wartość, np. 99.

Zdefiniujemy funkcję *SortZlicz* (niezwracającą wartości i bez parametrów), w której zdefiniujemy listę *liczniki* [*MAX_DANA* +1], służącą do zliczania wystąpień elementów zbioru. Liczba liczników, zgodnie z algorytmem, musi umożliwić zliczenie wystąpień każdej liczby z danego zakresu, więc przygotowujemy 100 liczników (dla liczb od 0 do 99).

Python

Zliczanie.py

File Edit Format Run Options Window Help

```
N = 5
a = [0] * N
MAX_DANA = 99

def WprowadzDane():
    print("Wprowadzaj liczby z przedziału < 0,", MAX_DANA, ">")
    for i in range(N):
        a[i] = int(input("Podaj liczbę: "))
        if a[i] < 0:
            a[i] = 0
        if a[i] > MAX_DANA:
            a[i] = MAX_DANA
```

Ln: 14 Col: 0

Rys. 13. Definicja funkcji wprowadzającej dane z określonego zakresu do listy – ćwiczenie 9.

W pierwszej pętli **`for`** (rys. 15., wiersze 3-4) dla każdej liczby wprowadzonej przez użytkownika zwiększamy odpowiadający jej licznik o 1.

Na rysunku 14. pokazano przykładową listę *liczniki* dla liczb {0, 2, 3, 5, 5}.

1	0	1	1	0	2
`liczniki [0]`	`liczniki [1]`	`liczniki [2]`	`liczniki [3]`	`liczniki [4]`	`liczniki [5]`

Rys. 14. Lista *liczniki* dla liczb {0, 2, 3, 5, 5}

W pętli **`for`** (rys. 15., wiersze 7-9) przeglądamy listę *liczniki* i wpisujemy do listy z danymi (*a*) daną wartość tyle razy, ile razy pojawiła się w zbiorze (liczbie wystąpień odpowiada element listy *liczniki*).

Pętlę z wierszy 7-9 komputer wykonuje dla każdej możliwej wartości danych wejściowych (rys. 15., pętla **`for`** – wiersz 6.). Mamy tu zatem do czynienia z **zagnieżdżeniem pętli.** Zmienna *poz* określa aktualną pozycję na liście *a*.

Python

```
1 def SortZlicz():
2     liczniki = [0] * (MAX_DANA +1)
3     for i in range(N):
4         liczniki[a[i]] = liczniki[a[i]] + 1
5     poz = 0
6     for i in range(MAX_DANA+1):
7         for j in range(liczniki[i]):
8             a[poz] = i
9             poz = poz + 1
```

definiujemy i równocześnie zerujemy listę liczników

dla każdej wprowadzonej liczby zwiększamy odpowiadający jej licznik o 1

układamy elementy listy *a* od najmniejszego do największego

Rys. 15. Definicja funkcji porządkującej elementy zbioru metodą przez zliczanie – ćwiczenie 9.

Python

```
WprowadzDane()
WyprowadzDane()
print("Dane posortowane metodą przez zliczanie")
SortZlicz()
WyprowadzDane()
```

Rys. 16. Wywołanie w programie głównym funkcji wprowadzania danych, porządkowania metodą przez zliczanie i wyprowadzania wyniku na ekran – ćwiczenie 9.

W programie głównym wywołamy trzy funkcje: wprowadzania danych (*WprowadzDane*), porządkowania (*SortZlicz*) i wyprowadzania danych na ekran (*WyprowadzDane*) – rys. 16.

Ćwiczenie 9. Zapisujemy w języku Python algorytm porządkowania metodą przez zliczanie

1. Przypomnij sobie działanie algorytmu porządkowania metodą przez zliczanie.
2. Otwórz program *Lista* zapisany w ćwiczeniu 7. Zmodyfikuj funkcję *WprowadzDane* tak, aby dodatkowo sprawdzała, czy liczba należy do określonego zakresu (rys. 13.).
3. Uzupełnij program, dodając definicję funkcji *SortZlicz* (rys. 15.) oraz wywołanie funkcji *SortZlicz* w programie głównym (rys. 16.).
4. Zapisz program w pliku pod nazwą *Zliczanie* i ponownie go uruchom. Sprawdź działanie programu dla różnych danych.

Algorytm wyszukiwania elementu w zbiorze uporządkowanym

Program ma wyszukiwać w zbiorze uporządkowanym daną podaną przez użytkownika i podawać jej pozycję na liście. Zakładamy, że wprowadzamy dane od najmniejszej do największej (inaczej program nie będzie działał poprawnie). Bez takiego założenia należałoby dodać również funkcję sortowania danych w określonym porządku. W programie wykorzystamy utworzoną wcześniej funkcję *WprowadzDane* (rys. 8.) do wprowadzenia danych do listy.

Zdefiniujemy funkcję *ZnajdzDana* z parametrem *wartosc* (rys. 17.). W funkcji wyznaczamy pozycję elementu środkowego (zmienna *srodek*).

Jeśli element środkowy (*a*[*srodek*]) jest szukaną daną (zmienna *wartosc*), to funkcja zwraca zmienną *srodek*, czyli pozycję danego elementu.

Jeśli szukana dana jest mniejsza od danej środkowej, odrzucamy prawą część zbioru (liczby większe od elementu środkowego) – ustalamy nowy koniec (`koniec = srodek - 1`).

W przeciwnym wypadku odrzucamy lewą część zbioru (liczby mniejsze od elementu środkowego) i ustalamy nowy początek (`poczatek = srodek + 1`).

Połowienie powtarzamy dopóki początek zbioru jest mniejszy lub równy końcowi zbioru (`poczatek <= koniec`) – o ile nie znajdziemy szukanego elementu (wtedy komputer wykonana instrukcję **`return`** `srodek`).

W algorytmie liczba iteracji zależy od spełnienia (lub niespełnienia) warunku, dlatego zamiast instrukcji iteracyjnej **`for`**, w której z góry określamy liczbę powtórzeń, użyjemy instrukcji iteracyjnej **`while`**, którą omówiliśmy w temacie 6.

Python

```
def ZnajdzDana(wartosc):
    poczatek = 0
    koniec = N - 1
    while poczatek <= koniec:
        srodek = (poczatek + koniec) // 2
        if a[srodek] == wartosc:
            return srodek
        else:
            if wartosc < a[srodek]:
                koniec = srodek - 1
            else:
                poczatek = srodek + 1
    return -1
```

wyznaczamy pozycję elementu środkowego; operator // oznacza dzielenie z obcięciem części ułamkowej

sprawdzamy, czy element środkowy jest szukaną liczbą

wyznaczamy nowy koniec lub początek

Rys. 17. Definicja funkcji wyszukującej daną w zbiorze uporządkowanym – ćwiczenie 10.

W funkcji *ZnajdzDana* (rys. 17.) instrukcja **`return`** występuje dwukrotnie. Niezależnie od miejsca, w którym występuje, zawsze kończy wykonanie funkcji. Jeśli element środkowy jest szukaną liczbą, funkcja zwraca wartość zmiennej *srodek* (**`return`** `srodek`).

Jeśli w przeszukiwanym zbiorze nie ma szukanej liczby, funkcja *ZnajdzDana* (rys. 17.) zwraca wartość -1 (**`return`** `-1`). Programiści często używają wartości niepoprawnych (w tym przypadku indeks listy nie może być liczbą ujemną) do zasygnalizowania niepowodzenia operacji lub błędu.

W programie głównym wywołamy funkcję wprowadzania danych (*WprowadzDane*), wprowadzimy z klawiatury daną do wyszukania (zmienna *wartosc*), wywołamy funkcję wyszukiwania danej (*ZnajdzDana*) i wypiszemy wynik. Uwzględnimy również sytuację, gdy podanej wartości nie ma w zbiorze (rys. 18.).

Python

```
WprowadzDane()
wartosc = int(input("Podaj daną do wyszukania: "))
pozycja = ZnajdzDana(wartosc)
if pozycja >= 0:
    print("Znaleziono wartość", wartosc, "na pozycji", pozycja)
    print("Pozycje liczone są od zera")
else:
    print("Nie znaleziono wartości", wartosc)
```

Rys. 18. Wywołanie w programie głównym funkcji wprowadzania danych i wyszukiwania określonej danej oraz wyprowadzenie wyniku na ekran – ćwiczenie 10.

Ćwiczenie 10. Zapisujemy w języku Python algorytm wyszukiwania elementu w zbiorze uporządkowanym

1. Przypomnij sobie działanie algorytmu wyszukiwania elementu w zbiorze uporządkowanym.
2. Otwórz program *Lista* zapisany w ćwiczeniu 7. Usuń funkcję *WyprowadzDane*.
3. Uzupełnij program, dodając definicję funkcji *ZnajdzDana* (rys. 17.). W programie głównym wywołaj funkcję *ZnajdzDana* i dodaj instrukcje wprowadzania szukanej danej i wyświetlania wyniku (rys. 18.).
4. Zapisz program w pliku pod nazwą *Wyszukiwanie*. Uruchom program. Sprawdź działanie programu dla różnych danych (pamiętaj o wprowadzaniu danych uporządkowanych – od najmniejszej do największej).

Warto zapamiętać

- Aby pisać przejrzyste programy, należy opracowywać oddzielnie problemy cząstkowe, stosując podprogramy (procedury i funkcje). Podprogram należy najpierw zdefiniować, a potem wywołać w programie głównym.
- W języku Python możemy zdefiniować funkcje:
  - niezwracające wartości, gdy chcemy np. wyprowadzić dane,
  - zwracające wartość, gdy celem funkcji jest obliczenie i zwrócenie pewnej wartości, np. wyniku sumowania.
- Parametr to wartość przekazywana do funkcji, a wartość zwracana to wartość przekazywana z funkcji.
- Możemy definiować funkcje z parametrami i bez parametrów. W momencie wywołania funkcji parametr formalny przyjmie wartość parametru aktualnego.
- Aby wywołać funkcję z parametrami, należy podać jej nazwę i parametry aktualne w nawiasach okrągłych. Gdy parametry nie występują, nawiasy pozostawiamy puste.
- Program główny komunikuje się z funkcją poprzez przekazywanie parametrów.
- W funkcji zwracającej wartość musi wystąpić instrukcja `return` z argumentem przedstawiającym zwracaną wartość. Wartość zwracaną przez funkcję (reprezentowaną przez nazwę funkcji i parametry umieszczone w nawiasach okrągłych) przypisujemy zmiennej.
- Aby zaprogramować algorytm, w którym w trakcie działania programu musimy pamiętać wszystkie elementy zbioru, możemy zapisywać te dane na liście.
- Do elementów listy (np. o nazwie *lista* i liczbie elementów *N*) odwołujemy się poprzez podanie nazwy listy i indeksu umieszczonego w nawiasach kwadratowych *lista*[0], *lista*[1], …, *lista*[N - 1].
- W języku Python nie określamy typu elementów listy. Interpreter określa typ elementów na podstawie rodzaju danych, jaki im przypiszemy.
- Definiowanie funkcji i stosowanie list możemy wykorzystać do zapisania w języku Python algorytmów porządkowania metodą przez wybieranie, porządkowania metodą przez zliczanie oraz wyszukiwania elementu w zbiorze uporządkowanym.

Pytania i polecenia

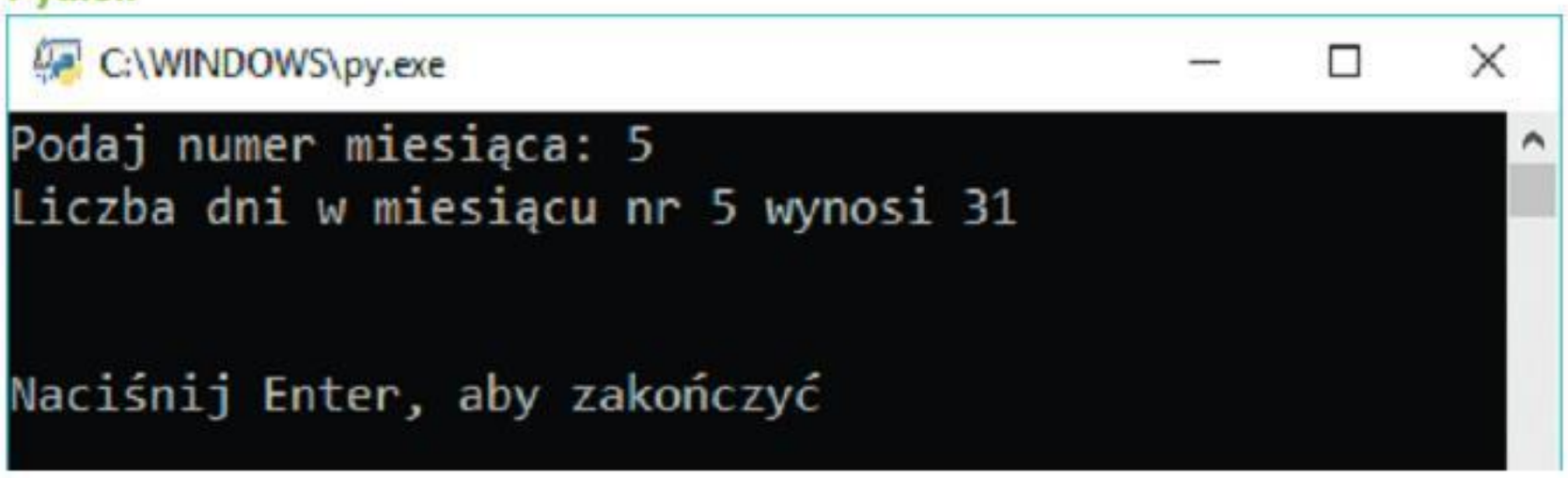
1. Dlaczego definiujemy podprogramy?
2. Czym jest parametr formalny, a czym aktualny w podprogramie (procedurze, funkcji)?
3. Kiedy definiujemy funkcję niezwracającą wartości, a kiedy zwracającą wartość w języku Python? Wyjaśnij na przykładzie.
4. W jaki sposób wywołujemy w języku Python funkcję bez parametrów, a w jaki – z parametrami?
5. W jakim celu definiujemy listę? Podaj przykład użycia listy.
6. Czym jest zmienna indeksowana? Wyjaśnij na przykładach sposób definiowania listy w języku Python.
7. Wyjaśnij sposób zapisania w języku Python algorytmu wyszukiwania metodą przez wybieranie. Do czego służą funkcje zdefiniowane w tym programie?
8. Dlaczego w programie realizującym algorytm porządkowania liczb od największej do najmniejszej metodą przez wybieranie musimy zapamiętywać pozycję znalezionej największej liczby w oddzielnej zmiennej?
9. Wyjaśnij sposób zapisania w języku Python algorytmu wyszukiwania metodą przez zliczanie. Do czego służą funkcje zdefiniowane w tym programie?

10. W jakim celu definiujemy listę *liczniki* w programie realizującym algorytm porządkowania metodą przez zliczanie?
11. Wyjaśnij sposób zapisania w języku Python algorytmu wyszukiwania elementu w zbiorze uporządkowanym. Do czego służą funkcje zdefiniowane w tym programie?
12. Dlaczego program realizujący algorytm wyszukiwania elementu w zbiorze uporządkowanym zaczyna się od sprawdzenia, czy środkowy element jest szukaną liczbą?

Zadania

1. W języku Scratch zdefiniuj dwie procedury: *Pięciokąt* z parametrem *bok_p* rysującą pięciokąt foremny i *Siedmiokąt* z parametrem *bok_s* rysującą siedmiokąt foremny, gdzie *bok_p* i *bok_s* określają odpowiednio długości boków pięciokąta i siedmiokąta. Korzystając z tych procedur, utwórz kompozycję z pięciokątów i siedmiokątów według własnego pomysłu. Wartości parametrów aktualnych (*p* i *s*) dla zdefiniowanych procedur podawaj z klawiatury po uruchomieniu programu. Zapisz program w pliku pod nazwą *Pięciokąty i siedmiokąty*.
2. Napisz w języku Python program umożliwiający wyprowadzenie na ekran monitora napisu „Gwiazdkowe pozdrowienia", a pod nim piętnastu gwiazdek (znaków *). Zdefiniuj funkcję *Gwiazdki* niezwracającą wartości i bez parametrów, wyprowadzającą w jednym wierszu piętnaście gwiazdek. Funkcję wywołaj w programie głównym. Zapisz program w pliku pod nazwą *Życzenia*.
3. Zmodyfikuj program *Życzenia* zapisany w zadaniu 2. tak, aby wyświetlił w kolejnym wierszu napis „Wszystkiego najlepszego", a pod nim ponownie piętnaście gwiazdek. Zapisz program w pliku pod tą samą nazwą.
4. Napisz w języku Python program umożliwiający obliczenie objętości sześcianu. Zdefiniuj funkcję *Szescian* z jednym parametrem o nazwie *b*, obliczającą sześcian liczby *b* i zwracającą do programu głównego wynik obliczenia. Wywołaj funkcję w programie głównym z parametrem aktualnym *bok*. Wartość parametru wprowadzaj z klawiatury. Zapisz program w pliku pod nazwą *Objętość sześcianu*.
5. Korzystając ze zdefiniowanej w ćwiczeniu 4. funkcji *Maksimum*, zdefiniuj funkcję *Minimum* i napisz program wyszukujący najmniejszy element w zbiorze *n*-elementowym. Zapisz program w pliku pod nazwą *Min_n_funkcja*.
6. Zdefiniuj listę składającą się z dwunastu elementów, będących liczbami dni kolejnych miesięcy (dla lutego przyjmij 28 dni). Utwórz program wypisujący w kolumnie elementy listy, czyli liczby dni kolejnych miesięcy. Zapisz program w pliku pod nazwą *Miesiące*.
   Wskazówka: Wykorzystaj przykład listy *L_DNI_W_MIESIACU* podanej w punkcie 3. tematu.
7. Korzystając z listy zdefiniowanej w zadaniu 6., utwórz program, który dla podanego z klawiatury numeru miesiąca wyświetli liczbę dni tego miesiąca. Zapisz program w pliku pod nazwą *Miesiąc*. Efekt wykonania programu powinien być taki, jak pokazano na rysunku 18.
8. Napisz w języku Python program realizujący algorytm wyszukiwania danego elementu w zbiorze nieuporządkowanym, korzystając z opisu podanego w temacie 7., w punkcie 1.3. Do wprowadzenia danych wykorzystaj funkcję *WprowadzDane()*, a do wyprowadzenia – funkcję *WyprowadzDane()*. Do wyszukania danej zastosuj funkcję *WyszukajDana()*, której wzór pokazano na rysunku 20. Wszystkie funkcje wywołaj w programie głównym. Szukaną wartość wprowadzaj z klawiatury. Zapisz program w pliku pod nazwą *Wyszukiwanie_nieuporządkowany*.

Python



Rys. 19. Efekt wykonania programu – zadanie 7.

Python

```
def WyszukajDana(wartosc):
    for i in range(N):
        if a[i] == wartosc:
            return i
    return -1
```

Rys. 20. Definicja funkcji wyszukującej dany element w zbiorze nieuporządkowanym i zwracającej pozycję elementu na liście – zadanie 8.

Dla zainteresowanych

9. Zmodyfikuj program zapisany w zadaniu 3. tak, aby liczba gwiazdek była parametrem *n* funkcji *Gwiazdki*. Wartość parametru aktualnego *liczba_gwiazdek* wprowadzaj w momencie wywołania funkcji w programie głównym. Zapisz program w pliku pod nazwą *Gwiazdki_n*.
10. Do programu zapisanego w zadaniu 4. dodaj funkcję *Kwadrat* z jednym parametrem o nazwie *b*, obliczającą kwadrat liczby *b* i zwracającą do programu głównego wynik obliczenia. Wywołaj funkcję w programie głównym z parametrem aktualnym *bok*, którego wartość wprowadzaj z klawiatury. Zapisz program w pliku pod nazwą *Objętość sześcianu i pole kwadratu*.
11. Otwórz program *Maks_n_funkcja* zapisany w ćwiczeniu 4. Dodaj wprowadzanie danych z wykorzystaniem listy. Zmodyfikuj funkcję *Maksimum* tak, aby zastosować zmienne indeksowane. Zapisz program w pliku pod nazwą *Maksimum_lista*.
12. Otwórz plik *Wybieranie* zapisany w ćwiczeniu 7. Zmodyfikuj program tak, aby porządkował wprowadzone dane od największej do najmniejszej.
13. Otwórz plik *Zliczanie* zapisany w ćwiczeniu 8. Zmodyfikuj program tak, aby porządkował wprowadzone dane od największej do najmniejszej.
14. Otwórz program *Wyszukiwanie* zapisany w ćwiczeniu 9. Chcielibyśmy wprowadzać również dane nieuporządkowane. Dodaj do programu funkcję porządkowania danych od najmniejszej do największej metodą przez wybieranie. Wywołaj ją w programie głównym. Zapisz program w pliku pod tą samą nazwą.
15. Znajdź w Internecie informacje na temat funkcji zwracającej wiele wartości w języku Python.
16. Korzystając z **Pomocy** do programu, dowiedz się, w jaki sposób utworzyć listę w języku Scratch. Następnie wykonaj w języku Scratch zadania 6. i 7.
    Wskazówka: Polecenia i elementy związane z listą znajdują się w grupie **Zmienne**.

Projekty multimedialne

III

Rozdział

Temat 10. Prezentacja multimedialna, modele 3D, druk 3D i obróbka filmów – projekt łączący poznane technologie

Temat 11. Historia i rozwój informatyki – projekt

Temat 10. Prezentacja multimedialna, modele 3D, druk 3D i obróbka filmów – projekt łączący poznane technologie

1. **Prezentacja multimedialna z modelami 3D i filmami**
   1.1. **Przygotowanie planu prezentacji i slajdu tytułowego**
   1.2. **Wstawianie obrazów, filmów i modeli 3D**
   1.3. **Wstawianie hiperłączy, wykresów i tabel**
   1.4. **Dodawanie animacji i przejść slajdów**
2. **Podstawowy montaż filmu**
3. **Projektowanie i druk 3D pomocy dydaktycznych**

Warto powtórzyć

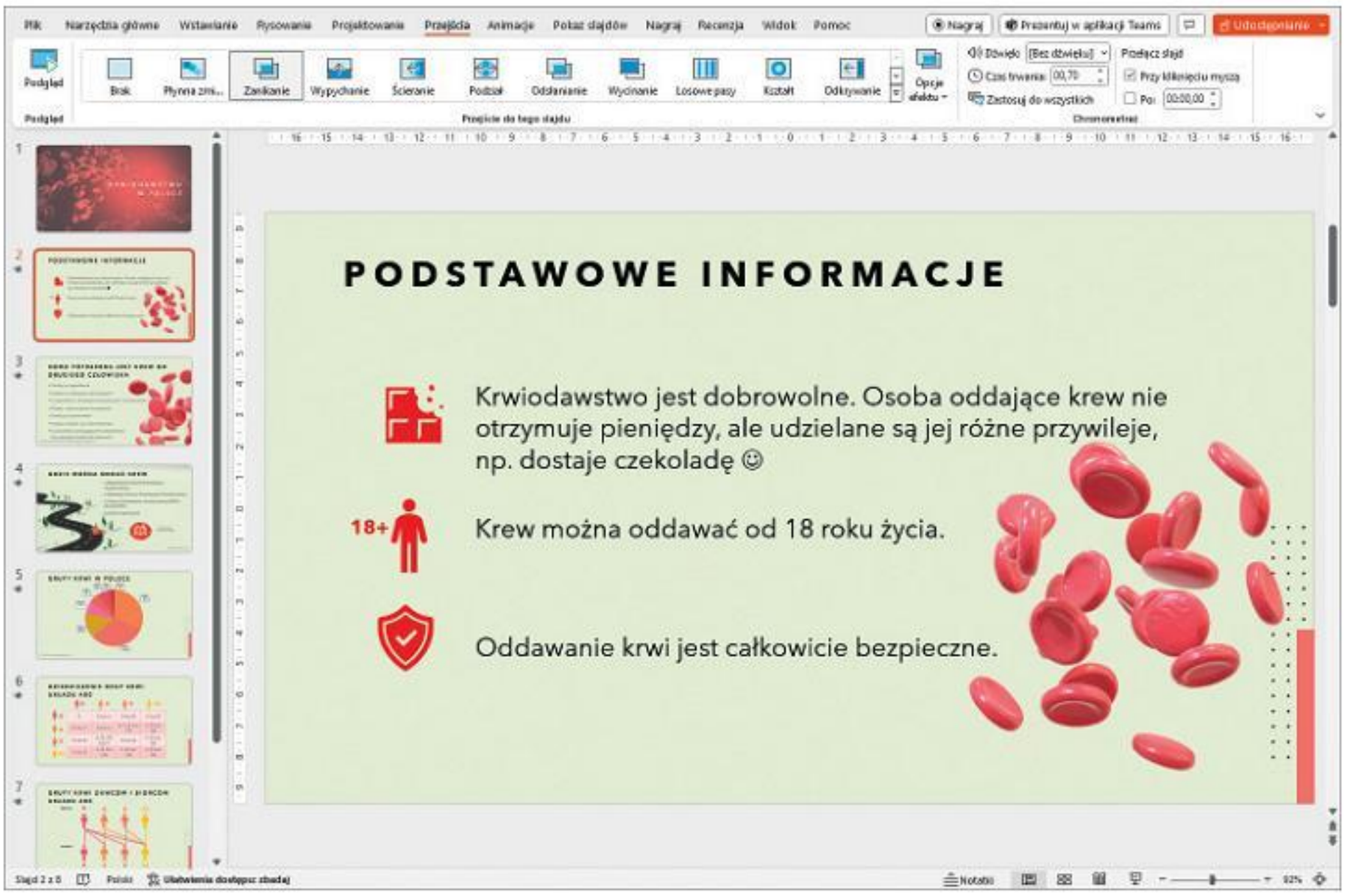
1. Jakie znasz zasady redagowania tekstu?
2. Czym są hipertekst i hiperłącze?
3. Jakie znasz programy umożliwiające tworzenie prezentacji multimedialnych?
4. W jaki sposób tworzy się wykres w programie Microsoft Excel lub LibreOffice Calc?
5. W jaki sposób w programie Tinkercad obraca się kształt 3D i zmienia jego rozmiar?
6. Jak wydrukować projekt 3D na drukarce 3D?

1. Prezentacja multimedialna z modelami 3D i filmami



Mamy przybliżyć osobom z klasy problem krwiodawstwa w Polsce. Jak wykonać prezentację na ten temat, aby zaciekawić słuchaczy?

Aby prezentacja mogła spełnić swoje zadanie, powinniśmy ją wykonać zgodnie z określonymi regułami.



Rys. 1. Przykładowy slajd prezentacji w oknie programu Microsoft PowerPoint. Po lewej stronie okna widać ponumerowane kolejno slajdy prezentacji

Cechy dobrej prezentacji

1. Jednolita oprawa graficzna – wszystkie slajdy prezentacji powinno się tworzyć w tym samym stylu i w jednolitej kolorystyce.
2. Umiarkowana ilość informacji – na jednym slajdzie nie należy wprowadzać zbyt wielu informacji (zwłaszcza tekstu). Na slajdach można umieszczać tekst, a także inne elementy, np. tabele, schematy, obrazy.
3. Poprawnie zredagowany tekst na slajdach.
4. Poprawnie dobrane parametry czcionek, m.in. odpowiedni rozmiar czcionki, krój (najlepiej bezszeryfowy) – aby tekst był czytelny i dobrze widoczny dla widzów.
5. Kolor i rodzaj tła dobrane tak, aby tekst był dobrze widoczny, np. ciemne litery na jasnym tle lub jasne litery na ciemnym tle.
6. Umiarkowane stosowanie animacji – nadmierna różnorodność animacji, zwłaszcza na jednym slajdzie, może sprawiać wrażenie chaosu.
7. Tempo wyświetlania slajdów powinno umożliwiać swobodne zapoznanie się z ich treścią.
8. Liczba slajdów wystarczająca do zaprezentowania danego tematu, jednak nie nadmiernie duża – długa prezentacja może znużyć widzów.

Ćwiczenie 1. Oglądamy gotową prezentację

Przyjrzyj się slajdowi prezentacji na rysunku 1. Zwróć uwagę na układ i dobór treści, obrazów, kolory tła oraz parametry czcionek.

1.1. Przygotowanie planu prezentacji i slajdu tytułowego

Zanim zaczniemy tworzyć prezentację, ustalamy zakres materiału, jaki zamierzamy przedstawić. Następnie planujemy, jak rozmieścić informacje na slajdach.

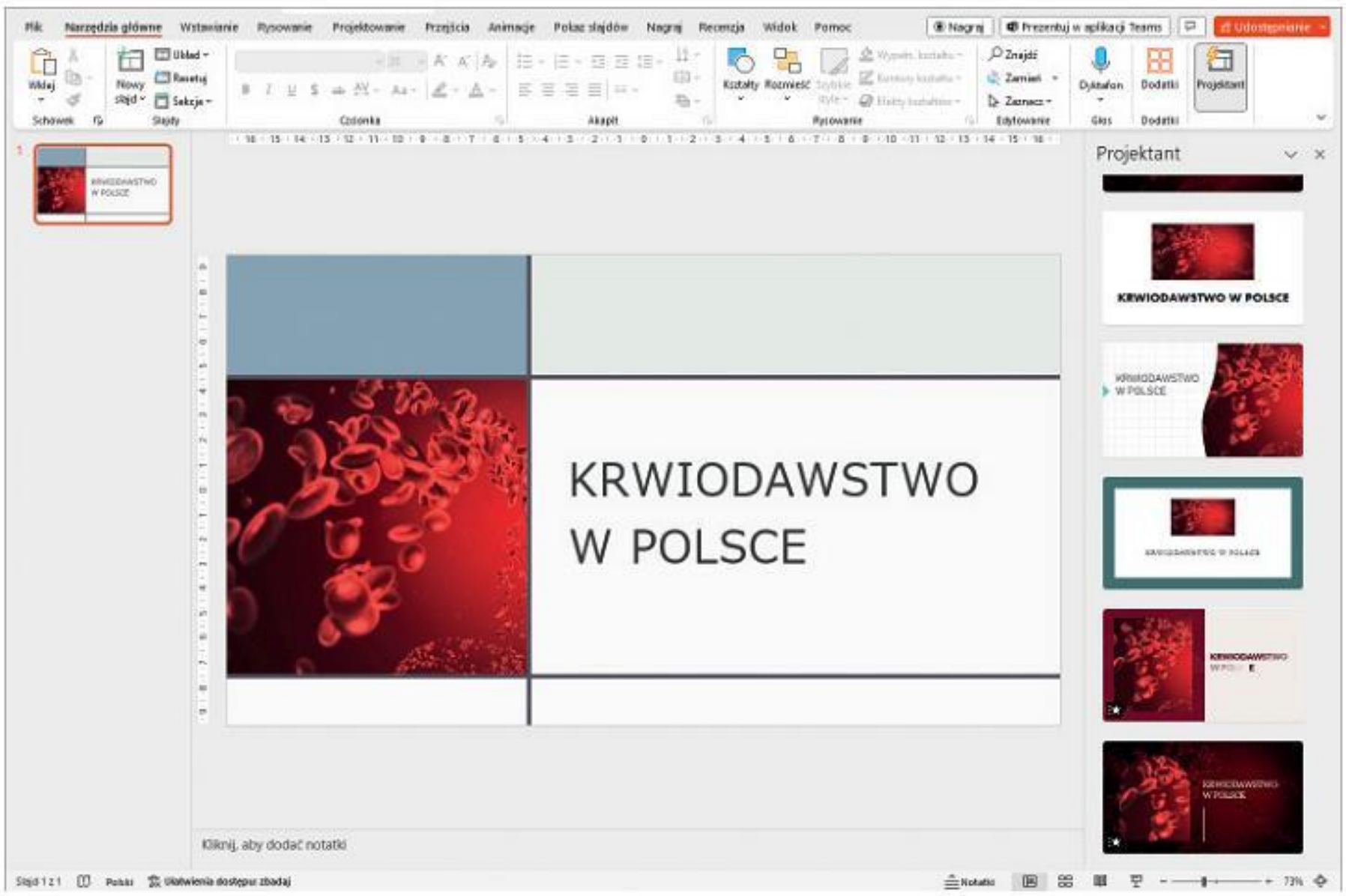
Ćwiczenie 2. Przygotowujemy plan prezentacji

Podaj w punktach zagadnienia, które przedstawisz w prezentacji na temat krwiodawstwa w Polsce. Do przygotowania zagadnień wykorzystaj informacje wyszukane w Internecie oraz wiedzę z innych przedmiotów, np. z biologii. Przykładowe zagadnienia do prezentacji:

1. *Podstawowe informacje.*
2. *Komu potrzebna jest krew od drugiego człowieka?*
3. *Gdzie można oddać krew?*
4. *Najczęściej i najrzadziej występujące grupy krwi w Polsce.*
5. *… itd.*

Prezentacja wygląda profesjonalnie, gdy slajdy mają podobne lub jednakowe tło albo zbliżoną tonację kolorystyczną. Programy do tworzenia prezentacji multimedialnych wyposażono w gotowe szablony (nazywane również motywami lub wzorcami stron). Można je zastosować do utworzenia nowej prezentacji. Możliwość tworzenia dokumentów na podstawie gotowych szablonów znamy z edytora tekstu.

Interesujący tytuł prezentacji może już na samym początku zachęcić oglądających do skupienia uwagi. W dobrym tonie jest podanie na slajdzie tytułowym lub końcowym imienia, nazwiska oraz np. adresu e-mailowego autora prezentacji. Na slajdzie tytułowym można dodać tło inne niż na pozostałych slajdach lub wstawić obraz dostosowany tematycznie do prezentacji.



Rys. 2. Przykładowy slajd tytułowy prezentacji z otwartym okienkiem **Projektant** w programie Microsoft PowerPoint – ćwiczenie 3.

W programie Microsoft PowerPoint pomocna w projektowaniu wyglądu slajdów jest opcja **Projektant**. Wybranie na karcie **Projektowanie** tej opcji otwiera okienko **Projektant**, w którym po dodaniu np. tytułu slajdu i wstawieniu obrazu program wyświetla pomysły rozmieszczenia tych elementów na slajdzie (rys. 2.). Zaakceptowany slajd możemy nadal edytować, np. zmienić tło, przesuwać elementy.

Ćwiczenie 3. Wybieramy motyw i przygotowujemy slajd tytułowy

1. Wybierz motyw (szablon) prezentacji.
2. Przygotuj slajd tytułowy prezentacji, wpisując jego tytuł i dobierając odpowiednią grafikę. Jeśli program, z którego korzystasz, ma taką możliwość, sprawdź, jakie propozycje slajdu tytułowego znajdują się w okienku **Projektant** i wybierz jedną z nich.
3. Zapisz prezentację pod nazwą *Krwiodawstwo*.

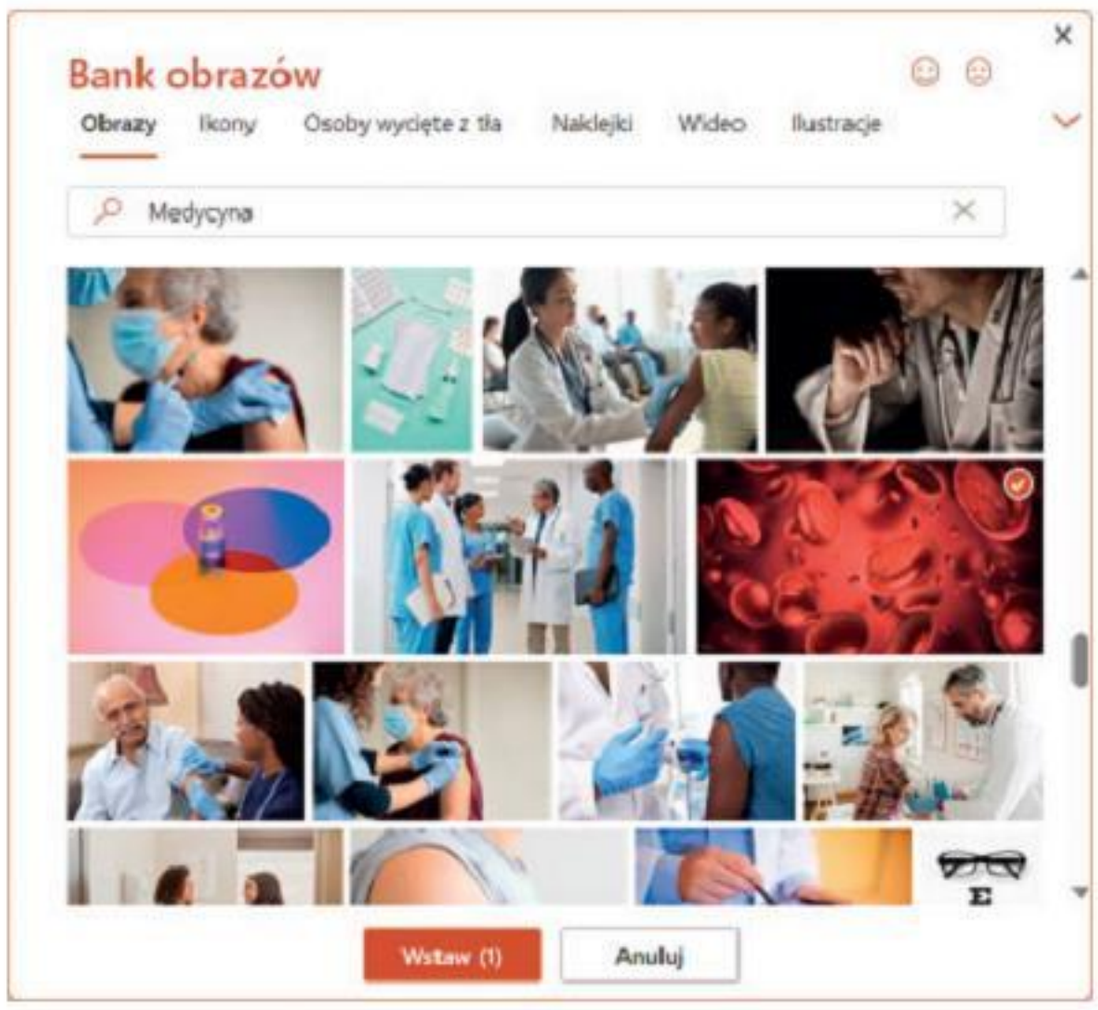
1.2. Wstawianie obrazów, filmów i modeli 3D

Aby prezentacja była interesująca, można ją urozmaicić, wstawiając obok tekstu obrazy, zdjęcia, ikony, ewentualnie dźwięki i filmy. Obrazy powinny ilustrować treść slajdu, mogą również pełnić funkcję ozdobnika lub tworzyć tło.

Obrazy (rysunki, zdjęcia) możemy wykonać samodzielnie lub skorzystać z ilustracji dostępnych w programie. Możemy również pobrać grafikę z Internetu i wstawić do prezentacji – z zachowaniem przepisów prawa dotyczących korzystania z cudzych materiałów. W programie Microsoft PowerPoint możemy posłużyć się gotowymi grafikami, takimi jak zdjęcia, ikony czy wideo z banku obrazów (rys. 3.), korzystając z opcji **Obrazy/Bank obrazów** na karcie **Wstawianie**.

Gdy na jednym slajdzie umieszczamy dużo tekstu, stosujemy czasem czcionkę o zbyt małym rozmiarze – tekst staje się nieczytelny, a widzowie szybko się nużą taką prezentacją. Nie przesadzajmy z ilością tekstu, jak również z liczbą obrazów umieszczanych na slajdzie.

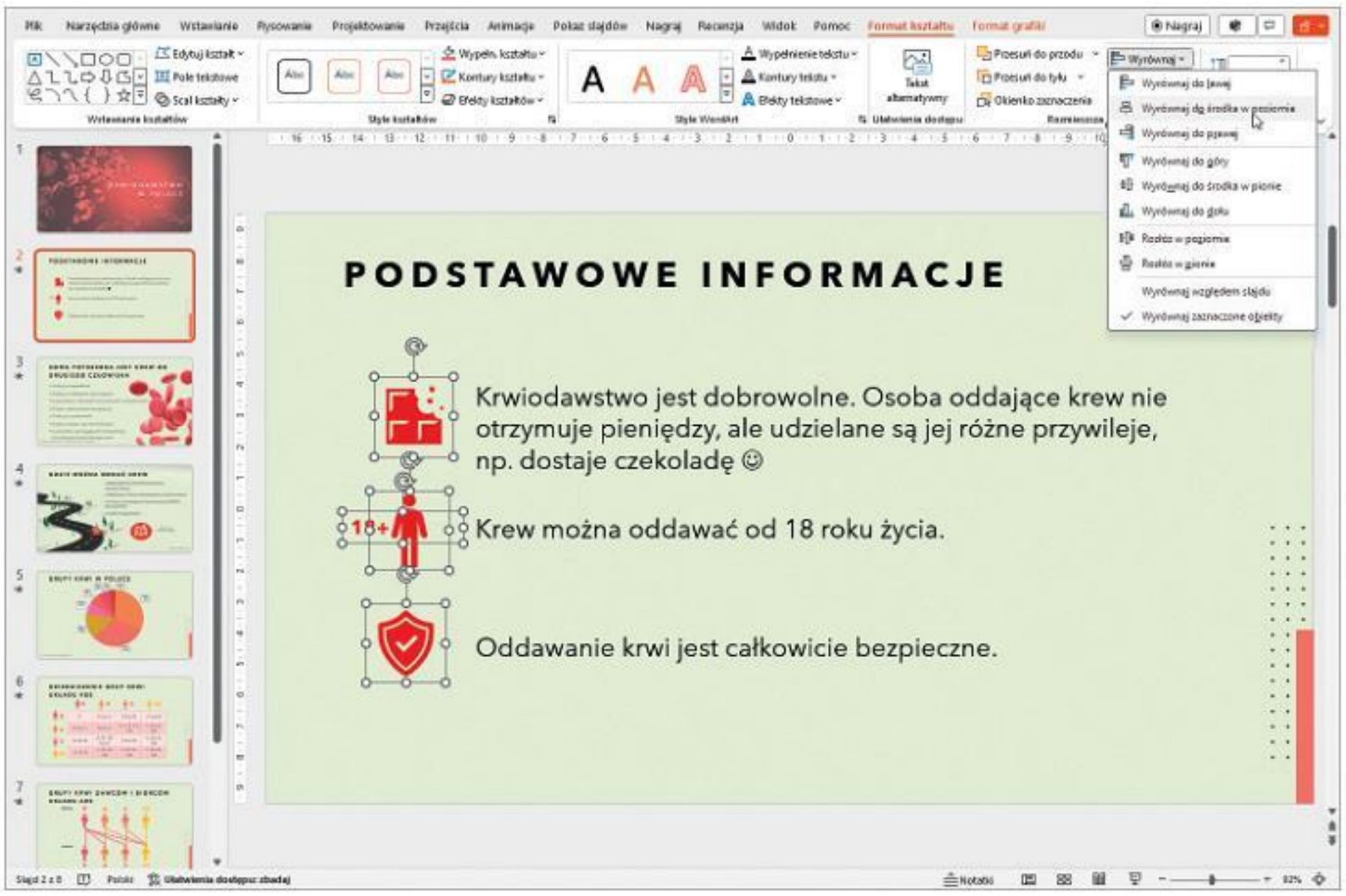
Jeśli na slajdzie jest więcej elementów niż jeden, powinny być one odpowiednio rozmieszczone i wyrównane względem siebie – w miarę jednolicie na podobnych slajdach.



Rys. 3. Okno **Bank obrazów** w programie Microsoft PowerPoint

Czy wiesz, że:

prostą prezentację można wykonać także za pomocą edytora graficznego? Np. darmowy program Zdjęcia Google pozwala na łatwe utworzenie ze zdjęć zgromadzonych w komputerze: albumu, pokazu, filmu lub nawet własnego wygaszacza ekranu.



Rys. 4. Wyrównywanie obrazów w programie Microsoft PowerPoint

Aby wyrównać kształty, obrazy lub zdjęcia umieszczone na slajdzie, należy zaznaczyć wybrane elementy, trzymając wciśnięty klawisz **Shift**, i skorzystać z odpowiedniej opcji wyrównania:

- w programie Microsoft PowerPoint – **Format kształtu**, **Format grafiki** lub **Format obrazu** na karcie **Wyrównaj** (rys. 4.),
- w programie LibreOffice Impress – **Wyrównaj obiekty** i **Rozmieść** z menu **Format**.

Ćwiczenie 4. Wstawiamy kształty, ikony i zdjęcia do slajdów

1. Dodaj kolejne slajdy zgodnie z planem prezentacji przygotowanym w ćwiczeniu 2. Pamiętaj, aby podać źródło wykorzystanych informacji.
2. Do slajdów wstaw kształty, ikony lub zdjęcia według własnego pomysłu. Skorzystaj z własnych zbiorów lub **Banku obrazów**.
3. Zapisz plik pod tą samą nazwą.

Wideo również może być częścią prezentacji, ozdobnikiem lub tłem dla slajdu. Podobnie jak obrazy, klipy wideo możemy nagrać samodzielnie lub skorzystać z dostępnych w programie lub Internecie.

Aby wstawić wideo w programie Microsoft PowerPoint, należy na karcie **Wstawianie** wybrać opcję **Wideo**, a następnie wybrać opcję określającą źródło pliku wideo, np.:

- **Wideo gotowe do użycia** oznacza korzystanie z gotowych klipów z **Banku obrazów**,
- **Wideo online** umożliwia podanie adresu klipu wideo z Internetu i wstawienie go zgodnie z warunkami użytkowania i zasadami ochrony prywatności danego dostawcy (m.in. YouTube, SlideShare, Vimeo).



Rys. 5. Klip wideo jako tło slajdu prezentacji w programie Microsoft PowerPoint – ćwiczenie 5.

Po wstawieniu i zaznaczeniu wideo pokażą się karty **Format wideo** i **Odtwarzanie** z opcjami do edytowania klipu, m.in., obracania, ustawiania klatki tytułowej, zmieniania opcji odtwarzania, określania czasu rozpoczęcia i zakończenia (rys. 5.).

Ćwiczenie 5. Wstawiamy wideo do prezentacji

1. Wybierz jeden slajd i wstaw do niego klip wideo ilustrujący treści zawarte na slajdzie. Inspirację możesz znaleźć na rysunku 5. Ustaw odtwarzanie wstawionego klipu w pętli (na karcie **Odtwarzanie**, w grupie **Opcje wideo** wybierz opcję **W pętli do zatrzymania**). Możesz też ustawić wideo jako tło slajdu.
2. Zapisz plik pod tą samą nazwą.

Wskazówka: Aby ustawić wideo jako tło slajdu, na karcie **Format wideo**, w grupie **Rozmieszczanie** wybierz opcję **Przesuń na spód**.

Atrakcyjnym elementem prezentacji, przyciągającym uwagę odbiorców, może być model 3D. Możemy go sami zaprojektować (np. w programie Paint 3D lub Tinkercad) lub skorzystać z gotowego projektu.

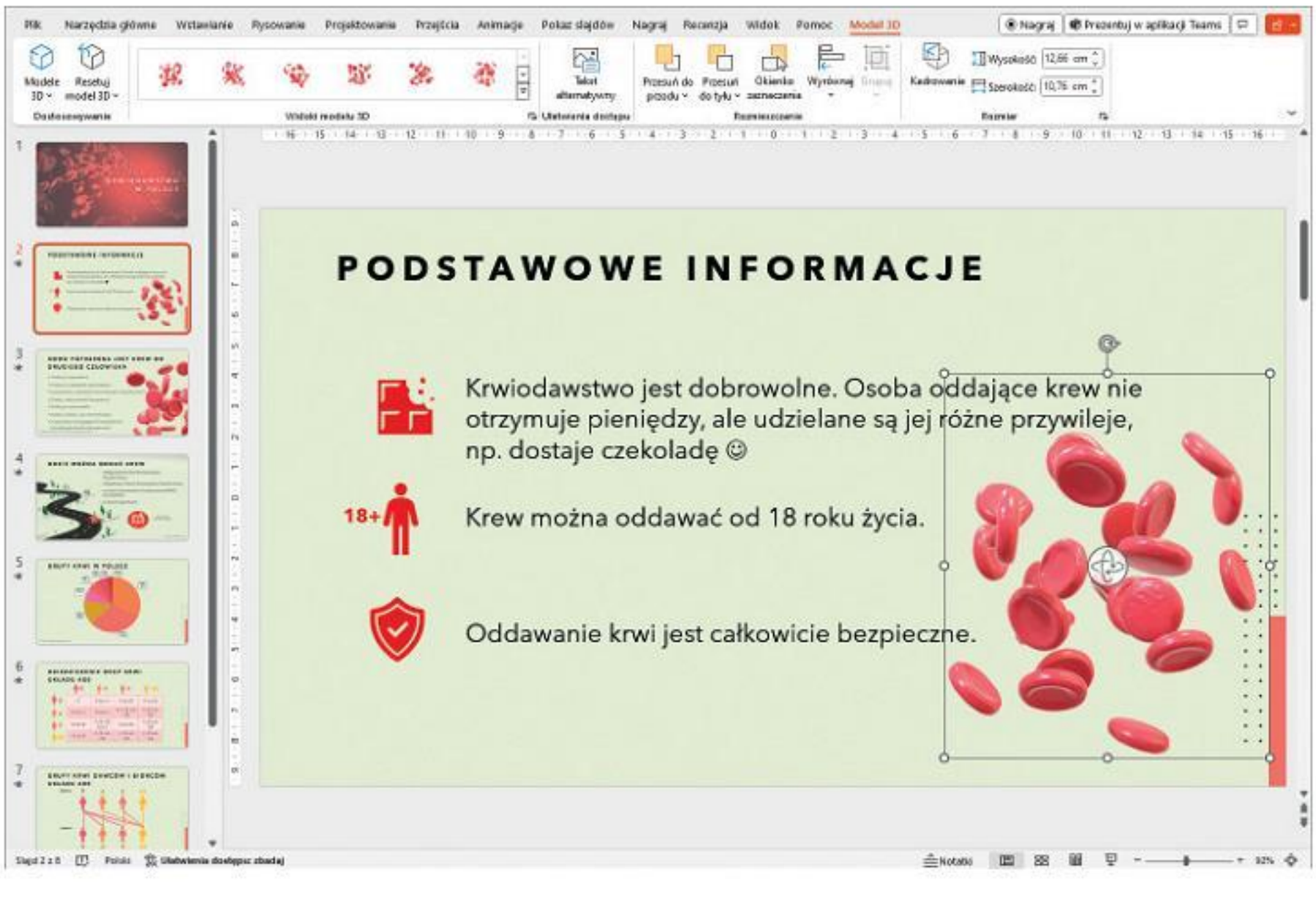
W programie Microsoft PowerPoint model 3D wstawiamy, wybierając opcję **Modele 3D** na karcie **Wstawianie**, w grupie **Ilustracje**. Po wstawieniu i zaznaczeniu modelu 3D możemy go obracać we wszystkich kierunkach i oglądać z każdej strony.



Rys. 6a. Okno **Modele 3D online** w programie Microsoft PowerPoint – kategoria **Medical**

Ćwiczenie 6. Wstawiamy model 3D jako ilustrację slajdu

1. Do prezentacji dodaj model 3D (jeśli program, z którego korzystasz, to umożliwia). Możesz wybrać model z gotowej biblioteki lub samodzielnie go zaprojektować, np. w programie Paint 3D.
2. Obróć model według uznania.
3. Zapisz plik pod tą samą nazwą.



Rys. 6b. Model 3D wstawiony na slajd prezentacji w programie Microsoft PowerPoint – ćwiczenie 6.

1.3. Wstawianie hiperłączy, wykresów i tabel

Hiperłączy używamy, gdy chcemy utworzyć prezentację interaktywną. Umożliwiają one na przykład szybkie przechodzenie do innych slajdów w tej samej prezentacji, wywołanie strony internetowej lub otwarcie pliku.

Hiperłącze może być reprezentowane przez tekst lub obiekt, na przykład obraz, ikonę, wykres, tekst WordArt. Tekst reprezentujący hiperłącze (link, odsyłacz) wyświetla się z podkreśleniem.

Aby wstawić hiperłącze, należy uaktywnić dany obiekt lub zaznaczyć fragment tekstu i wybrać opcję **Link** na karcie **Wstawianie** (Microsoft PowerPoint) lub **Hiperłącze** w menu **Wstaw** (LibreOffice Impress). Następnie należy określić obiekt docelowy, do którego tworzymy przejście, np. wybrany slajd lub link.

Ćwiczenie 7. Wstawiamy hiperłącza

1. Znajdź na mapie Google najbliższy punkt krwiodawstwa w regionie, w którym mieszkasz. Do odpowiedniego slajdu dodaj ikonę mapy i wstaw hiperłącze prowadzące do mapy. Kliknięcie ikony powinno powodować przejście na stronę z mapą z zaznaczonym punktem krwiodawstwa (rys. 5.).

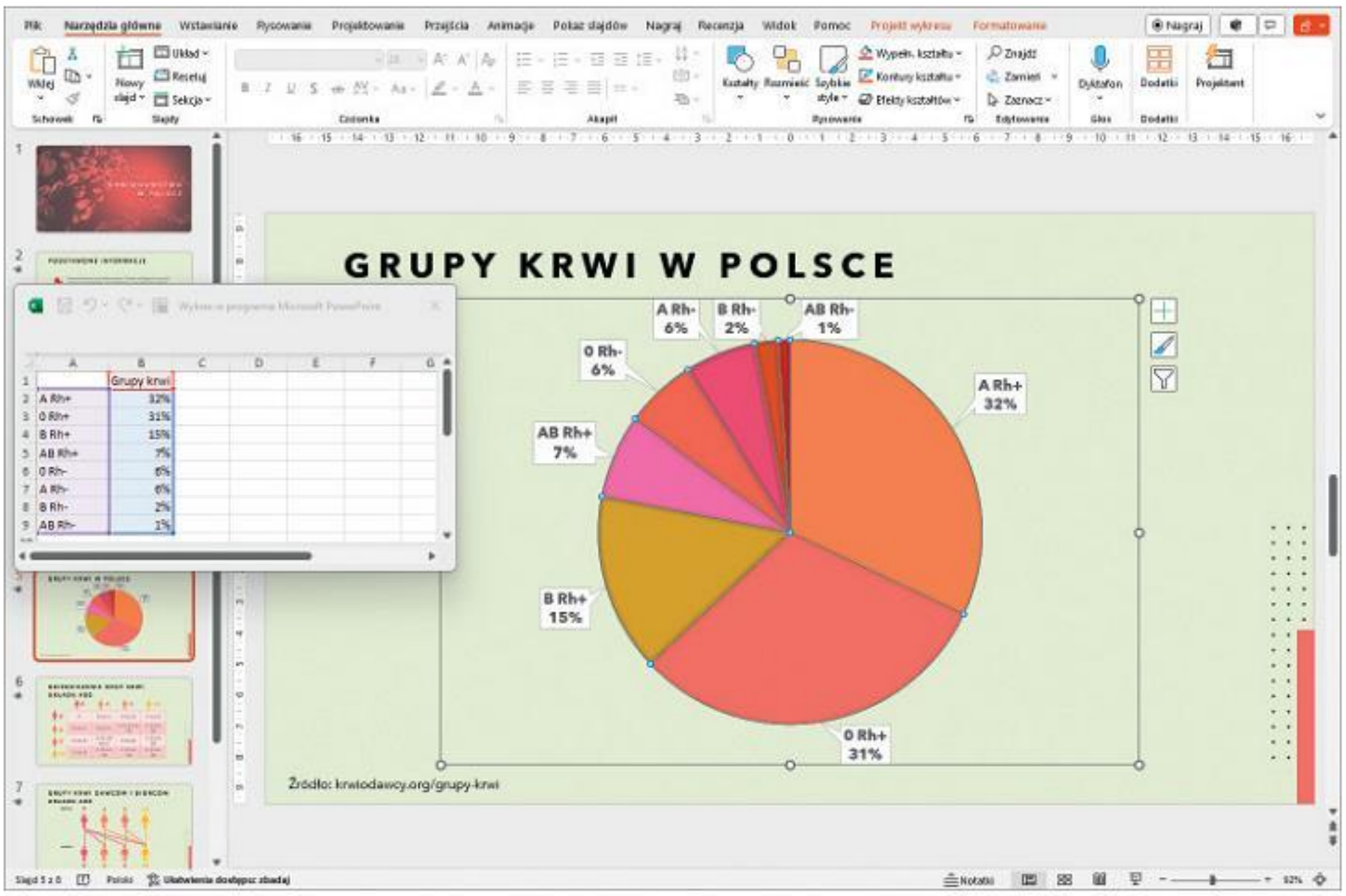
2. Jeśli masz problemy, sprawdź w **Pomocy**, w jaki sposób można wstawić hiperłącza na tekst lub obiekty umieszczone na slajdzie.
3. Zapisz plik pod tą samą nazwą.

Wskazówka: Link do danego miejsca na mapie Google znajdziesz po kliknięciu przycisku **Udostępnij**.

Na slajdzie prezentacji możemy zaprezentować dane w postaci wykresu, wykorzystując odpowiednie opcje programu.

Aby wstawić wykres do prezentacji, można wybrać opcję wstawiania wykresu na karcie **Wstawianie** (Microsoft PowerPoint) lub w menu **Wstaw** (LibreOffice Impress), wybrać typ wykresu i zaakceptować wybór, następnie umieścić dane w komórkach tabeli w otwartym oknie **Wykres** w programie Microsoft PowerPoint (rys. 7.) lub **Tabela danych** w LibreOffice Impress i zamknąć to okno.

Utworzony w ten sposób wykres komputer pamięta jako obiekt wstawiony do prezentacji i nie trzeba go oddzielnie zapisywać. Aby zmodyfikować dane, należy z menu kontekstowego wykresu wybrać polecenie **Edytuj dane** (Microsoft PowerPoint) lub **Tabela danych** (LibreOffice Impress).



Rys. 7. Wstawianie wykresu na slajd prezentacji w programie Microsoft PowerPoint – ćwiczenie 8.

Ćwiczenie 8. Wstawiamy wykres do prezentacji multimedialnej

1. Wstaw wykres dotyczący częstotliwości występowania grup krwi w Polsce. Dane na ten temat znajdź w Internecie.
2. Zapisz plik pod tą samą nazwą.

Połączenie dokumentów **P**

Wstawienie do dokumentu (np. tekstowego) obiektu lub jego fragmentu pochodzącego z innej aplikacji (np. arkusza kalkulacyjnego) w ten sposób, by zachować trwałe połączenie między obiektem wstawionym a obiektem źródłowym i jego aplikacją macierzystą. Każdą zmianę danych w pliku źródłowym widać w obiekcie wstawionym.

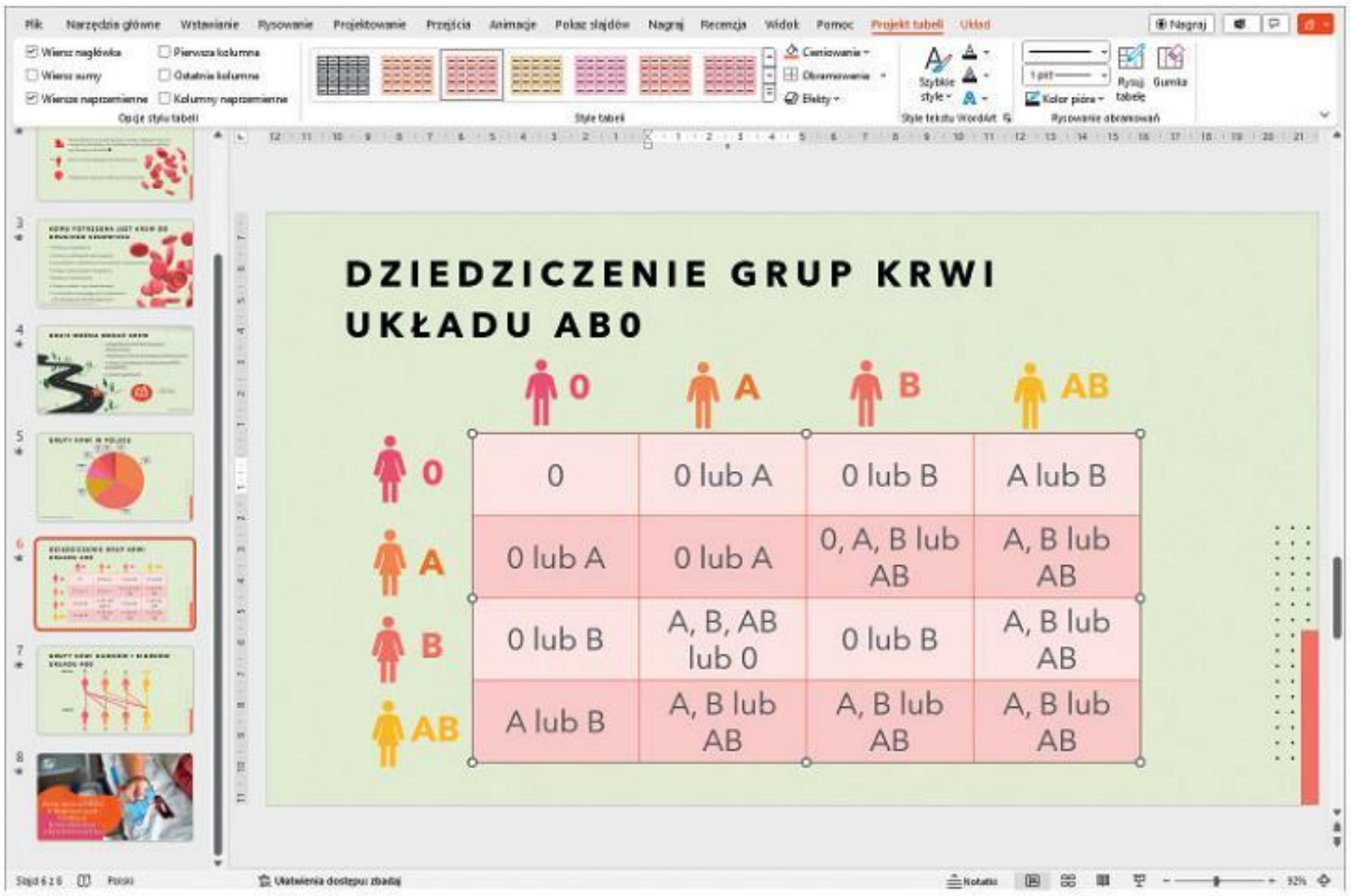
Na slajd prezentacji możemy również wstawić wykres utworzony w arkuszu kalkulacyjnym Microsoft Excel lub LibreOffice Calc. W tym celu należy zaznaczyć wykres, skopiować go i wkleić na slajd (więcej o tworzeniu wykresów – temat 13.).

Gdy w programie Microsoft Excel do wklejania użyjemy opcji **Wklej specjalnie/Zachowaj łącze**, zachowamy dodatkowo **połączenie** pomiędzy plikiem źródłowym (arkuszem kalkulacyjnym) a docelowym (prezentacją). W programie LibreOffice Impress takie połączenie uzyskamy po wybraniu z menu **Wstaw** opcji **Obiekt/Obiekt OLE/Utwórz z pliku**, następnie wybraniu pliku arkusza i zaznaczeniu opcji **Łącze do pliku.**

Jeśli chcemy, aby odbiorca prezentacji mógł łatwo przeanalizować dane, możemy przedstawić je w postaci tabeli.

Aby wstawić tabelę do prezentacji, należy:
- wybrać opcję wstawiania tabeli na karcie **Wstawianie** (Microsoft PowerPoint) lub w menu **Wstaw** (LibreOffice Impress),
- określić liczbę kolumn i wierszy i zaakceptować wybór.

Po wstawieniu tabeli można ją edytować, wybierając odpowiednie opcje na karcie **Projekt tabeli** (Microsoft PowerPoint) lub na pasku narzędzi **Tabela** (LibreOffice Impress).



Rys. 8. Wstawianie tabeli na slajd prezentacji w programie Microsoft PowerPoint – ćwiczenie 9.

Ćwiczenie 9. Wstawiamy tabelę na slajd prezentacji multimedialnej

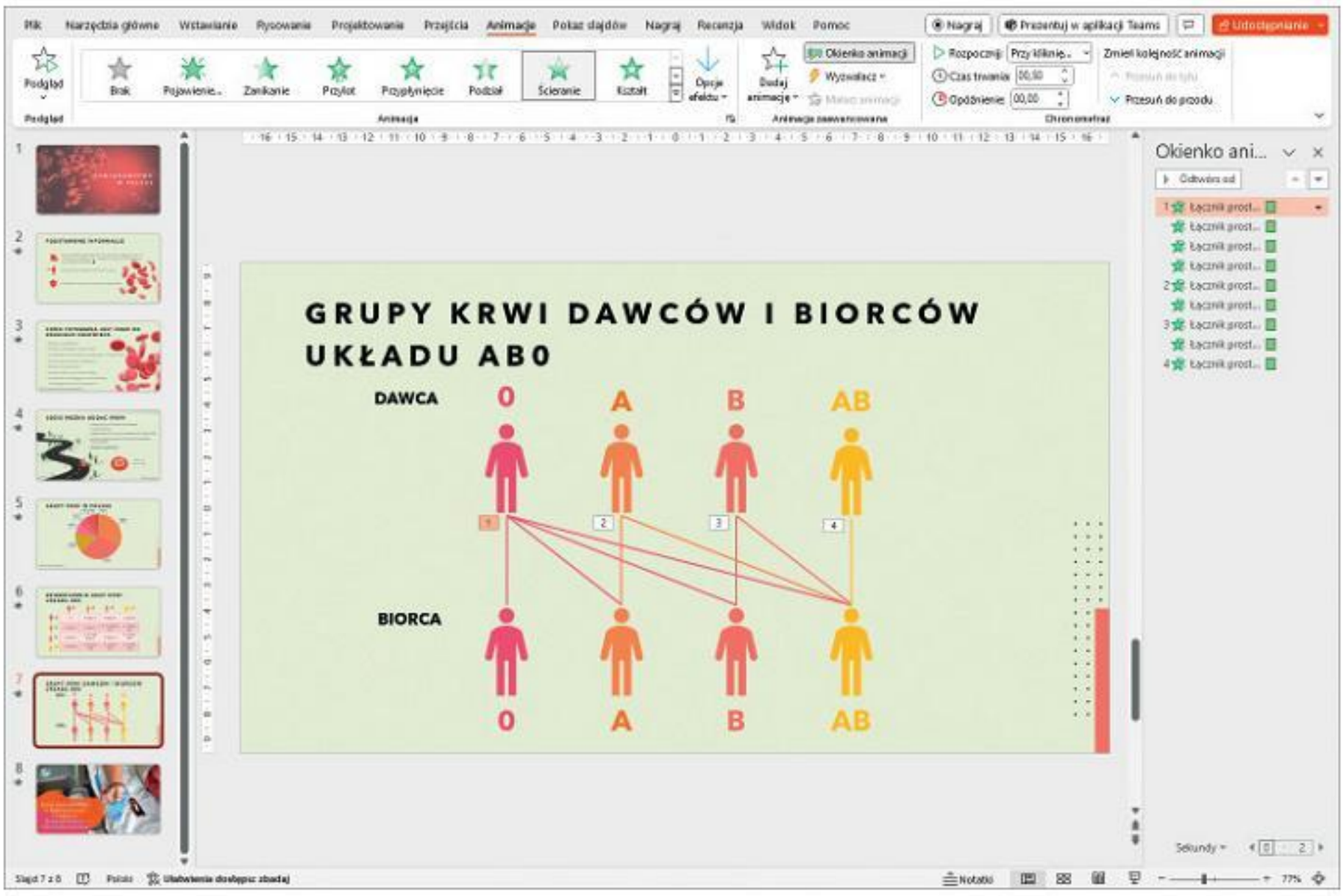
1. Wstaw tabelę przedstawiającą dziedziczenie grup krwi. Odpowiednio ją sformatuj i wypełnij.
2. Zapisz plik pod tą samą nazwą.

1.4. Dodawanie animacji i przejść slajdów

Animacje bardzo uatrakcyjniają prezentację. Dzięki zastosowaniu animacji można zwrócić uwagę oglądających na istotne kwestie. Animację można przypisać dowolnemu elementowi slajdu – na przykład tekst może pojawiać się z lewej strony po jednym słowie, a model 3D może się obracać. Efekty tego typu należy jednak stosować z umiarem. Jeśli będzie ich zbyt dużo, mogą rozpraszać uwagę odbiorców.

Aby uzyskać efekty animacji obiektu, należy uaktywnić dany obiekt i na karcie **Animacje** w grupie **Animacja** kliknąć wybrany efekt (Microsoft PowerPoint) lub w menu **Widok** wybrać opcję **Animacja** i dodać efekt, klikając przycisk **Dodaj efekt** (LibreOffice Impress).

Zanim uruchomimy cały pokaz, warto sprawdzić działanie wybranego efektu (przycisk **Podgląd**). Jeśli na slajdzie jest więcej animowanych obiektów, można w dowolny sposób ustawiać i zmieniać kolejność animacji w **Okienku animacji** (rys. 9.).



Rys. 9. Okienko animacji otwarte po prawej stronie okna programu Microsoft PowerPoint

Ćwiczenie 10. Dodajemy do slajdów animacje

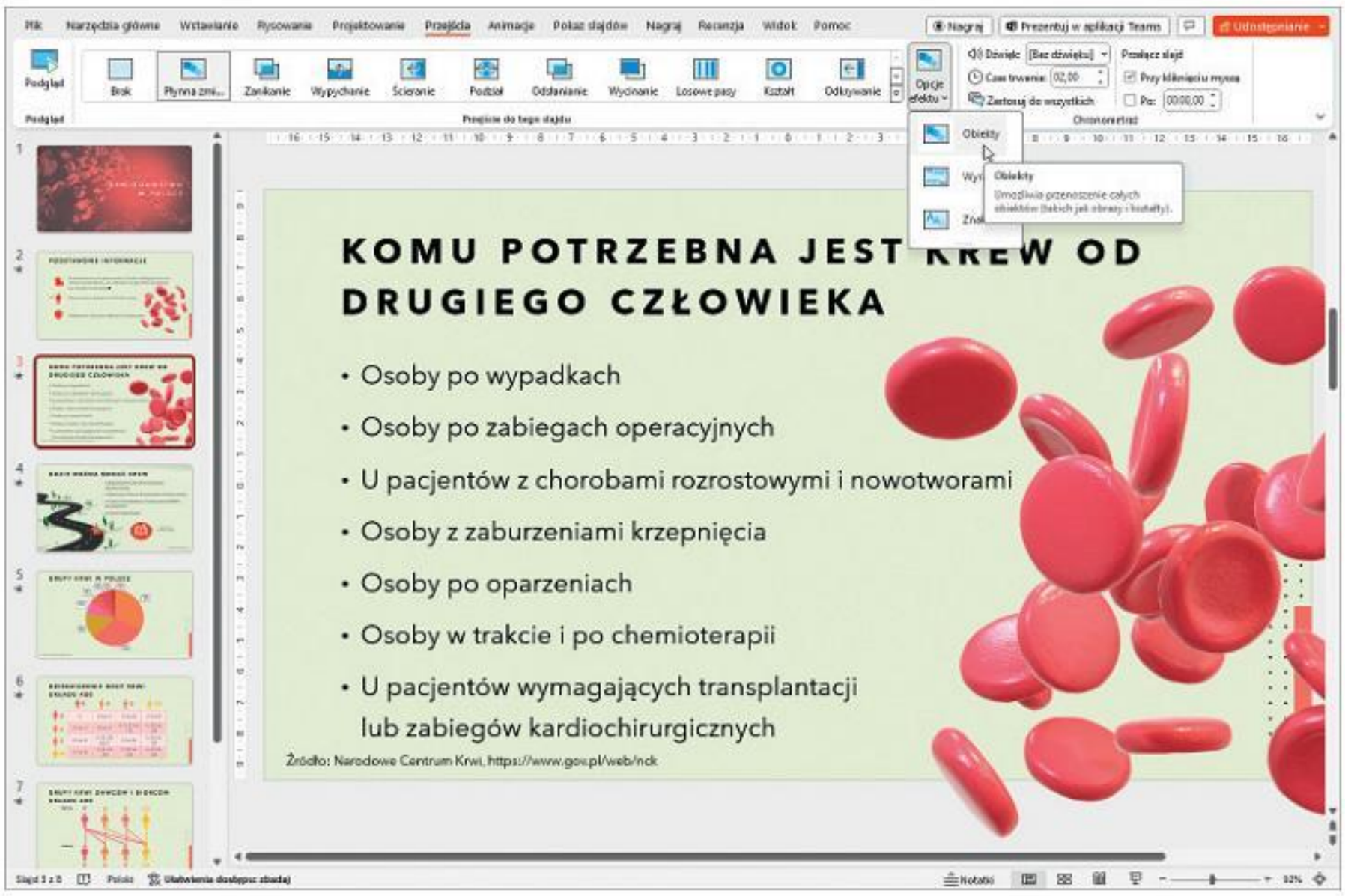
1. Dodaj animacje do utworzonych slajdów.
2. Sprawdź, jakie opcje animacji można zastosować dla modelu 3D dodanego w ćwiczeniu 6. Wybierz jedną animację dla tego modelu według własnego uznania.
3. Sprawdź, w jaki sposób w programie, z którego korzystasz, uzyskuje się efekty animacji. Przejrzyj wykaz możliwych efektów. W przypadku problemów skorzystaj z **Pomocy**. Dodaj animacje do pozostałych slajdów – według własnego uznania.
4. Zapisz plik pod tą samą nazwą.

Ciekawym rodzajem animacji w trakcie przechodzenia na kolejny slajd w programie Microsoft PowerPoint jest zastosowanie efektu przejścia **Płynna zmiana**. Najlepsze rezultaty otrzymamy, jeśli skopiujemy dany obiekt i umieścimy go na następnym slajdzie w innym miejscu i/lub zmienimy jego rozmiar. W ustawieniach **Opcje efektu** można wybrać zastosowanie tego przejścia do obiektów, wyrazów lub znaków.

Ćwiczenie 11. Stosujemy przejście **Płynna zmiana** na slajdzie z modelem 3D

1. Skopiuj model 3D wstawiony w ćwiczeniu 6. i wklej na kolejny slajd. Zmień położenie modelu (np. obróć go) i jego rozmiar.
2. Na slajdzie, na który został skopiowany model 3D, zastosuj przejście **Płynna zmiana.** Uruchom **Podgląd.** Jaki widzisz efekt?
3. Zastosuj odpowiednie przejścia na pozostałych slajdach.
4. Sprawdź, czy twoja prezentacja spełnia warunki dobrej prezentacji opisane w punkcie 1. tego tematu.
5. Zapisz plik pod tą samą nazwą.

Wskazówka: Jeśli na wklejonym modelu 3D ustawiono animację typu **Wejście**, należy ją usunąć. W przeciwnym wypadku zostanie ona wykonana i efekt przejścia nie będzie widoczny.



Rys. 10. Zastosowanie przejścia **Płynna zmiana** dla modelu 3D w programie Microsoft PowerPoint – ćwiczenie 11.

2. Podstawowy montaż filmu

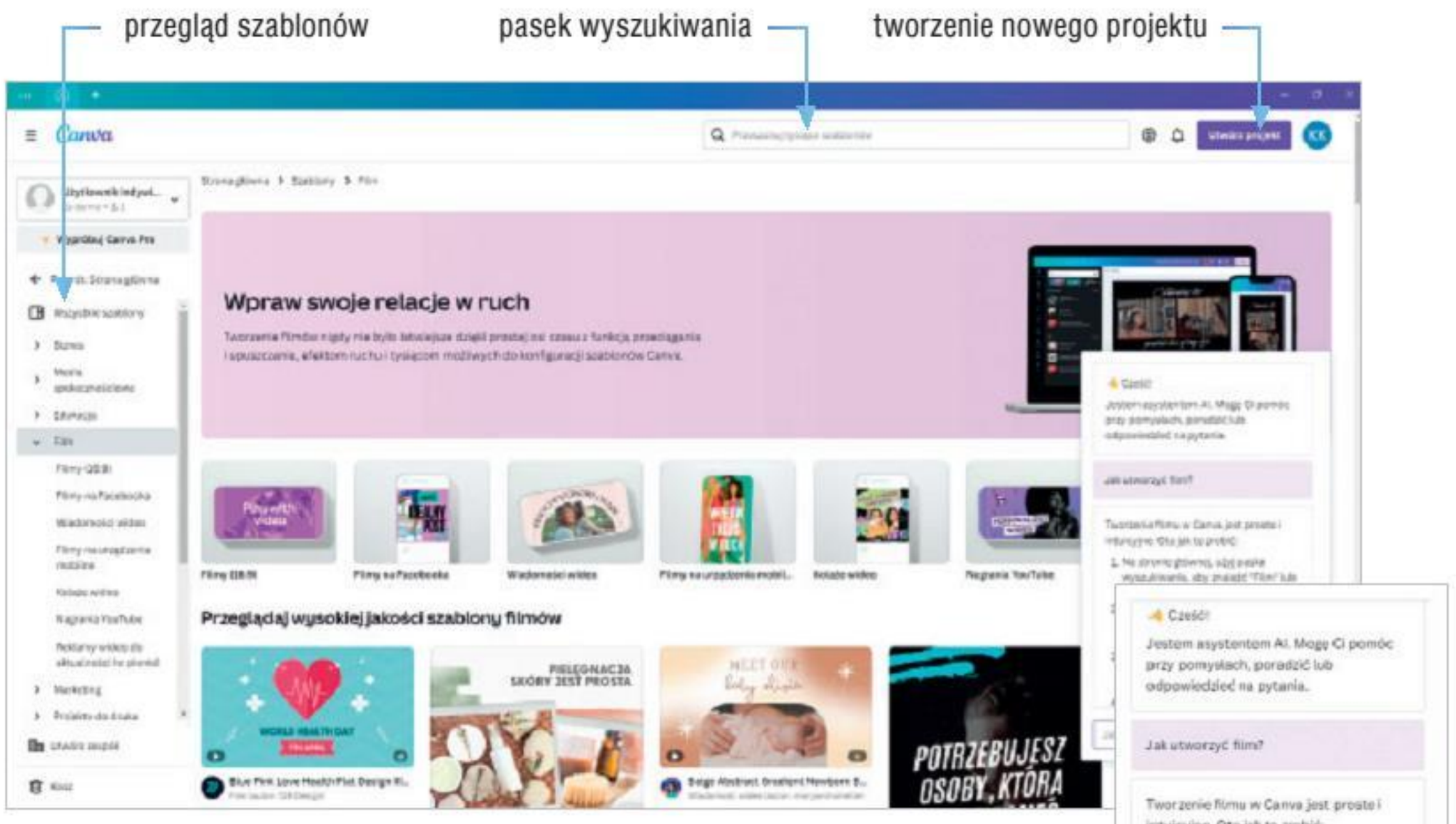
Na ostatnim slajdzie prezentacji chcemy wyświetlać krótki, półminutowy film, podsumowujący wystąpienie i zachęcający do oddawania krwi. Jak go przygotować?

Film zapisany w wersji cyfrowej (np. nagrany smartfonem) można edytować. Podstawową jednostką w opracowywaniu filmu jest **scena**. Scena to fragment filmu ograniczony czasowo lub np. zmianą ujęcia. Sceny można ze sobą łączyć, dzielić na mniejsze, ustawiać w różnej kolejności. Można także wstawić sceny z innego filmu lub stworzyć własne, np. z ruchomymi napisami i grafikami.

Istnieje wiele programów umożliwiających opracowywanie filmów. Edycję filmów pokażemy na przykładzie serwisu **Canva**, z którego można bezpłatnie korzystać w Internecie lub pobrać aplikację na komputer, również w ramach wersji edukacyjnej, dostępnej dla szkół. Scena w serwisie Canva jest przedstawiana jako **Strona**.

Platforma Canva umożliwia nie tylko opracowywanie filmów, ale też tworzenie prezentacji, infografik, postów i filmów do mediów społecznościowych, stron internetowych i wielu innych.

W celu uzyskania pomocy należy skorzystać z **Asystenta AI** (znak ? lub ? w prawym dolnym rogu ekranu). Pytania zadajemy w okienku **Pomocy** w formie chatu (rys. 11.).



Rys. 11. Widok aplikacji Canva dla systemu Windows z otwartym oknem Asystenta AI

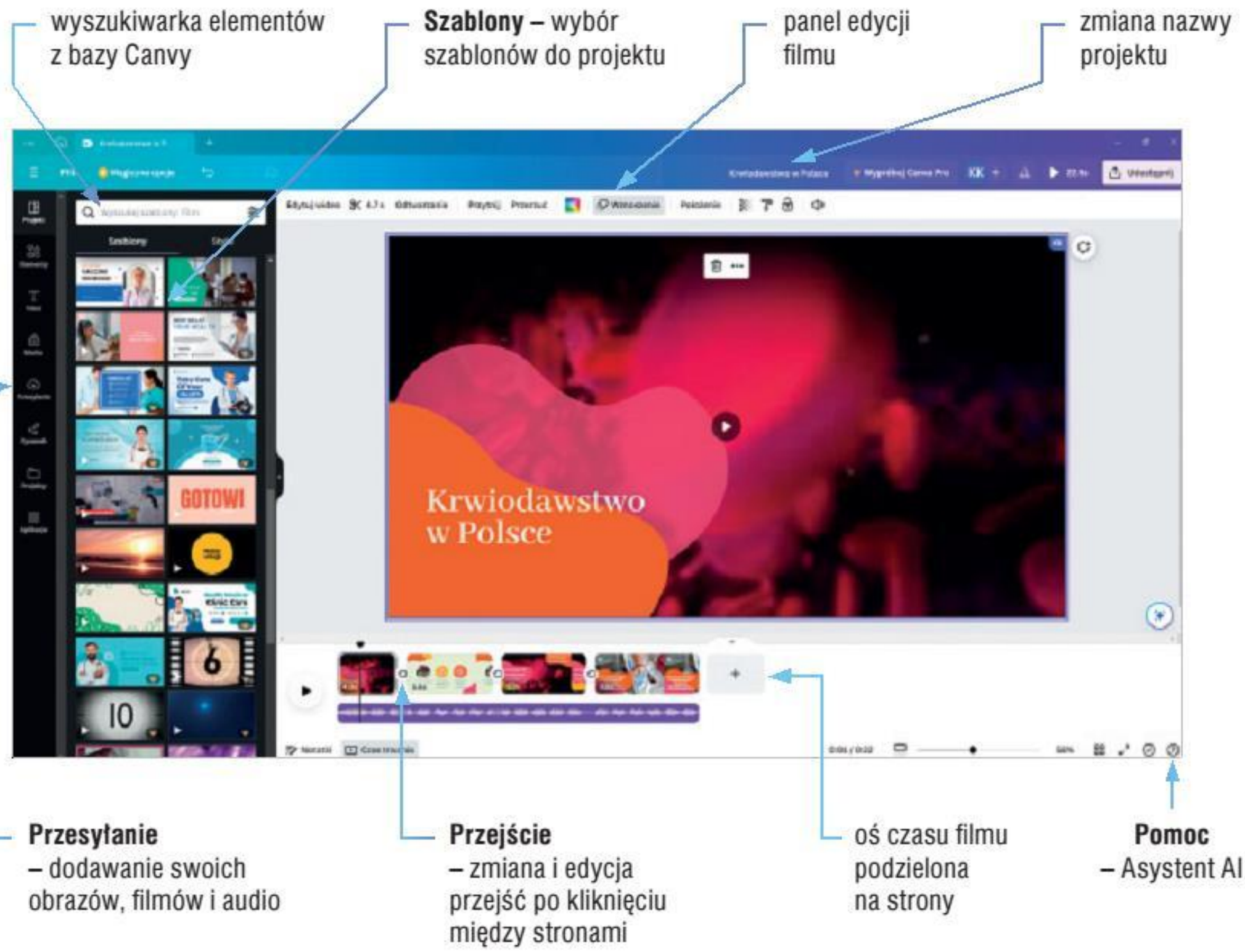
Sztuczna inteligencja (w skrócie SI, ang. *artificial intelligence* – AI)

W informatyce to dziedzina badań naukowych oraz zbiorcza nazwa technologii służących modelowaniu i symulowaniu działania naturalnej (zwierzęcej lub ludzkiej) inteligencji.

Odpowiedź jest generowana przez **sztuczną inteligencję** i może zawierać błędy. Należy upewnić się, czy odpowiedź jest poprawna, np. wykonując kolejno kroki instrukcji podanej przez Asystenta AI. Jeśli nie uzyskamy satysfakcjonującej odpowiedzi na nasze pytanie, należy spróbować zadać je w inny sposób.

Ćwiczenie 12. Poznajemy platformę Canva

1. Wejdź na stronę *www.canva.com* i postępuj zgodnie z instrukcjami nauczyciela w celu zalogowania się.
2. Zapoznaj się z szablonami filmów, prezentacji, grafik do mediów społecznościowych, stron www i innych.
3. Zadaj Asystentowi AI pytanie dotyczące tworzenia filmów. Zapoznaj się z odpowiedzią.



Rys. 12. Podstawowe opcje aplikacji Canva – edycja filmu

Aby utworzyć projekt filmu w dowolnym programie, w tym na platformie Canva, należy:

- utworzyć pusty projekt (na platformie Canva po kliknięciu przycisku **Utwórz projekt** wyszukać typ projektu, np. film), ewentualnie wybrać szablon, jeśli program posiada taką opcję,
- dodać wybrane pliki do projektu – pliki wideo, zdjęcia, grafikę z własnych zbiorów lub z banku grafik programu,
- dopracować film, m.in.: przyciąć klip wideo, podzielić klip wideo na mniejsze części i zmienić ich kolejność, zmniejszyć lub zwiększyć szybkość odtwarzania,
- dodać ścieżkę dźwiękową; można dodać również inne efekty specjalne,
- dodać tytuły, opisy do dowolnego fragmentu filmu oraz animacje do nich,
- zapisać projekt i wyeksportować film do innego formatu.

Ćwiczenie 13. Tworzymy film podsumowujący prezentację z wykorzystaniem szablonu

1. W aplikacji Canva utwórz nowy projekt filmu w formacie 1920 x 1080 px.
2. Wybierz szablon filmu według własnego uznania i zastosuj go do projektu.
3. Skorzystaj z wyszukiwarki Canvy i znajdź filmy, które pasują do tematu krwiodawstwa. Dodaj je do swojego projektu i usuń filmy proponowane w szablonie, jeśli są nieodpowiednie. Wykonaj podstawową edycję filmu, m.in. dostosuj jego długość i szybkość odtwarzania.
4. Zmień nazwę projektu na *Krwiodawstwo w Polsce*.

Wskazówka: W razie potrzeby skorzystaj z **Pomocy** programu.



Rys. 13. Podstawowe opcje aplikacji Canva – dodatkowe elementy

Aby zapisać film (lub inny projekt) w programie Canva, należy klinąć **Udostępnij** na górnym pasku menu, a następnie **Pobierz** i wybrać typ pliku, np. MP4.

Ćwiczenie 14. Zmieniamy elementy graficzne i dodajemy ścieżkę dźwiękową do filmu

1. Do projektu dodaj nowe elementy graficzne (np. naklejki, pola tekstowe, nagłówki, tytuły) i dostosuj istniejące według własnego pomysłu, np. zmień teksty, kolory i animacje. Dodaj ścieżkę dźwiękową z zakładki **Elementy** i dostosuj jej długość.
2. Pobierz film w formacie MP4.
3. Wstaw film jako ostatni slajd prezentacji zapisanej w ćwiczeniu 11.

3. Projektowanie i druk 3D pomocy dydaktycznych

Na zakończenie wykładu chcemy przekazać odbiorcom prezentacji breloczki do kluczy z symbolem ich grupy krwi wydrukowane na drukarce 3D. W jaki sposób je zaprojektować?

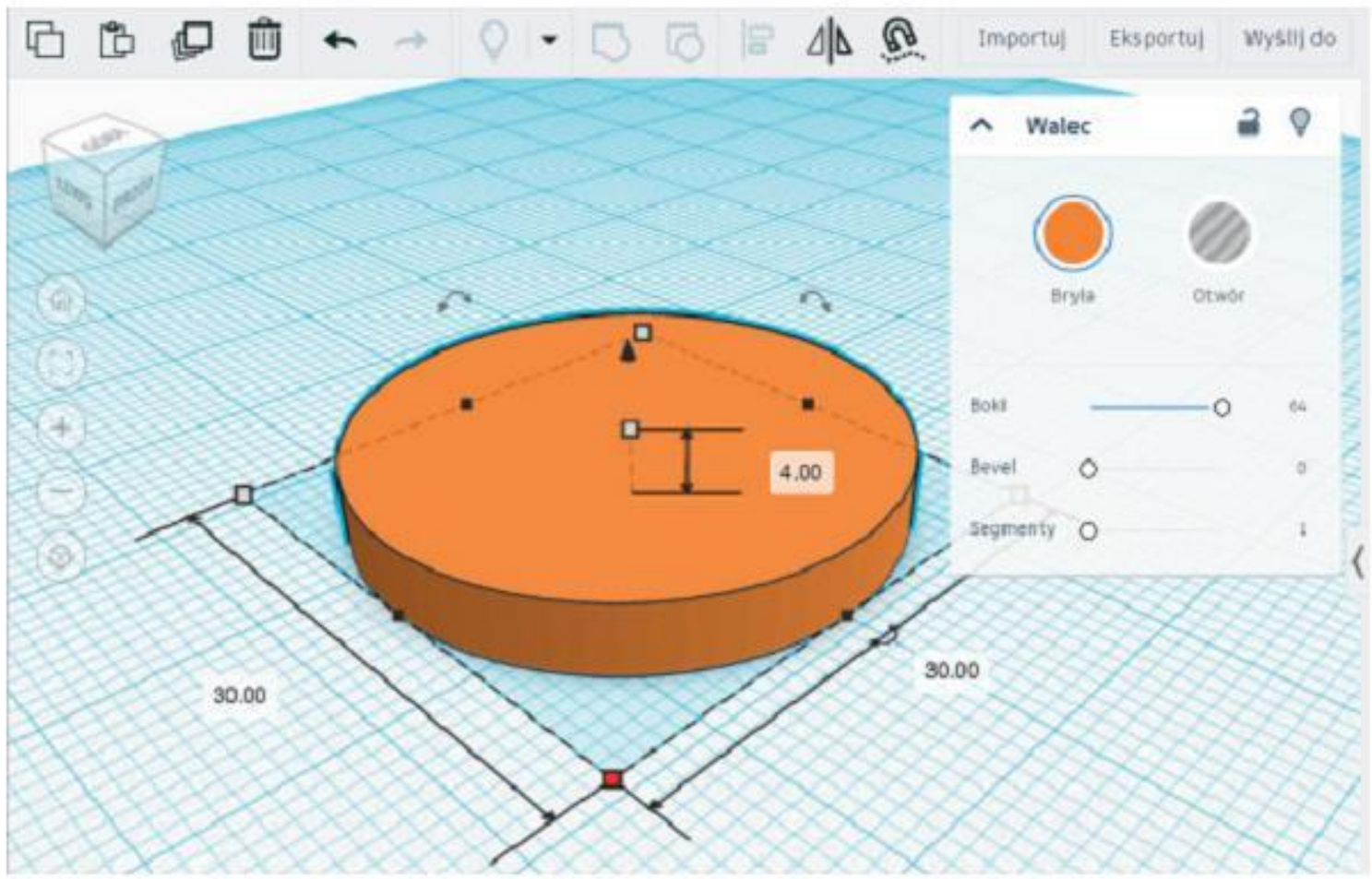
Projekt breloczka wykonamy w programie Tinkercad. Breloczek ma mieć kształt kropli o grubości 4 mm. Do wykonania projektu użyjemy znanych nam przekształceń: wstawianie kształtu, zmienianie jego wymiarów, obracanie, grupowanie, zmienianie koloru oraz dodawanie tekstu. Zdobędziemy też nowe umiejętności z zakresu projektowania 3D i drukowania 3D.



Ćwiczenie 15. Wstawiamy walec i zmieniamy jego wymiary

1. Zaloguj się na stronie programu Tinkercad zgodnie z instrukcją nauczyciela.
2. Na płaszczyznę roboczą wstaw kształt **Walec**. Ustaw wymiary kształtu: podstawa – 30 mm x 30 mm i wysokość – 4 mm (rys. 14.).
3. Zmień nazwę pliku, w którym zapisano projekt, z domyślnej na *Breloczek*.

Wskazówka: Zmień wartość parametru **Boki** w panelu **Inspektor** na 64 – dzięki temu walec będzie miał bardziej gładki obrys.



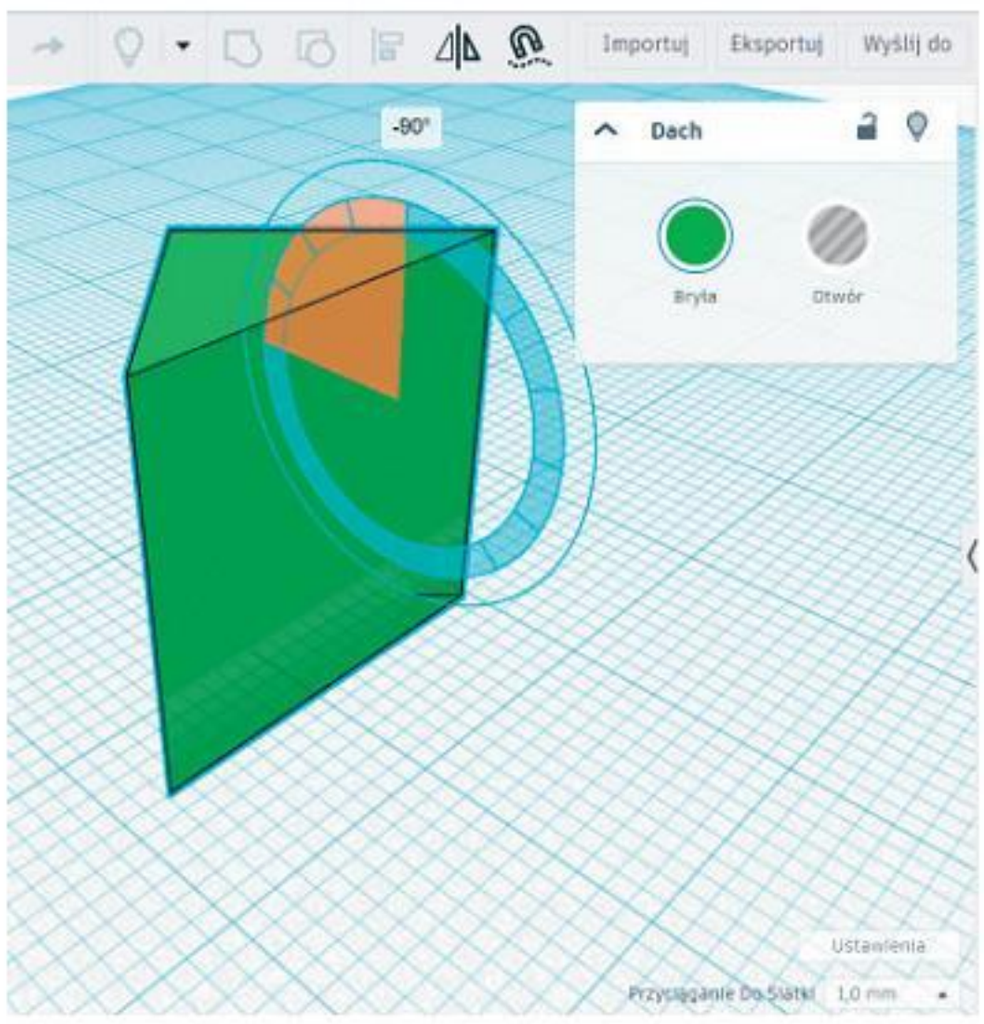
Rys. 14. Projekt walca – zmiana wymiarów – ćwiczenie 15.

Kształt kropli możemy uzyskać, łącząc ze sobą dwa kształty: **Walec** oraz **Dach**, przy czym **Dach** musimy wcześniej obrócić o 90 stopni, wykorzystując w tym celu **uchwyt obracania** (zaokrąglona strzałka z podwójnym grotem).

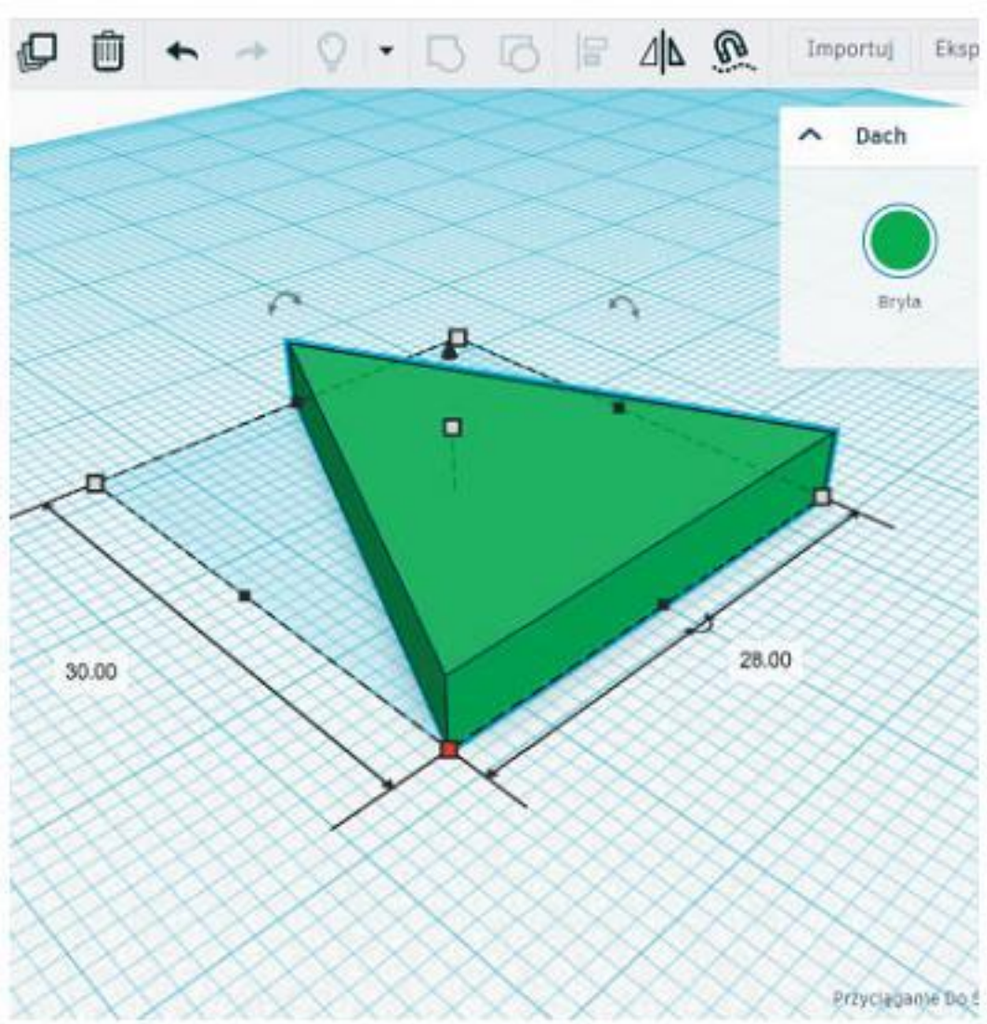
Ćwiczenie 16. Wstawiamy kształt **Dach** i przekształcamy go

1. Na tę samą płaszczyznę roboczą wstaw kształt **Dach**. Obróć go o 90 stopni, aby płaska trójkątna płaszczyzna była na górze (rys. 15a).
2. Ustaw wymiary kształtu: podstawa – 28 mm x 30 mm i wysokość – 4 mm (rys. 15b).

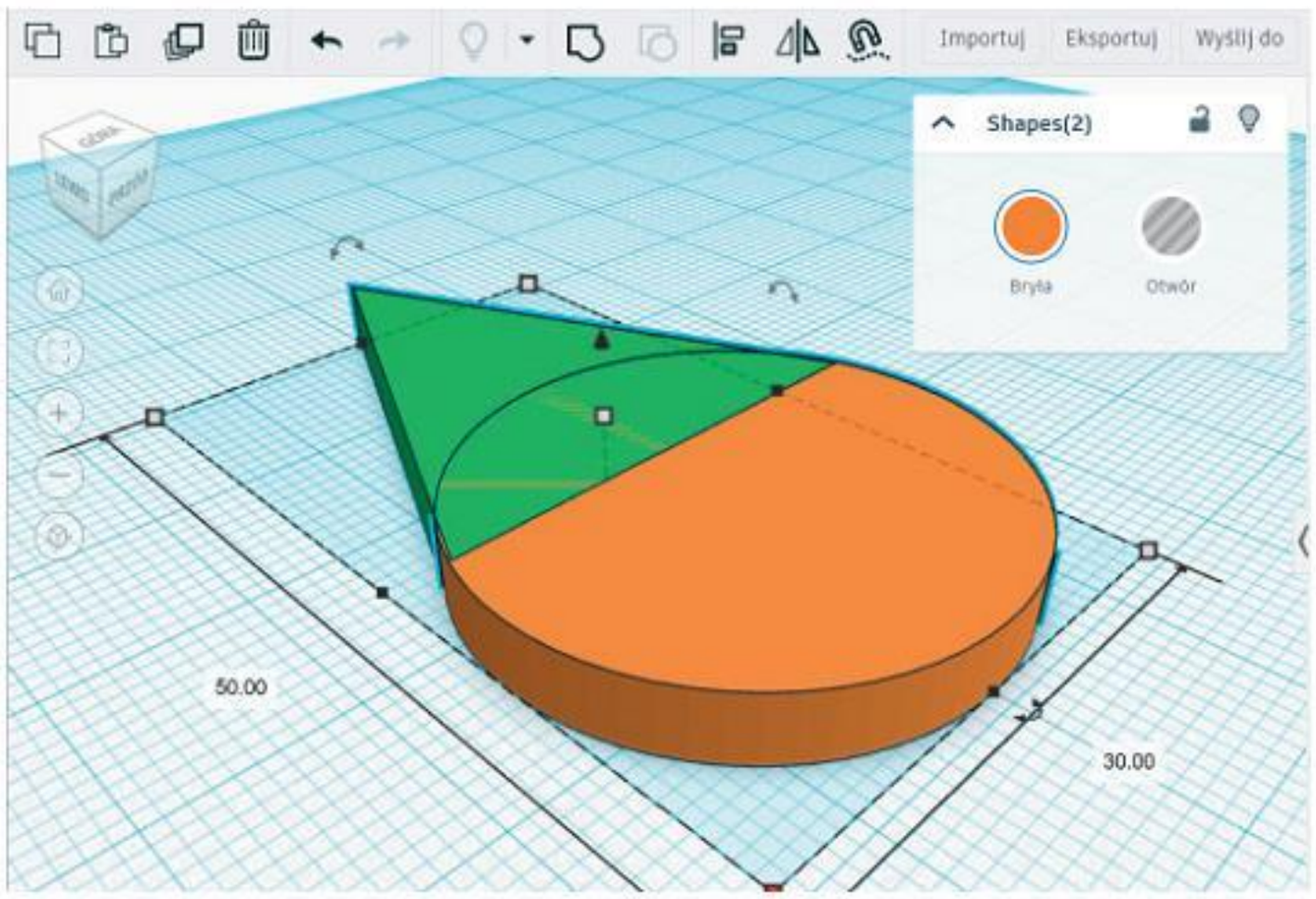
Wskazówka: Upewnij się, że kształt jest umieszczony na płaszczyźnie roboczej, a nie pod nią. W razie potrzeby podnieś kształt za pomocą uchwytu podnoszenia/upuszczania.



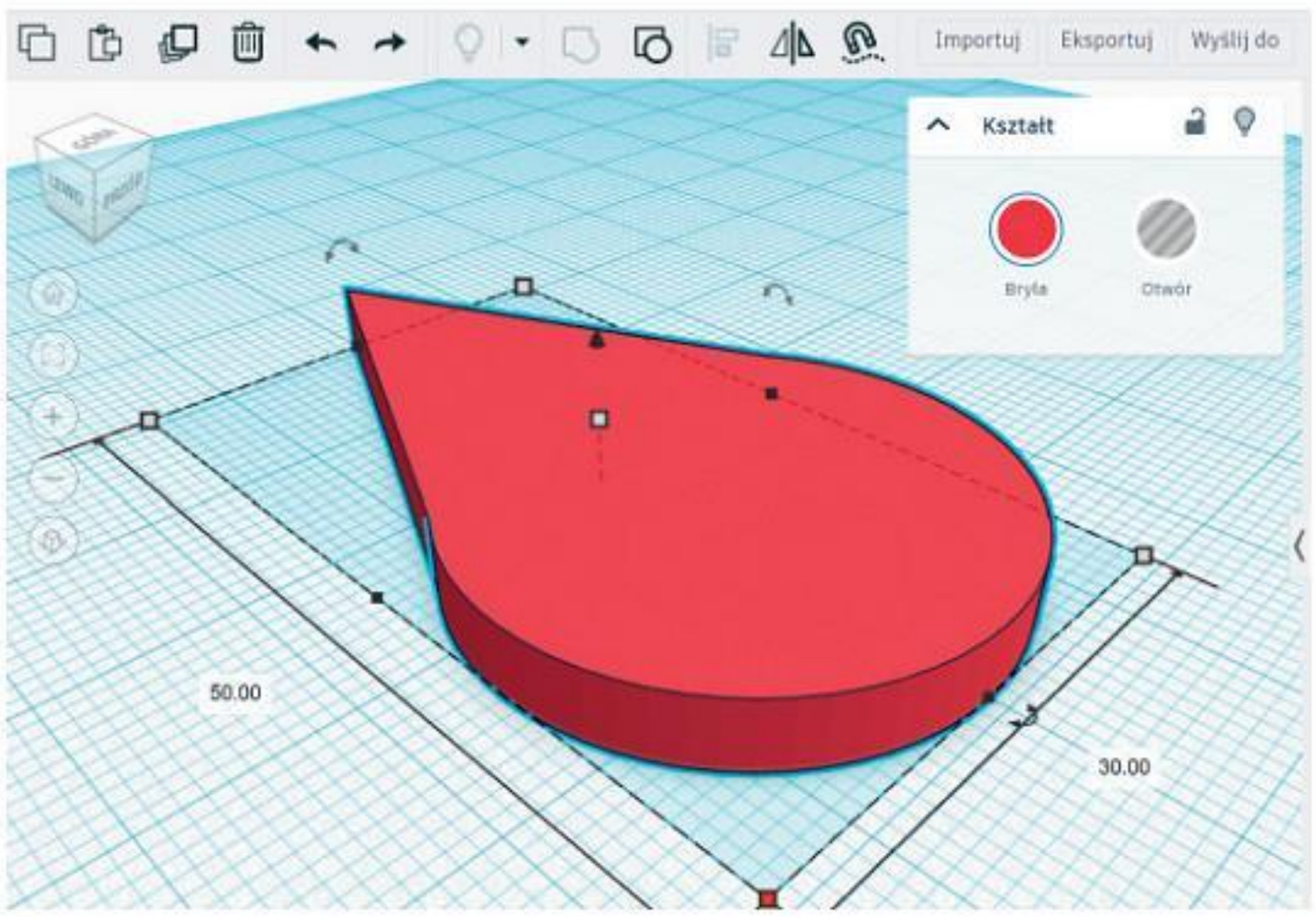
Rys. 15a. Obrót kształtu **Dach** o 90 stopni – ćwiczenie 16.



Rys. 15b. Obrócony kształt **Dach** – zmiana wymiarów – ćwiczenie 16.



Rys. 16a. Umieszczanie dwóch kształtów obok siebie – ćwiczenie 17.



Rys. 16b. Zgrupowane kształty – ćwiczenie 17.

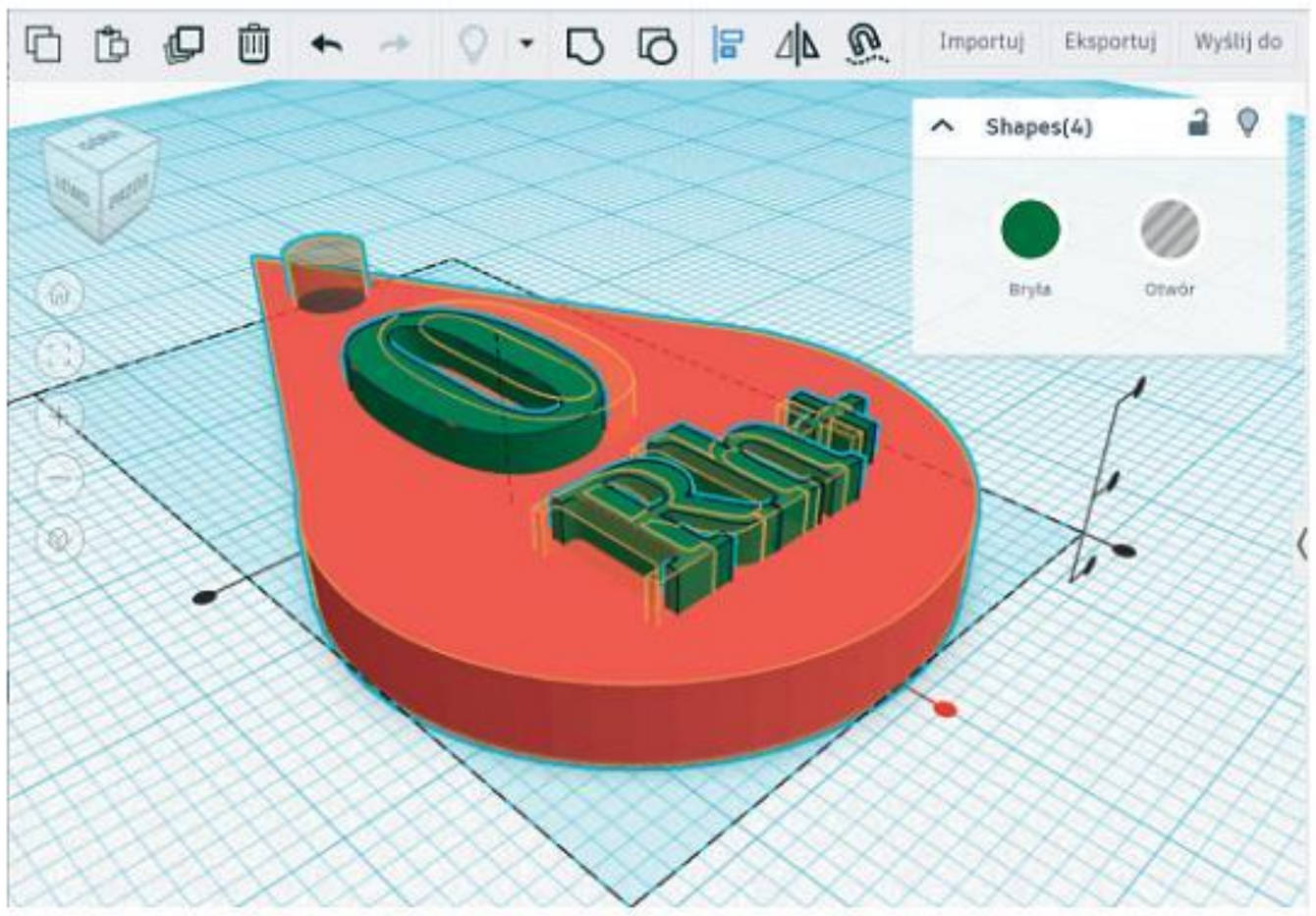
Ćwiczenie 17. Grupujemy kształty

1. Umieść kształty **Walec** i **Dach** tak jak pokazano na rysunku 16a, aby utworzyły nowy kształt o wymiarach 50 mm x 30 mm x 4 mm (rys. 16a).
2. Zaznacz obydwa kształty i zgrupuj je. Zmień kolor powstałej bryły na czerwony (rys. 16b).

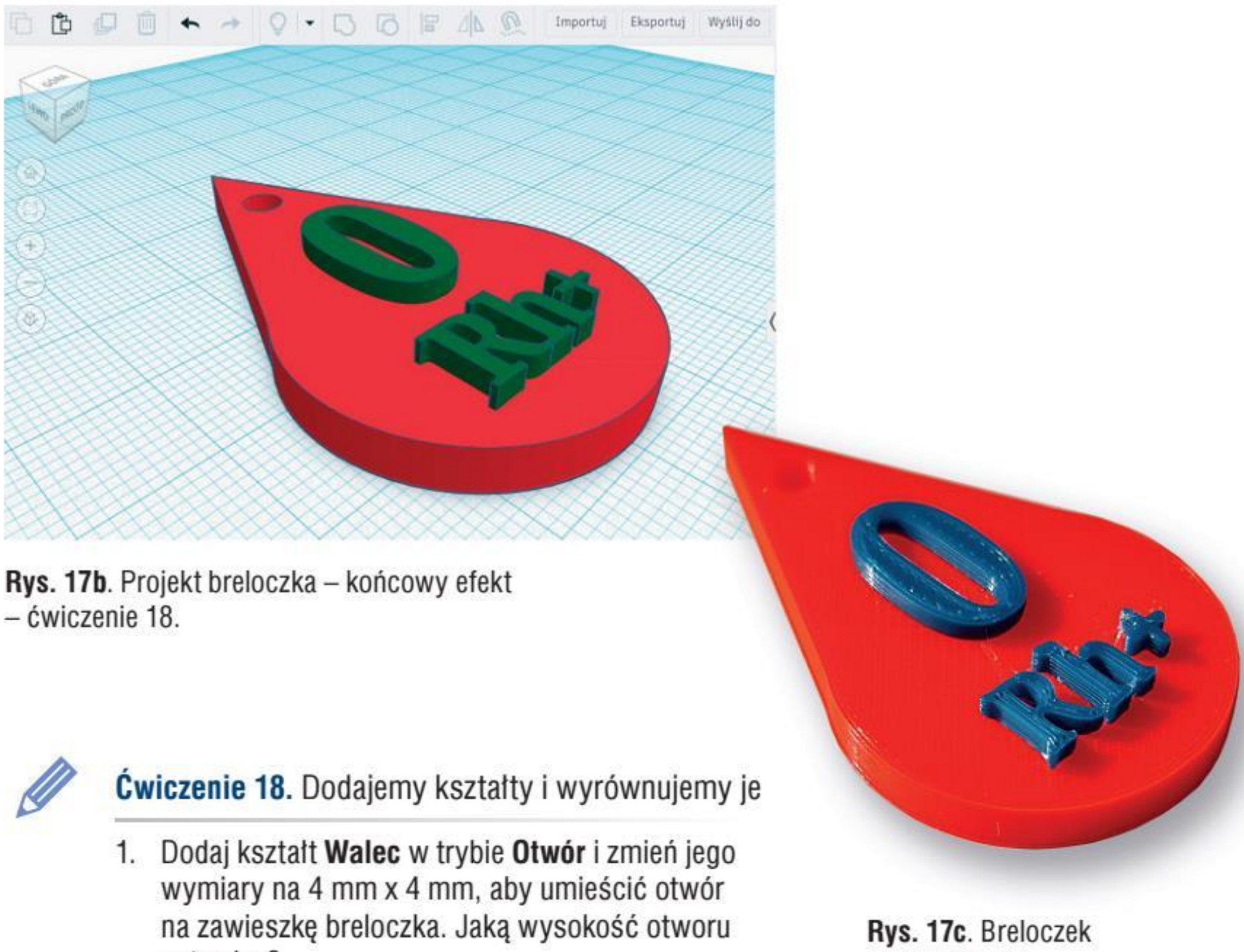
Wskazówka: Przyciągnij kształt **Dach** w kierunku **Walca** bokiem o długości 28 mm.

W breloczku potrzebne są otwór na zawieszkę od kluczy oraz napis z grupą krwi. Po umieszczeniu odpowiednich kształtów należy je wyrównać do środka.

Aby wyrównać kształty, należy je zaznaczyć, trzymając wciśnięty klawisz **Shift**, a następnie wybrać opcję **Wyrównaj** . Po wybraniu tej opcji pojawią się osie oznaczone czarnymi punktami. Należy wybrać oś, względem której mają być wyrównane kształty (rys. 17a).



Rys. 17a. Wyrównywanie kształtów – wybrana oś wyrównania (czerwona kropka) – ćwiczenie 18.



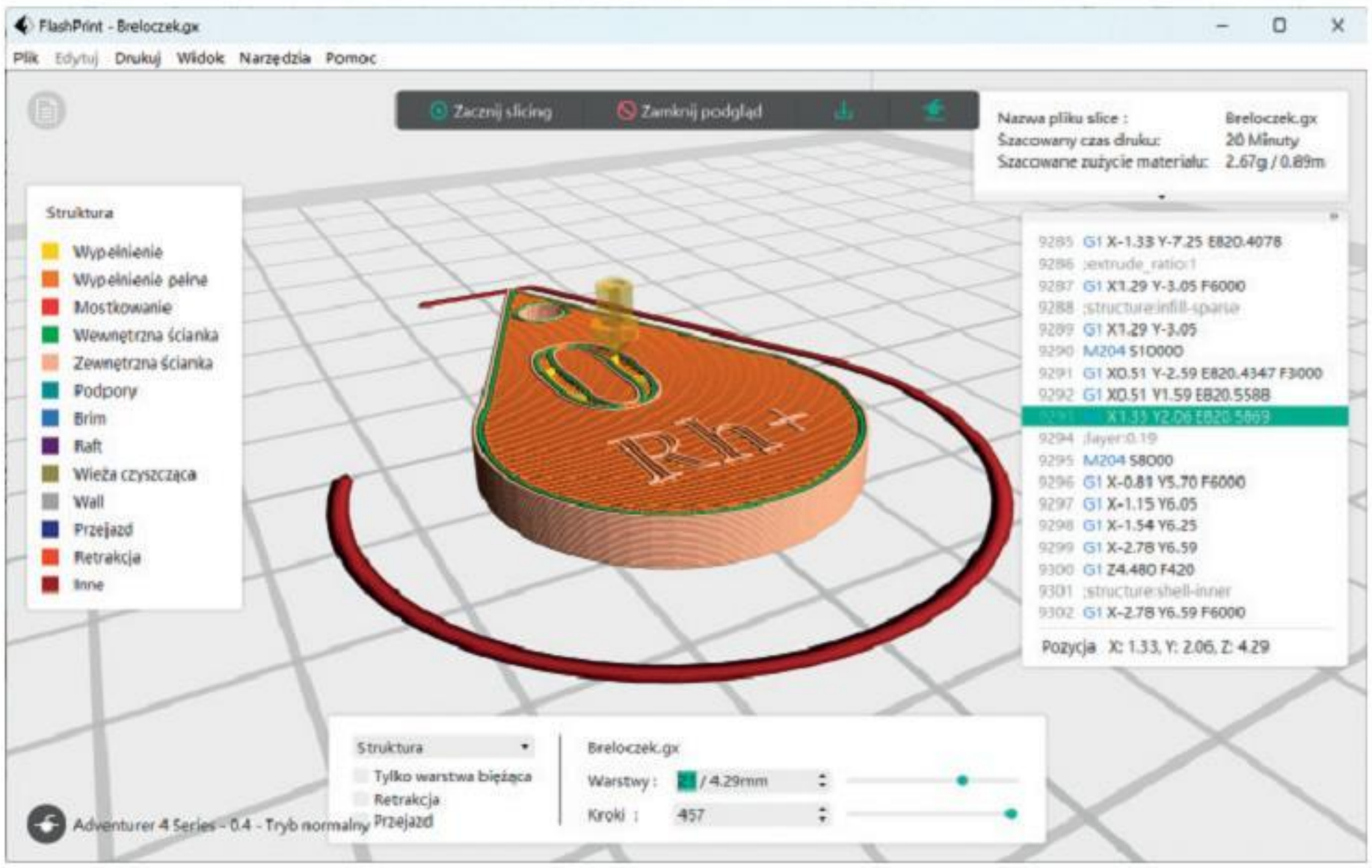
Rys. 17b. Projekt breloczka – końcowy efekt – ćwiczenie 18.

Rys. 17c. Breloczek wydrukowany na drukarce 3D

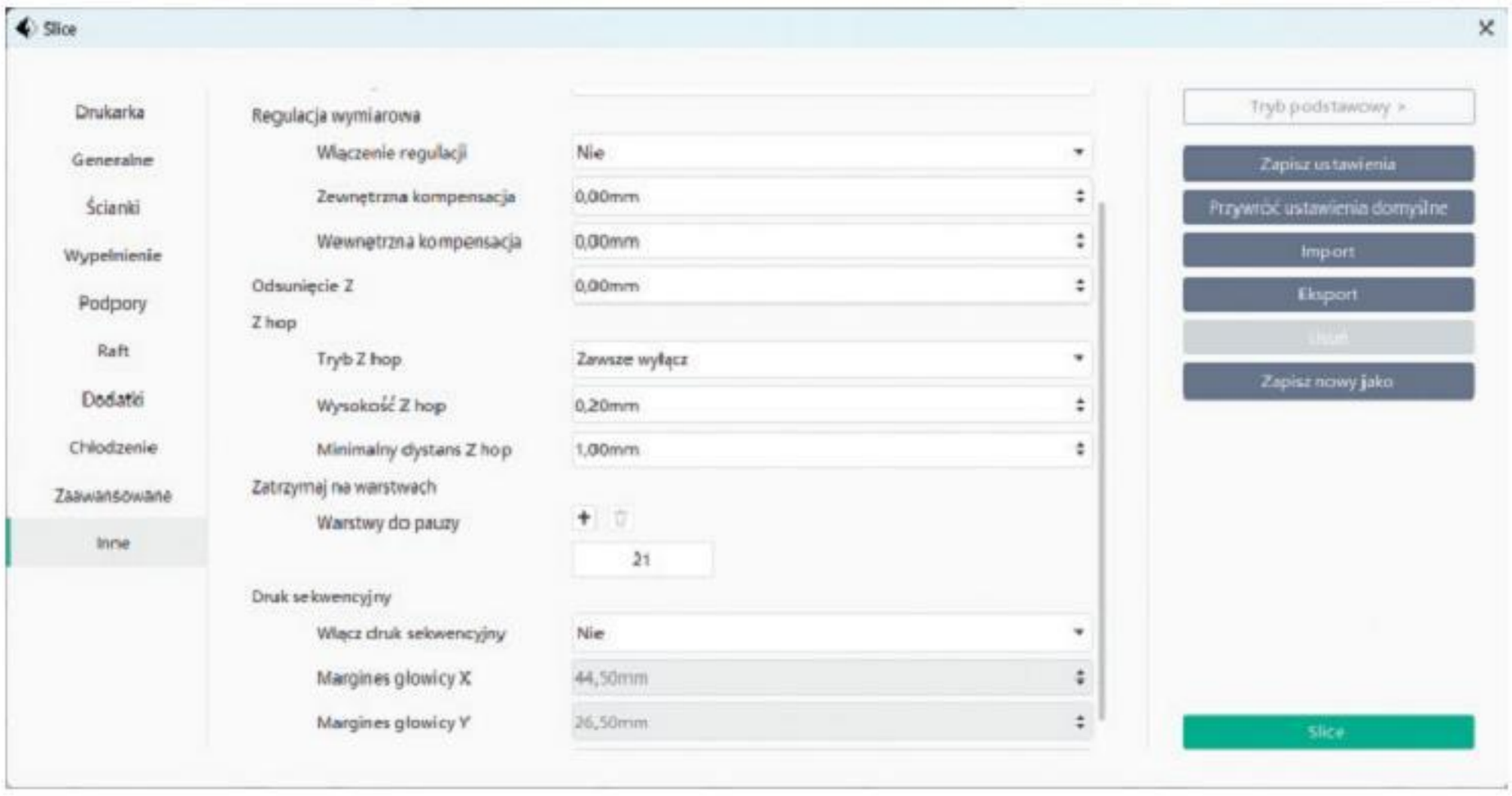
Ćwiczenie 18. Dodajemy kształty i wyrównujemy je

1. Dodaj kształt **Walec** w trybie **Otwór** i zmień jego wymiary na 4 mm x 4 mm, aby umieścić otwór na zawieszkę breloczka. Jaką wysokość otworu ustawisz?
2. Dodaj kształt **Tekst** i zapisz symbol swojej grupy krwi. Jeśli go nie znasz, wpisz „0 Rh+”. Wysokość napisu ustaw na 2 mm.
3. Wyśrodkuj wszystkie elementy, stosując opcję **Wyrównaj**.
4. Pobierz plik *Breloczek* na swój komputer i zapisz go z rozszerzeniem *stl*.

Projekt możemy teraz przetworzyć w programie typu slicer, a następnie wydrukować na drukarce 3D. Aby uzyskać dwukolorowy efekt, jak na rys. 17c, należy w programie typu slicer sprawdzić, po której warstwie następuje druk napisów. Następnie podczas druku na tej warstwie należy zatrzymać druk i zmienić filament. Można to zrobić ręcznie lub w programie typu slicer dodać odpowiednie ustawienia, np. „Zatrzymaj na warstwach".



Rys. 18a. Przykład podglądu pociętych warstw w slicerze



Rys. 18b. Ustawienia slicera – zatrzymanie druku po wydrukowaniu 21. warstwy w celu zmiany koloru filamentu

Ćwiczenie 19. Sprawdzamy pocięte warstwy projektu 3D

1. Uruchom wybrany program typu slicer. Otwórz plik *Breloczek.stl* zapisany w ćwiczeniu 18.
2. Przygotuj do druku model i sprawdź, przy której warstwie drukarka zacznie drukować tekst.
3. Zapisz pocięty na warstwy model w pliku z odpowiednim rozszerzeniem.
4. Wydrukuj model, jeśli masz taką możliwość.

Warto zapamiętać

- Prezentacja multimedialna składa się ze slajdów, z których każdy jest osobnym ekranem pokazu. Slajdy mogą zawierać różne obiekty, m.in. dźwięki, obrazy, filmy. Tekst na slajdach umieszcza się w polach tekstowych.
- Należy dbać o poprawność redakcyjną tekstu oraz dobierać parametry czcionki tak, aby tekst był czytelny i dobrze widoczny na slajdzie.
- Prezentację można wzbogacić efektami wizualnymi i dźwiękowymi. Trzeba jednak stosować je z umiarem. Jeśli będzie ich zbyt dużo, mogą rozpraszać uwagę oglądających.
- Podstawową jednostką w opracowywaniu filmu jest scena, która jest fragmentem filmu ograniczonym czasowo lub np. zmianą ujęcia.
- Film nagrany np. smartfonem i zapisany w postaci cyfrowej można edytować, używając odpowiedniej aplikacji lub serwisu internetowego (np. programu Canva), m.in.: przyciąć klip wideo, podzielić klip wideo na mniejsze części oraz zmienić ich kolejność i szybkość odtwarzania, dodać tytuły, ścieżkę dźwiękową i inne efekty specjalne.
- W programie Tinkercad możemy zaprojektować model 3D i wyeksportować go do druku na drukarce 3D.

Pytania i polecenia

1. Masz 15 minut na przedstawienie prezentacji multimedialnej na temat programowania. Z ilu maksymalnie slajdów powinna się składać piętnastominutowa prezentacja? Uzasadnij odpowiedź.
2. Czym różnią się animacje z grupy **Wejścia** od animacji z grupy **Wyjścia**? Podaj po jednym przykładzie zastosowania animacji z każdej grupy.
3. Jaką funkcję w prezentacji multimedialnej pełnią hiperłącza?
4. W jaki sposób należy dodać tytuł i napisy w filmie opracowywanym w serwisie Canva? Odszukaj odpowiednie opcje.
5. Jak wyrównać obiekty umieszczone na slajdzie prezentacji?

Zadania

1. Przygotuj prezentację multimedialną na wybrany temat, np.: moja rodzina, moja klasa, moja szkoła, wycieczka szkolna, wakacje.
2. W programie Tinkercad utwórz model 3D pamiątki, która przypomni ci wakacje, np. muszelkę znad morza, szyszkę z lasu, kapelusz góralski, kuter rybacki. Jeśli masz taką możliwość, wydrukuj zaprojektowany model 3D.

Przykłady innych projektów grupowych

Utwórz prezentację multimedialną lub film na wybrany temat:

1. Unia Europejska – cele integracji europejskiej, członkowie Unii Europejskiej, instytucje UE (Rada Europejska, Rada Unii Europejskiej, Parlament Europejski, Komisja Europejska).
2. Walory turystyczne i kulturowe wybranego regionu geograficznego, np. regionu, w którym mieszkasz.
3. Parowozownia w Wolsztynie.
   Uwaga: Jest to czynna od ponad 100 lat parowozownia, w której można zobaczyć lokomotywy parowe (parowozy) nadal wykorzystywane w transporcie. Znajdź w Internecie więcej informacji na ten temat.
4. Sąsiedzi Polski.
5. Wybrane regiony świata.
6. Wielkie odkrycia geograficzne, m.in. wyprawy Krzysztofa Kolumba, Vasco da Gamy, Ferdynanda Magellana.
7. Polska pierwszych Piastów.
8. Budowa i funkcjonowanie organizmu człowieka.
9. Figury płaskie – pola i obwody.
10. Ruch prostoliniowy.
11. Internet jako sieć o zasięgu globalnym.
12. Warunki życia w średniowiecznym mieście i na wsi.

Temat 11. Historia i rozwój informatyki – projekt

1. Przykłady dawnych przyrządów i maszyn do wykonywania obliczeń
2. Pierwsze komputery
3. Zadanie projektowe – historia i rozwój informatyki

Warto powtórzyć

1. Z czego składa się podstawowy zestaw komputerowy?
2. Na czym polega działanie komputera?
3. W jaki sposób dane liczbowe są reprezentowane w komputerze?

1. Przykłady dawnych przyrządów i maszyn do wykonywania obliczeń

Zanim powstało pierwsze urządzenie, które można nazwać komputerem, konstruowano różne urządzenia do wykonywania obliczeń, m.in. liczydła i kalkulatory. Potrzeba i umiejętność liczenia pojawiły się, kiedy nasi przodkowie żyli w grupach plemiennych, a rozwinęły wraz z powstaniem pierwszych państw – zorganizowanych form społecznych. Wtedy też powstały pierwsze urządzenia ułatwiające wykonywanie obliczeń.

Już od X wieku przed naszą erą znano przyrządy do liczenia – **abakusy** (rys. 1.), zbliżone wyglądem do liczydeł. Abakusy to gliniane tabliczki z wyżłobionymi rowkami, w których umieszczano przesuwane kamyki. Są one maszynami liczącymi używanymi najdłużej w historii.

Odmianą abakusa jest **soroban** (rys. 2.), który stosowano od XV wieku w Japonii. Liczymy na nim, przesuwając koraliki w określony sposób – do lub od belki poprzecznej. Mimo powszechnego zastosowania komputerów, w Japonii nadal korzysta się z sorobanów do wykonywania obliczeń.

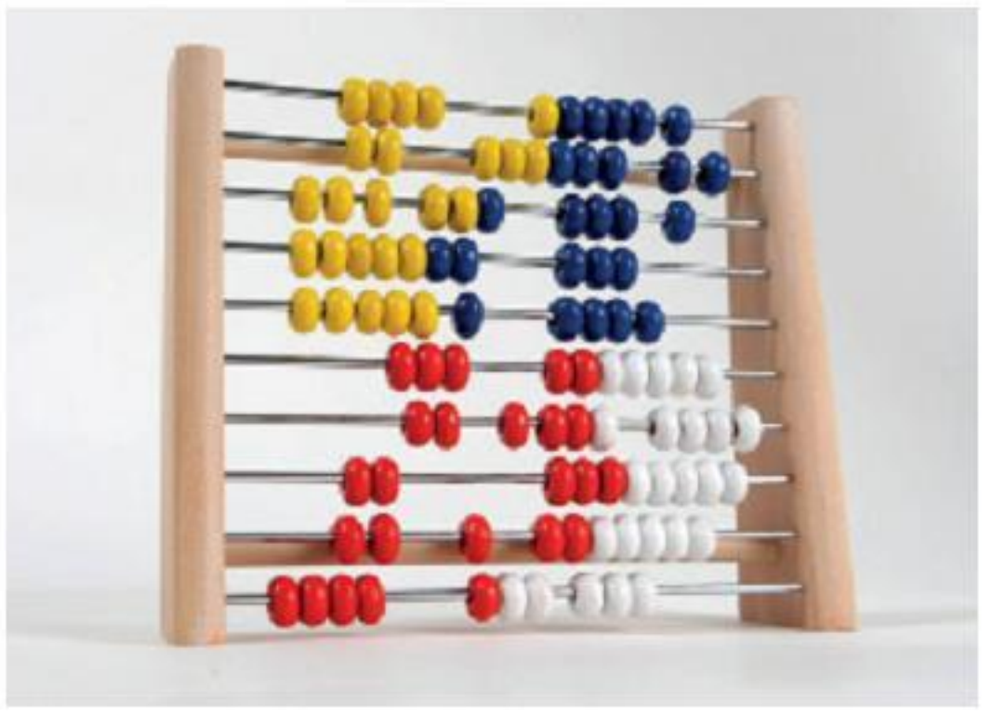
Na podobnej zasadzie jak abakusy i sorobany działa popularne do dziś liczydło (rys. 3.), na którym liczy się, przesuwając w określony sposób koraliki na prętach.



Rys. 1. Rekonstrukcja rzymskiego abakusa



Rys. 2. Japoński soroban



Rys. 3. Liczydło

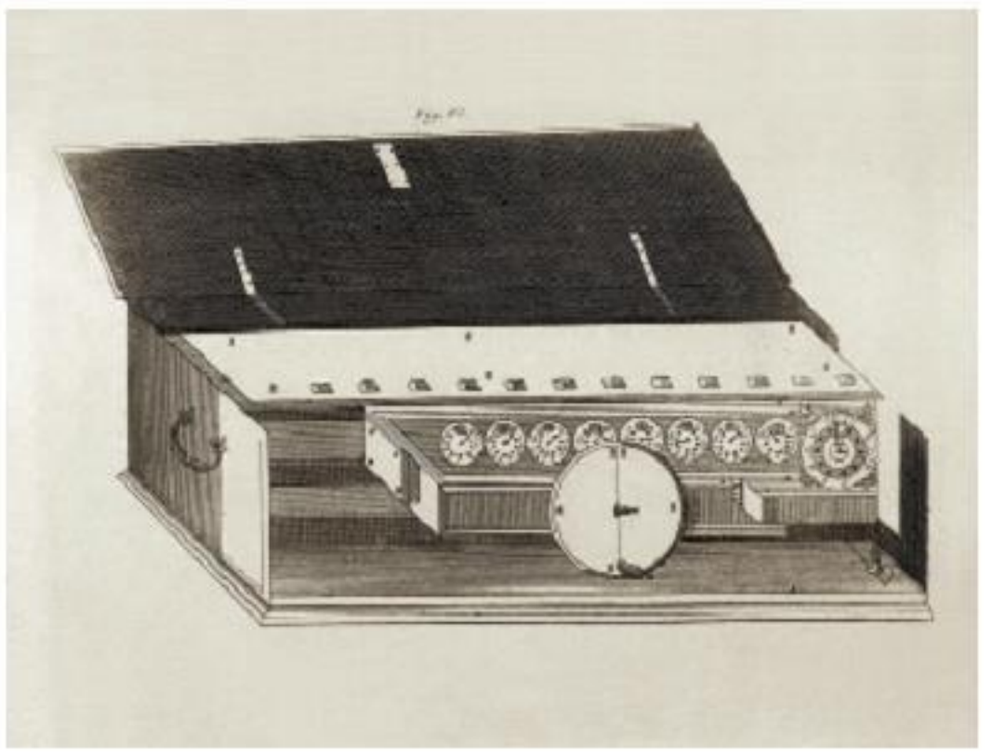
Nowe urządzenia liczące zaczęły powstawać w XVII wieku dzięki rozwojowi techniki, a szczególnie mechaniki precyzyjnej. Najważniejszym elementem ich konstrukcji były obracające się koła zębate.

Jedną z pierwszych na świecie maszyn wykonujących cztery działania (dodawanie, odejmowanie, mnożenie i dzielenie liczb całkowitych) wynalazł w 1623 roku niemiecki matematyk **William Schickard** – urządzenie wykonał z drewnianych elementów (rys. 4.).

Niedługo później (w latach 1672-1694) powstał mechanizm zastosowany w kalkulatorze opracowanym przez niemieckiego uczonego **Wilhelma Leibniza** (rys. 5.) – wykorzystywano go do budowy różnych urządzeń aż do XX wieku.



Rys. 4. Replika maszyny Williama Schickarda



Rys. 5. Kalkulator Wilhelma Leibniza

Kalkulatory z XVII i XVIII wieku zadziwiały nie tylko precyzją wykonania, ale również elegancką formą.

Wynalazek elektryczności i rozwój maszyn elektrycznych pozwoliły na zastąpienie korbek i dźwigni zespołem silników i przekaźników elektrycznych.

2. Pierwsze komputery

Co te dawne maszyny mają wspólnego z dzisiejszymi komputerami wykonującymi nie tylko bardzo skomplikowane obliczenia, lecz także generującymi ruchome obrazy czy komponującymi muzykę? U podstaw działania współczesnych komputerów leżą te same proste operacje matematyczne i logiczne. Komputer potrafi jednak wykonywać je bardzo szybko (miliardy operacji na sekundę).

Wszystko, co wykonuje komputer, wymaga opisu językiem matematyki i logiki. Opis ten stanowią programy, podprogramy, procedury sprowadzone do poziomu tysięcy czy milionów najprostszych działań.

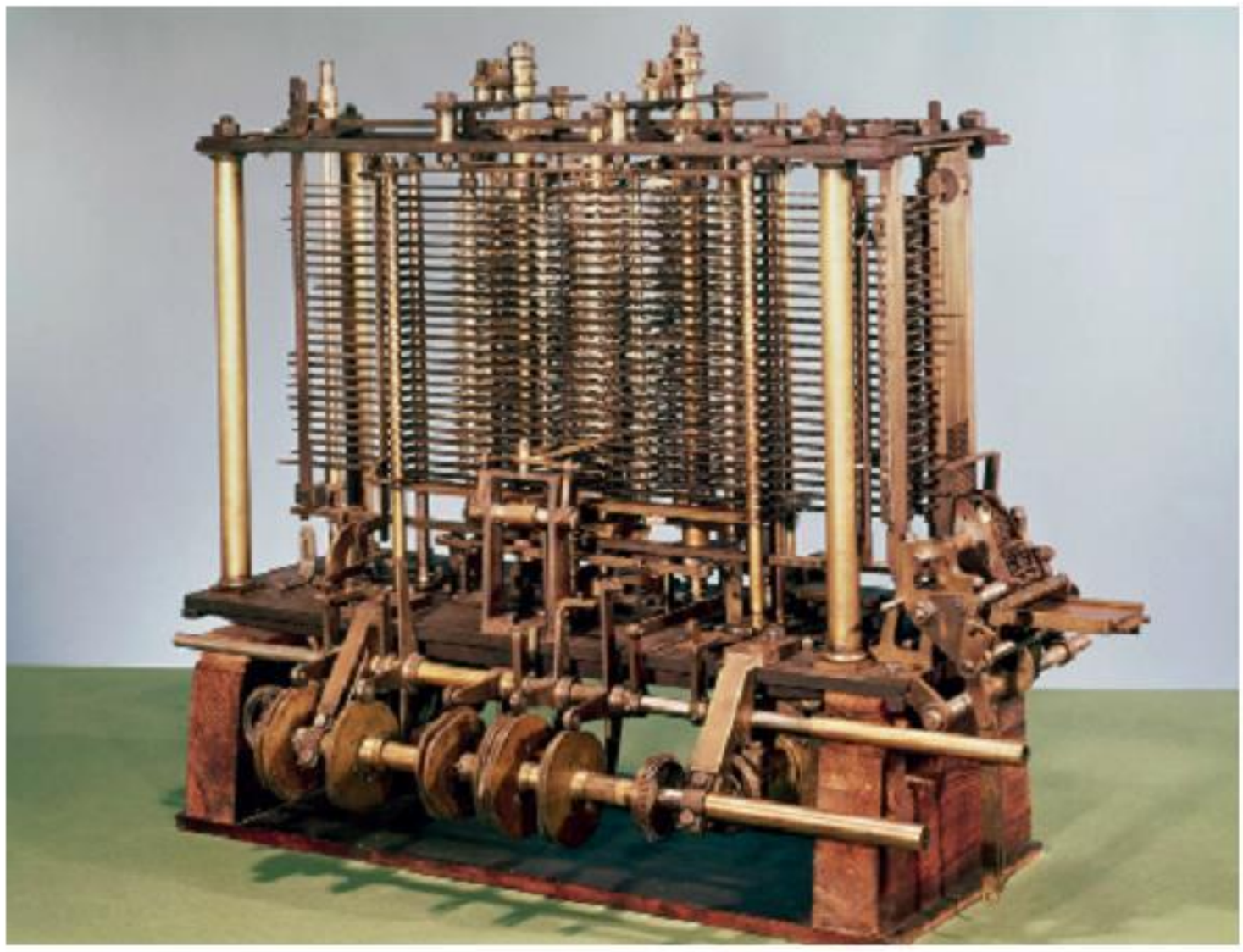
1833

Angielski matematyk **Charles Babbage** opracowuje model maszyny analitycznej napędzanej silnikiem lokomotywy parowej (rys. 6.).

Maszyna miała wykonywać dowolne ciągi instrukcji oraz operować na danych. Wprowadzanie i wyprowadzanie danych przebiegało na **kartach dziurkowanych**.

Mimo że maszyna nigdy nie została wykonana w całości, jej model pomógł w opracowaniu koncepcji współczesnych komputerów.

Z Charlesem Babbage'em współpracowała Ada Lovelace, którą wielu uważa za pierwszą programistkę komputerów. Tworzone przez lady Lovelace opisy rozwiązywania konkretnych zadań obliczeniowych uznaje się za pierwsze programy. Nazwa języka programowania Ada (który powstał w latach 70. XX wieku) pochodzi właśnie od jej imienia.



Rys. 6. Model fragmentu maszyny analitycznej Charlesa Babbage'a

1911

W roku 1911 powstaje amerykańskie przedsiębiorstwo **IBM**, na początku znane pod nazwą CTR. Nazwę IBM przedsiębiorstwo przyjęło dopiero w 1924 roku.

IBM to jedno z najstarszych przedsiębiorstw informatycznych na świecie. Co ciekawe, początkowo firma nie zajmowała się produkcją komputerów, bo ich jeszcze nie wynaleziono. Przedsiębiorstwo wytwarzało różne produkty, m.in. systemy pomiaru czasu pracy, wagi, krajalnice do mięsa i **karty perforowane**.

Poza wyprodukowaniem komputera osobistego IBM PC (1981), firma opracowała różne technologie używane w informatyce, tj. dysk twardy Winchester, kursor oraz dyskietkę, a także tworzyła systemy operacyjne.

1936

Angielski matematyk **Alan Turing** opracowuje model teoretyczny maszyny do wykonywania algorytmów.

Model ten składał się z nieskończenie długiej taśmy zawierającej komórki z przetwarzanymi symbolami, głowicy odczytująco-zapisującej i układu sterowania. Obliczenia wykonywane za pomocą tej maszyny zależały od układu symboli wpisanych na taśmie oraz od przyjętego zestawu instrukcji. Podobnie działają dzisiejsze komputery – wyniki obliczeń zależą od danych zapisanych w pamięci komputera i od zestawu wykonanych instrukcji.

1941

Niemiecki inżynier **Konrad Zuse** konstruuje programowalną maszynę **Z3** (rys. 7.), w której zastosował **przekaźniki elektromagnetyczne**.

Z3 wykonywała operacje matematyczne (tj. dodawanie, odejmowanie, mnożenie, dzielenie i pierwiastkowanie) w **systemie binarnym**. Działaniem maszyny sterował program zapisany na **taśmie perforowanej**. Dane wprowadzano z **klawiatury**, a wynik odczytywano z **wyświetlacza**.

Maszynę Z3 określa się czasem jako pierwszy komputer, choć naprawdę był to programowalny kalkulator.



Rys. 7. Komputer Z3 Konrada Zuse

1944

W Stanach Zjednoczonych powstaje komputer **Harvard Mark I** (rys. 8.) – największy w historii kalkulator **elektromechaniczny** o długości prawie 16 m i wysokości 2,5 m. Ważył ponad 5 ton i zawierał ponad 800 km przewodów z trzema milionami połączeń.

Była to pierwsza maszyna, której pracą sterował **wbudowany program**. Jedną z programistek maszyny Harvard Mark I była Grace Hopper, która wprowadziła do słownictwa informatycznego pojęcie „bug" oznaczające dziś błąd w kodzie programu. Gdy pracowała przy kolejnej wersji komputera – Mark II, do wnętrza urządzenia dostała się ćma i spowodowała awarię. Naprawę tej usterki pracownicy określili jako „debugging", czyli odpluskwianie.



Rys. 8. Komputer Harvard Mark I

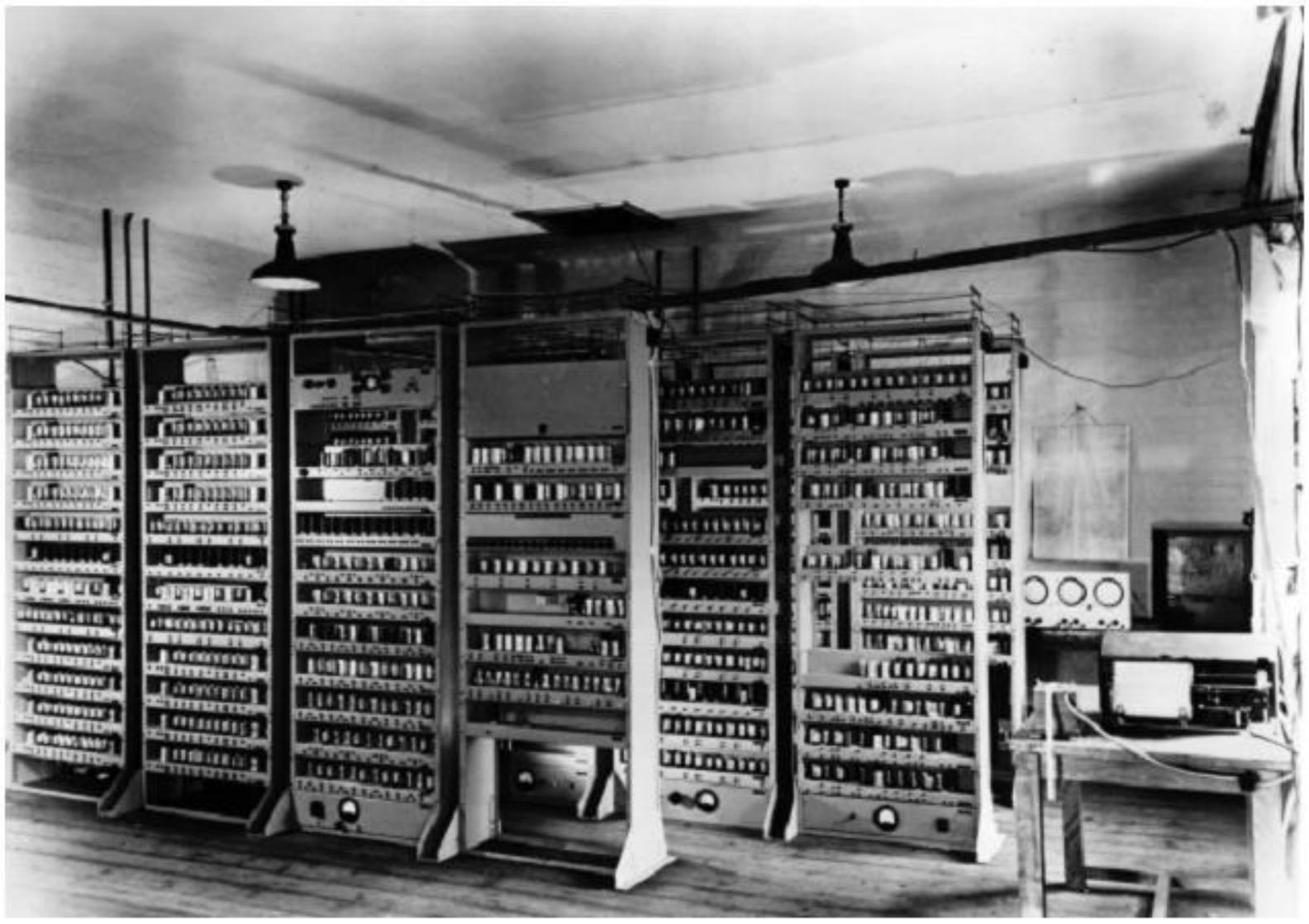
Po wielu latach, w których powstawały maszyny mechaniczne, wreszcie pojawia się **komputer elektroniczny**.

1945

W Stanach Zjednoczonych powstaje pierwszy w historii **komputer elektroniczny** o nazwie **ENIAC** (rys. 9.).

Dotychczas używane koła zębate, przekaźniki i silniki zastąpiono **lampowymi układami elektronicznymi**. Do budowy wykorzystano prawie 20 tysięcy **lamp elektronowych**, a do zasilania tego komputera zbudowano oddzielną elektrownię. Średni bezawaryjny czas pracy maszyny wynosił… 2 godziny, ponieważ po tym czasie któraś z lamp ulegała przepaleniu i należało ją wymienić.

ENIAC ważył ponad 27 ton i zajmował powierzchnię ok. 140 metrów kwadratowych. Komputer ten nie miał pamięci operacyjnej. Na zdjęciach w Internecie można zobaczyć plątaninę kabli na ścianie komputera, ponieważ ENIAC był programowany przez przełączanie wtyków kablowych.



Rys. 9. Pierwszy komputer elektroniczny – ENIAC

1948

Powstaje **tranzystor**, którego masową produkcję rozpoczęto w **1954** roku. Pozwoliło to na znaczną miniaturyzację kolejnych maszyn.

Na przerzutnikach tranzystorowych budowano pierwsze polskie komputery z **serii Odra**.

1948

Amerykański matematyk **John von Neumann** opracowuje (korzystając z modelu Charlsa Babbage'a) koncepcję logiczną i proponuje architekturę komputerów stosowaną do dziś.

Idea maszyny von Neumanna:

- programy i dane mogą posiadać taką samą postać,
- programy i dane mogą być przechowywane w tej samej pamięci.

1954

Powstaje **FORTRAN** – pierwszy język programowania wysokiego poziomu.

1968

Powstaje firma **Intel** zajmująca się opracowywaniem i produkcją **układów scalonych** (integrujących w jedną funkcjonalną całość proste elementy logiczne: przerzutniki, rejestry i bramki logiczne) różnego zastosowania.

Opracowanie pierwszych układów scalonych to kolejny krok postępu technologicznego.

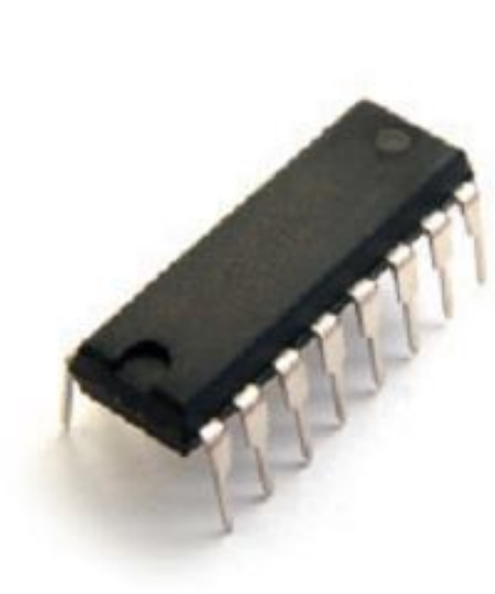
1971

Firma Intel opracowuje pierwszy mikroprocesor **Intel 4004**.

W 1971 roku pewna japońska firma zleciła Intelowi opracowanie i wyprodukowanie kilkunastu układów scalonych do produkowanych przez siebie kalkulatorów. Konstruktorzy zespołu, któremu zlecono projekt, zauważyli, że zamiast kilkunastu różnych układów można wykonać jeden. Natomiast wymagane funkcje można realizować za pomocą krótkiego programu umieszczonego w wewnętrznej pamięci typu ROM. Rozwiązanie to znacznie uprościło proces technologiczny i obniżyło koszty, a na dodatek przez modyfikację programu można było przystosować to rozwiązanie do wielu różnych funkcji. Tak powstał pierwszy na świecie czterobitowy **mikrokontroler 4004**, protoplasta współczesnych mikroprocesorów (rys. 10.). Zawierał ok. 2300 tranzystorów.

Ani zleceniodawca, ani firma Intel nie docenili jednak wagi tego wynalazku. Było to powodem rozstania się zespołu konstruktorów z Intelem. Założyli oni własną firmę pod nazwą ZILOG, tę samą, która później produkowała procesory Z80, m.in. do popularnych mikrokomputerów Sinclair.

Intel powrócił jednak do swego wynalazku. Szybko powstała nowsza, ośmiobitowa wersja procesora, oznaczona symbolem 8008, a w roku **1974** – uniwersalny mikroprocesor **8080** – serce wielu przyszłych komputerów.



Rys. 10. Mikrokontroler Intel 4004 – protoplasta współczesnych mikroprocesorów



Rys. 11. Współczesny procesor Intel Core i7

Rozpoczyna się nowa era w historii komputerów – era **komputerów osobistych**.

1975

Firma MITS produkuje **Altair 8800** (rys. 12.) – pierwszy mikrokomputer oparty na procesorze 8080.

Początkowo był on dostępny w zestawach do samodzielnego montażu. Kosztował znacznie mniej niż wcześniejsze urządzenia – poniżej 500 dolarów. Komputer ten nie posiadał klawiatury. Programowano go za pomocą przełączników na płycie czołowej.

Komputer Altair to również początek kariery **Billa Gatesa**, późniejszego założyciela Microsoftu. Wówczas, jako 18-letni student, był współautorem prostej wersji języka programowania BASIC, napisanej dla Altaira.



Rys. 12. Komputer Altair 8800

1976

Powstaje firma **Apple**. Pierwszym produktem firmy był komputer osobisty **Apple I** (rys. 13.). Funkcję monitora pełnił domowy odbiornik telewizyjny.



Rys. 13. Pierwszy komputer firmy Apple

1981

Firma SINCLAIR produkuje komputer **ZX81** (rys. 14a) z 16 KB pamięci operacyjnej i zaimplementowanym interpreterem języka BASIC.

Komputery ZX81 i następny model **ZX Spectrum** (rys. 14b) były bardzo popularne w Polsce.



Rys. 14a. Komputer ZX81



Rys. 14b. Komputer ZX Spectrum

1981

Firma IBM przedstawia pierwszy komputer osobisty z serii **IBM PC** (rys. 15.), który wyprodukowała w ciągu roku. Potem został on zastąpiony bardziej rozbudowanymi modelami: XT i AT.

Komputer miał od 16 do 256 KB pamięci operacyjnej, monochromatyczny monitor oraz stację dyskietek 5,25 cala (nie miał jeszcze dysku twardego). System operacyjny DOS do tego komputera napisała firma Microsoft, której właścicielem był 25-letni Bill Gates.



Rys. 15. Pierwszy komputer IBM PC 5150



Rys. 16. Komputer Macintosh firmy Apple z interfejsem graficznym

1982-1985

Początki **Internetu** – do powszechnego użytku wchodzi protokół TCP/IP.

1984

Firma Apple zaczyna masowo sprzedawać pierwszy komputer z interfejsem graficznym – Macintosh (rys. 16.) i… odnosi komercyjny sukces.

Od 2001 komputery Apple sprzedaje się z systemem operacyjnym Mac OS X.

1990

Pojawia się **MS Windows 3.0** – graficzna nakładka na system operacyjny DOS.

1991

Powstają pierwsze strony WWW oraz pojawia się pierwsza wersja systemu operacyjnego **Linux**.

Wielozadaniowy, 32-bitowy system, dzięki udostępnieniu kodu źródłowego w Internecie, szybko zyskał entuzjastów. Tysiące programistów na całym świecie zaczęło bezinteresownie pracować nad jego rozwojem.

1995-2000

Następuje eksplozja usług internetowych. Pojawiają się e-sklepy, e-banki, e-szkoły, kawiarenki internetowe, komunikatory internetowe, Internet mobilny, telefonia VoiP.

Minęło wiele lat od chwili, kiedy von Neumann przedstawił koncepcję, według której do dziś buduje się komputery. Zmieniły się technologie, ale komputery w istocie pozostały jedynie liczydłem, tyle że niezwykle rozbudowanym i bardzo szybkim, a inteligencja komputerów ciągle jest domeną literatury SF.

Czy wiesz, że:

w 1998 roku rzecznik Biura Patentów USA Charles H. Duell powiedział: „Wszystko, co było do wynalezienia, już wynaleziono"? Wkrótce potem nastąpił burzliwy rozwój nauki i techniki, który trwa do dziś…

Od 2001 do… dziś

W przytoczonych faktach dotyczących historycznego rozwoju informatyki i technologii nie uwzględniono XXI wieku. Nie ujęto m.in. zagadnień, z którymi spotkaliście się na zajęciach z informatyki w ostatnich latach nauki, a niektóre z nich poznaliście też w praktyce. Możecie je opracować samodzielnie w ramach zadań projektowych. Są to m.in.:

- komputery przenośne (laptopy, tablety),
- urządzenia mobilne,
- chmura obliczeniowa,
- urządzenia GPS,
- roboty,
- portale społecznościowe,
- sztuczna inteligencja.

3. Zadanie projektowe „Historia i rozwój informatyki"

Przykładowy podział zadania na zadania szczegółowe

Uwagi:

- Jeśli wykorzystujesz zasoby pobrane z Internetu (zwłaszcza zdjęcia), pamiętaj o przestrzeganiu zasad korzystania z cudzych materiałów. Bez ograniczeń możesz korzystać z grafiki i innych utworów w tzw. domenie publicznej.
- Możesz zaplanować również dodatkowe zadania szczegółowe i wybrać odmienny sposób realizacji zadań.

Zadanie 1. Odszukaj w Internecie więcej informacji na temat abakusa i wykonywania obliczeń za jego pomocą. Przygotuj krótką prezentację multimedialną na ten temat (pięć slajdów).

Zadanie 2. Odszukaj w Internecie więcej informacji na temat sorobanu i wykonywania obliczeń za jego pomocą. Przygotuj krótką prezentację multimedialną na ten temat (pięć slajdów). Zastanów się, dlaczego wciąż używa się tych przyrządów do liczenia, skoro istnieją szybkie komputery.

Zadanie 3. Dowiedz się, jak liczyć na liczydle. Przynieś na lekcję liczydło i pokaż, jak go używać. Porównaj sposób liczenia na liczydle z wykonywaniem obliczeń na abakusie i sorobanie. Wskaż podobieństwa i różnice.

Zadanie 4. Opracuj w wybranej formie temat: „Maszyny liczące Schickarda i Leibniza a współczesne kalkulatory". Znajdź w Internecie więcej informacji na temat pierwszych kalkulatorów (zdjęcia, opisy działania). Porównaj działanie dawnych kalkulatorów do współczesnych. Możesz przynieść na lekcje przykładowy kalkulator.

Zadanie 5. Kalkulatorów znanych pod nazwą „kręciołki", wynalezionych w 1930 r., używano w biurach jeszcze pod koniec lat 60. XX wieku. Znajdź w Internecie lub innych źródłach więcej informacji na temat tych urządzeń (zdjęcia, opis działania). Wyjaśnij, dlaczego nazywano je „kręciołkami".

Przygotuj krótką prezentację multimedialną na ten temat. Dobierz odpowiednią liczbę slajdów do prezentowanego materiału.

Zadanie 6. W niektórych opracowaniach Charles Babbage nazywany jest „ojcem komputerów", a Ada Lovelace – „pierwszą programistką komputerów". Korzystając z Internetu i innych źródeł, uzasadnij celowość tych określeń. Poszukaj również informacji na temat karty dziurkowanej (opis, zdjęcia). Dowiedz się, na czym polegał zapis na takiej karcie. Przygotuj krótką prezentację multimedialną na ten temat. Dobierz odpowiednią liczbę slajdów do prezentowanego materiału.

Zadanie 7. Skorzystaj z Internetu i innych źródeł, żeby dowiedzieć się więcej na temat maszyn, których nazwy zaczynają się na literę Z, a ich konstruktorem był Konrad Zuse. Wyjaśnij, czym są przekaźniki elektromagnetyczne. Znajdź również informacje (opis, zdjęcia) na temat taśmy perforowanej. Dowiedz się, na czym polegał zapis na takiej taśmie. Przygotuj krótką prezentację multimedialną na ten temat. Dobierz odpowiednią liczbę slajdów do prezentowanego materiału.

Zadanie 8. Skorzystaj z Internetu i innych źródeł, żeby dowiedzieć się więcej na temat komputerów Mark I i ENIAC. Porównaj oba urządzenia. Uwzględnij w porównaniu wielkość, budowę, ogólny sposób działania. Przedstaw wyniki porównania w tabeli.

Zadanie 9. Sprawdź wymiary i wagę komputera, którego używasz, a następnie oblicz, ile razy są mniejsze od wymiarów i wagi pierwszych komputerów. Do tabeli z zadania 8. dołącz informacje o współczesnym komputerze. Pod tabelą umieść wnioski (trzy zdania).

Zadanie 10. Przedstaw na kilku slajdach prezentacji multimedialnej rozwój informatyki w latach 70. XX wieku.

Zadanie 11. Przedstaw na kilku slajdach prezentacji multimedialnej rozwój informatyki w latach 80. XX wieku. Dodaj na końcu wnioski – porównaj osiągnięcia z lat 70. i 80.

Zadanie 12. Znajdź w Internecie (i innych źródłach) informacje na temat różnych pomysłów na komputer, np. o komputerze kwantowym. Opisz w edytorze tekstu (na jednej stronie), na czym polega dany pomysł. Jeśli znajdziesz w Internecie zdjęcia, dołącz je do dokumentu.

Zadanie 13. Utwórz prezentację multimedialną o historii rozwoju Internetu. Uwzględnij daty powstania: sieci ARPANET (1969), poczty elektronicznej (1972), protokołu TCP/IP (1983), serwisu WWW (1991). Znajdź w Internecie więcej informacji na ten temat. Uwzględnij również współczesną powszechność dostępu do usług internetowych, podając przykłady. Dobierz odpowiednią liczbę slajdów do prezentowanego materiału.

Zadanie 14. Przedstaw w wybranej postaci (np. w formie referatu, osi czasu, prezentacji multimedialnej) chronologicznie uporządkowane najważniejsze wydarzenia z rozwoju informatyki i technologii po 2000 roku.

Zadanie 15. Znajdź w Internecie więcej informacji na temat sztucznej inteligencji. Opracuj to zagadnienie w wybranej formie. Uwzględnij w opracowaniu ciekawostki podane na końcu tematu i rozwiń ich treść.

Zadanie 16. Opracuj w wybranej formie temat coraz powszechniej stosowanych robotów i wykorzystania komputerów do sterowania nimi.

Uwagi do realizacji zadań szczegółowych

1. Podczas rozwiązywania zadań 1-3 warto ze sobą współpracować, wymieniając się informacjami.
2. Zadania 4. i 5. można wykonywać wspólnie, dzieląc się informacjami i tworząc jedną prezentację na temat różnych kalkulatorów.
3. Zadania 8. i 9. uczniowie powinni wykonywać wspólnie, łącząc wyszukane informacje w jedną tabelę.
4. Prezentacje z zadań 10. i 11. należy połączyć w jedną prezentację.
5. Zadanie 12. może realizować większa liczba uczniów, którzy połączą poszczególne opracowania w jeden dokument. Każdy może opracować inny pomysł na komputer.
6. Zadanie 14. może przygotować większa liczba uczniów, którzy podzielą się wyszukiwaniem informacji na konkretny temat.

Warto zapamiętać

- Historia urządzeń do liczenia zaczęła się od kamyków przesuwanych w rowkach glinianych tabliczek, potem były liczydła i kalkulatory mechaniczne.
- Pierwsze komputery działały w oparciu o lampowe układy elektroniczne, które następnie zastąpiono tranzystorami, a w końcu – układami scalonymi.
- Pierwszym w historii komputerem elektronicznym był ENIAC, który powstał w 1945 roku.
- Wynalezienie układu scalonego było przełomem w historii rozwoju informatyki – zaczęły dzięki temu powstawać pierwsze komputery osobiste.

Przeczytaj, jeśli chcesz wiedzieć więcej…

- Już w **1961** roku powstał pierwszy komputer zbudowany w oparciu o układy scalone produkowane przez firmę Texas Instruments. Mimo postępu miniaturyzacji, komputery były duże i drogie. Ich ceny przekraczały kilkaset tysięcy dolarów. W **1965** roku firma Digital Equipment Corporation wyprodukowała pierwszy tani minikomputer, który kosztował… „tylko” 18 000 dolarów.
- W **1952** roku po raz pierwszy komputer pokonał człowieka w grze w kółko i krzyżyk. W **1994** roku program komputerowy okazał się lepszy od profesjonalnego gracza w szachy, a w **1997** roku komputer Deep Blue pokonał w szeroko komentowanej rozgrywce szachowego mistrza świata, Garriego Kasparowa.
- W październiku **2017** roku robot Sophia został pierwszym robotem w historii, który dostał obywatelstwo Arabii Saudyjskiej. Robota obdarzono **sztuczną inteligencją**, aby mógł uczyć się, dostosowywać do ludzkich zachowań i pracować z ludźmi. Robot ten udzielił wielu wywiadów na całym świecie.

Obliczenia w arkuszu kalkulacyjnym

Rozdział IV

Temat 12. Więcej o pracy w arkuszu kalkulacyjnym

Temat 13. Przedstawianie danych w postaci wykresu i zadania projektowe

Temat 14. Zastosowanie arkusza kalkulacyjnego



Temat 12. Więcej o pracy w arkuszu kalkulacyjnym

1. Adresowanie mieszane
2. Linie siatki w arkuszu kalkulacyjnym a obramowania komórek
3. Drukowanie tabeli arkusza kalkulacyjnego

Warto powtórzyć

1. Jak się tworzy formuły?
2. Na czym polega adresowanie względne?
3. Kiedy stosujemy adres bezwzględny?

1. Adresowanie mieszane

Zamierzamy przygotować w arkuszu kalkulacyjnym procentowe zestawienie liczby dziewcząt i chłopców w poszczególnych klasach pewnej szkoły (rys. 1.). Do wszystkich obliczeń użyjemy danych z kolumny *B*. W jaki sposób utworzyć formułę, aby po jej skopiowaniu do innej kolumny nadal zawierała adresy komórek z kolumny *B*?

Podczas tworzenia i kopiowania formuł w arkuszu kalkulacyjnym ważne jest ustalenie, jakiego rodzaju adresowanie należy zastosować, aby nie pojawiły się błędy w kopiowanych formułach.

Jeśli w formule musimy użyć danych z konkretnej kolumny lub z konkretnego wiersza arkusza kalkulacyjnego, warto utworzyć formułę, stosując **adresowanie mieszane**. Przed literą oznaczającą kolumnę albo liczbą oznaczającą numer wiersza należy wpisać znak „$”.

Adres mieszany A

Adres, w którym występuje odwołanie względne do kolumny i bezwzględne do wiersza lub odwrotnie: odwołanie bezwzględne do kolumny i względne do wiersza.

	A	B	C	D	E	F	G
1	**nazwa klasy**	**liczba uczniów**	**liczba dziewcząt**	**% dziewcząt**	**liczba chłopców**	**% chłopców**	
2	**I**	31	12				
3	**II**	30	13				
4	**III**	28	11				
5	**IV**	29	8				
6	**V**	26	15				
7	**VI**	30	14				
8	**VII**	26	13				
9	**VIII**	27	10				
10	**SUMA**						
11							

Rys. 1. Tabela – ćwiczenie 1.

Ćwiczenie 1. Stosujemy adres mieszany

1. W arkuszu kalkulacyjnym utwórz tabelę pokazaną na rysunku 1.
2. Oblicz liczbę wszystkich uczniów szkoły oraz liczbę wszystkich dziewcząt.
3. Oblicz, ilu jest chłopców w poszczególnych klasach, wpisując odpowiednią formułę do komórki *E2*. Skopiuj ją do komórek od *E3* do *E9*. Czy możesz skopiować tę formułę również do komórki *E10*, aby obliczyć liczbę chłopców w całej szkole?
4. Wpisz do komórki *D2* formułę obliczającą procent dziewcząt w danej klasie (rys. 2a). Skopiuj ją do odpowiednich komórek w kolumnie *D*. Sprawdź, jaka jest postać formuły w komórkach *D3:D9* (po skopiowaniu). Czy możesz skopiować tę formułę również do komórki *D10*, aby obliczyć procent dziewcząt w całej szkole?
5. Skopiuj formułę z komórki *D2* do kolumny *F* – począwszy od komórki *F2*.
6. Sprawdź, jaka jest postać formuły w komórkach *F2:F10*. Przyjrzyj się również rysunkowi 2b, na którym w komórce *F2* widać postać formuły skopiowanej z komórki *D2*. Czy formuła jest poprawna? Wyjaśnij, dlaczego po skopiowaniu pojawiła się taka postać formuły.

	A	B	C	D	E	F
1	nazwa klasy	liczba uczniów	liczba dziewcząt	% dziewcząt	liczba chłopców	% chłopców
2	I	31	12	=C2/B2*100	19	

Rys. 2a. Przykładowa formuła – ćwiczenie 1. (punkt 4.)

	A	B	C	D	E	F
1	nazwa klasy	liczba uczniów	liczba dziewcząt	% dziewcząt	liczba chłopców	% chłopców
2	I	31	12	38,71	19	=E2/D2*100

Rys. 2b. Przykładowa formuła – ćwiczenie 1. (punkt 6.)

7. Zastanów się, jak powinna wyglądać formuła w komórce *D2* (i kolumnie *D*), aby można ją było skopiować do komórki *F2* i kolumny *F*. Zmień formułę w komórce *D2* (rys. 3a) i skopiuj ją do odpowiednich komórek.

	A	B	C	D	E	F
1	nazwa klasy	liczba uczniów	liczba dziewcząt	% dziewcząt	liczba chłopców	% chłopców
2	I	31	12	=C2/$B2*100	19	

Rys. 3a. Przykładowa formuła – ćwiczenie 1. (punkt 7.)

8. Po wstawieniu w formule w komórce *D2* znaku „$” przed literą *B* formuła skopiowana do komórki *F2* jest poprawna (rys. 3b).

	A	B	C	D	E	F
1	nazwa klasy	liczba uczniów	liczba dziewcząt	% dziewcząt	liczba chłopców	% chłopców
2	I	31	12	38,71	19	=E2/$B2*100

Rys. 3b. Przykładowa formuła – ćwiczenie 1. (punkt 8.)

Dobra rada

Przed utworzeniem formuły należy ustalić, które adresy komórek musimy w niej umieścić i jaki rodzaj adresowania zastosować, aby formuła była poprawna także po skopiowaniu i wklejeniu do innych komórek.

9. W kolumnach *D* i *F* ustal format wyświetlania liczb z dwoma miejscami po przecinku.
   Uwaga: Nie ustalaj w tym przypadku formatu procentowego, bo program pokaże wartości pomnożone po raz drugi przez 100.
10. Zapisz plik pod nazwą *Szkoła*.

Wskazówka: W formule obliczającej procent dziewcząt i procent chłopców musimy zastosować dane z kolumny *B* (liczba uczniów). Do oznaczenia stałej kolumny używa się znaku „$" przed literą kolumny. Formuła w komórce *D2* powinna mieć postać: =C2/$B2*100, co oznacza: „weź wartość z komórki najbliższej kolumny z lewej strony, podziel przez wartość z komórki z kolumny *B* i pomnóż przez 100" (rys. 3a).



Ćwiczenie 2. Stosujemy adresowanie mieszane

1. Do tabeli zapisanej w ćwiczeniu 1. w pliku *Szkoła* wstaw kolejne kolumny danych dla poszczególnych klas:
   - liczbę uczniów z poszczególnych klas, mieszkających w tej samej miejscowości, w której mieści się szkoła: 28, 20, 21, 16, 12, 20, 24, 19,
   - procent uczniów z poszczególnych klas, mieszkających w tej samej miejscowości, w której mieści się szkoła,
   - liczbę uczniów mieszkających poza miejscowością, w której mieści się szkoła,
   - procent uczniów mieszkających poza miejscowością, w której mieści się szkoła.

   Czy w każdym przypadku musisz tworzyć nową formułę?
2. Zapisz plik pod tą samą nazwą.



Jeśli w adresie komórki nazwa kolumny lub numer wiersza nie powinny się zmieniać wraz ze zmianą położenia formuły zawierającej ten adres, należy zastosować **adres mieszany**.

2. Linie siatki w arkuszu kalkulacyjnym a obramowania komórek

Linie siatki L

Pionowe i poziome linie oddzielające od siebie poszczególne komórki tabeli arkusza kalkulacyjnego.

Po otwarciu tabeli arkusza kalkulacyjnego widzimy na ekranie monitora **linie siatki**, które pokazują granice poszczególnych komórek.

Linii siatki nie ma na wydruku, ponieważ program standardowo wyłącza opcję ich drukowania. W **Ustawieniach strony** można wybrać opcję drukowania linii siatki.

Aby tabela była bardziej czytelna (szczególnie na wydruku), należy obramować komórki, w których umieszczamy dane i wyniki obliczeń. Możemy wybrać różne rodzaje obramowań.



Aby obramować komórki arkusza kalkulacyjnego, należy je zaznaczyć, a następnie kliknąć strzałkę przy przycisku **Obramowanie** i wybrać z listy odpowiedni rodzaj obramowania.

Ćwiczenie 3. Oglądamy dokument arkusza kalkulacyjnego w **Podglądzie wydruku**

1. Utwórz nowy dokument arkusza kalkulacyjnego. Wpisz pięć liczb w kolumnie *A*. Obejrzyj dokument w **Podglądzie wydruku** (**Plik/Drukuj**). Zwróć uwagę, czy widzisz linie siatki.
2. Zapisz plik pod nazwą *Obramowanie*.

Ćwiczenie 4. Dodajemy obramowania komórek arkusza kalkulacyjnego

1. Otwórz plik *Szkoła* zapisany w ćwiczeniu 2.
2. Uzupełnij obramowania komórek tabeli, dodaj podwójną linię pod komórkami z nagłówkami kolumn, dodaj kolorowe tło w nagłówkach kolumn, zmień kolor czcionki.
3. Zapisz plik pod tą samą nazwą.
4. Obejrzyj dokument w **Podglądzie wydruku**.

Wskazówka: W domyślnym widoku dokumentu arkusza kalkulacyjnego nie widać w całości obramowania pierwszego wiersza i pierwszej kolumny (przesłaniają je pola z oznaczeniami kolumn i wierszy). Jeśli włączysz **Podgląd wydruku**, zobaczysz całe obramowanie.

Dobra rada

Podczas tworzenia tabeli najpierw dobrze zaprojektuj formuły, m.in. użyj adresów odpowiednich komórek i właściwych sposobów adresowania, zastosuj funkcje, a dopiero potem upiększaj tabelę – dodawaj obramowania i cieniowania komórek tabeli czy zmieniaj krój, rodzaj i rozmiar czcionki.

3. Drukowanie tabeli arkusza kalkulacyjnego

Tabela arkusza kalkulacyjnego może się składać nawet z kilkuset kolumn i tysięcy wierszy. Wydrukowanie takiego dokumentu na jednej kartce nie ma sensu, ponieważ będzie on nieczytelny.

Aby wydrukować tabelę arkusza kalkulacyjnego – podobnie jak w przypadku innych dokumentów – wybieramy opcję **Drukuj**. Aby zobaczyć, jak tabela będzie wyglądała na wydruku, należy wybrać opcję **Podgląd wydruku** (**Plik/Drukuj**).

Przed wydrukowaniem dokumentu arkusza kalkulacyjnego należy skorzystać z opcji **Podgląd podziału stron** na karcie **Widok** i **Ustawienia strony** (**Plik/Drukuj**) w programie Microsoft Excel lub Podział strony w menu **Widok** i **Układ strony** (**Plik/Drukuj**) w programie LibreOffice Calc. W widoku **Podglądu podziału stron** granicę strony program oznacza linią przerywaną, narysowaną wzdłuż linii siatki. Można się zorientować, które kolumny i wiersze znajdą się na jednej stronie. Jeśli zmienimy szerokość lub wysokość kolumn, to linie określające granice strony odpowiednio zmienią swoje położenie. Linie określające granice strony można przesuwać, klikając je i przeciągając myszą, np. aby umieścić na jednej stronie dodatkowe wiersze lub kolumny. Wtedy cała tabela zostanie proporcjonalnie pomniejszona.

Jeśli postać wydruku nam nie odpowiada, można ją dostosować do wymagań, zmieniając odpowiednie opcje w oknie **Ustawienia strony**.

Ćwiczenie 5. Przeglądamy strony i drukujemy tabelę arkusza kalkulacyjnego

1. Otwórz plik *Szkoła* zapisany w ćwiczeniu 4.
2. Odszukaj w programie opcje **Podgląd podziału stron** i **Ustawienia strony**.
3. Sprawdź, z ilu stron składa się utworzona tabela i gdzie przebiegają granice podziału stron (które kolumny i wiersze znajdują się na danej stronie).
4. Sprawdź, jak dokument wygląda w **Podglądzie wydruku** i go wydrukuj. Jakie możliwości zmian ustawień strony oferuje program, z którego korzystasz. Jeśli twoja tabela nie mieści się w pionowej orientacji strony, zmień orientację na poziomą.
5. Zapisz plik pod tą samą nazwą i wydrukuj gotową tabelę.

Warto zapamiętać

- Adres mieszany umożliwia zablokowanie adresu komórki tak, że podczas kopiowania formuły zawierającej ten adres albo zmienia się względnie kolumna, a numer wiersza pozostaje ten sam, albo zmienia się względnie numer wiersza, a kolumna pozostaje ta sama. Adres mieszany tworzymy przez dodanie znaku „$” przed literą kolumny lub przed numerem wiersza.
- Linie siatki to pionowe i poziome linie oddzielające poszczególne komórki tabeli arkusza kalkulacyjnego.
- Zazwyczaj nie widzimy linii siatki na wydruku. Aby wydrukowana tabela była czytelna, stosuje się formatowanie komórek (np. obramowanie) lub włącza w **Ustawieniach strony** opcję drukowania linii siatki.
- Przed wydrukowaniem dokumentu utworzonego w arkuszu kalkulacyjnym należy dopracować wygląd strony, korzystając z możliwości zmiany jej ustawień.

Pytania i polecenia

1. Kiedy stosujemy adresowanie mieszane? Wyjaśnij na przykładzie.
2. Jakie korzyści daje stosowanie różnego rodzaju adresowania w arkuszu kalkulacyjnym?
3. Co to są linie siatki i czym różnią się od obramowania?
4. Podaj kilka zasad drukowania tabeli arkusza kalkulacyjnego.

Zadania

1. W arkuszu kalkulacyjnym utwórz tabelę pokazaną na rysunku 4. W tabeli podane są informacje o liczbie poszczególnych ocen z wszystkich przedmiotów w klasach IV-VIII. Do tabeli dodaj formuły obliczające, ile ocen celujących (6), bardzo dobrych (5), dobrych (4), dostatecznych (3), dopuszczających (2) i niedostatecznych (1) przypada na jednego ucznia w poszczególnych klasach. Obliczenia umieść w kolejnych kolumnach. Dodaj odpowiednie nagłówki. Jaki format liczb (ile miejsc po przecinku) jest najbardziej odpowiedni dla otrzymanych wyników? Zapisz plik pod nazwą *Oceny*.

	A	B	C	D	E	F	G	H
1			Oceny z wszystkich przedmiotów na koniec roku szkolnego					
2	klasa	liczba uczniów	liczba szóstek	liczba piątek	liczba czwórek	liczba trójek	liczba dwójek	liczba jedynek
3	IVA	30	16	55	98	113	70	8
4	IVB	29	21	40	99	114	65	9
5	IVC	28	10	54	112	99	44	17
6	VA	31	15	68	97	121	51	20
7	VB	29	11	71	100	111	44	11
8	VC	29	6	69	109	113	38	13
9	VIA	27	9	60	89	131	27	8
10	VIB	26	13	79	110	88	18	4
11	VIC	28	5	70	130	101	23	7
12	VIIA	29	12	67	101	120	39	9
13	VIIB	25	14	72	99	90	14	11
14	VIIC	26	8	85	105	96	8	10
15	VIIIA	27	10	76	113	98	15	12
16	VIIIB	29	11	75	115	119	20	8
17	VIIIC	30	17	100	153	68	16	6

Rys. 4. Tabela – zadanie 1.

2. Otwórz plik *Oceny* zapisany w zadaniu 1. Skopiuj do **Schowka** zawartość komórek z kolumn *A* i *B*, począwszy od drugiego wiersza (czyli zakres komórek *A2:B17* – rys. 4.). Utwórz nowy dokument arkusza kalkulacyjnego i wklej skopiowane komórki do kolumn *A* i *B*. Następnie:
   a. Oblicz sumę wszystkich uczniów szkoły.
   b. Utwórz formuły, które obliczą, jaki procent uczniów szkoły stanowią uczniowie poszczególnych klas.
   c. Odpowiedz na pytania: *Ile nowych formuł musisz utworzyć? Czy sposób umieszczenia kolumn z danymi procentowymi ma tu znaczenie?*

   Zapisz plik pod nazwą *Statystyka szkolna*.
3. Do tabeli z zadania 2. dodaj formuły na obliczenie, jaki procent uczniów szkoły stanowią uczniowie klas czwartych, piątych, szóstych, siódmych i ósmych. Zapisz plik pod tą samą nazwą.

Temat 13. Przedstawianie danych w postaci wykresu i zadania projektowe

1. Podstawowe pojęcia i zasady tworzenia wykresów
2. Tworzenie wykresu dla jednej serii danych
3. Tworzenie wykresu dla kilku serii danych
4. Dostosowywanie typu wykresu do rodzaju danych
5. Projekt grupowy – impreza sportowa

Warto powtórzyć

1. Do czego służy arkusz kalkulacyjny?
2. Jak zbudowany jest dokument arkusza kalkulacyjnego?
3. Jakie są zasady projektowania tabeli w arkuszu kalkulacyjnym?

1. Podstawowe pojęcia i zasady tworzenia wykresów

Arkusz kalkulacyjny umożliwia przedstawianie danych w postaci wykresów, co znacznie ułatwia analizę i interpretację tych danych.

Pojęcia związane z tworzeniem wykresów:

- **Seria danych** to zakres komórek arkusza kalkulacyjnego, obejmujący jeden wiersz lub jedną kolumnę, których wartości chcemy umieścić na wykresie. Każdą serię danych program zaznacza na wykresie innym kolorem.
- **Nazwa serii danych** to opis serii (zwykle nagłówek kolumny lub wiersza), który pojawi się na wykresie w legendzie lub jako tytuł.
- **Etykiety osi poziomej (kategorii)** to opisy osi poziomej – zwykle nagłówki kolumn lub wierszy.
- **Etykiety osi pionowej (wartości)** to wartości odpowiadające kategoriom umieszczonym na osi poziomej.
- **Etykiety danych** to nazwy serii, nazwy kategorii lub wartości danych umieszczone np. nad kolumną (na wykresie kolumnowym) lub obok wycinka koła (na wykresie kołowym). Można zrezygnować z ich wyświetlania, np. jeśli zasłaniają wykres.
- **Typ wykresu** to sposób prezentacji danych i wyników na wykresie, np. za pomocą słupków, linii, wycinków koła.

Aby utworzyć wykres w arkuszu kalkulacyjnym, należy wskazać dane, które chcemy zilustrować, czyli serię (lub serie) danych, a następnie wybrać opcję wstawiania wykresów na karcie **Wstawianie** (Microsoft Excel) lub w menu **Wstaw** (LibreOffice Calc).

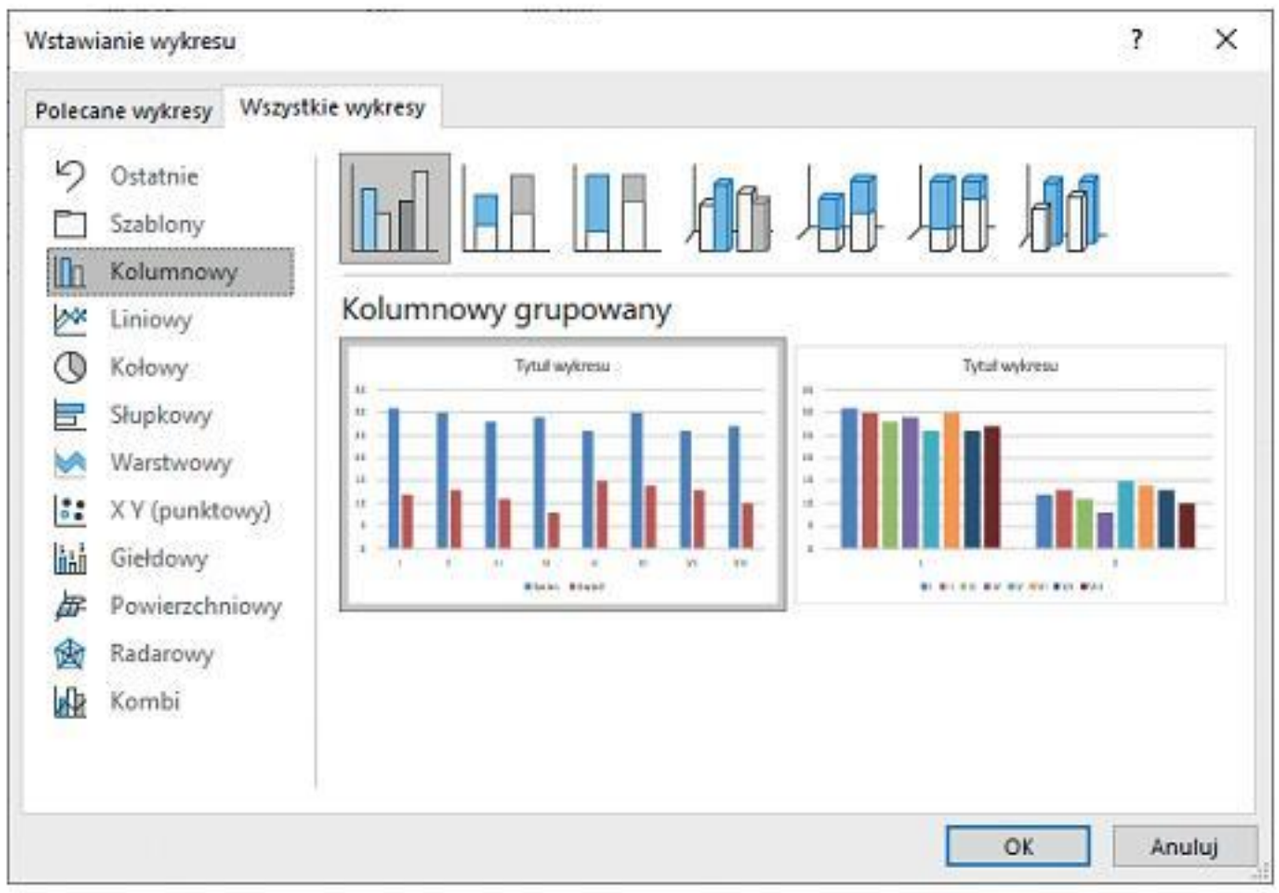
Wykresy możemy tworzyć dla jednej lub dla kilku serii danych. Niektóre typy wykresów (np. wykres kołowy) stosuje się tylko dla jednej serii danych.

2. Tworzenie wykresu dla jednej serii danych

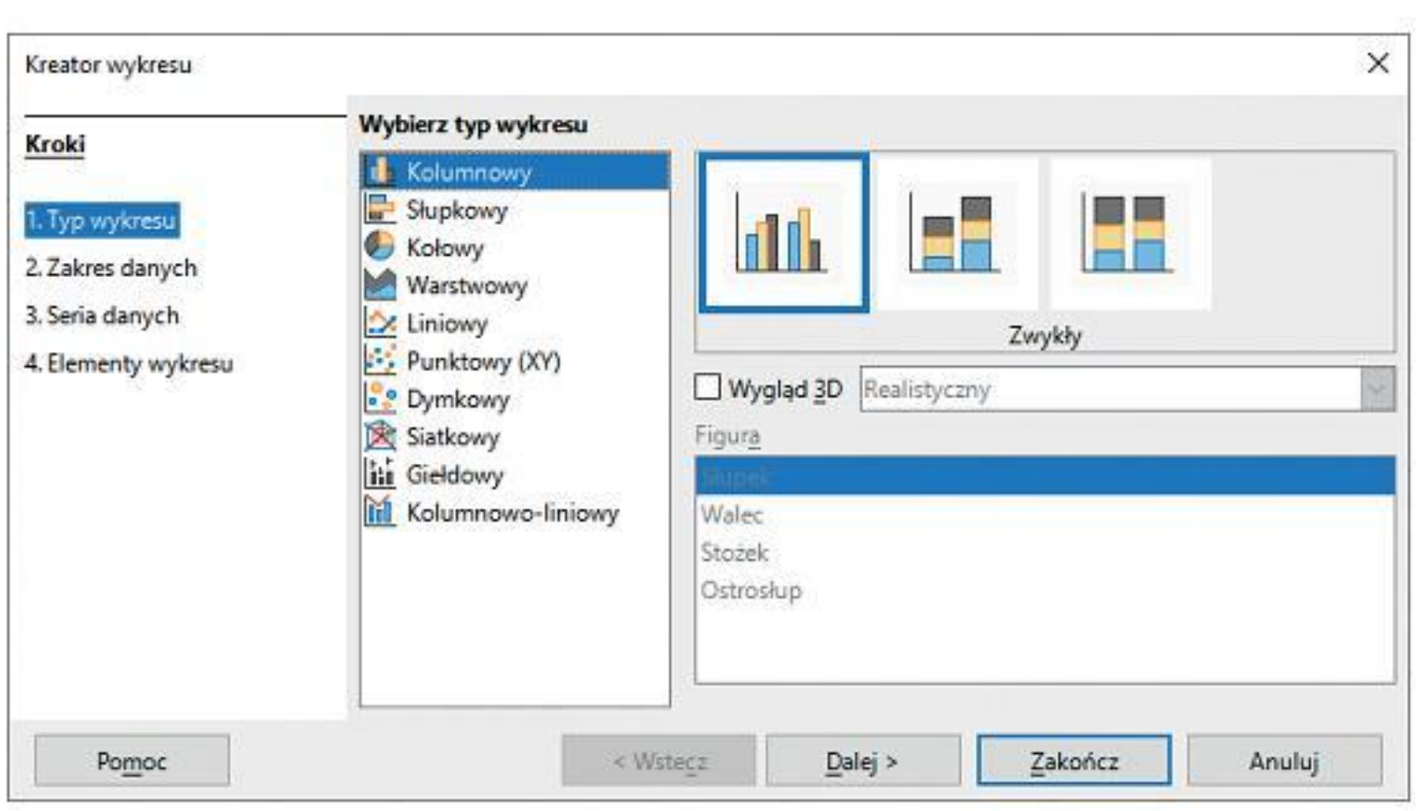
Zamierzamy przedstawić na wykresie liczbę uczniów w poszczególnych klasach pewnej szkoły. Dodatkowo chcemy, aby nazwy klas pojawiły się na osi poziomej. W jaki sposób to zrobić w arkuszu kalkulacyjnym?

Aby rozpocząć tworzenie wykresu dla jednej serii danych, należy:

- zaznaczyć dane, które mają się znaleźć na wykresie, w tym nagłówki kolumn lub etykiety wierszy (dane powinny znajdować się w jednej kolumnie lub w jednym wierszu),
- wybrać opcję **Polecane wykresy** na karcie **Wstawianie** (Microsoft Excel) lub **Wykres** w menu **Wstaw** (LibreOffice Calc),
- wybrać typ wykresu (rys. 1a, 1b),
- określić nazwę serii danych (która następnie pojawi się w legendzie lub w tytule wykresu) oraz etykiety osi poziomej.



Rys. 1a. Przykład wyboru typu wykresu (Microsoft Excel)



Rys. 1b. Przykład wyboru typu wykresu (LibreOffice Calc)

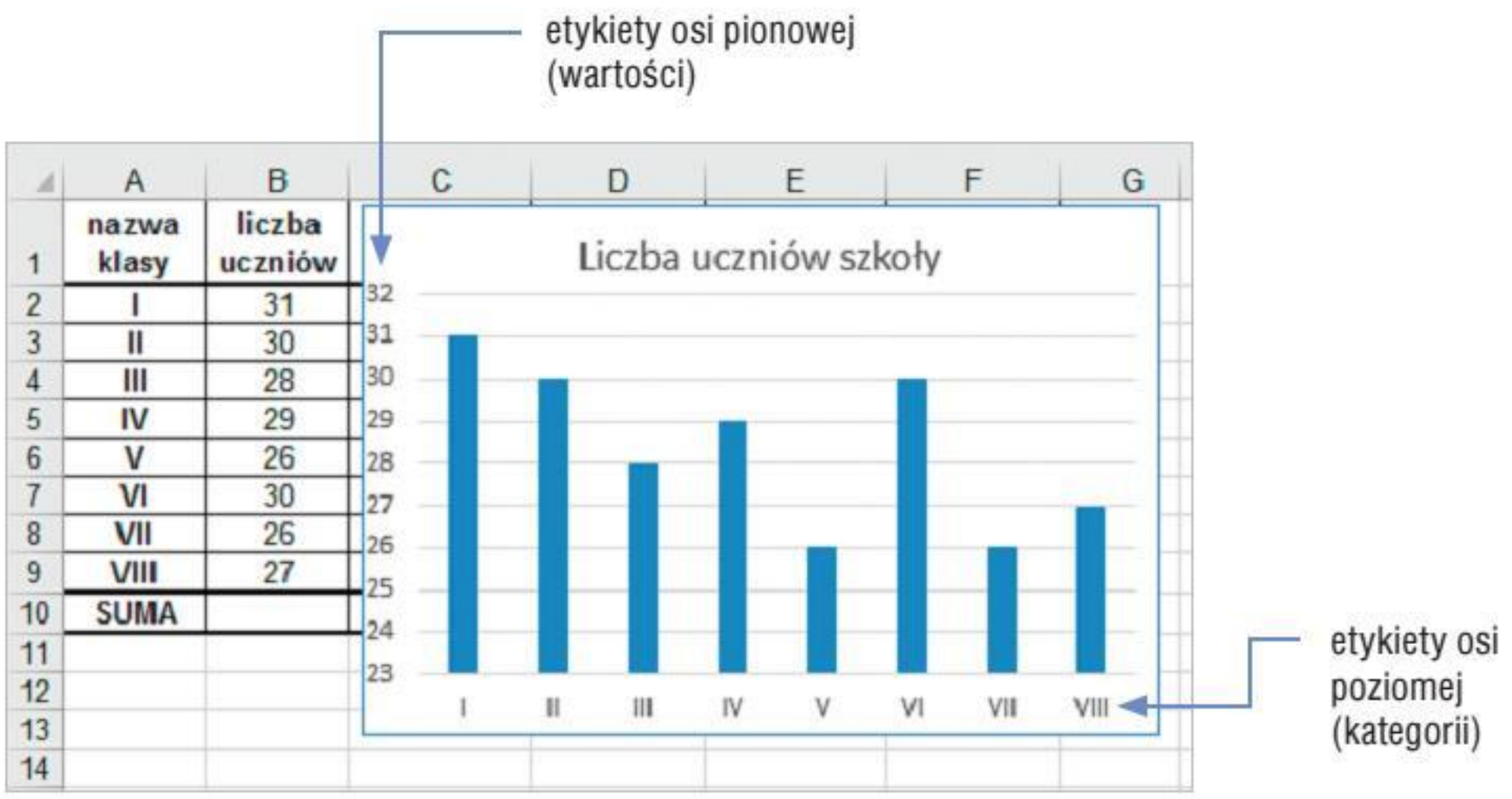
Najlepiej zaznaczyć dane łącznie z nagłówkami kolumn i etykietami wierszy. Wtedy nagłówki kolumn i etykiety wierszy posłużą do opisania tytułu, osi poziomej (kategorii) i legendy.

Ćwiczenie 1. Tworzymy wykres dla jednej serii danych

1. Otwórz plik *Szkoła* zapisany w ćwiczeniu 5., w temacie 12. W tabeli umieszczono informacje o liczbie uczniów pewnej szkoły.
2. Przygotuj wykres kolumnowy obrazujący liczbę uczniów w poszczególnych klasach.
3. Zapisz plik pod tą samą nazwą.

Wskazówki:

- Zaznacz zakres komórek *A1:B9*. Zaznaczenie zakresu komórek łącznie z nagłówkami kolumn i nazwami klas (etykietami wierszy) spowoduje, że nagłówek „liczba uczniów" posłuży do zapisania tytułu wykresu, a etykiety wierszy z nazwami klas – do opisu osi poziomej (kategorii).
- Wybierz typ wykresu – kolumnowy (rys. 2.). Podczas tworzenia wykresu zawsze masz możliwość podglądu jego postaci. Zauważ, że w etykietach osi kategorii pojawiły się nazwy klas. Wprowadzone automatycznie nazwy, np. tytuł wykresu czy opis legendy, można zmienić, uaktywniając i edytując obiekt, w którym umieszczono daną informację.



	A	B
1	nazwa klasy	liczba uczniów
2	I	31
3	II	30
4	III	28
5	IV	29
6	V	26
7	VI	30
8	VII	26
9	VIII	27
10	SUMA	

Rys. 2. Fragment tabeli i przykład wykresu kolumnowego dla jednej serii danych – ćwiczenie 1.

Ćwiczenie 2. Sprawdzamy, czy po zmianie danych w tabeli arkusza kalkulacyjnego trzeba ponownie tworzyć wykres

1. W tabeli zapisanej w pliku *Szkoła* (ćwiczenie 1.) błędnie wpisano niektóre liczby uczniów. W klasie IV powinno być 27 uczniów, w V – 24, w VII – 32. Zmień dane w tabeli. Co można zauważyć na wykresie?
2. Zapisz plik pod tą samą nazwą.

Zmiana danych w tabeli powoduje automatyczną aktualizację wykresu utworzonego na podstawie tej tabeli.

3. Tworzenie wykresu dla kilku serii danych

Wykres dla kilku serii danych można utworzyć, zaznaczając zakres wszystkich komórek, których wartości chcemy na nim umieścić.

W większości programów możemy zaznaczyć serie danych zapisanych w kolumnach lub wierszach niesąsiadujących ze sobą. W tym celu po zaznaczeniu pierwszej serii należy zaznaczyć kolejną serię, przytrzymując wciśnięty klawisz **Ctrl**.

Do utworzonego wykresu można także dodać nową serię danych, korzystając z opcji wykresu. W niektórych programach można zaznaczony zakres danych skopiować do **Schowka** (**Ctrl+C**), uaktywnić wykres i wkleić w jego obszarze zaznaczoną serię danych (**Ctrl+V**).



Aby utworzyć wykres dla kilku serii danych, należy:

- zaznaczyć wszystkie serie danych lub tylko pierwszą serię (jeśli zaznaczenie wszystkich naraz nie było możliwe),
- wybrać polecenie wstawiania wykresu (na karcie **Wstawianie** lub w menu **Wstaw**),
- wybrać typ wykresu,
- dodać kolejne serie danych (jeśli nie wybraliśmy ich od razu),
- określić nazwy serii danych (pojawią się w legendzie) oraz etykiety osi kategorii.

Możemy stosować różne typy wykresów. Na przykład do porównywania i grupowania danych liczbowych można zastosować wykres kolumnowy.



Ćwiczenie 3. Tworzymy wykres dla dwóch serii danych

1. Sprawdź, w jaki sposób można w używanym przez ciebie programie wstawić na wykresie kolejną serię danych. W razie problemów skorzystaj z **Pomocy**.
2. Otwórz plik *Szkoła* zapisany w ćwiczeniu 2. Przygotuj wykres kolumnowy przedstawiający liczbę dziewcząt i chłopców w poszczególnych klasach szkoły.

Program Microsoft Excel (lub LibreOffice Calc) domyślnie wstawia wykres do dokumentu arkusza kalkulacyjnego jako obiekt obok tabeli z danymi. Jeśli uaktywnimy wykres, w menu pojawią się opcje, które pozwalają go edytować: dodać serię danych, tytuły, opisy osi, legendę, kolory, desenie itp. Uaktywnienie wybranego elementu wykresu (np. legendy, tytułu, obszaru kreślenia) również umożliwia jego modyfikowanie. Po dwukrotnym kliknięciu danego elementu otwiera się okno, w którym możemy zmieniać różne parametry.

Polecenia formatowania wykresu można też wybrać z menu kontekstowego, klikając wybrane miejsce wykresu prawym przyciskiem myszy.



Ćwiczenie 4. Dodajemy do wykresu tytuł i legendę

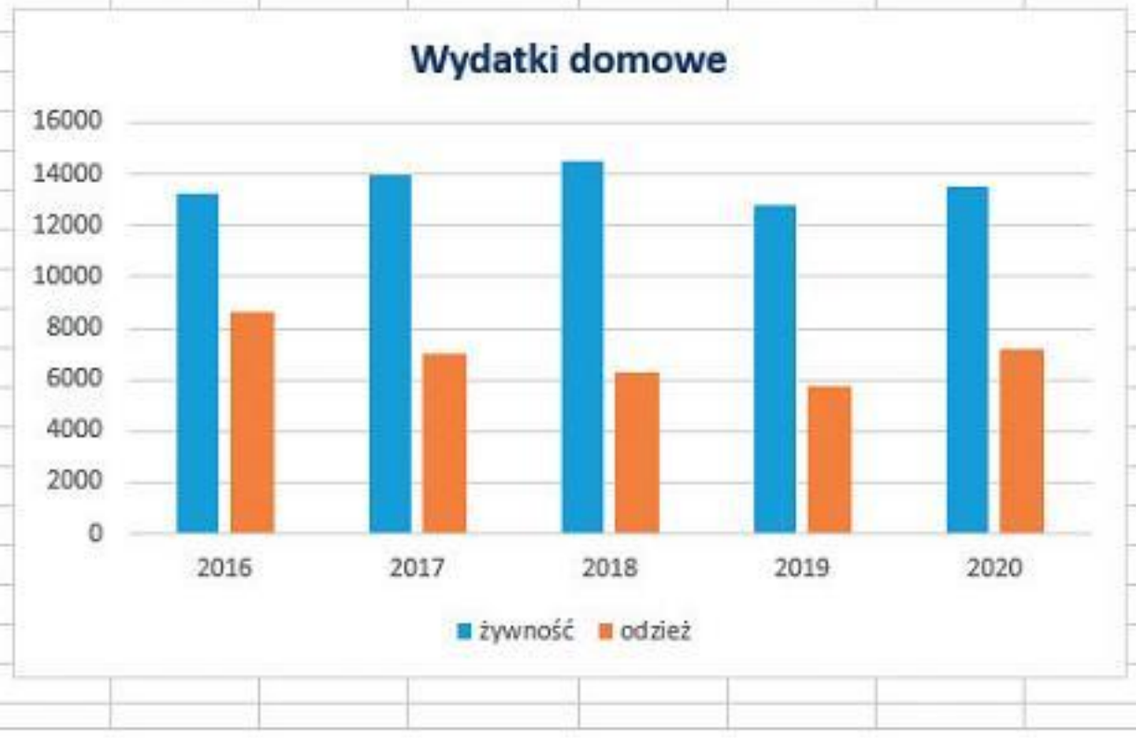
Do wykresu utworzonego w ćwiczeniu 3. dodaj tytuł (lub zmodyfikuj tytuł wstawiony automatycznie): „Liczba dziewcząt i chłopców w szkole". Sprawdź, czy pojawiła się legenda. Jeśli nie – dodaj ją. Zmień kolory wykresu: kolumny oznaczające liczbę dziewcząt powinny mieć kolor czerwony, a chłopców – niebieski.

Dla jednej tabeli arkusza kalkulacyjnego można przygotować więcej niż jeden wykres. Tabela i wykresy zapisują się w tym samym pliku.

Ćwiczenie 5. Przykład wykresu dla kilku serii danych

1. Otwórz nowy plik arkusza kalkulacyjnego.
2. Zaprojektuj tabelę przedstawiającą wydatki domowe w kolejnych latach (rys. 3.). Wpisz przykładowe rodzaje wydatków i odpowiednie kwoty.
3. Zsumuj wydatki każdego rodzaju oraz wydatki w poszczególnych latach.
4. Przygotuj wykres porównujący wszystkie wydatki w kolejnych latach. Na rysunku 3. pokazano wykres porównujący wydatki na odzież i żywność. Czy możesz od razu umieścić na wykresie wszystkie serie danych oraz etykiety osi kategorii?
5. Ile serii danych znajdzie się na tym wykresie? Jakie nazwy należy umieścić jako etykiety osi kategorii?

	A	B	C	D	E	F	G
1	Wydatki domowe						
2	lata	żywność	odzież	opłaty stałe	podróże	inne	suma
3	2016	13200	8600	6000	4800	1000	33600
4	2017	14000	7000	6100	5000	1500	33600
5	2018	14500	6300	6150	3700	2000	32650
6	2019	12800	5700	6200	4000	1900	30600
7	2020	13500	7200	5700	3800	900	31100
8	suma	68000	34800	30150	21300	7300	161550



Rys. 3. Przykład tabeli oraz wykresu dla dwóch serii danych – ćwiczenie 5.

4. Dostosowywanie typu wykresu do rodzaju danych

Nie każdy wykres jest odpowiedni do przedstawienia określonego rodzaju danych czy wyników. Za pomocą **wykresu liniowego** przedstawia się zwykle tendencje wzrostowe lub spadkowe (wzrost produkcji, spadek zatrudnienia itp.). Wykres **XY** (**punktowy**) często wykorzystuje się na lekcjach matematyki lub fizyki (np. wykres funkcji liniowej czy wykres prędkości). Wykres **kolumnowy** lub **słupkowy** służy do porównywania wartości dla różnych kategorii, np. liczby ocen niedostatecznych, dopuszczających, dostatecznych, dobrych, bardzo dobrych i celujących w poszczególnych klasach, liczby urodzeń w krajach Europy w danym roku.

Do przedstawienia proporcji, np. jaki procent uczniów szkoły uczestniczy w olimpiadach przedmiotowych, używa się **wykresu kołowego**. Wykres kołowy wykonuje się tylko dla jednej serii danych i nie występują na nim osie kategorii i wartości.

Ćwiczenie 6. Przykład wykresu kołowego

1. Przedstaw w postaci wykresu kołowego udział uczniów szkoły w olimpiadach i konkursach przedmiotowych. Umieść na wykresie etykiety danych (wartości procentowe). Przykładowe dane oraz propozycję wykresu przedstawia rysunek 4.
2. Zapisz plik pod nazwą *Konkursy i olimpiady*.

Wskazówka: Aby umieścić nazwy przedmiotów w legendzie, zaznacz kolumny z nazwami przedmiotów i z liczbami uczniów.

	A	B
1	Udział uczniów w olimpiadach i konkursach przedmiotowych	
2	**przedmiot**	**liczba uczniów**
3	matematyka	120
4	język polski	60
5	historia	40
6	język angielski	50
7	inne	40
8	nie uczestniczyli	239
9	**liczba uczniów ogółem**	549
10		
11		
12		

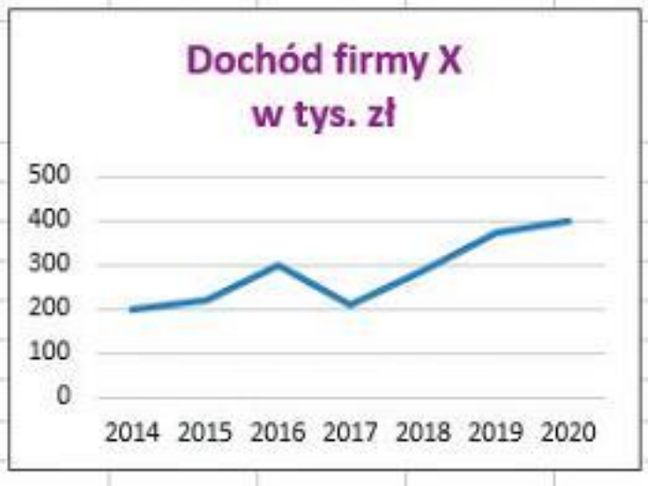


Rys. 4. Tabela i wykres kołowy – ćwiczenie 6.

Ćwiczenie 7. Przykład wykresu liniowego

1. Przedstaw na wykresie zmiany wartości dochodów pewnej firmy w kolejnych latach. Umieść przykładowe dane w tabeli (kwoty podaj w tysiącach złotych). Dane z kolumny *A* mają posłużyć do opisu osi poziomej. Na rysunku 5. przedstawiono tabelę i propozycję wykresu.
2. Zapisz plik pod nazwą *Dochody firmy*.

	lata	dochód w tys. zł
4	2014	200
5	2015	220
6	2016	300
7	2017	210
8	2018	290
9	2019	370
10	2020	400



Rys. 5. Tabela i wykres liniowy – ćwiczenie 7.

5. Projekt grupowy – impreza sportowa

Zadanie projektowe – przygotowanie dokumentacji imprezy sportowej

Poznane możliwości pracy w chmurze i w arkuszu kalkulacyjnym wykorzystamy podczas pracy nad projektem. W pracy zespołowej należy dobrze zaplanować działania i rozdzielić zadania szczegółowe.

Przykładowy podział zadania na zadania szczegółowe

Zadanie 1. Przygotuj ogólny program kilkudniowej imprezy sportowej.

Zadanie 2. Przygotuj szczegółowy (godzinowy) harmonogram rozgrywek.

Zadanie 3. Zaprojektuj tabelę do zapisywania list uczniów biorących udział w zawodach w poszczególnych dyscyplinach sportu, według klas. Uwzględnij podział na dziewczęta i chłopców. Utwórz formuły obliczające udział procentowy uczniów z poszczególnych klas (z podziałem na dziewczęta i chłopców) w rozgrywanych dyscyplinach.

Uwaga: W projekcie użyj fikcyjnych danych (imion i nazwisk) uczniów. Można dla uproszczenia przyjąć, że w szkole są trzy klasy biorące udział w imprezie sportowej i w każdej jest po piętnastu uczniów. Dyscypliny sportowe wybierz według własnego pomysłu.

Zadanie 4. Zaprojektuj tabelę, do której wpiszesz wyniki w poszczególnych dyscyplinach. Na przykład w skoku w dal tabela powinna zawierać nazwiska i imiona zawodników oraz wyniki osiągnięte przez nich w trzech próbach skoków. W kolejnej kolumnie umieść najlepszy wynik każdego ucznia, a następnie wyróżnij wiersze zawierające dane trzech najlepszych zawodników w danej dyscyplinie.

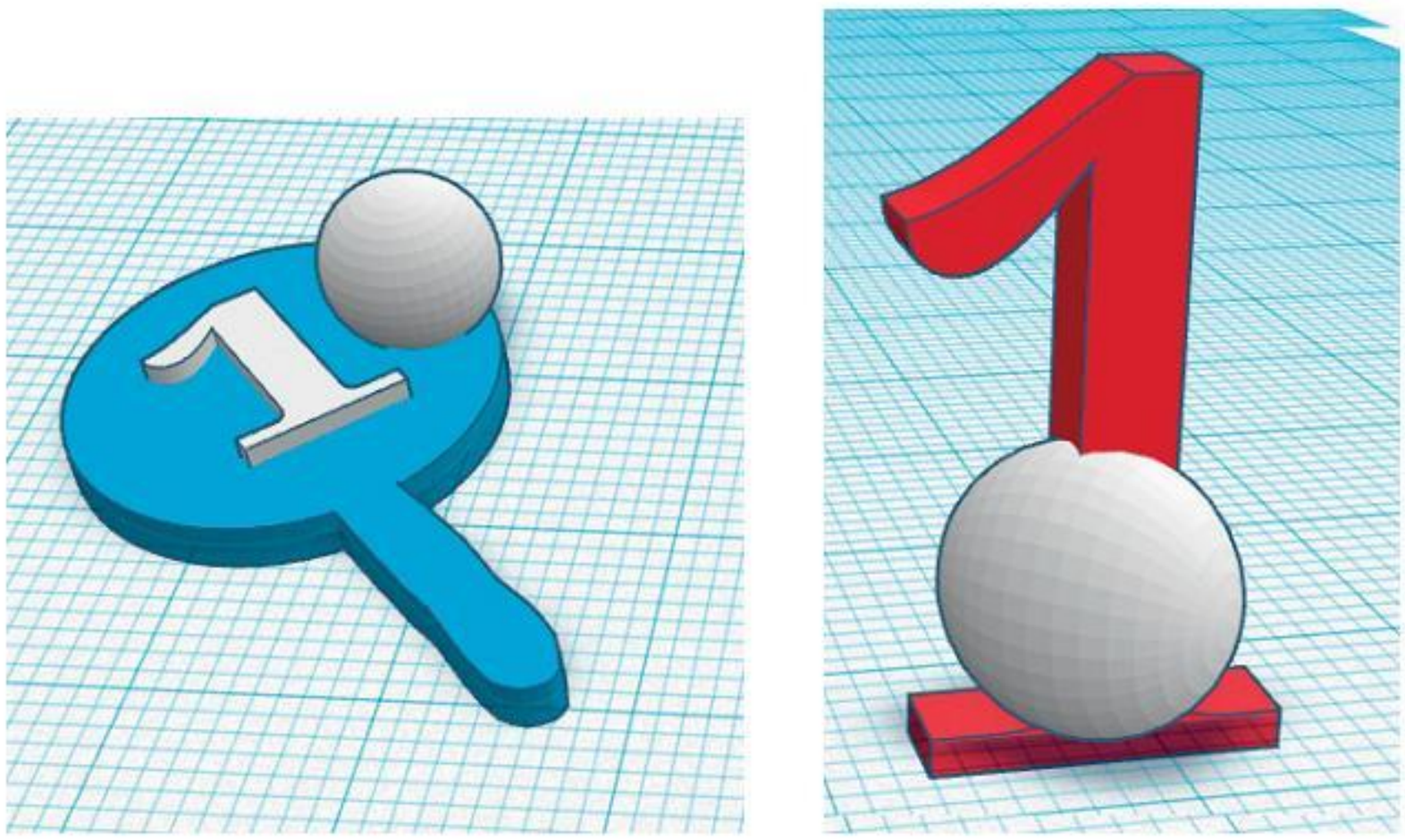
Wskazówka: Zadanie 4. może wykonywać kilka osób (grup). Każda zaprojektuje tabelę dla innej dyscypliny sportu, ale wcześniej należy ustalić szczegółowo układ tabeli oraz prowadzone obliczenia.

Zadanie 5. Przygotuj zestawienie wyników dla wszystkich klas w poszczególnych dyscyplinach: w tabeli i na wykresie, np. uwzględniając liczbę zdobytych pierwszych miejsc.

Zadanie 6. Wydrukuj wszystkie dokumenty.

Zadanie 7. Zaprojektuj w programie Tinkercad nagrody dla zwycięzców (zdobywców pierwszego miejsca) symbolizujące daną dyscyplinę w postaci modeli 3D , np. statuetki lub puchary. Wydrukuj utworzone modele na drukarce 3D.

Wskazówka: Zadanie 7. może wykonywać kilka osób (grup). Każda zaprojektuje statuetkę lub puchar dla innej dyscypliny sportu, np. paletka do tenisa stołowego, piłkę do siatkówki, koszykówki czy do gry w piłkę nożną z cyfrą 1 (rys. 6.). Możesz wykazać się kreatywnością i zaprojektować pamiątki dla zwycięzców według własnego pomysłu.



Rys. 6. Pomysły na pamiątki-nagrody dla zwycięzcy w turnieju tenisa stołowego – modele 3D wykonane w programie Tinkercad – zadanie 7.

Przykłady łączenia dokumentów

1. Do ogólnego programu imprezy (zadanie 1.) można dołączyć szczegółowy harmonogram (zadanie 2.).
2. Do każdego szczegółowego harmonogramu (przygotowanego w zadaniu 2.) można dołączyć tabele z wynikami w danej konkurencji (zadanie 4.).

Uwaga: Możesz zaplanować również inne dodatkowe zadania i wybrać odmienny sposób łączenia dokumentów.

Warto zapamiętać

- Tworzenie wykresu należy rozpoczynać od zaznaczenia komórek zawierających serie danych, które chcemy przedstawić.
- Warto umieścić na wykresie opisy osi poziomej i pionowej.
- Na jednym wykresie można umieścić więcej niż jedną serię danych, ale nie powinno ich być zbyt wiele, gdyż wykres stanie się nieczytelny.
- Trzeba dostosować typ wykresu do rodzaju danych, które chcemy zobrazować za jego pomocą, i do stawianego celu.
- Jeśli chcemy:
  - przedstawić tendencje (wzrostowe lub spadkowe) – najlepiej stosować wykres liniowy;
  - narysować wykres funkcji – najlepiej zastosować wykres XY (punktowy);
  - porównać dane – najlepiej zastosować wykres słupkowy lub kolumnowy;
  - zobrazować udziały procentowe – najlepiej zastosować wykres kołowy.
- Jeżeli wydruk wykresu ma być czarno-biały, należy ustalić użyte kolory i desenie pod kątem takiego wydruku.
- Należy opisać wykres – wstawić tytuł wykresu, legendę (zwłaszcza gdy na wykresie przedstawiamy kilka serii danych).

Pytania i polecenia

1. Jakie są zalety prezentowania danych w postaci wykresów? Wymień przynajmniej trzy.
2. Jakie są kolejne kroki postępowania przy tworzeniu wykresów dla jednej serii danych?
3. Jak dodać kolejną serię danych do wykresu?
4. Na wykresie nie umieściliśmy opisu osi poziomej. Czy można go uzupełnić? Wyjaśnij sposób postępowania.
5. Kiedy powinniśmy posługiwać się wykresem kolumnowym, kołowym, XY (punktowym), a kiedy liniowym? Podaj przykłady.
6. Pomyliliśmy się, wprowadzając dane, i już przygotowaliśmy na ich podstawie wykres. Czy po poprawieniu danych należy sporządzić wykres od nowa? Uzasadnij odpowiedź.
7. Sprawdź, jakiego typu wykresy zawiera arkusz kalkulacyjny, z którego korzystasz. Czy są one tylko dwuwymiarowe?
8. Wyjaśnij, dlaczego wykresu kołowego nie można stosować dla kilku serii danych.

Zadania

1. Otwórz plik *Oceny* zapisany w zadaniu 1., w temacie 12. Przygotuj wykres, na którym porównasz liczby ocen niedostatecznych i dopuszczających w poszczególnych klasach. Zapisz plik pod tą samą nazwą.
2. Otwórz plik *Oceny* zapisany w zadaniu 1. Przygotuj wykres, na którym porównasz liczby ocen celujących, bardzo dobrych, dobrych, dostatecznych, dopuszczających i niedostatecznych w klasach ósmych. Czy możesz skorzystać z wykresu wykonanego w zadaniu 1.? Zapisz plik pod tą samą nazwą.
3. Otwórz plik *Oceny* zapisany w zadaniu 2. Przedstaw na wykresie liczbę ocen celujących, bardzo dobrych, dobrych, dostatecznych, dopuszczających i niedostatecznych przypadających na jednego ucznia w szkole. Jaki typ wykresu będzie najbardziej odpowiedni? Skorzystaj z obliczeń wykonanych w zadaniu 1., w temacie 12. Wykres umieść w osobnym arkuszu. Zapisz plik pod tą samą nazwą.
4. Wielu uczniów uprawia różne dyscypliny sportu, ale są i tacy, którzy nie uprawiają żadnej (w tym zadaniu zakładamy, że dany uczeń uprawia maksymalnie jedną dyscyplinę). Zaprojektuj tabelę dla danych: liczba uczniów – 900, piłka nożna – 250, koszykówka – 20, pływanie – 40, lekka atletyka – 50; pozostali nie uprawiają żadnej dyscypliny sportu.
   Wykonaj wykres przedstawiający procentowe udziały uczniów uprawiających poszczególne dyscypliny sportu. Czy musisz tworzyć nowy wykres, jeśli zmienisz dane liczbowe?
5. Przygotuj prezentację graficzną danych zapisanych w pliku *Statystyka szkolna* w zadaniu 3., w temacie 12. Zastanów się nad wyborem typu wykresu i liczbą serii danych. Możesz utworzyć kilka wykresów różnych typów.

Dla zainteresowanych

6. Przygotuj prezentację graficzną danych zapisanych w pliku *Oceny* w zadaniu 1., w temacie 12. Na wykresie przedstaw liczby ocen od 1 do 6 dla każdej klasy od IVA do VIIIC. Nazwy klas umieść jako etykiety osi poziomej (kategorii). Dobierz odpowiedni typ wykresu.

Przeczytaj, jeśli chcesz wiedzieć więcej…

Aby otrzymać wykres funkcji matematycznej, wystarczy wpisać wzór tej funkcji w wyszukiwarce Google, np. $y = 2x + 1$. W ten sposób można tworzyć wykresy funkcji jednej zmiennej (dwuwymiarowe) lub dwóch zmiennych (trójwymiarowe).

Temat 14. Zastosowanie arkusza kalkulacyjnego

1. Wykorzystanie arkusza kalkulacyjnego do realizacji algorytmów
2. Fizyka – badanie ruchu jednostajnego prostoliniowego
3. Geografia – przygotowanie referatu o województwach w Polsce
4. Modelowanie i symulacja
   - 4.1. Tworzenie modelu na przykładzie rzutu sześcienną kostką do gry
   - 4.2. Symulacja modelu rzutu sześcienną kostką do gry
5. Porządkowanie i filtrowanie danych

Warto powtórzyć

1. Jakie zasady adresowania w arkuszu kalkulacyjnym poznaliśmy?
2. Jakie są zasady projektowania tabeli?
3. Co to jest algorytm?
4. Jaki algorytm nazywamy liniowym, a jaki – algorytmem z warunkami?
5. Jaki algorytm nazywamy iteracyjnym?
6. W jaki sposób tworzy się wykresy przedstawiające jedną serię danych, a w jaki – przedstawiające kilka serii?
7. Co należy uwzględniać podczas wybierania typu wykresu?
8. W jaki sposób można wstawić wykres do dokumentu tekstowego?

1. Wykorzystanie arkusza kalkulacyjnego do realizacji algorytmów

W arkuszu kalkulacyjnym można rozwiązywać zadania rachunkowe z różnych przedmiotów szkolnych, dziedzin nauki i życia codziennego. Możemy zrealizować w nim wybrane algorytmy i proste doświadczenia losowe z rachunku prawdopodobieństwa (np. polegające na rzucie kostką do gry czy monetą), a także uporządkować i wyszukać dane. W poprzednich tematach wielokrotnie rozwiązywaliśmy zadania z matematyki, m.in. obliczaliśmy sumę i średnią arytmetyczną liczb, wykonywaliśmy obliczenia procentowe oraz tworzyliśmy różne wykresy.

Podstawą obliczeń wykonywanych za pomocą funkcji arkusza kalkulacyjnego, takich jak sumowanie, potęgowanie, pierwiastkowanie, zaokrąglanie, obliczanie średniej, jest algorytm liniowy.

Ćwiczenie 1. Realizujemy algorytm liniowy w arkuszu kalkulacyjnym

1. W arkuszu kalkulacyjnym utwórz tabelę pokazaną na rysunku 1.
2. Oblicz liczbę uczniów szkoły oraz liczbę uczniów z wszystkich klas biorących udział w wycieczce szkolnej.
3. Zapisz plik pod nazwą *Wycieczka*.

	A	B	C	D
1	klasa	liczba uczniów	liczba uczniów biorących udział w wycieczce	procent uczniów biorących udział w wycieczce
2	I	30	15	
3	II	27	8	
4	III	26	7	
5	IV	29	10	
6	V	29	5	
7	VI	24	5	
8	VII	27	3	
9	VIII	30	11	
10	RAZEM			

Rys. 1. Tabela – ćwiczenie 1.

W arkuszu kalkulacyjnym iterację realizuje się za pomocą kopiowania formuł. Formułę można skopiować tyle razy, ilu powtórzeń wymaga dane zadanie obliczeniowe.

Ćwiczenie 2. Realizujemy algorytm iteracyjny w arkuszu kalkulacyjnym

1. Otwórz plik *Wycieczka* zapisany w ćwiczeniu 1.
2. Oblicz:
   a) jaki procent uczniów z każdej klasy bierze udział w wycieczce szkolnej,
   b) jaki procent uczniów całej szkoły bierze udział w wycieczce szkolnej.
3. Zapisz plik pod tą samą nazwą.

Wskazówka: Zamiast formuły =C2/B2*100 możesz w komórce *D2* wpisać formułę =C2/B2 i ustalić dla tej komórki format procentowy.

W arkuszu kalkulacyjnym możemy realizować algorytmy z warunkami, np. stosując funkcję JEŻELI.

Ćwiczenie 3. Realizujemy algorytm z warunkami w arkuszu kalkulacyjnym

Oblicz różnicę liczb zapisanych w kolejnych wierszach, w kolumnach *A* i *B*. Wyniki umieść w odpowiednich wierszach w kolumnie *C*. Jeśli różnica jest dodatnia, to w kolumnie *D* ma się pojawić napis „dodatnia". W przeciwnym przypadku – napis „niedodatnia".

1. W pierwszym wierszu (zaczynając od komórki *A1*) umieść napisy: „liczba *a*", „liczba *b*", „różnica *a* – *b*", „kategoria".
2. W kolumnie *A* (zaczynając od komórki *A2*) wpisz liczby: 234, 78, 100, 23, 8, 34, 98, 50, 20.
3. W kolumnie *B* (zaczynając od komórki *B2*) wpisz liczby: 789, 23, 120, 7, 12, 123, 40, 13, 10.

4. W komórce *C2* umieść formułę obliczającą różnicę liczb z komórek *A2* i *B2*.
5. Skopiuj formułę do komórek C3:C10 (program wykona obliczenia automatycznie).
6. W komórce *D2* umieść formułę: =JEŻELI(C2>0;"dodatnia";"niedodatnia").
7. Skopiuj formułę do komórek D3:D10.

2. Fizyka – badanie ruchu jednostajnego prostoliniowego

Na lekcjach fizyki wyniki przeprowadzonych doświadczeń często wpisuje się do tabeli oraz przedstawia na wykresie. Czasami pomiary powtarza się kilka razy. Arkusz kalkulacyjny ułatwia opracowanie danych uzyskanych w doświadczeniu. Obliczenia dla nowych danych program wykonuje automatycznie oraz przedstawia je na wykresie.

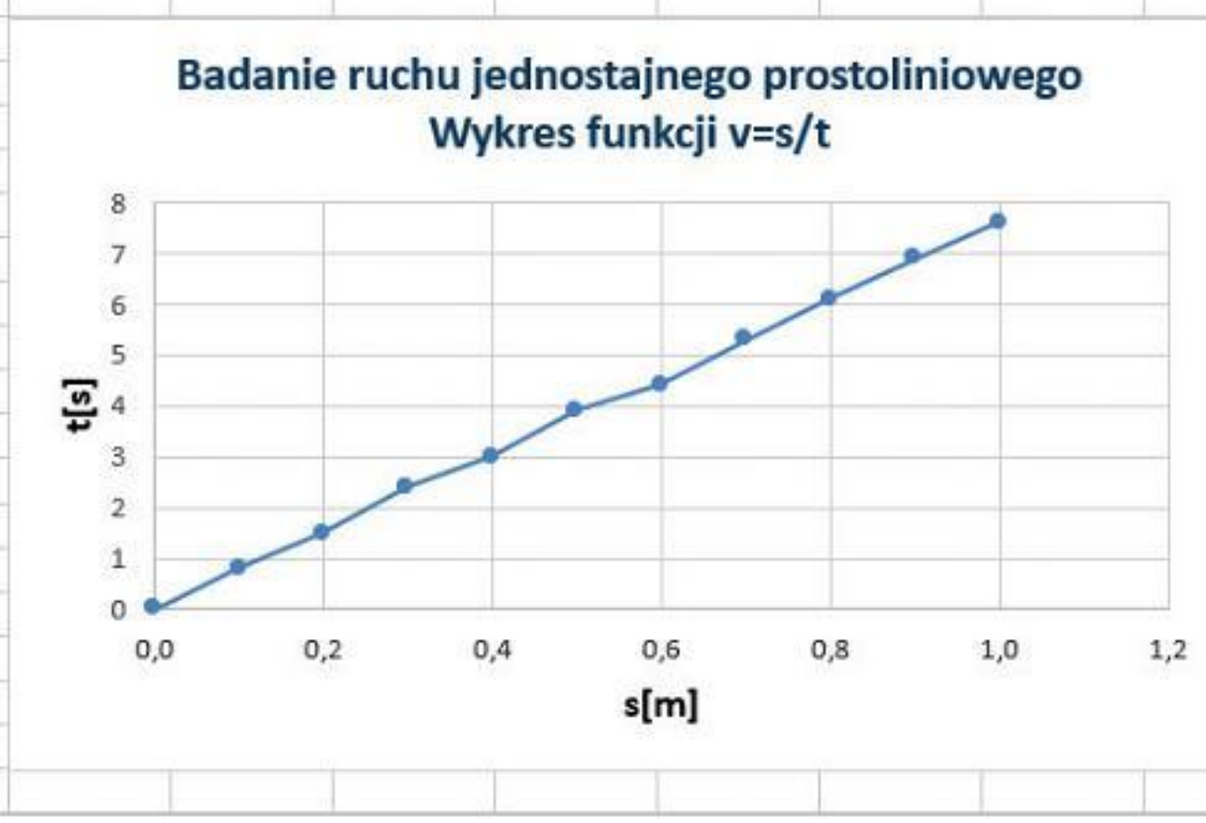
Ćwiczenie 4. Badamy ruch jednostajny prostoliniowy

Opis doświadczenia: Obserwowano ruch pęcherzyka powietrza znajdującego się w szklanej rurce napełnionej cieczą. Na rurce zaznaczono odcinki takiej samej długości: co 0,1 m (rurka ma długość 1 m). Pęcherzyk powietrza utrzymuje się zawsze u góry rurki ustawionej pionowo. Jeśli odwrócimy rurkę, pęcherzyk znajdzie się na dole i ruchem jednostajnym zacznie przesuwać się do góry. W tabeli (rys. 2.) przedstawiono wyniki pomiaru czasu, w którym pęcherzyk powietrza przebył odcinki drogi tej samej długości. Doświadczenie można powtarzać wiele razy i za każdym razem wpisywać wyniki pomiarów do tabeli arkusza kalkulacyjnego. Wartości prędkości otrzymamy automatycznie, o ile poprawnie utworzymy formułę do obliczania.

1. Zaprojektuj w arkuszu kalkulacyjnym tabelę do wpisywania wyników pomiarów w tym doświadczeniu.
2. Wpisz przykładowe wyniki.
3. Zaprojektuj formuły, przeprowadź obliczenia oraz przygotuj odpowiedni wykres.

Podobne doświadczenie, niekoniecznie z pęcherzykiem powietrza, można wykonać na lekcji fizyki lub w domu, wykorzystując do obliczeń i sporządzenia wykresu arkusz kalkulacyjny.

	A	B	C
1	s [m]	t [s]	v [m/s]
2	0,0	0	
3	0,1	0,8	0,13
4	0,2	1,5	0,13
5	0,3	2,4	0,13
6	0,4	3,0	0,13
7	0,5	3,9	0,13
8	0,6	4,4	0,14
9	0,7	5,3	0,13
10	0,8	6,1	0,13
11	0,9	6,9	0,13
12	1,0	7,6	0,13



Rys. 2. Tabela i wykres przygotowane w arkuszu kalkulacyjnym do doświadczenia – badanie ruchu jednostajnego prostoliniowego – ćwiczenie 4.

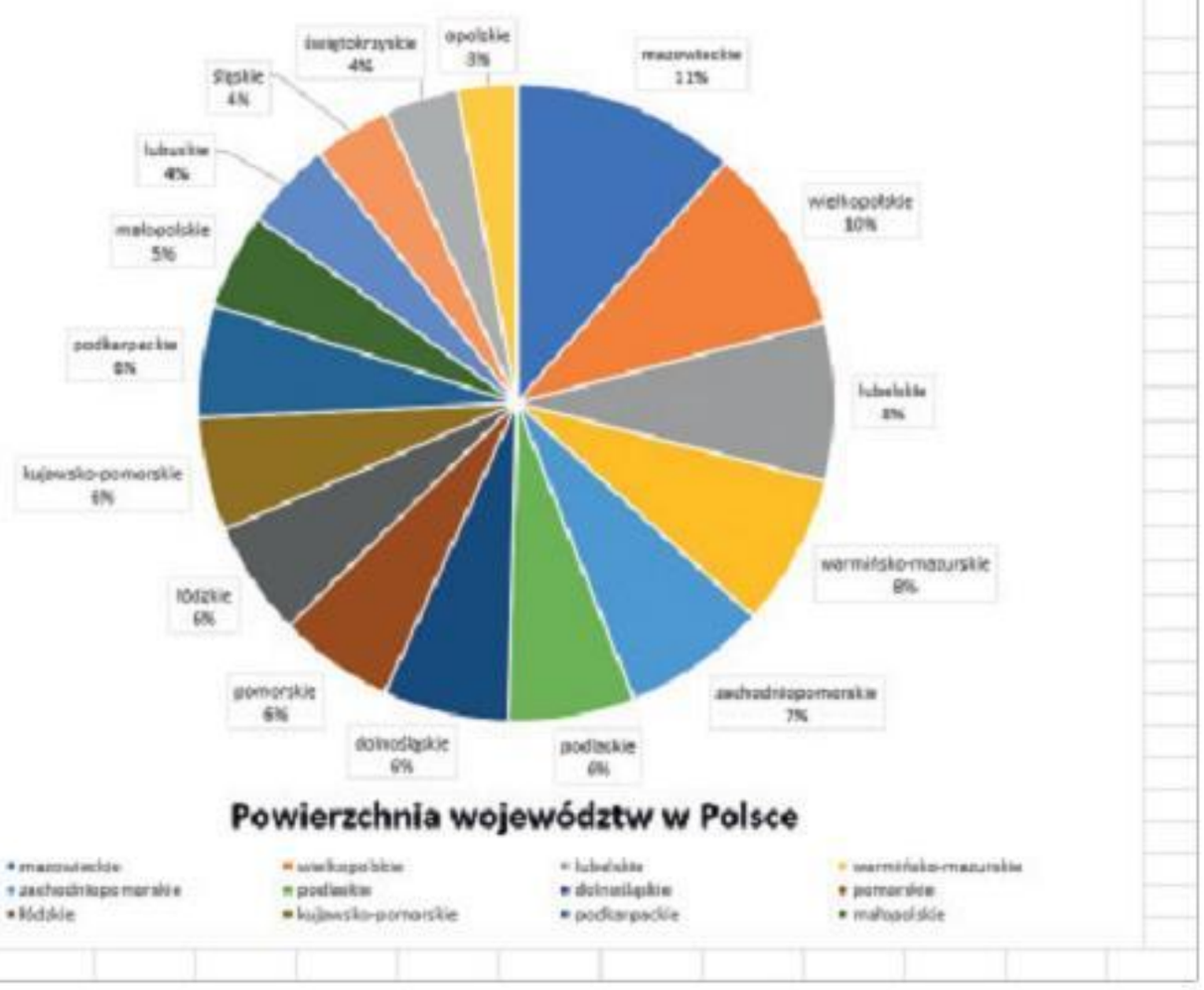
3. Geografia – przygotowanie referatu o województwach w Polsce

Niektóre informacje z geografii można zamieszczać w tabelach lub ilustrować je za pomocą wykresów. Najczęściej są to wykresy kołowe lub kolumnowe.

Ćwiczenie 5. Wykonujemy wykres kołowy

1. Tabela na rysunku 3. pokazuje zestawienie powierzchni poszczególnych województw w kilometrach kwadratowych w kolejności malejącej. Przedstaw w postaci wykresu kołowego powierzchnię Polski według województw. Umieść na wykresie etykiety danych (nazwy województw i wartości procentowe).
2. Zapisz plik pod nazwą *Województwa*.

Lp.	Województwo	Powierzchnia w km²
1	mazowieckie	35559
2	wielkopolskie	29827
3	lubelskie	25122
4	warmińsko-mazurskie	24174
5	zachodniopomorskie	22907
6	podlaskie	20187
7	dolnośląskie	19947
8	pomorskie	18336
9	łódzkie	18218
10	kujawsko-pomorskie	17972
11	podkarpackie	17845
12	małopolskie	15184
13	lubuskie	13988
14	śląskie	12334
15	świętokrzyskie	11709
16	opolskie	9411
	Polska	**312720**



Rys. 3. Przykład wykresu kołowego obrazującego powierzchnię Polski według województw. Dane z 2022 roku. Źródło: *http://stat.gov.pl/* – ćwiczenie 5.

Ćwiczenie 6. Przygotowujemy referat o województwach w Polsce

Masz przygotować referat na temat województw w Polsce.

1. Utwórz dokument tekstowy, w którym podasz podstawowe informacje o każdym województwie, m.in. o powierzchni (rys. 3.), stolicy, liczbie ludności, gęstości zaludnienia, położeniu, atrakcjach turystycznych, przyrodniczych i krajobrazowych.
2. Przygotuj tabele z odpowiednimi danymi i formułami oraz przedstaw ważniejsze dane na wykresach.
3. Wstaw wybrane tabele i wykresy do dokumentu tekstowego (m.in. wykres utworzony w ćwiczeniu 5.). Zachowaj połączenie z plikiem źródłowym.

Wskazówka: Ćwiczenie można wykonać grupowo – rozdzielić zadania szczegółowe, a potem połączyć dokumenty w jeden referat. Wyszukaj informacje w Internecie lub innych źródłach.

4. Modelowanie i symulacja

Wiele zjawisk obrazuje się za pomocą **modeli**. Dzięki obserwowaniu zachowania modelu możemy wyciągać wnioski dotyczące opisywanej rzeczywistości. Na przykład można modelować zależność oporu powietrza od kształtu samochodu.

Najpierw należy zbadać zjawisko, które chcemy opisać za pomocą modelu. Następnie, korzystając z narzędzia informatycznego, np. arkusza kalkulacyjnego lub języka programowania, można zbudować model.

4.1. Tworzenie modelu na przykładzie rzutu sześcienną kostką do gry

Modele to często bardzo skomplikowane programy (np. symulatory lotu samolotem). Przedstawimy prosty model rzutu sześcienną kostką do gry, który wykonamy w arkuszu kalkulacyjnym (rys. 4.).

Skorzystamy z funkcji losowej LOS(), generującej losowe liczby z przedziału ⟨0; 1) (widoczne w kolumnie *A* arkusza kalkulacyjnego).

Jeśli do komórki *B2* wprowadzimy formułę: = LICZBA.CAŁK(A2*6+1) i skopiujemy ją do odpowiedniej liczby komórek w kolumnie *B*, to otrzymamy liczby z przedziału ⟨1; 6⟩, czyli możliwe wyniki rzutu kostką.

Obliczamy, ile razy dla różnej liczby rzutów wypadła szóstka, piątka, czwórka itd. Korzystamy ze standardowej funkcji LICZ.JEŻELI, która oblicza liczbę komórek we wskazanym zakresie, spełniających określone kryteria, np. formuła wprowadzona do komórki *E2*: =LICZ.JEŻELI(B2:B601;6) policzy, ile jest szóstek w komórkach od *B2* do *B601*, czyli dla 600 rzutów.

4.2. Symulacja modelu rzutu sześcienną kostką do gry

Ćwiczenie 7. Badamy model

1. Korzystając z przedstawionego opisu i rysunku 4., utwórz w arkuszu kalkulacyjnym tabelę przedstawiającą model rzutu sześcienną kostką do gry.
2. Losuj kolejne wyniki rzutów (naciskając klawisz **F9**) i obserwuj, jak zmieniają się wartości w tabeli – ile razy wypadły jedynka, dwójka, trójka, czwórka, piątka, szóstka.

Wskazówki:

- W kolumnie *A* wygeneruj losowo 6000 liczb z zakresu ⟨0; 1), stosując w kolumnie *A* formułę =LOS().
- W komórce *B2* umieść formułę =LICZBA.CAŁK(A2*6+1), która przekształca liczby z kolumny *A* na liczby całkowite z zakresu ⟨1; 6⟩. Skopiuj tę formułę do pozostałych komórek w kolumnie *B*.
- W komórce *E2* umieść formułę =LICZ.JEŻELI(B$2:B$7;6), skopiuj ją do komórek *E10*, *H2* i *H10* i odpowiednio zmodyfikuj zakres danych (z kolumny *B*), aby dostosować go do symulowanej liczby rzutów.
- Formuły z komórek *E2*, *E10*, *H2* i *H10* skopiuj do komórek, w których zliczane będą liczby piątek, czwórek, trójek, dwójek, jedynek i odpowiednio zmodyfikuj kryterium (wynik rzutu).

Czy wiesz że…

w 6000 losowaniach Lotto (dawniej: Dużego Lotka) najczęściej wypadała liczba 34 (792 razy), a najrzadziej – liczba 43 (657 razy)? Możesz się o tym przekonać samodzielnie, korzystając z arkusza kalkulacyjnego oraz informacji o wynikach losowań znalezionych w Internecie. Może nawet uda ci się znaleźć formułę na trafienie „szóstki" ☺.

E2 =LICZ.JEŻELI(B$2:B$7;6)

	A	B	C	D	E	F	G	H
1	losowanie liczby	wynik rzutu kostką		dla 6 rzutów			dla 600 rzutów	
2	0,7010529	5		liczba szóstek	0		liczba szóstek	109
3	0,6619574	4		liczba piątek	1		liczba piątek	101
4	0,0741211	1		liczba czwórek	2		liczba czwórek	107
5	0,3083488	2		liczba trójek	1		liczba trójek	84
6	0,6017089	4		liczba dwójek	1		liczba dwójek	98
7	0,4707522	3		liczba jedynek	1		liczba jedynek	101
8	0,8076906	5						
9	0,410798	3		dla 60 rzutów			dla 6000 rzutów	
10	0,3114939	2		liczba szóstek	11		liczba szóstek	1031
11	0,0585451	1		liczba piątek	12		liczba piątek	1001
12	0,9053252	6		liczba czwórek	9		liczba czwórek	1018
13	0,260231	2		liczba trójek	10		liczba trójek	988
14	0,1012728	1		liczba dwójek	7		liczba dwójek	1000
15	0,7058908	5		liczba jedynek	11		liczba jedynek	962
16	0,560424	4						
17	0,8420833	6						
18	0,0904261	1						

Rys. 4. Przykład modelu rzutu kostką do gry zrealizowany w arkuszu kalkulacyjnym – ćwiczenie 7.

Obserwacja zachowania modelu rzutu sześcienną kostką (czyli jego **symulacja**) nasuwa następujące wnioski: im więcej rzutów, tym liczby poszczególnych wyników są do siebie bardziej zbliżone. Dla kogoś, kto zna rachunek prawdopodobieństwa, nie jest to niespodzianką. Można powiedzieć, że skonstruowany model wiernie odwzorowuje rzeczywistość. Dla osoby nieznającej teorii prawdopodobieństwa model przybliża to zjawisko w sposób mniej pracochłonny niż rzucanie kostką (trudno przecież rzucać kostką np. 6000 razy, zliczać i zapisywać wyniki).

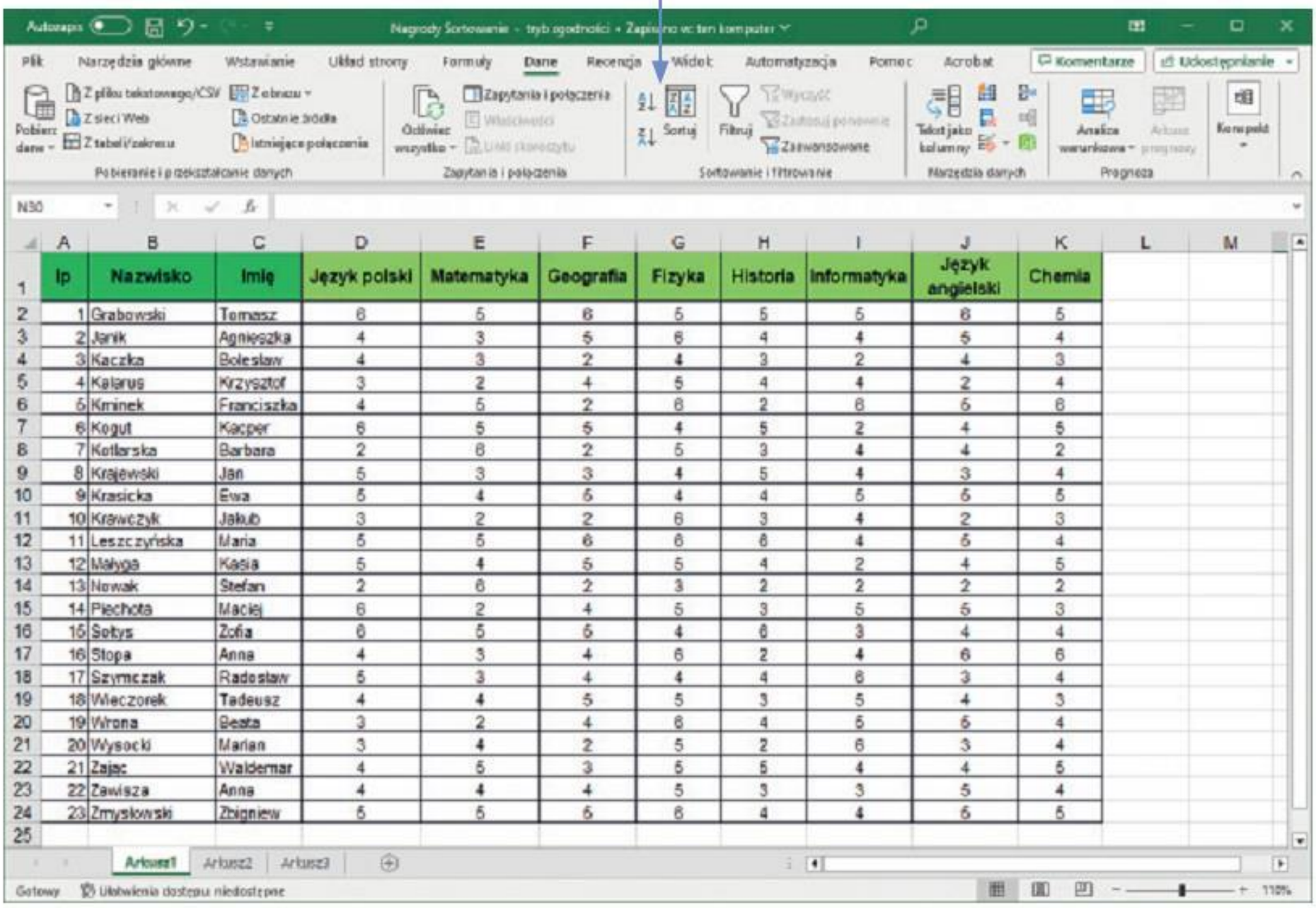
W językach programowania (np. C++, Python) również istnieje możliwość generowania liczb losowych. Można zatem, używając niewielu instrukcji, napisać program, który będzie przedstawiał model rzutu kostką do gry.

5. Porządkowanie i filtrowanie danych

W tabeli arkusza kalkulacyjnego (rys. 5.) chcemy uporządkować dane uczniów w kolejności od najwyższej do najniższej oceny z informatyki i znaleźć nazwiska uczniów, którzy z matematyki otrzymali szóstki. Jak to zrobić?

Ręczne wykonanie takich operacji, zwłaszcza dla dużej liczby uczniów, jest pracochłonne. W arkuszu kalkulacyjnym tego typu zadania możemy wykonać, stosując opcje sortowania i filtrowania. Ważne jest, aby kolumny w tabeli miały nagłówki, szczególnie przy stosowaniu opcji filtrowania.

Aby uporządkować dane według określonego kryterium, należy:
- kliknąć dowolną komórkę w kolumnie, w której chcemy posortować dane,
- z karty lub menu **Dane** wybrać opcję (kryterium) sortowania: **Sortuj od najmniejszych do największych** lub **Sortuj od największych do najmniejszych** (rys. 5.).



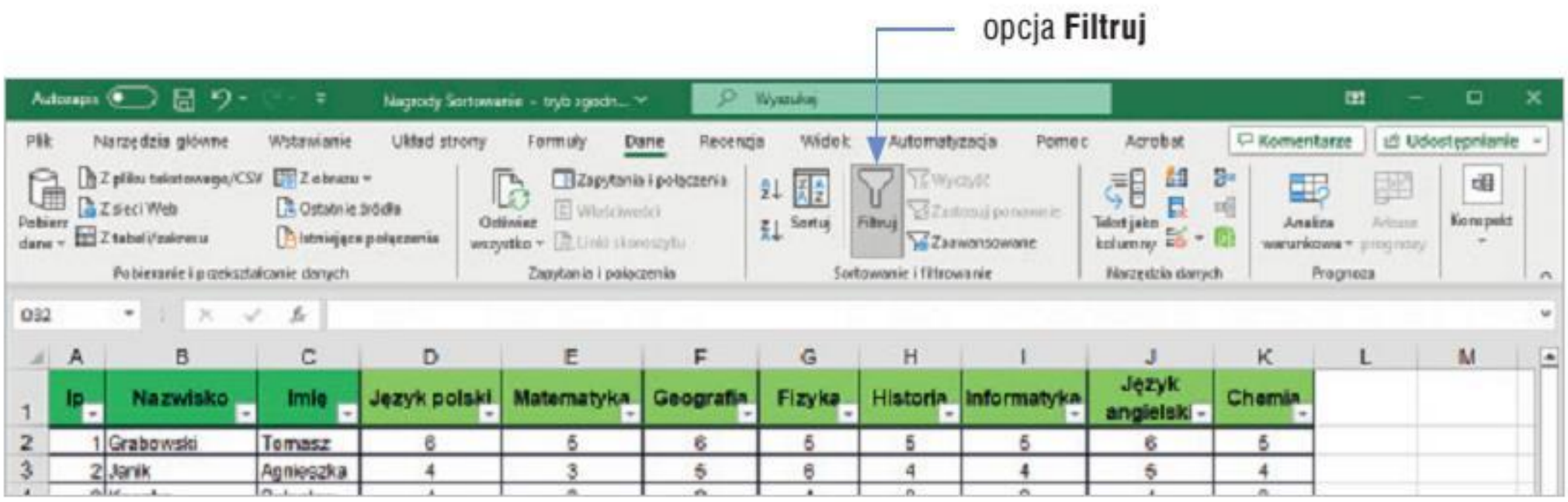
	A	B	C	D	E	F	G	H	I	J	K	L	M
1	lp	Nazwisko	Imię	Język polski	Matematyka	Geografia	Fizyka	Historia	Informatyka	Język angielski	Chemia		
2	1	Grabowski	Tomasz	6	5	6	5	5	5	6	5		
3	2	Janik	Agnieszka	4	3	5	6	4	4	5	4		
4	3	Kaczka	Bolesław	4	3	2	4	3	2	4	3		
5	4	Kalarus	Krzysztof	3	2	4	5	4	4	2	4		
6	5	Kminek	Franciszka	4	5	2	6	2	6	6	6		
7	6	Kogut	Kacper	6	5	5	4	5	2	4	5		
8	7	Kotlarska	Barbara	2	6	2	5	3	4	4	2		
9	8	Krajewski	Jan	5	3	3	4	5	4	3	4		
10	9	Krasicka	Ewa	5	4	5	4	4	5	5	5		
11	10	Krawczyk	Jakub	3	2	2	6	3	4	2	3		
12	11	Leszczyńska	Maria	5	5	6	6	6	4	5	4		
13	12	Małyga	Kasia	5	4	5	5	4	2	4	5		
14	13	Nowak	Stefan	2	6	2	3	2	2	2	2		
15	14	Piechota	Maciej	6	2	4	5	3	5	5	3		
16	15	Sołtys	Zofia	6	5	5	4	6	3	4	4		
17	16	Stopa	Anna	4	3	4	6	2	4	6	6		
18	17	Szymczak	Radosław	5	3	4	4	4	6	3	4		
19	18	Wieczorek	Tadeusz	4	4	5	5	3	5	4	3		
20	19	Wrona	Beata	3	2	4	6	4	5	5	4		
21	20	Wysocki	Marian	3	4	2	5	2	6	3	4		
22	21	Zając	Waldemar	4	5	3	5	5	4	4	5		
23	22	Zawisza	Anna	4	4	4	5	3	3	5	4		
24	23	Zmysłowski	Zbigniew	5	5	5	6	4	4	5	5		
25													

Rys. 5. Przykładowa tabela – ćwiczenie 8.

Po wybraniu kryterium sortowania dla określonej kolumny program uporządkuje wszystkie wiersze tabeli według wartości znajdujących się w tej kolumnie.

Ćwiczenie 8. Porządkujemy dane w tabeli

1. Tabela na rysunku 5. zawiera fikcyjne (uporządkowane alfabetycznie) nazwiska uczniów i ich oceny z poszczególnych przedmiotów. Utwórz podobną tabelę.
2. Uporządkuj dane uczniów według ocen z informatyki – od najwyższej do najniższej.
3. Zapisz plik pod nazwą *Nagrody sortowanie*.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	lp	Nazwisko	Imię	Język polski	Matematyka	Geografia	Fizyka	Historia	Informatyka	Język angielski	Chemia		
2	1	Grabowski	Tomasz	6	5	6	5	5	5	6	5		
3	2	Janik	Agnieszka	4	3	5	6	4	4	5	4		

Rys. 6. W nagłówkach tabeli widać włączoną opcję **Filtruj** – ćwiczenie 9.

Aby znaleźć dane spełniające określone kryterium, należy:

- kliknąć dowolną komórkę tabeli,
- z karty lub menu menu **Dane** wybrać opcję filtrowania (**Filtr** w Microsoft Excel lub **Autofiltr** w LibreOffice Calc) – w nagłówkach kolumn pojawią się przyciski ze strzałkami (rys. 6.),
- kliknąć strzałkę w nagłówku kolumny, w której chcemy wyszukać dane, np. w polu *Matematyka* – rozwinie się lista z możliwościami wyboru różnych kryteriów filtrowania (rys. 7.),
- wybrać wartość (wartości) i kliknąć przycisk **OK** – w tabeli zobaczymy wiersze spełniające wybrane kryterium.



Po wybraniu określonej liczby w kolumnie *Matematyka* ograniczymy wyświetlanie tabeli do wierszy zawierających tę liczbę w wybranej kolumnie, ponieważ program ukrywa wiersze, które nie spełniają wybranego kryterium.

Rys. 7. Stosowanie filtrów do wyszukiwania danych – ćwiczenie 9.

Aby wyłączyć widok filtru, należy wybrać tę samą opcję, która włącza ten widok (**Filtr** lub **Autofiltr**).

Aby ponownie wyświetlić wszystkie wiersze, należy przy polach, dla których ustawiono filtry, zaznaczyć opcję wyświetlenia wszystkich wierszy lub opcję wyczyszczenia filtru (np. **Wyczyść**).

Aby wybrać dodatkowe filtry (np. Mniejsze niż, Mniejsze niż lub równe), należy w oknie wyboru kryterium (rys. 7.) kliknąć polecenie **Filtry liczb** (Microsoft Excel) lub **Filtrowanie wg warunku/Filtr standardowy** (LibreOffice Calc).

Ćwiczenie 9. Stosujemy filtrowanie do wyszukiwania danych

1. Otwórz plik *Nagrody sortowanie* zapisany w ćwiczeniu 8.
2. Uporządkuj dane uczniów według liczby porządkowej – rosnąco (od najmniejszej do największej).
3. Zapisz plik pod nazwą *Nagrody filtrowanie*.
4. Wyświetl w tabeli tylko dane uczniów, którzy otrzymali z matematyki piątki lub szóstki. Nie zapisuj zmian.

5. Wyświetl ponownie wszystkie wiersze i wyszukaj dane uczniów, którzy otrzymali z informatyki oceny niższe niż 4. Nie zapisuj zmian.
6. Wyświetl ponownie wszystkie wiersze i wyszukaj dane uczniów, którzy otrzymali z języka polskiego oceny wyższe od 4 lub równe 4. Nie zapisuj zmian.

Warto zapamiętać

- W arkuszu kalkulacyjnym możemy realizować algorytmy liniowe, z warunkami (korzystając z funkcji JEŻELI) i iteracyjne (wykorzystując kopiowanie formuł).
- Przykłady zastosowania arkusza kalkulacyjnego:
  - do rysowania wykresu i analizy funkcji liniowej, np. na lekcjach matematyki,
  - do pokazania w postaci wykresu ruchu jednostajnego, np. na lekcji fizyki,
  - do zilustrowania danych w postaci tabel i wykresów, np. na lekcji geografii,
  - do tworzenia modeli zjawisk.
- Zadaniem modelu jest umożliwienie wyciągania wniosków dotyczących opisywanej rzeczywistości.
- W arkuszu kalkulacyjnym można stosować sortowanie i filtrowanie, które umożliwiają porządkowanie danych umieszczonych w tabeli i wyszukiwanie danych według określonych kryteriów.

Pytania i polecenia

1. W jaki sposób można zrealizować w arkuszu kalkulacyjnym algorytm z warunkami? Wyjaśnij na przykładzie.
2. Dlaczego do narysowania wykresu prędkości w ruchu jednostajnym prostoliniowym w arkuszu kalkulacyjnym najlepiej wybrać wykres **XY** (**punktowy**), a do przedstawienia danych dotyczących powierzchni województw – wykres kołowy?
3. W jakim celu buduje się modele rzeczywistych zjawisk? Na przykładzie tworzenia modelu rzutu kostką omów, na czym polegają modelowanie i symulacja.
4. W jaki sposób uporządkować malejąco dane umieszczone w tabeli (rys. 5.) według średniej ocen ze wszystkich przedmiotów?
5. Jaki filtr zastosujesz, aby wybrać z tabeli (rys. 5.) uczniów, którzy otrzymali szóstkę lub piątkę z języka angielskiego?

Zadania

1. Przedstaw w arkuszu kalkulacyjnym algorytm obliczania pola powierzchni całkowitej sześcianu o krawędzi *a*. Obliczenia wykonaj dla dwudziestu liczb.
2. Zmodyfikuj zadanie 1. tak, aby dla *a* dodatniego program obliczył pole powierzchni całkowitej i wyprowadził wynik, a w przeciwnym przypadku wyświetlił napis „niepoprawna wartość".
3. Uczniowie mierzyli temperaturę wody z jeziora. Bezpośrednio po przyniesieniu do laboratorium woda miała temperaturę 11 °C. Po upływie czterech minut termometr wskazywał 14 °C, a po następnych czterech minutach pokazywał 17 °C. Cztery kolejne pomiary, przeprowadzane co 4 minuty, wskazywały temperaturę 20 °C. Przedstaw wyniki pomiarów w tabeli i sporządź wykres zależności temperatury próbki wody od czasu pomiaru. Jeśli masz taką możliwość, to wykonaj kilkukrotnie podobne pomiary, wyniki wpisz do tabeli i odpowiedz na pytanie: *Czy otrzymasz podobne wykresy w każdym doświadczeniu?* Spróbuj uzasadnić odpowiedź.

4. Zaprojektuj tabelę i wykonaj wykres obrazujący strukturę użytków rolnych w Polsce, biorąc pod uwagę kolejne lata od roku 2010 do roku ubiegłego. Utwórz formułę obliczającą procentowy udział danego rodzaju użytków w ogólnej powierzchni użytków rolnych. Czy wykres kołowy będzie tu odpowiedni? Uzasadnij odpowiedź. Dane znajdź w Internecie, rocznikach statystycznych lub w innych źródłach.
5. Przygotuj tabelę i wykres z danymi dotyczącymi struktury zatrudnienia w Polsce. Uwzględnij zatrudnienie m.in. w przemyśle, rolnictwie, budownictwie, komunikacji, handlu, oświacie, służbie zdrowia. Jaki typ wykresu najlepiej zastosować? Dane odszukaj w Internecie, podręcznikach szkolnych, encyklopedii lub innych źródłach.
6. Przygotuj tabelę z danymi dotyczącymi liczby ludności kilku większych miast Polski i przedstaw dane na odpowiednim wykresie. Uwzględnij lata przedwojenne, zaraz po wojnie i stan obecny. Przygotuj w edytorze tekstu krótką analizę porównawczą tych danych, a następnie włącz do tekstu tabelę oraz wykres. Dane znajdź w Internecie, rocznikach statystycznych lub w innych źródłach.
7. Korzystając z arkusza kalkulacyjnego, wykonaj model rzutu monetą.
8. Korzystając z arkusza kalkulacyjnego, wykonaj model rzutu dwiema i trzema kostkami do gry. Jaka suma oczek wypada najczęściej przy rzucie dwiema kostkami, a jaka – przy rzucie trzema kostkami?
9. Otwórz plik *Nagrody_filtrowanie* zapisany w ćwiczeniu 9.
   a. Oblicz średnią ocen każdego ucznia (wyniki umieść w kolumnie *L*). Wyświetl w tabeli tylko dane uczniów, którzy mają średnią ocen wyższą niż 4,5.
   b. Wyświetl ponownie wszystkie wiersze i wyszukaj dane uczniów, którzy mają średnią ocen niższą niż 4.
   c. Wyświetl ponownie wszystkie wiersze i wyszukaj wybrane wiersze, ustalając kryterium wyszukiwania według własnego pomysłu.

Dla zainteresowanych

10. Przedstaw w arkuszu kalkulacyjnym algorytm obliczania objętości prostopadłościanu o krawędziach *a*, *b*, *c*. Uwzględnij warunek: dla *a*, *b* i *c* dodatnich powinna zostać obliczona i wyświetlona objętość, w przeciwnym przypadku powinien pojawić się napis „niepoprawne dane”.
11. Przygotuj w edytorze tekstu referat z geografii o polskim przemyśle. Umieść w dokumencie odpowiednie tabele i wykresy.
12. Zaprojektuj doświadczenie badające ruch jednostajnie zmienny (np. staczanie się piłki lub wózka po równi pochyłej). Przygotuj w arkuszu kalkulacyjnym odpowiednie tabele i wykresy, np. wykres prędkości piłeczki na równi pochyłej, wykres położenia piłeczki (wózka) w zależności od czasu.
13. Dokonaj szczegółowego porównania dokumentu utworzonego w arkuszu kalkulacyjnym z dokumentem utworzonym za pomocą edytora tekstu – uwzględnij podobieństwa i różnice. Opracuj to porównanie w edytorze tekstu.
14. Otwórz plik *Oceny* zapisany w zadaniu 1., w temacie 12. Ustal trzy kryteria wyszukiwania według własnego pomysłu i wyświetl odpowiednio dane.

Przeczytaj, jeśli chcesz wiedzieć więcej…

Aby do wybranej funkcji wprowadzić adresy komórek, które ze sobą nie sąsiadują, należy klikać te komórki, trzymając wciśnięty klawisz **Ctrl** (Microsoft Excel, LibreOffice Calc).

Internet

Rozdział V

Temat 15. Tworzenie strony internetowej z wykorzystaniem znaczników HTML – projekt

Temat 16. Systemy zarządzania treścią – projekt



Temat 15. Tworzenie strony internetowej z wykorzystaniem znaczników HTML – projekt

1. Narzędzia do tworzenia stron internetowych
2. Ogólne zasady projektowania stron internetowych
3. Kod źródłowy strony internetowej
4. Tworzenie strony internetowej „Podróże po stolicach Europy”
   - 4.1. Formatowanie tekstu
   - 4.2. Wypunktowania i tabele
   - 4.3. Hiperłącza do innych stron
   - 4.4. Wstawianie grafiki

Warto powtórzyć

1. Do czego służy przeglądarka internetowa?
2. W jaki sposób informacje wyświetlają się na stronach internetowych? Omów sposób rozmieszczenia tekstu, grafiki i innych elementów.
3. Jaki mechanizm umożliwia przechodzenie od jednej strony internetowej do drugiej?
4. Jakiego rodzaju dokumenty komputerowe tworzyliśmy?

1. Narzędzia do tworzenia stron internetowych

Język znaczników HTML jest podstawowym (choć nie jedynym) narzędziem do przygotowywania stron WWW. Trzeba zdawać sobie sprawę, że ciągle powstają coraz nowsze technologie internetowe i różnego rodzaju narzędzia do profesjonalnego tworzenia stron internetowych.

Znacznik języka HTML Ciąg znaków zapisany w nawiasach kątowych, np. `<img src="rysunek.jpg">`.

Strona WWW – podobnie jak tabela w arkuszu kalkulacyjnym, rysunek w edytorze grafiki czy tekst w edytorze tekstu – jest dokumentem komputerowym, zapisanym zwykle w pliku z rozszerzeniem *html* lub *htm*. Jest to plik tekstowy, w którym umieszcza się specjalne **znaczniki języka HTML**.

Do utworzenia prostej strony internetowej można zastosować:

- edytory plików tekstowych, w których sami wstawiamy znaczniki HTML, np. Notatnik,
- edytory plików tekstowych dla programistów, np. Notepad++, Microsoft Visual Studio Code, Notepad3 (dostępne bezpłatnie w Internecie),

- programy pozwalające na pracę z dokumentem bez znajomości języka HTML (na ekranie edytora widzimy stronę tak, jak będzie wyglądać) oraz na zapis dokumentu w formacie HTML.

Edytory tekstu zawierają narzędzia pozwalające na szybkie przygotowanie strony WWW na podstawie szablonu lub istniejącego dokumentu. Korzystamy wówczas ze specjalnych poleceń (umieszczonych na pasku narzędzi), związanych z HTML, które automatycznie tworzą odpowiednie znaczniki HTML.

Do tworzenia rozbudowanej witryny stosuje się:

- specjalistyczne narzędzia przeznaczone dla osób zajmujących się tworzeniem stron internetowych; podczas tworzenia witryny wymagana jest zazwyczaj znajomość baz danych i skryptowych języków programowania,
- systemy zarządzania treścią (CMS) – narzędzia pozwalające na szybkie i łatwe utworzenie strony internetowej (nawet rozbudowanej) przeznaczone dla użytkowników, którzy nie mają szczegółowej wiedzy technicznej (np. Wordpress, Joomla, Drupal).

Oprócz języka HTML do opisywania wyglądu i funkcjonowania witryn internetowych wykorzystuje się kaskadowe arkusze stylów (CSS) oraz język programowania JavaScript.

2. Ogólne zasady projektowania stron internetowych

Profesjonalna strona WWW składa się zwykle nie z jednej, lecz z kilku połączonych stron tworzących tzw. witrynę. Najważniejsza jest strona główna, z której – dzięki odsyłaczom – można przejść do innych stron (podstron). Stanowi ona wejście do innych części witryny. Jeśli witrynę właściwie zaprojektowano i wykonano, do strony głównej można wrócić z dowolnej podstrony.

Pierwszym krokiem w projektowaniu strony WWW jest określenie jej tematyki. Następnie należy zaplanować:

- treści, w tym elementy graficzne (rysunki, zdjęcia), które chcemy zamieścić na stronie, sposób rozmieszczenia grafiki względem tekstu,
- parametry formatowania tekstu, stosowanie numeracji, wyliczeń, tabel,
- kolorystykę strony (tła, tekstu, elementów graficznych).

Na stronie możemy umieścić wiele elementów graficznych, nie należy jednak przesadzać z ich liczbą, ponieważ strona stanie się nieczytelna, a właściwa zawartość zejdzie na drugi plan.

Wykonanie dobrej strony internetowej nie jest łatwe. Takimi zadaniami zajmują się często specjalistyczne firmy. Dlatego nie zniechęcajmy się, jeśli pierwsze tworzone przez nas strony nie będą doskonałe.

Ćwiczenie 1. Przeglądamy i porównujemy strony WWW

1. Porównaj kilka witryn; wejdź kolejno na ich strony główne. Zobacz m.in. strony: *www.sejm.gov.pl*, *cke.gov.pl*.
2. Zwróć uwagę na strukturę strony: tytuł, rozmieszczenie tekstu, przejrzystość, grafikę, sposób przechodzenia do innych stron.

3. Kod źródłowy strony internetowej

Dobra rada

Zanim umieścimy materiały w Internecie, warto, aby przejrzały je osoby, które poprawią błędy i doradzą ewentualne zmiany.

Zaczniemy od zaprojektowania i utworzenia jednej strony. Wykorzystamy w tym celu prosty edytor plików tekstowych i język HTML. Aby lepiej wyobrazić sobie, jak wygląda **kod źródłowy strony WWW**, czyli jak umieszcza się w tekście znaczniki HTML, przeanalizujemy kod HTML przykładowej strony. Funkcję podglądu kodu oferują prawie wszystkie przeglądarki internetowe. Oglądane przez nas źródła stron będą zazwyczaj bardzo rozbudowane i skomplikowane.

Dobra rada

Efektem tworzenia strony internetowej jest zestaw plików HTML i towarzyszących mu plików pomocniczych (graficznych, multimedialnych). Dlatego warto znać znaczniki HTML – wykonujemy wówczas wszystkie działania bardziej świadomie.

Aby zobaczyć kod HTML (źródło) strony WWW, można wskazać wolny od tekstu i grafiki obszar strony i wybrać z menu kontekstowego opcję **Wyświetl źródło strony** (**Pokaż źródło strony**).

Uwaga

Aby kod strony był przejrzysty, należy stosować w nim wcięcia (tak aby znacznik początkowy i końcowy zaczynały się w tej samej kolumnie, a kod pomiędzy nimi był przesunięty w prawo).

Znaczniki HTML występują zwykle parami: **znacznik początkowy** i **końcowy**. Tekst umieszczony między znacznikiem początkowym a końcowym przeglądarka interpretuje (np. wyświetla) w sposób określony przez znacznik.

Ogólny sposób stosowania znaczników jest następujący: `<znacznik>tekst</znacznik>`.

Znaczniki z reguły zapisujemy małymi literami.

Opis całej strony zawieramy między znacznikami `<html>` ... `</html>` – ten znacznik powinien zawsze wystąpić. Nagłówek strony wpisujemy w znacznikach `<head>` ... `</head>`, a w nim – między znacznikami `<title>` ... `</title>` – umieszczamy tytuł strony, który pojawia się na pasku tytułowym okna przeglądarki. Zawartość strony (teksty, rysunki) umieszczamy między znacznikami `<body>` ... `</body>`.

Ułożenie znaczników w podstawowej strukturze strony

początek struktury — `<html>`
początek nagłówka — `<head>`
tytuł strony — `<title>`*Tytuł_strony*`</title>`
koniec nagłówka — `</head>`
początek zawartości strony — `<body>`
treść strony (tekst, grafika, hiperłącza)
koniec zawartości strony — `</body>`
koniec struktury — `</html>`

4. Tworzenie strony internetowej „Podróże po stolicach Europy”

Zamierzamy utworzyć stronę internetową poświęconą stolicom państw europejskich. Na stronie chcemy umieścić fotografie i informacje o najciekawszych zabytkach lub ważnych obiektach, wskazówki dla turystów dotyczące godzin otwarcia muzeów oraz opłat za zwiedzanie, ciekawostki historyczne itp. Jak wykonać taką stronę, korzystając z edytora plików tekstowych, np. Notatnika?

Podczas projektowania strony należy zastanowić się nad wyborem jej układu i kolorystyki oraz nad elementami, które chcemy umieścić. Wprowadzenie zbyt wielu elementów może się okazać zbyt trudne dla początkującego twórcy stron WWW, więc postaramy się je ograniczyć.



Aby utworzyć stronę WWW w edytorze plików tekstowych, należy w oknie nowego dokumentu wpisać tekst wraz z odpowiednimi znacznikami. Następnie, za pomocą opcji **Zapisz jako**, należy zapisać dokument w pliku w formacie HTML, czyli z rozszerzeniem *html* lub *htm*.



Ćwiczenie 2. Tworzymy podstawową strukturę strony WWW

1. Zobacz kod źródłowy wybranej strony internetowej.
2. Utwórz folder o nazwie *Strony*, w którym będziesz zapisywać swoje strony WWW. Uruchom edytor plików tekstowych, np. Notatnik, Notepad++.
3. Wprowadź znaczniki `<html>`, `<head>`, `<title>`, `</title>`, `</head>`, `<body>`, `</body>`, `</html>` według wzoru przedstawionego w punkcie 3. tego tematu. W nagłówku wpisz tytuł strony: „Podróże”. Dodaj również polecenia widoczne w pierwszym i piątym wierszu kodu źródłowego na rysunku 1.
4. W zawartości strony wpisz: „Podróże po stolicach Europy”. Dodaj cztery zdania o tym, co znajdzie się na stronie. Każde zdanie wpisz w oddzielnym wierszu.
5. Zapisz plik pod nazwą *index.html*.

Wskazówki: Skorzystaj z podanego wcześniej opisu znaczników. Zobacz też wzór strony internetowej przedstawiony na rysunku 1. Zapisując plik w Notatniku, należy wybrać typ pliku: **Wszystkie pliki**.

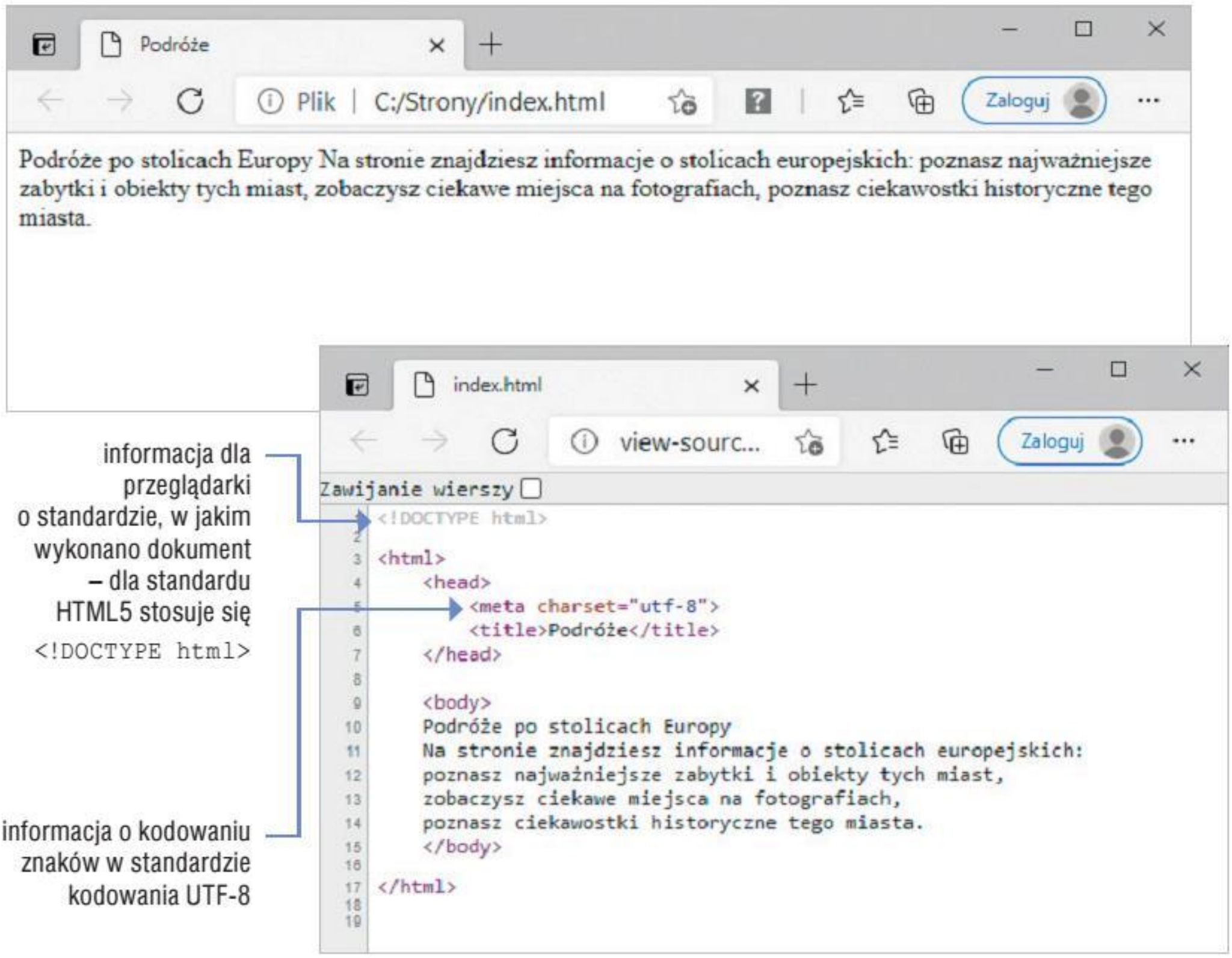


Aby zobaczyć, jak strona WWW będzie wyglądała w Internecie, należy w przeglądarce otworzyć plik, w którym zapisano stronę (opcja **Plik/Otwórz plik**). Można też dwukrotnie kliknąć ikonę pliku HTML, aby uruchomić przeglądarkę internetową i otworzyć plik.



Ćwiczenie 3. Oglądamy stronę w przeglądarce

1. Obejrzyj w przeglądarce internetowej stronę utworzoną w ćwiczeniu 2.
2. Sprawdź, czy zdania wyświetlają się w oddzielnych wierszach oraz czy w pasku tytułu okna pojawił się napis *Podróże*.
3. Wyświetl kod źródłowy strony.



Rys. 1. Przykład realizacji ćwiczeń 2. i 3. – strona w przeglądarce i jej kod źródłowy w oknie podglądu kodu źródłowego

4.1. Formatowanie tekstu

Tekst strony WWW – podobnie jak tekst w redagowanych dokumentach – możemy formatować. Tekst, który wyświetla przeglądarka, znajduje się w jednym akapicie, mimo że każde zdanie zapisywaliśmy w oddzielnym wierszu. Przeglądarka zignorowała przejścia do nowego akapitu, czyli naciśnięcia klawisza **Enter**. Aby tekst zawierał akapity, należy skorzystać z odpowiednich znaczników. Podobnie jest w przypadku zmiany parametrów czcionki, np. kroju, rozmiaru. Jeśli chcemy wyróżnić pewien tekst, np. pogrubić tytuł „Podróże po stolicach Europy", musimy zastosować znaczniki formatowania.

Akapity tworzymy, otaczając tekst znacznikami i .

Na przykład:
tekst_akapitu1
tekst_akapitu2

Warto zauważyć, że pomiędzy akapitami przeglądarka wyświetla pustą linię odstępu. Aby w ramach akapitu umieścić dwa napisy, jeden bezpośrednio pod drugim, należy skorzystać ze znacznika <br>. Znacznik <br> odpowiada ręcznemu podziałowi wiersza za pomocą klawiszy **Shift** + **Enter** w edytorze tekstu.

Na przykład:

```
<p>
    linia1<br>
    linia2
</p>
```

Znaczniki mogą występować z **atrybutami**, określającymi szczegółowo działanie danego znacznika. Aby wyśrodkować tekst „Warszawa jest stolicą Polski", należy ustawić atrybut `align` znacznika `<p>` na wartość `center`:

`<p align="center">`*Warszawa jest stolicą Polski*`</p>`.

Do formatowania tekstu można stosować m.in. znaczniki:

`<h1> ... </h1>` – nagłówek pierwszego poziomu,
`<h2> ... </h2>` – nagłówek drugiego poziomu,
...
`<h6> ... </h6>` – nagłówek szóstego poziomu,
`<b> ... </b>` – tekst pogrubiony,
`<i> ... </i>` – tekst pochylony (kursywa),
`<u> ... </u>` – tekst podkreślony,
`<s> ... </s>` – tekst przekreślony,
`<sup> ... </sup>` – indeks górny,
`<sub> ... </sub>` – indeks dolny.

Znaczniki można umieszczać wewnątrz innych znaczników. Ważne jest jednak, aby przed zamknięciem danego znacznika zamknąć wszystkie znaczniki otwarte wewnątrz niego.

Na przykład:
`<p><b>`*tekst*`</b></p>` – poprawnie,
`<p><b>`*tekst*`</p></b>` – niepoprawnie.

Uwaga

Nie wszystkie znaczniki mają znacznik zamykający, nie ma go np. znacznik `<br>`.

Dobra rada

Nie trzeba za każdym razem na nowo tworzyć podstawowej struktury strony. Wystarczy skopiować zawartość z poprzedniego pliku źródłowego i wkleić do nowego dokumentu. Potem trzeba tylko wymienić treść i zmodyfikować niektóre znaczniki.

Ćwiczenie 4. Formatujemy tekst na stronie WWW

1. W kodzie źródłowym strony WWW utworzonej w ćwiczeniu 2.:
   - sformatuj tekst „Podróże po stolicach Europy" jako nagłówek pierwszego poziomu – zastosuj znaczniki `<h1>`*tekst*`</h1>`,
   - rozdziel zdania opisujące zawartość strony znacznikami przejścia do nowego wiersza.
2. Zapisz plik pod tą samą nazwą.
3. Obejrzyj stronę w przeglądarce.

Wskazówka: Aby zobaczyć poprawioną stronę, jeśli masz otwarte okno przeglądarki ze swoją stroną, kliknij przycisk **Odśwież**.

4.2. Wypunktowania i tabele

W tekście strony WWW można stosować numerowania i wypunktowania. W przypadku języka HTML mówimy o:

- listach uporządkowanych (odpowiednik numerowania) – znaczniki `<ol>` ... `</ol>`,
- listach nieuporządkowanych (odpowiednik wypunktowania) – znaczniki `<ul>` ... `</ul>`.

  Kolejne elementy listy umieszczamy pomiędzy znacznikami `<li>` ... `</li>`.



Ćwiczenie 5. Wprowadzamy wypunktowania

Dodaj wypunktowanie do czterech pierwszych akapitów opisujących stronę.
Zapisz plik pod tą samą nazwą. Obejrzyj zaktualizowaną stronę w przeglądarce.



Tabele tworzymy za pomocą znacznika `<table>`. Poszczególne wiersze tabeli oznacza się znacznikiem `<tr>`, a komórki – znacznikiem `<td>`. Każdy z tych znaczników występuje w parze ze znacznikiem zamykającym.

Na przykład:

```
<table>
        <tr>
            <td>wiersz1, komórka1</td><td>wiersz1, komórka2</td>
        </tr>
        <tr>
            <td>wiersz2, komórka1</td><td>wiersz2, komórka2</td>
        </tr>
</table>
```

Aby było widać obramowanie tabeli, należy w znaczniku `<table>` zastosować atrybut `border="1"` (rys. 3.): `<table border="1">`.

Gdy wartość atrybutu `border` będzie równa 0 lub atrybut nie wystąpi, obramowanie będzie niewidoczne.



Ćwiczenie 6. Tworzymy tabelę

Przedstaw w tabeli wykaz państw i ich stolic, o których informacje umieścisz na tworzonej stronie.

Wskazówka: Aby było widać obramowanie tabeli, ustaw wartość atrybutu: `border="1"`.

4.3. Hiperłącza do innych stron

Istota poruszania się po stronach WWW polega na tym, że z każdej z nich można w prosty sposób przenieść się na inną (w obrębie tej samej bądź innej witryny). Służą do tego **hiperłącza**, inaczej **odsyłacze** lub, żargonowo, **linki** (z ang. *link* – łącze, połączenie). Hiperłącze to adres strony lub pliku ukryty pod wyróżnionym (zwykle podkreślonym i oznaczonym innym kolorem) elementem, np. tekstem. Gdy klikniemy ten element, hiperłącze przenosi nas na inną stronę.

Hiperłącze tworzymy, stosując znaczniki `<a></a>` z atrybutem `href`. Hiperłącze może mieć postać `<a href="adres_strony">tekst</a>`.

! Uwaga

Wartości atrybutów niebędące liczbami należy umieszczać w cudzysłowach. Kolejność występowania atrybutów znacznika nie jest istotna.

Na przykład wpisanie w dokumencie HTML kodu:

```
Na stronie o <a href="http://www.paryz.pl/">Paryżu</a> można znaleźć wiele informacji na temat stolicy Francji.
```

spowoduje wyświetlenie w przeglądarce zdania:

„Na stronie o Paryżu można znaleźć wiele informacji na temat stolicy Francji". Kliknięcie podkreślonego wyrazu otworzy stronę o Paryżu.

Ćwiczenie 7. Dodajemy hiperłącza prowadzące do innych stron

Dodaj do tworzonej przez siebie strony odsyłacz do strony o stolicy Polski (rys. 2. i 3.). Zapisz plik pod tą samą nazwą. Obejrzyj zaktualizowaną stronę w przeglądarce.

4.4. Wstawianie grafiki

Strony WWW wzbogaca się zazwyczaj rysunkami i zdjęciami. Odpowiednio dobrana grafika sprawia, że strona jest bardziej atrakcyjna. Aby zapewnić szybkie ładowanie strony, należy stosować odpowiednie formaty plików graficznych, np. JPG, PNG (nie należy korzystać z plików w formacie BMP). Aby uniknąć problemów z umieszczeniem strony w Internecie, w nazwach plików nie powinno się używać polskich liter i spacji.

Aby wstawić obraz na stronę, należy zastosować znacznik `<img>` z atrybutem `src`. Ogólna postać znacznika jest następująca: `<img src="plik_graficzny">`, gdzie *plik_graficzny* określa nazwę pliku z grafiką (wraz z rozszerzeniem nazwy pliku).

Na przykład, gdy wprowadzimy w dokumencie HTML znacznik `<img src="rysunek.jpg">`, na stronie WWW wyświetli się rysunek zapisany w pliku *rysunek.jpg* i umieszczony w tym samym folderze co plik HTML. Rysunek pojawi się w miejscu, gdzie umieściliśmy znacznik. Jeśli chcemy umieścić rysunek pod tekstem, trzeba pamiętać o znaczniku `<br>`.

Ćwiczenie 8. Wstawiamy zdjęcie na stronę WWW

1. Odszukaj we własnych zbiorach zdjęcie wieży Eiffla lub ją narysuj i zapisz rysunek w pliku. Wstaw obraz na stronę, umieszczając go po prawej stronie wypunktowanego tekstu.
2. Zapisz plik pod tą samą nazwą i obejrzyj stronę w przeglądarce.

Wskazówka: Aby umieścić zdjęcie po prawej stronie tekstu, należy do znacznika `<img>` dodać atrybut `align` z wartością `"right"`, a w kodzie HTML przed danym tekstem umieścić zapis: `<img src="Francja-Paryz-Wieza.jpg" align="right">` (rys. 2. i 3.).

Podróże

Plik | D:/Dysk%20C/Podręcznik_TB8(3...

Podróże po stolicach Europy

Na stronie znajdziesz informacje o stolicach europejskich:

- poznasz najważniejsze zabytki i obiekty tych miast,
- zobaczysz ciekawe miejsca na fotografiach,
- poznasz ciekawostki historyczne tego miasta.

Więcej informacji o stolicy Polski: strona internetowa Warszawy.

Państwo	Stolica
Austria	Wiedeń
Dania	Kopenhaga
Francja	Paryż
Grecja	Ateny
Polska	Warszawa



Rys. 2. Przykład realizacji ćwiczeń 6., 7. i 8.

Zawijanie wierszy

```
<!DOCTYPE html>

<html>
    <head>
        <meta charset="utf-8">
        <title>Podróże</title>
    </head>

    <body>
    <h1>Podróże po stolicach Europy</h1>
    <img src="Austria-Wieden-Strauss.jpg" align = "right">
    <p>Na stronie znajdziesz informacje o stolicach europejskich:</p>
    <ul>
        <li>poznasz najważniejsze zabytki i obiekty tych miast,</li>
        <li>zobaczysz ciekawe miejsca na fotografiach,</li>
        <li>poznasz ciekawostki historyczne tego miasta.</li>
    </ul>
    <p>Więcej informacji o stolicy Polski: strona internetowa <a href="http://www.warszawa.pl/">Warszawy</a>.</p>
    <table border=1>
        <tr>
            <td><b>Państwo</b></td><td><b>Stolica</b></td>
        </tr>
        <tr>
            <td>Austria</td><td>Wiedeń</td>
        </tr>
        <tr>
            <td>Dania</td><td>Kopenhaga</td></tr>
        <tr>
            <td>Francja</td><td>Paryż</td>
        </tr>
        <tr>
            <td>Grecja</td><td>Ateny</td>
        </tr>
        <tr>
            <td>Polska</td><td>Warszawa</a></td>
        </tr>
    </table>
    </body>

</html>
```

Rys. 3. Źródło strony – realizacja ćwiczeń 6., 7. i 8.

Funkcje hiperłącza może pełnić nie tylko tekst, ale także obraz. Wewnątrz znaczników hiperłącza wystarczy wstawić znacznik obrazu. Można przygotować własny obraz (ikonę) i umieścić go na stronie w odpowiednich znacznikach. Na przykład umieszczenie zapisu: `<a href="adres_strony_WWW"><img src="rysunek.jpg"></a>` spowoduje, że kliknięcie na stronie rysunku (zapisanego w pliku *rysunek.jpg*) otworzy stronę, której adres podano jako wartość atrybutu `href`.



Warto zapamiętać

- Witryna jest zbiorem powiązanych ze sobą hiperłączami stron WWW, umieszczonych pod tym samym adresem internetowym.
- Strona internetowa powinna być: przejrzysta, czytelna, atrakcyjna graficznie (m.in. nieprzeładowana elementami graficznymi, z odpowiednio dobraną kolorystyką tła i tekstu oraz innych elementów) oraz poprawna pod względem merytorycznym i redakcyjnym.
- Strony można tworzyć w edytorach tekstu lub w specjalnych edytorach HTML.
- Standardowym sposobem opisu strony WWW jest język znaczników HTML.
- Korzystając ze znaczników, możemy formatować tekst na stronie, wstawiać obrazy, hiperłącza do innych stron, tabele.



Pytania i polecenia

1. Jakie narzędzie wybierzesz, aby utworzyć prostą stronę internetową? Uzasadnij odpowiedź.
2. Jakich podstawowych zasad powinno się przestrzegać podczas projektowania stron WWW? Na przykład, dlaczego na stronie nie powinno się umieszczać zbyt wielu elementów graficznych?
3. Czym są znaczniki języka HTML?
4. Jakie są zasady stosowania znaczników języka HTML?
5. Przedstaw podstawowe kroki tworzenia strony WWW za pomocą prostego edytora plików tekstowych, np. Notatnika.
6. W jaki sposób wstawia się na stronę grafikę?
7. Do czego służy znacznik `<tr>`, a do czego znacznik `<td>`?
8. Jak można utworzyć hiperłącze do innej strony WWW?
9. Czy zapis `<p><i>Wrocław<p>` jest poprawny? Uzasadnij odpowiedź.



Zadania

1. Przygotuj prostą stronę internetową o dyscyplinie sportu, jaką jest pływanie.
2. Dopisz do strony utworzonej w zadaniu 1. tekst: „Więcej informacji na temat zawodów pływackich możesz znaleźć na stronie Polskiego Związku Pływackiego". Na określenie „Polskiego Związku Pływackiego" wstaw hiperłącze do odpowiedniej strony. Znajdź tę stronę, stosując poznane metody wyszukiwania informacji w Internecie.
3. Umieść na swojej stronie dobrą radę dla innych użytkowników: „Uważaj na wirusy komputerowe. Jeśli chcesz wiedzieć, jak sobie z nimi radzić, kliknij na obrazek". Przygotuj w edytorze grafiki mały rysunek humorystycznie przedstawiający wirusa i umieść go na stronie. Następnie wstaw odpowiedni znacznik, tak aby kliknięcie rysunku przenosiło nas do witryny, na której znajdują się informacje o wirusach komputerowych.

Dla zainteresowanych

4. Rozbuduj stronę o podróżach po stolicach Europy, m.in.:
   a. W tabeli utworzonej w ćwiczeniu 6. do nazw stolic państw europejskich dodaj hiperłącza prowadzące do stron z opisem tych miast.
   b. Dobierz odpowiednio kolory czcionek. Dodaj tło.
   c. Utwórz dodatkowe strony z opisami stolic. Informacje o miastach znajdź w Internecie lub innych źródłach.

   Wskazówka: W dodatkowych źródłach lub Internecie znajdź informacje na temat ustalania kolorów czcionek.

Przeczytaj, jeśli chcesz wiedzieć więcej…

Oprócz języka HTML do tworzenia nowoczesnych stron internetowych wykorzystuje się inne technologie:

- CSS – do opisywania wyglądu strony internetowej;
- JavaScript – do programowania interakcji pomiędzy użytkownikiem a stroną internetową;
- języki skryptowe po stronie serwera (np. PHP, ASP) – do programowania, przetwarzania i zapisywania danych na serwerze;
- bazy danych (np. MySQL) – do przechowywania danych na serwerze.



Temat 16. Systemy zarządzania treścią – projekt

1. Czym są blogi?
2. Czym są systemy zarządzania treścią?
3. Przykładowy CMS – WordPress
   3.1. Tworzenie blogu w systemie WordPress
   3.2. Inne możliwości systemu WordPress
4. Projekty grupowe
   4.1. Dookoła świata z Internetem
   4.2. Strona internetowa o naszej miejscowości



Warto powtórzyć

1. Za pomocą jakich narzędzi można tworzyć strony internetowe?
2. Czym charakteryzuje się język znaczników HTML?
3. Czym jest witryna internetowa?
4. Jakie są zasady przygotowania projektu grupowego, w tym korzystania z cudzych materiałów?
5. Jakie są etapy pracy nad projektem grupowym?
6. Czym jest chmura? Jakie korzyści płyną z używania chmury?
7. W jaki sposób udostępnia się pliki umieszczone w chmurze innym osobom?

1. Czym są blogi?

Blog możemy założyć w prosty sposób, korzystając z serwisów WWW oraz portali internetowych.

Blog może mieć ściśle określony temat (np. gotowanie, zdrowe odżywianie, podróże, moda, sport, polityka) lub zawierać wpisy o ogólnej tematyce. Na blogu umieszcza się uwagi, komentarze, rysunki, nagrania (w tym wideo – tzw. **wideoblog**).

Blogi mogą mieć osobisty charakter i zawierać przemyślenia przedstawiające światopogląd osoby tworzącej wpisy. Tego rodzaju blogi mają zazwyczaj jednego autora. W innych przypadkach blog może mieć wielu autorów.

B **Blog**

Rodzaj strony internetowej, na której umieszcza się opatrzone datami notatki (zwane wpisami).

Ćwiczenie 1. Szukamy blogów tematycznych

Planujesz dziesięciodniową wycieczkę po Europie. Chcesz uzyskać informacje bezpośrednio od osób, które odbyły taką podróż.

1. Odszukaj w Internecie przykładowe blogi o tej tematyce.
2. Sprawdź informacje na temat trasy, jaką można wybrać: który kraj (kraje) odwiedzić, czym podróżować, jakie miejsca zwiedzić.
3. Oceń, który blog okazał się najbardziej przydatny.

Istnieje wiele serwisów umożliwiających darmowe prowadzenie blogów. Założenie własnego bloga wymaga podania kilku danych, w szczególności: nazwy bloga (będącej potem częścią jego adresu), adresu e-mail właściciela i hasła dostępowego.

Po potwierdzeniu rejestracji w serwisie możemy wybrać wygląd blogu, korzystając z dostępnych szablonów. Potem pozostaje już tylko wypełnić blog treścią.

Blog możemy założyć, korzystając m.in. z systemu WordPress.

2. Czym są systemy zarządzania treścią?

Chcemy utworzyć blog na temat nauki programowania. Czy język znaczników HTML jest jedynym narzędziem do przygotowywania stron WWW? Jakie oprogramowanie można wykorzystać do tego celu?

Uwaga

Aby korzystać z systemu Wordpress (lub innego systemu CMS), należy pobrać aplikację z Internetu i zainstalować ją na serwerze WWW (np. szkolnym). Można również skorzystać z wybranego bezpłatnego serwera, który umożliwia pracę z aplikacją.

Systemy zarządzania treścią (**CMS**) to oprogramowanie do tworzenia stron internetowych, które może być z powodzeniem wykorzystywane przez osoby niemające szczegółowej wiedzy dotyczącej technologii internetowych. Systemy tego typu pozwalają na utworzenie witryny internetowej i zarządzanie nią za pomocą przeglądarki internetowej, bez konieczności poznawania języka HTML.

Wygląd serwisu definiujemy za pomocą szablonów (inaczej zwanych motywami), co ułatwia dokonywanie szybkich zmian szaty graficznej – wystarczy wybrać inny szablon. Systemy zarządzania treścią mogą być komercyjne, ale w Internecie są również dostępne bezpłatne programy, np. **Joomla!**, **Drupal**, **WordPress** – rozpowszechniane na licencji GNU GPL.

Niektóre programy są przeznaczone do konkretnych celów, np. MediaWiki – do tworzenia encyklopedii, a WordPress – do obsługi blogów.

System WordPress umożliwia tworzenie rozbudowanych blogów, ale również innych stron internetowych, m.in. sklepu internetowego, forum czy portfolio (stron, na których prezentuje się własne prace, np. obrazy).

W ramach systemu CMS (np. WordPress) jedną witrynę może tworzyć wiele osób. Każda osoba może mieć przydzielone inne uprawnienia – najwyższe ma **administrator**, który może zmieniać wygląd witryny i uprawnienia użytkowników.

Na przykład niektórzy użytkownicy mogą tworzyć treści wpisów, ale nie mogą ich publikować, inna osoba – tzw. redaktor – czyta wszystkie wpisy (ewentualnie poprawia) i je publikuje.

Informacje (artykuły) wprowadzane do systemu i edytowane przez autora strony (lub osobę uprawnioną do redakcji tekstów, w przypadku większej liczby osób) zapisują się w bazie danych. Na podstawie szablonu oraz treści pochodzących z bazy danych system dynamicznie generuje stronę internetową w języku HTML, w chwili gdy odwiedza ją użytkownik.

3. Przykładowy CMS – WordPress

3.1. Tworzenie blogu w systemie WordPress

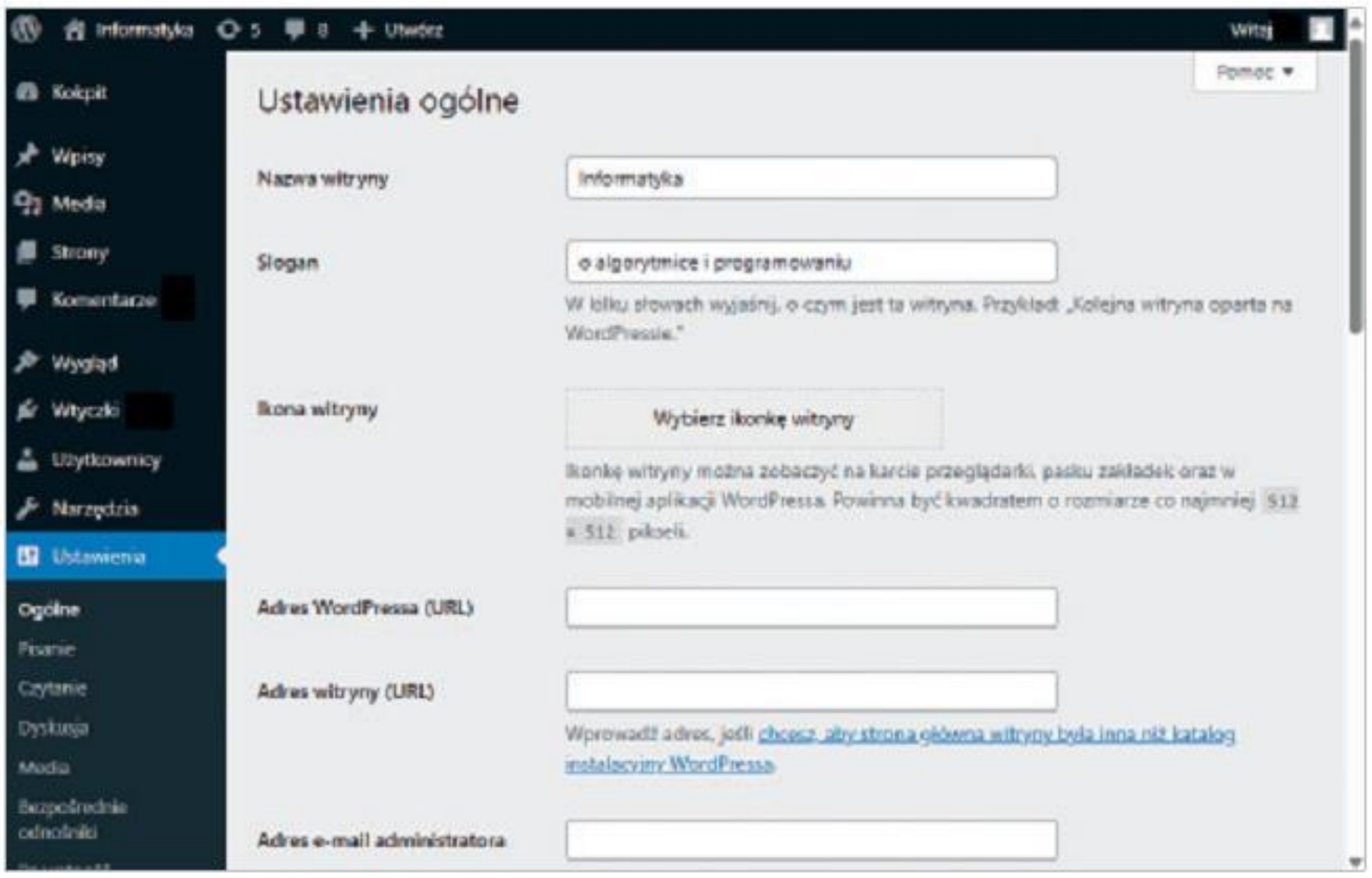
Podstawowe możliwości systemów zarządzania treścią pokażemy na przykładzie systemu WordPress. Po zainstalowaniu systemu WordPress możemy rozpocząć pracę i zacząć tworzyć blog. Korzystamy w tym celu z wybranej przeglądarki internetowej.



Ćwiczenie 2. Zmieniamy tytuł i opis strony

Zastanów się nad tematyką strony (blogu) i ustal tytuł odpowiadający planowanej treści. Zmień odpowiednio tytuł i opis strony (ustawione domyślnie przez program). Zapisz zmiany.

Wskazówka: Aby zmienić tytuł i opis strony, należy wybrać opcję **Ustawienia/Ogólne** (rys. 1.).



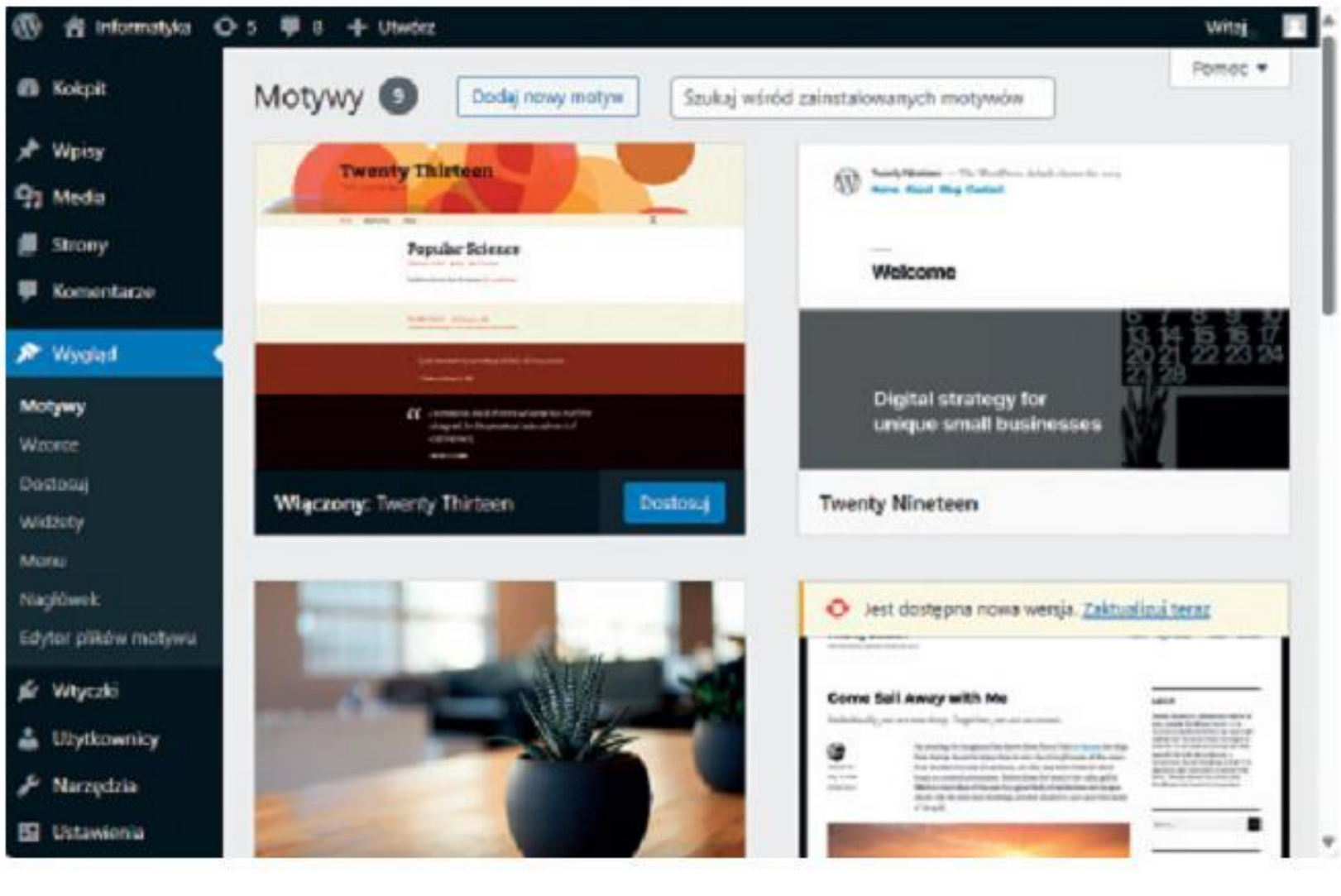
Rys. 1. Zmienianie tytułu i opisu strony – ćwiczenie 2.

Ćwiczenie 3. Przeglądamy dostępne motywy

Przejrzyj motywy dostępne w systemie WordPress i skorzystaj z możliwości zmiany motywu strony (opcja **Wygląd/Motywy**). WordPress proponuje szablon domyślny. Możesz go pozostawić lub wybrać inny.

Wskazówki:

- Aby zmienić motyw strony, należy wybrać opcję **Wygląd/Motywy** (rys. 2.).
- Aby zmieniać wygląd witryny, musisz mieć uprawnienia administratora.



Rys. 2. Określenie wyglądu (wybór motywu w systemie WordPress) – ćwiczenie 3.

Tag

Słowo kluczowe przypisane do danego wpisu i określające tematykę wpisu. Do jednego wpisu można dodać więcej niż jeden tag.

Dobra rada

Można już na początku pracy nad blogiem utworzyć kilka kategorii, aby od razu porządkować pierwsze wpisy. Potem, gdy blog będzie się rozrastał, można dodawać kolejne kategorie. Tagi najlepiej dodawać podczas edycji wpisu.

Wpisy na blogu powinny być w określony sposób uporządkowane, co znacznie ułatwia wyszukiwanie informacji i zarządzanie nimi.

Wpisy możemy przydzielać do grup tematycznych, zwanych **kategoriami**. Nazwa kategorii powinna odnosić się do treści danego wpisu. Na przykład, jeśli blog dotyczy programowania, możemy stworzyć kategorie: *Programowanie*, *Algorytmika*; jeśli blog dotyczy miejscowości, w której mieszkamy – *Zabytki*, *Turystyka*, *Historia*.

Wyszukiwanie wpisów dotyczących konkretnego zagadnienia ułatwia również dodawanie tzw. **tagów** (etykietek) – np. opatrzenie tagiem „algorytm" kilku wpisów pozwoli na późniejsze wyszukanie i wyświetlenie tych wpisów przez kliknięcie na tag.

Aby dodać nową kategorię lub tag, należy skorzystać odpowiednio z opcji **Kategorie** lub **Tagi** w menu **Wpisy**.

Po zatwierdzeniu nowej nazwy kategorii lub tagu można ją przypisywać do nowych lub istniejących wpisów. Kategorie lub tagi można również dodawać podczas tworzenia lub modyfikowania wpisu.

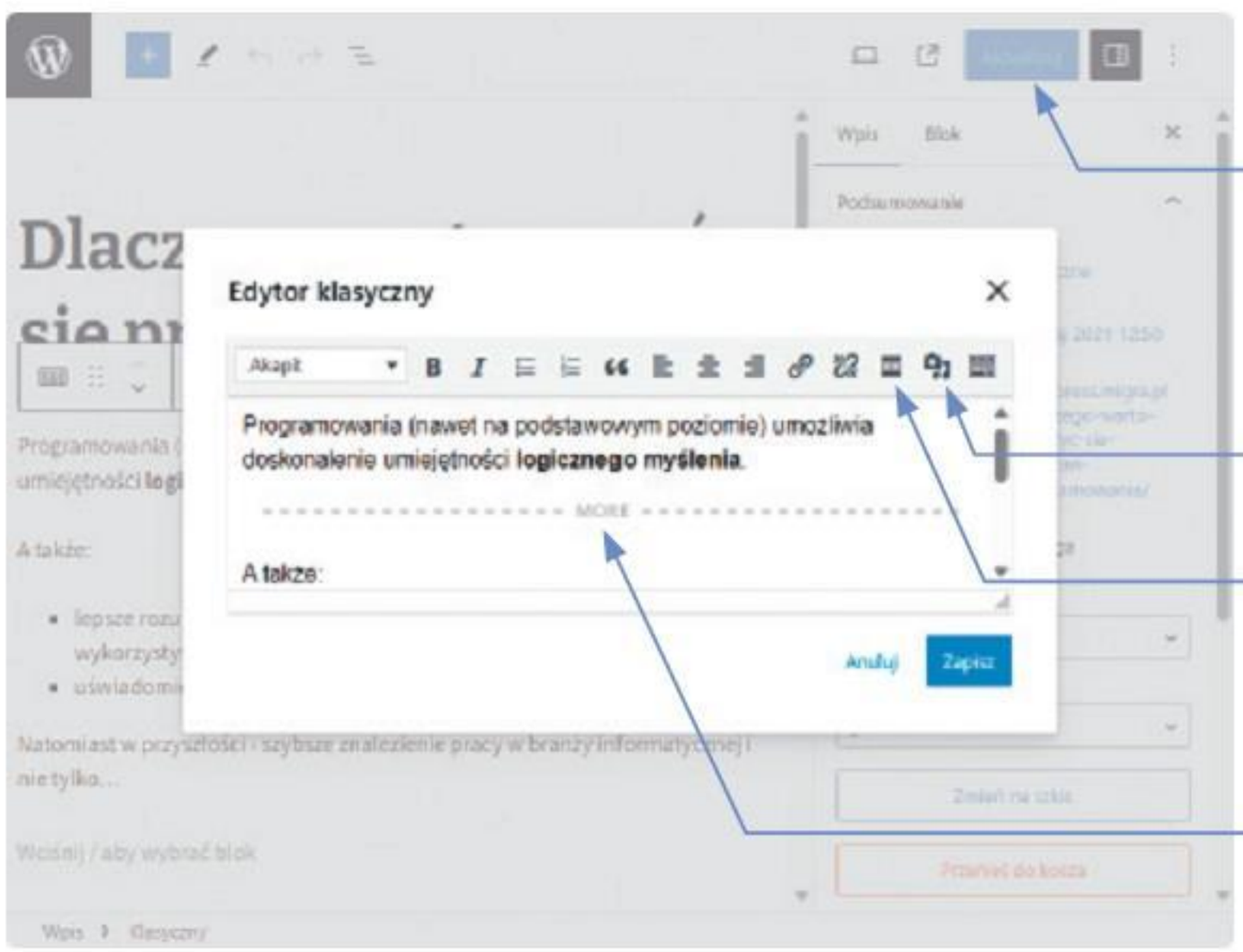
Ćwiczenie 4. Dodajemy kategorie i tagi

Utwórz trzy kategorie odpowiadające tematyce tworzonego blogu. Zapisz zmiany.

Aby dodać nowy wpis, należy wybrać opcje menu **Wpisy/Dodaj nowy**.

Należy podać tytuł wpisu i jego treść. Po napisaniu tekstu warto podejrzeć wygląd wpisu na stronie, a dopiero potem go opublikować. Jeśli chcemy zaktualizować wpis, należy kliknąć jego tytuł. Otworzy się strona, na której możemy edytować wpis (rys. 3.), przyporządkować go do konkretnej kategorii, przydzielić wpisowi tagi, utworzyć nowe kategorie i tagi oraz opublikować zaktualizowany wpis.

Jeśli wpis jest długi i zajmuje dużo miejsca na stronie głównej, można podzielić go na dwie części (wprowadzenie i rozwinięcie). W tym celu dodajemy odpowiedni znacznik do tekstu, tak aby przed znacznikiem wyświetlała się tylko pierwsza część wpisu, a za znacznikiem umieszczamy część tekstu, która ma być początkowo niewidoczna (rys. 3.). Gdy opublikujemy wpis, w miejscu wstawienia znacznika pojawi się link **Czytaj dalej** (rys. 4.). Po jego kliknięciu zobaczymy pozostałą część wpisu.



Rys. 3. Strona edytowania wpisu – ćwiczenia 4. i 5.

Aby podzielić wpis, należy wstawić do tekstu znacznik **Czytaj dalej**: .

Ćwiczenie 5. Dodajemy nowy wpis

1. Dodaj wpis składający się z trzech zdań (pamiętaj o wpisaniu tytułu). Na stronie głównej ma się wyświetlać tylko jedno zdanie, a pozostałe powinny się pokazać po naciśnięciu linku **Czytaj dalej**.
2. Wpis dodaj do jednej z kategorii. Dodaj nowy tag i przyporządkuj go do tego wpisu. Podejrzyj wpis – jeśli wszystko wyświetla się poprawnie, opublikuj go. Obejrzyj wpis na stronie głównej.

Widżet W
Element graficznego interfejsu użytkownika (np. okno, pole edycji, suwak, przycisk).

Po wybraniu opcji **Wygląd/Widgety** będziemy mogli zapoznać się z dostępnymi widżetami i zdecydować, które z nich mają się wyświetlać, np. w panelu bocznym czy w stopce strony. W tym celu należy umieścić wybrany widżet w tzw. **strefie widżetów**. Zależnie od wybranego szablonu mamy do wyboru różne widżety.

Dodane kategorie i tagi mogą się wyświetlać na stronie głównej. Na rysunku 4. widać je w panelu po prawej stronie okna.

Ćwiczenie 6. Modyfikujemy wygląd strony

Zmodyfikuj wygląd strony tak, aby w panelu bocznym po prawej stronie było widać archiwa, kategorie, tagi. Dodaj też kalendarz i sprawdź, jakie daty w nim wyróżniono. Zapisz zmiany.

Do wpisu na blogu możemy dodać **media** (np. obraz, film – zapisane w pliku), klikając przycisk **Dodaj medium** na stronie edytowania wpisu (rys. 3.).

Ćwiczenie 7. Dodajemy do wpisu obraz

Dodaj do wybranego wpisu zdjęcie lub wcześniej przygotowany rysunek zapisany w pliku. Sprawdź, czy wstawiony obraz znalazł się w bibliotece mediów (opcja menu **Media**).

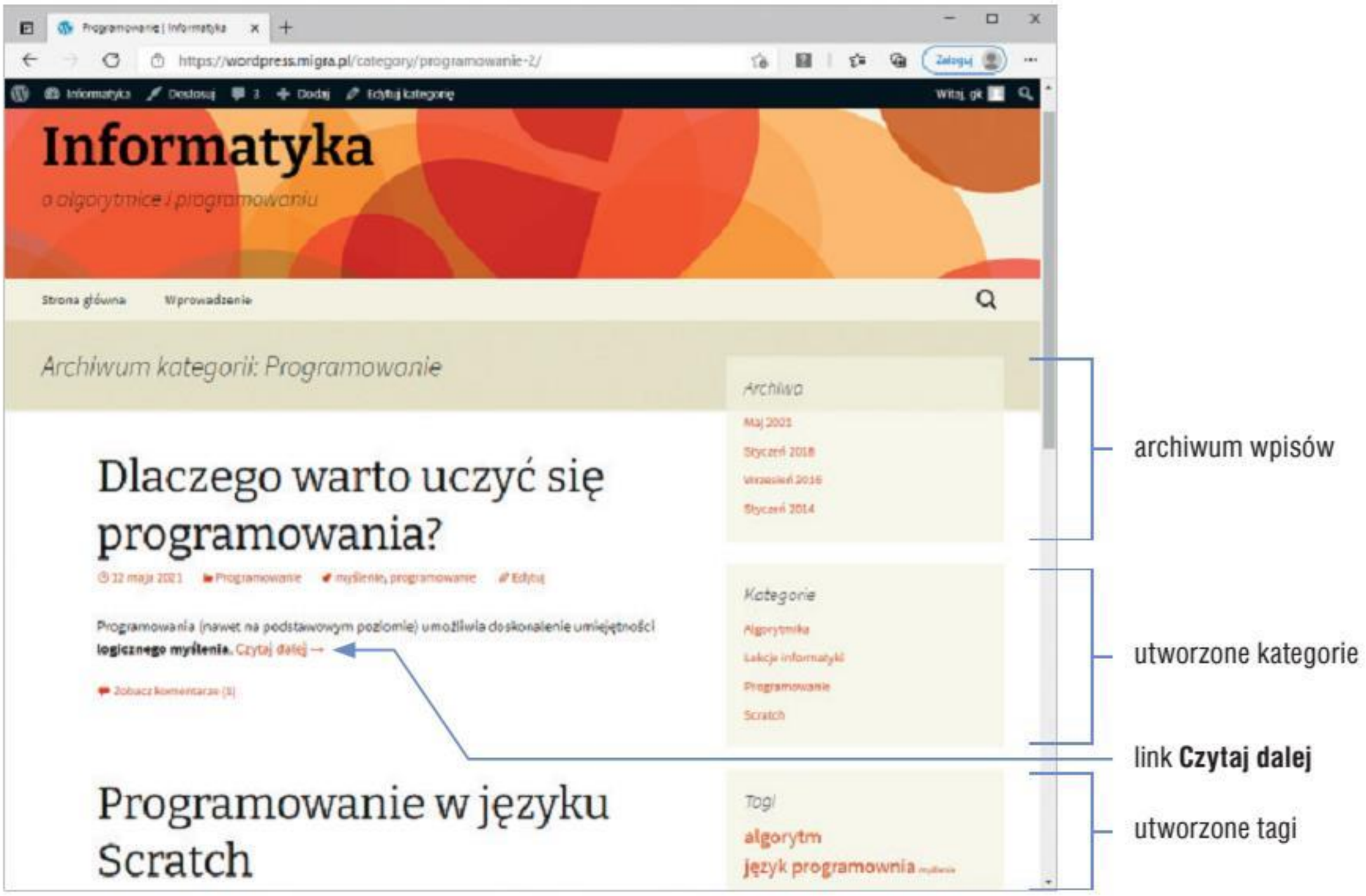
Ćwiczenie 8. Dodajemy nowego użytkownika

Odszukaj opcje umożliwiające dodanie nowego użytkownika. Sprawdź, jakie role można mu przypisać. Dodaj jednego nowego użytkownika (koleżankę lub kolegę z klasy), z którym będziesz wspólnie pracować nad blogiem.

3.2. Inne możliwości systemu WordPress

Poza cyklicznymi wpisami (tworzącymi blog) na witrynie można również umieszczać ogólne informacje (teksty, obrazy), np. dotyczące autora lub tematyki strony. Korzystając z gotowych motywów, możemy zmienić strukturę menu lub wygląd strony.

Po wybraniu opcji dodania nowej strony otworzy się strona umożliwiająca wprowadzenie informacji, które będą umieszczone na stałe w jednym miejscu i pokazane np. w panelu nawigacyjnym. Należy podać tytuł strony, np. „Wprowadzenie”, „O blogu”, „O mnie”, „O nas”.



Rys. 4. Wpis z rysunku 3. widoczny na stronie głównej. Wybrano wpisy z kategorii ***Programowanie***

Aby dodać nową stronę do witryny, należy wybrać opcję **Strony/Dodaj nową**.

Ćwiczenie 9. Dodajemy stronę

Dodaj nową stronę, która będzie informować o tematyce blogu.

Tytuł strony może się pojawić w menu na stronie głównej. Wygląd i strukturę menu możemy zmienić, dodając – według uznania – odpowiednie podmenu. Poza utworzeniem nowej strony możemy również dodać do menu link do innej witryny.

Do tworzenia menu oraz wprowadzania zmian w jego strukturze wykorzystujemy opcję **Wygląd/Menu** oraz odpowiednie opcje z zakładek **Edytuj menu** i **Zarządzaj rozmieszczeniem.**

Ćwiczenie 10. Zmieniamy strukturę menu

Zapoznaj się z możliwościami zarządzania menu. Do menu, które widać pod nagłówkiem strony głównej, dodaj tytuł utworzonej strony.

4. Projekty grupowe

Możliwości pracy w chmurze oraz wyszukiwania informacji w Internecie, a także poznane zasady tworzenia strony internetowej wykorzystamy podczas pracy nad projektami.

Zanim przystąpimy do wykonywania zadań projektowych, powinniśmy się zastanowić, jakich narzędzi użyjemy. Na przykład do tworzenia strony internetowej możemy wykorzystać edytor plików tekstowych Notepad++ i język HTML, utworzyć stronę w postaci blogu, albo skorzystać z serwisu Canva, dostępnego bezpłatnie w Internecie (*www.canva.com*). Do zaplanowania podróży możemy wykorzystać wyszukiwarkę internetową, edytor tekstu i arkusz kalkulacyjny.

4.1. Trzy atrakcje w mojej miejscowości

Zadanie projektowe – wykonujemy stronę internetową o naszej miejscowości

Do twojej szkoły przyjeżdżają uczniowie z innej miejscowości. Chcesz zachęcić ich do przyjazdu, przygotowując dla nich stronę internetową, na której pokażesz i krótko opiszesz trzy wybrane atrakcje z twojej miejscowości.

Przykładowy podział zadania na zadania szczegółowe

Zadanie 1. Zaplanuj, jakie informacje oraz elementy graficzne umieścisz na stronie.

Zadanie 2. Wybierz odpowiednie narzędzie do tworzenia stron.

Zadanie 3. Wspólnie z innymi członkami grupy zaprojektuj (np. na tablicy) ogólny wygląd i strukturę strony, m.in. określ, jak rozmieścić tekst i elementy graficzne, jakie informacje znajdą się na stronie lub wybierz gotowy szablon strony WWW proponowany w wybranym narzędziu do tworzenia stron.

Zadanie 4. Ustal parametry formatowania tekstu, ewentualne stosowanie numeracji i wyliczeń, kolorystykę strony (tła, tekstu, elementów graficznych). W przypadku korzystania z szablonu, możesz zaakceptować te proponowane przez szablon lub je zmodyfikować.

Zadanie 5. Zgromadź potrzebne informacje dotyczące kilku atrakcji turystycznych i wybierz z nich trzy, które uznasz za warte polecenia.

Zadanie 6. Zgromadź (lub wykonaj samodzielnie) zdjęcia.

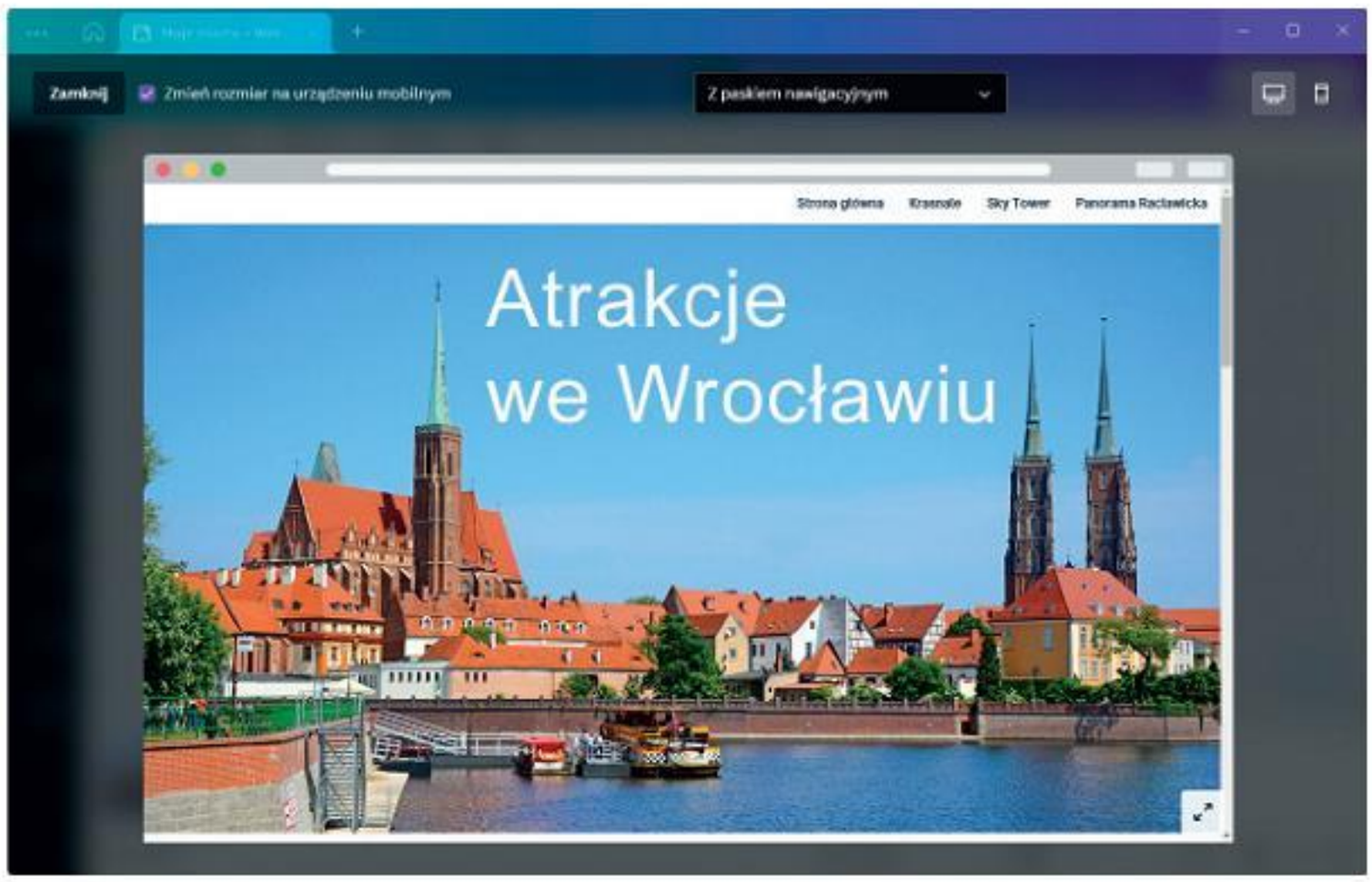
Zadanie 7. Uporządkuj zgromadzone materiały: przeanalizuj je, posegreguj, odrzuć nieprzydatne.

Zadanie 8. Wybierz adresy stron internetowych, do których warto zrobić odwołania.

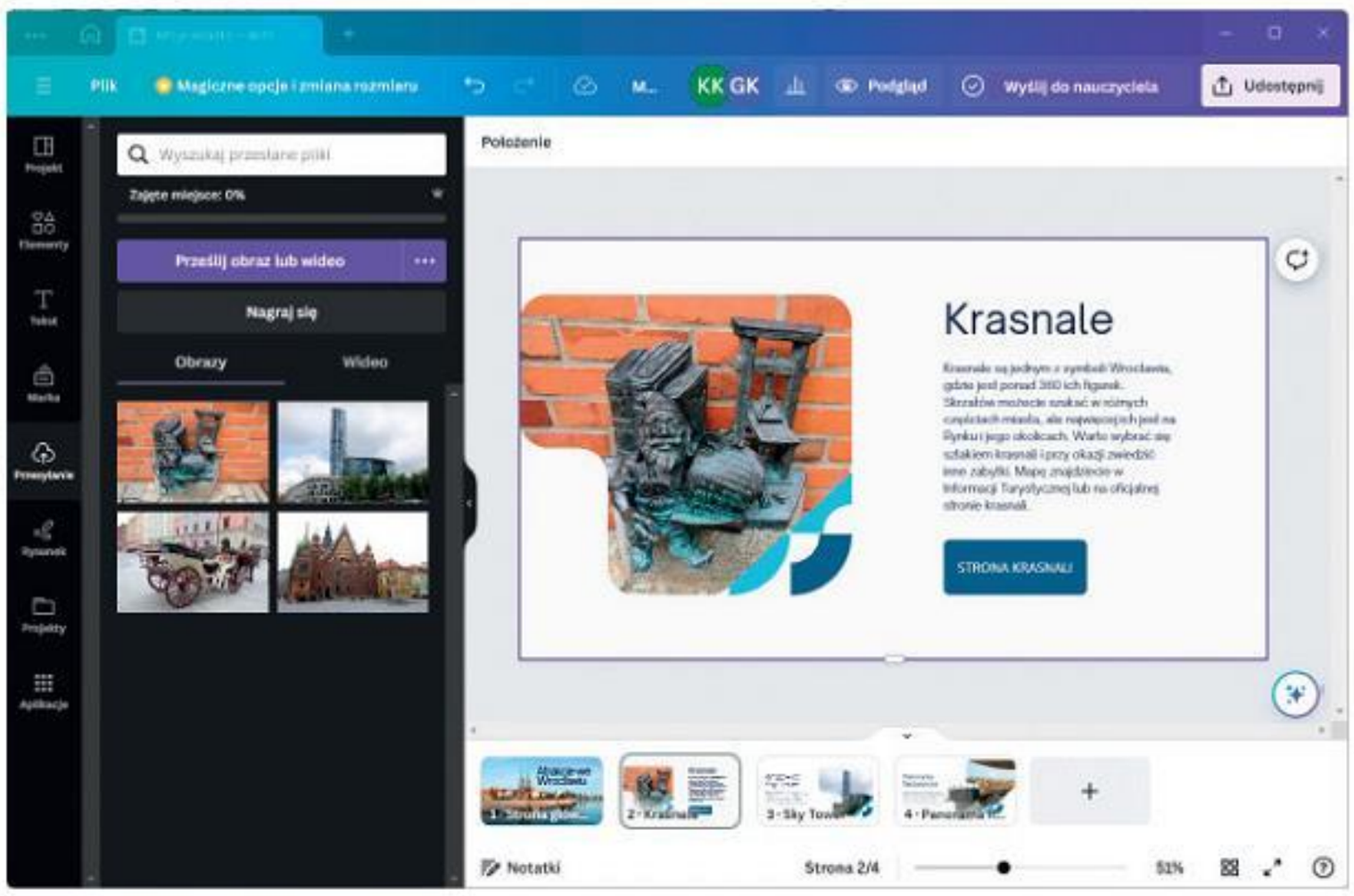
Wskazówki do zadań szczegółowych

1. W Internecie może już istnieć strona poświęcona twojej miejscowości, dlatego warto ją przejrzeć i postarać się umieścić inne, niepodane tam informacje, np. własne zdjęcia, własne spostrzeżenia i wskazówki dla twoich rówieśników.

2. Stronę możesz przygotować w Canvie (temat 10.):
   - na stronie głównej Canvy kliknij link **Strony WWW**,
   - w polu wyszukiwania wpisz np. „miasto” i wybierz szablon, który ci odpowiada,
   - w pasku tytułu zmień nazwę strony, np. „Moje miasto”,
   - pobierz własne zdjęcia ze swojego folderu (w panelu bocznym kliknij przycisk **Przesyłanie**) lub wybierz z darmowych zbiorów Canvy (w polu wyszukiwania wpisz np. „Wrocław”),
   - umieść wybrane zdjęcie na stronie głównej (kliknij je w panelu **Obrazy**) i z menu kontekstowego zdjęcia wybierz polecenie **Ustaw obraz jako tło** (rys. 5.),
   - umieść pozostałe zdjęcia na kolejnych podstronach (rys. 6.),



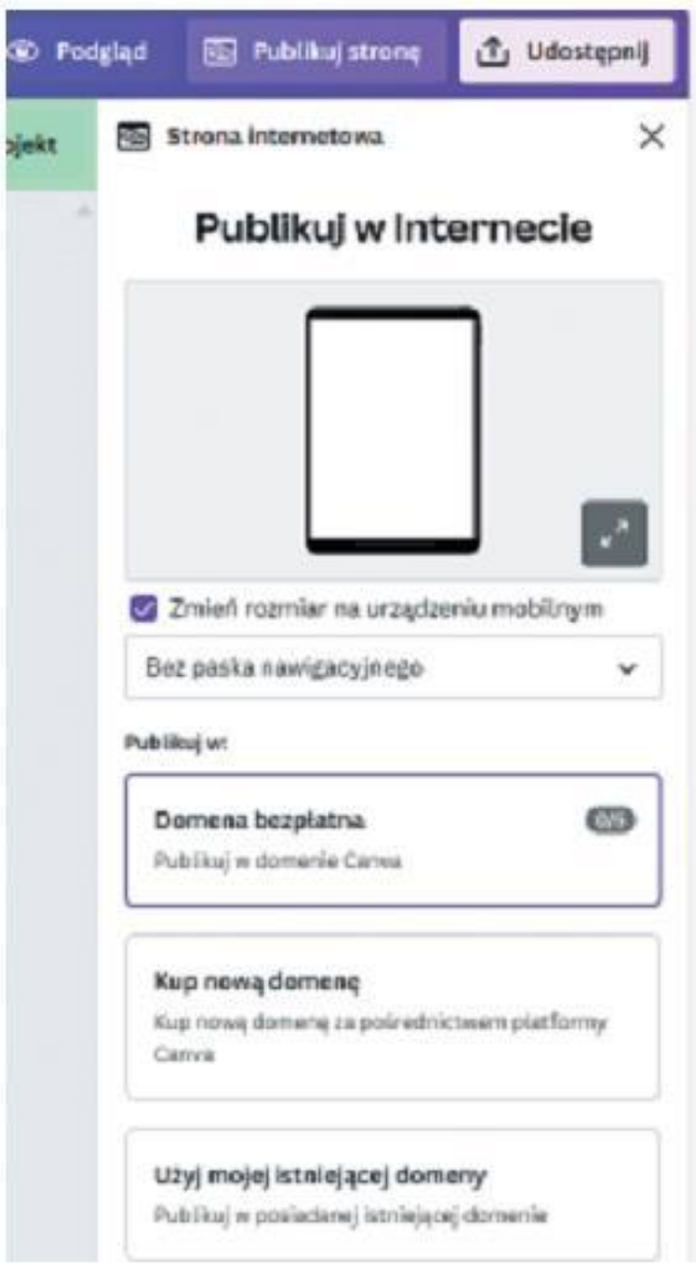
Rys. 5. Przykładowa strona główna projektu „Trzy atrakcje w mojej miejscowości” wykonana w Canvie (widok **Podgląd**)



Rys. 6. Przykładowa podstrona projektu „Trzy atrakcje w mojej miejscowości” wykonana w Canvie



Rys. 7. Podgląd strony WWW na urządzeniu mobilnym



Rys. 8. Możliwości publikowania strony WWW na darmowym koncie użytkownika indywidualnego

- na podstronach dodaj własne opisy poszczególnych atrakcji (rys. 6.),
- dodaj linki do stron internetowych, na których można znaleźć więcej informacji na dany temat (zaznacz tekst, do którego ma być dodane hiperłącze – pojawi się przycisk **Link** ; kliknij go i wklej odpowiedni adres strony WWW),
- dodaj nazwy strony głównej i podstron (kliknij przycisk **Notatki** w dole okna i w polu tekstowym nazwy wpisz nazwę strony, np. „Krasnale" – nazwa pojawi się na stronie głównej jako menu (rys. 5.),
- zobacz stronę w **Podglądzie** (rys. 5.), w tym również w wersji mobilnej (przycisk **Wersja mobilna** – rys. 7.)
- jeśli uznasz, że wszystko jest w porządku, prześlij stronę do swojego nauczyciela (kliknij opcję **Wyślij do nauczyciela**).

Uwaga: W wersji edukacyjnej Canvy możemy stronę przesłać do nauczyciela czy udostępniać na różne sposoby (przycisk **Udostępnianie**), ale nie ma możliwości publikacji. Natomiast, jeśli zalogujemy się jako indywidualny użytkownik, mamy taką możliwość (przycisk **Publikuj stronę** – rys. 8.), ale jest to wersja bardziej ograniczona (ma mniej możliwości niż wersja edukacyjna).

4.2. Dookoła świata z Internetem

Zadanie projektowe – planujemy podróż dookoła świata z Internetem

Przykładowy podział zadania na zadania szczegółowe

Zadanie 1. Zaplanuj przebieg podróży: określ czas i trasę podróży (państwa, miasta).

Zadanie 2. Znajdź miejsca noclegów (schroniska, pola namiotowe, hotele) oraz sprawdź ceny i możliwość rezerwacji, np. przez Internet.

Zadanie 3. Wybierz środki transportu, którymi będziesz się przemieszczać (statek, prom, pociąg, samolot, autobus). Sprawdź ceny biletów oraz dowiedz się, czy istnieje możliwość rezerwacji, np. przez Internet.

Zadanie 4. Odszukaj informacje o krajach, w których będziesz się zatrzymywać, m.in. na temat:

- historii danego kraju,
- klimatu,
- ukształtowania powierzchni,
- struktury gospodarki,
- liczby ludności.

Zadanie 5. Wybierz przynajmniej jeden ważny obiekt lub miejsce warte zwiedzenia (zdjęcie, opis) w danym kraju (mieście) i wyszukaj informacje o nim.

Warto zapamiętać

- Systemy zarządzania treścią (CMS) umożliwiają tworzenie stron internetowych (w tym zakładanie blogów) również osobom bez szczegółowej wiedzy na temat technologii internetowych.
- Wpisy umieszczane na blogach lub w systemach CMS warto grupować w kategoriach określających tematykę wpisu.
- Dodanie tagów ułatwia wyszukiwanie wpisów dotyczących konkretnego zagadnienia.

Pytania i polecenia

1. Czym jest blog?
2. W jakim celu tworzone są blogi? Co jest przyczyną ich popularności?
3. Czym są systemy zarządzania treścią? Do czego można je wykorzystać?
4. Na przykładzie systemu WordPress przedstaw w kilku punktach sposób zakładania blogu.
5. Dlaczego warto przydzielać wpisy do kategorii i dodawać do nich tagi?
6. Jakie są inne możliwości systemu WordPress?
7. Jakie jest twoje zdanie na temat blogu jako środka komunikacji?

Zadania

1. Do tworzonego w tym temacie blogu dodaj trzy nowe wpisy (pamiętaj o wpisaniu tytułu). Na stronie głównej wyświetlaj po dwa zdania, a pozostałe udostępnij po naciśnięciu linku **Czytaj dalej**. Wpisy dodaj do jednej z kategorii lub utwórz nowe kategorie. Dodaj nowe tagi i przyporządkuj je do nowych wpisów. Podejrzyj wpisy – jeśli wszystko wyświetla się poprawnie, opublikuj je. Obejrzyj wpisy na stronie głównej.
2. Odszukaj w Internecie przykładowe blogi o tematyce kulinarnej, żużlowej i dotyczącej programowania. Zwróć uwagę na liczbę blogów dotyczących każdego z tych tematów. Co zauważasz?

Dla zainteresowanych

3. Utwórz blog na temat wyjazdu wakacyjnego (planowanego lub zakończonego). Dodaj zdjęcia i opisy ciekawych miejsc.

Przykłady innych projektów grupowych

1. Podróż dookoła Polski.
2. Nasza szkoła.
3. Znaczenie bakterii w przyrodzie i dla człowieka.
4. Ryby akwariowe.
5. Budowa komórek bakterii, roślin i zwierząt.
6. Krainy geograficzne Polski.
7. Ruch obrotowy i obiegowy Ziemi.
8. Układ planetarny.
9. Rodzaje lasów w Polsce.
10. Walory przyrodnicze Wisły i Odry.
11. Polska Piastów, m.in. Bolesława Chrobrego, Bolesława Krzywoustego, Kazimierza Odnowiciela i Bolesława Śmiałego.
12. Cywilizacje starożytnej Grecji i Rzymu.
13. Osiągnięcia Komisji Edukacji Narodowej.
14. Twórcy klasyki literatury polskiej, m.in. Adam Mickiewicz, Juliusz Słowacki, Henryk Sienkiewicz, Bolesław Prus, Ignacy Krasicki, Jan Brzechwa, Konstanty Ildefons Gałczyński.
15. Ulubiony zespół muzyczny.
16. Mój klub sportowy.



Słownik pojęć

A

Adres mieszany
Adres, w którym występuje odwołanie względne do kolumny i bezwzględne do wiersza lub odwrotnie: odwołanie bezwzględne do kolumny i względne do wiersza (str. 174).

B

Blog
Jest to rodzaj strony internetowej, na której umieszcza się opatrzone datami notatki (zwane wpisami) (str. 221).

I

Interpretacja
Tłumaczenie programu tworzonego w języku programowania instrukcja po instrukcji, tak aby komputer mógł wykonać każdą z nich. Tłumaczenie następuje każdorazowo przy uruchomieniu programu (str. 35, 50).

K

Kompilacja
Przetłumaczenie programu w całości na język zrozumiały dla procesora, tak by mógł go wykonać komputer. Jeśli kompilacja przebiegnie poprawnie, można uruchomić program. Raz skompilowany program nie wymaga już powtórnej operacji tłumaczenia (str. 35, 50).

Korekta
Poprawianie błędów literowych, interpunkcyjnych, ortograficznych (str. 32).

L

Layout
(z ang. – układ)
Graficzny wzorzec publikacji, obejmujący: wielkość strony, szerokość szpalt (kolumn), krój czcionek, rozmieszczenie stałych elementów strony (np. numerów stron), kolorystykę (str. 31).

Linie siatki
Pionowe i poziome linie oddzielające od siebie poszczególne komórki tabeli arkusza kalkulacyjnego (str. 176).

Ł

Łamanie
Wstawienie do odpowiednich kolumn: tekstów, tytułów, zdjęć, rysunków oraz ich rozmieszczenie według makiety (str. 31).

M

Makieta
Wzorzec rozmieszczenia tekstów oraz ilustracji w szpaltach (kolumnach) na stronie (str. 31).

N

Nagłówek i stopka
Odpowiednio górna i dolna część strony dokumentu, których zawartość pojawia się automatycznie na każdej stronie dokumentu (str. 19).

NWD
Największy wspólny dzielnik dwóch lub więcej liczb naturalnych (różnych od zera) to największa liczba naturalna, przez którą dzieli się bez reszty każda z tych liczb (str. 67).

O

Operator
Symbol określający rodzaj operacji wykonywanej przez komputer (str. 73).

P

PDF
(ang. *Portable Document Format* – przenośny format dokumentu)
W formacie PDF można zapisać dowolny dokument (np. dokument tekstowy utworzony w edytorze tekstu, dokument arkusza kalkulacyjnego, prezentację multimedialną). Dokumenty w formacie PDF są wyświetlane i drukowane tak samo niezależnie od użytego komputera i systemu operacyjnego (str. 30).

Połączenie dokumentów
Wstawienie do dokumentu (np. tekstowego) obiektu lub jego fragmentu (np. tabeli arkusza kalkulacyjnego, pliku multimedialnego) pochodzącego z innej aplikacji w ten sposób, by zachować trwałe połączenie między obiektem wstawionym a obiektem źródłowym i jego aplikacją macierzystą. Każdą zmianę danych w pliku źródłowym widać w obiekcie wstawionym (str. 148).

Prezentacja multimedialna
Przedstawienie danego zagadnienia w postaci dokumentu elektronicznego z elementami multimedialnymi, takimi jak: rysunki, zdjęcia, animacje, dźwięki, filmy (str. 144).

S

Sekcja
Część dokumentu, dla której możemy ustalać inne od użytych w pozostałych częściach dokumentu parametry formatowania strony, np. układ pozio-my (str. 23).

Słowo kluczowe
Słowo mające szczególne znaczenie w danym języku programowania, np. oznaczające określony rozkaz, instrukcję lub polecenie (str. 37).

Spacja nierozdzielająca
Stosowana na potrzeby rozmieszczania tekstu w obrębie jednego wiersza. Wyrazy rozdzielone taką spacją są traktowane jak jeden wyraz. (str. 11).

Sztuszna inteligencja
(w skrócie SI, ang. *artificial intelligence* – AI)
W informatyce dziedzina badań naukowych oraz zbiorcza nazwa technologii służących modelowaniu i symulowaniu działania naturalnej (zwierzęcej lub ludzkiej) inteligencji. (str. 152).

T

Tag
Słowo kluczowe przypisane do danego wpisu i określające tematykę wpisu. Do jednego wpisu można dodać więcej niż jeden tag (str. 224).

W

Widżet
Element graficznego interfejsu użytkownika (np. okno, pole edycji, suwak, przycisk) (str. 226).

Z

Znacznik języka HTML
Ciąg znaków zapisany w nawiasach kątowych, np.
`<img src="rysunek.gif">`
(str. 200).

Podręcznik jest przeznaczony do nauczania informatyki w klasie 8 szkoły podstawowej.

Prezentowane w podręczniku metody, pojęcia i przykłady uwzględniają różnorodność oprogramowania, jaka występuje w szkołach podstawowych.

Każdy temat to gotowy scenariusz lekcji o wyrazistej strukturze merytorycznej i metodycznej. Uczeń poznaje zagadnienia w sposób systematyczny i uporządkowany, a rozwiązując niebanalne zadania, rozwija wyobraźnię twórczą i uczy się logicznego myślenia.

Podręcznik zawiera:

- ponad 270 ćwiczeń i zadań sprawdzających
- ponad 110 pytań sprawdzających
- 38 zadań projektowych

