Grażyna Koba

TERAZBAJTY

Informatyka

dla szkół ponadpodstawowych

Zakres podstawowy

Klasa III

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
Marta Skała-Kowalczyk
Michał Łętowski
Marcin Mytych
Piotr Babij

Bolesław Gremblewski

Redakcja i korekta: dr Halina Kubicka

Opracowanie graficzne: Roman Jankowski, Grażyna Koba

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

Cyfrowe odzwierciedlenie podręcznika dopuszczonego do użytku szkolnego przez ministra właściwego do spraw oświaty i wychowania i wpisanego do wykazu podręczników przeznaczonych do kształcenia ogólnego do nauczania informatyki, na podstawie opinii rzeczoznawców: prof. dr hab. Liliany Byczkowskiej-Lipińskiej, dr. hab. Henryka Nogi, dr Iwony Wandy Grygiel.

Zakres kształcenia: podstawowy.
Etap edukacyjny: III.
Typ szkoły: liceum ogólnokształcące i technikum.
Rok dopuszczenia: 2021.

Numer ewidencyjny w wykazie: 1042/3/2021

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

ISBN 978-83-964794-8-8

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

Wydanie I

Fotografie: Krzysztof Fortuna – str. 20 (rys. 1), str. 21 (rys 2. z lewej); SSPL/Getty Images – str. 21 (rys. 2 z prawej); Krzysztof Fortuna – str. 22 (rys. 3.), str. 23 (rys. 4a, 4b, 5a i 5b), str. 24 (rys. 6.); SSPL/Getty Images – str. 25 (rys. 7. i 8.), str. 26 (rys. 9.), str. 27 (rys. 10); Audioklan/Yamaha – str. 28 (rys. 11.); Audioklan/Yamaha – str. 29 (rys. 14.); Audioklan/Yamaha – str. 30 (rys. 15. i 16.); SSPL/Getty Images – str. 31 (rys. 17.), str. 32 (rys. 18. i 19.), str. 34 (rys. 21); Michał Koba – str. 35 (rys. 22); SSPL/Getty Images – str. 39 (rys. 1.), str. 42 (zawody pływackie); Marcin Mytych – str. 50 (gabinet stomatologiczny); SSPL/Getty Images – str. 84 (rys. 1.); Krzysztof Fortuna – str. 90 (rys. 7b); Marcin Mytych – str. 90 (ławka); SSPL/Getty Images – str. 98 (rys. 16.), str. 135 (rys. 11.); Grażyna Koba – str. 144 (pieniądze); SSPL/Getty Images – str. 149 (stos książek), str. 151 (króliki); Grażyna Koba – str. 169 (ogród); Marcin Mytych – str. 171 (rys. 3.), str. 172 (tabela 1.), str. 173 (rys. 4.), str. 174 (rys. 5.), str. 175 (rys. 6.), str. 181 (rys. 12.); Michał Koba – str. 187 (rys. 1.), str. 188 (rys. 2.), str. 193 (rys. 8b), str. 195 (morze).

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

Moduł A Wokół komputera i sieci komputerowych 7

Temat A1. Reprezentacja danych w komputerze 8

1. Logiczny model komputera 8
   1.1. Na czym polega działanie procesora? 9
   1.2. W jaki sposób procesor dodaje liczby? 11
2. System dwójkowy (binarny) 11
   2.1. Wprowadzenie do systemów liczbowych 11
   2.2. Wartość liczby zapisanej w systemie dwójkowym 13
   2.3. Wyznaczanie rozwinięcia dwójkowego liczby dziesiętnej 14
3. System szesnastkowy 15
   3.1. Wartość liczby zapisanej w systemie szesnastkowym 15
   3.2. Wyznaczanie rozwinięcia szesnastkowego liczby dziesiętnej 15
   3.3. Zależność między systemem dwójkowym a szesnastkowym 16

Temat A2. Wybrane urządzenia cyfrowe 19

1. Wprowadzenie 19
2. Urządzenia cyfrowe w szkole 20
   2.1. Drukarki 20
   2.2. Tablice i monitory 25
   2.3. Kamery do monitoringu 26
3. Urządzenia cyfrowe w domu 27
   3.1. Płyta grzejna i okap kuchenny 27
   3.2. Domowy system audio 27
   3.3. Dysk sieciowy 30
4. Motoryzacja i turystyka 31
   4.1. System eCall 31
   4.2. Technologie Apple CarPlay i Android Auto 32
   4.3. Systemy nawigacji 32
   4.4. Lokalizatory 34
5. Smartwatch 35

Moduł B Wokół dokumentów komputerowych 37

Temat B1. Tworzenie baz danych 38
1. W gąszczu informacji… 38
2. Podstawowe pojęcia związane z bazami danych 39
3. Etapy tworzenia bazy danych 40
4. Przykład bazy danych – obsługa szkolnych zawodów pływackich 42
   4.1. Określenie wymagań bazy danych *Zawody* 42
   4.2. Wstępne projektowanie formularzy i raportów w bazie danych *Zawody* 43
   4.3. Przygotowanie tabel w bazie danych *Zawody* 44
   4.4. Definiowanie relacji w bazie danych *Zawody* 46
5. Tworzenie bazy danych – obsługa gabinetu stomatologicznego 50
   5.1. Określenie wymagań bazy danych *Stomatolog* 50
   5.2. Wstępne projektowanie formularzy i raportów w bazie danych *Stomatolog* 51
   5.3. Tworzenie tabel w bazie danych *Stomatolog* 51
   5.4. Definiowanie relacji w bazie danych *Stomatolog* 53

Temat B2. Tworzenie i stosowanie formularzy w relacyjnej bazie danych 57
1. Korzystanie z formularzy w bazie *Zawody* *57*
2. Zasady tworzenia formularzy na przykładzie bazy *Zawody* *61*
   2.1. Formularz standardowy 61
   2.2. Formularz z podformularzem 62
3. Przygotowanie formularzy w bazie *Stomatolog* *64*
4. Importowanie danych z innych dokumentów do tabeli bazy danych 66

Temat B3. Wykonywanie podstawowych operacji na relacyjnej bazie danych 69
1. Wprowadzenie 69
2. Zastosowanie filtrów do wyszukiwania danych 70
3. Zasady tworzenia kwerend na przykładzie bazy *Zawody* 70
   3.1. Tworzenie kwerend wybierających 70
   3.2. Tworzenie kwerend parametrycznych 72
4. Zasady tworzenia raportów na przykładzie bazy *Zawody* 74
5. Przygotowywanie kwerend i raportów w bazie *Stomatolog* 77
6. Przygotowywanie korespondencji seryjnej 78

Temat B4. Projektowanie modeli dwuwymiarowych i trójwymiarowych 83
1. Rodzaje grafiki – przypomnienie 83
2. Określenie problemu – zaprojektowanie ogrodu 84
3. Wybrane możliwości programu SketchUp przydatne do projektowania ogrodu 86
   3.1. Program SketchUp 86
   3.2. Przekształcanie modelu 2D w 3D 88
   3.3. Projektowanie trójwymiarowego modelu ławki 89
4. Tworzenie projektu ogrodu 91
   4.1. Model 2D ogrodu 91
   4.2. Model 3D ogrodu 92
5. Projektowanie własnego pokoju 94

Moduł C Wokół algorytmiki i programowania 99

Temat C1. Algorytmy na tekstach w językach C++ i Python 100
1. Wybrane algorytmy na tekstach 100
   1.1. Porównywanie tekstów 101
   1.2. Szukanie i zliczanie wystąpień wybranego znaku w tekście 104
   1.3. Wyszukiwanie wzorca w tekście metodą naiwną 105

2. Praca z danymi tekstowymi w językach C++ i Python 107
2.1. Dane tekstowe w języku C++ 107
2.2. Dane tekstowe w języku Python 109
2.3. Używanie danych tekstowych w językach C++ i Python 112
3. Programowanie algorytmów na tekstach w językach C++ i Python 114
3.1. Zapisywanie algorytmu porównywania tekstów w językach C++ i Python 114
3.2. Zapisywanie algorytmu szukania i zliczania wystąpień znaku w tekście w językach C++ i Python 115
3.3. Zapisywanie algorytmu wyszukiwania wzorca w tekście metodą naiwną w językach C++ i Python 118

Temat C2. Algorytmy szyfrowania 123
1. W jakim celu stosujemy szyfry? 123
2. Podstawowe pojęcia dotyczące szyfrowania 124
3. Szyfrowanie tekstu metodą podstawieniową na przykładzie szyfru Cezara 125
4. Szyfrowanie tekstu metodą przestawieniową 126
5. Programowanie algorytmów szyfrowania w językach C++ i Python 127
5.1. Miniszyfrowanie – tworzenie anagramów 127
5.2. Programowanie algorytmu szyfrowania podstawieniowego 128
5.3. Programowanie algorytmu szyfrowania przestawieniowego 131
6. Więcej o szyfrowaniu 134
6.1. Szyfrowanie symetryczne i asymetryczne 134
6.2. Maszyna Enigma 135
6.3. Algorytm RSA 136

Temat C3. Metoda połowienia, podejście zachłanne i rekurencja 138
1. Metoda połowienia 138
1.1. Stosowanie metody połowienia do wyszukiwania elementu w zbiorze uporządkowanym 138
1.2. Zapisywanie algorytmu wyszukiwania elementu w zbiorze uporządkowanym metodą połowienia w językach C++ i Python 141
2. Metoda zachłanna 143
2.1. Jak znaleźć najlepszy sposób osiągnięcia celu? 143
2.2. Stosowanie metody zachłannej do wydawania reszty 144
2.3. Zapisywanie algorytmu wydawania reszty metodą zachłanną w językach C++ i Python 145
3. Na czym polega rekurencja? 147
4. Algorytm obliczania silni 148
4.1. Iteracyjne obliczanie silni liczby naturalnej 148
4.2. Rekurencyjne obliczenie silni liczby naturalnej 149
5. Rekurencyjne obliczanie wartości elementów ciągu Fibonacciego 151

Temat C4. Algorytmy na liczbach w językach C++ i Python 154
1. Zamiana liczby dwójkowej na dziesiętną 154
2. Zamiana liczby dziesiętnej na dwójkową 155
3. Zamiana liczb między systemami pozycyjnymi 156
3.1. Obliczanie wartości dziesiętnej liczby 157
3.2. Zapisywanie liczby dziesiętnej w systemie liczbowym o określonej podstawie 159

4. Działania na ułamkach z wykorzystaniem NWD i NWW 160
4.1. NWD i NWW 160
4.2. Reprezentacja ułamka w programie komputerowym 161
4.3. Skracanie ułamków zwykłych 163
4.4. Dodawanie i odejmowanie ułamków zwykłych 164
4.5. Mnożenie i dzielenie ułamków zwykłych 165

Moduł D Wokół Internetu i projektów 167

Temat D1. Elementy robotyki – projekty 168
1. Wprowadzenie 168
1.1. Przygotowanie do wykonania projektu 168
1.2. Czym jest Arduino? 169
2. Projekt sterowania nawadnianiem ogrodu 171
3. Zadania projektowe 179

Temat D2. Więcej na temat tworzenia stron internetowych – projekt 183
1. Strony internetowe – usystematyzowanie wiadomości 183
1.1. Blogi 184
1.2. Systemy zarządzania treścią 185
1.3. Szablony do tworzenia stron 185
2. Przygotowanie projektu 186
3. Etapy tworzenia strony internetowej 186
3.1. Projekt graficzny strony 186
3.2. Wdrożenie – realizacja projektu 187
3.3. Testowanie 192
4. Zadania projektowe 194

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



Słowniczek – wyjaśnienie ważnych terminów

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

Na końcu każdego tematu 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



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

Wokół komputera i sieci komputerowych

A

Moduł

Temat A1. Reprezentacja danych w komputerze

Temat A2. Wybrane urządzenia cyfrowe

Temat A1. Reprezentacja danych w komputerze

1. Logiczny model komputera
   1.1. Na czym polega działanie procesora?
   1.2. W jaki sposób procesor dodaje liczby?
2. System dwójkowy (binarny)
   2.1. Wprowadzenie do systemów liczbowych
   2.2. Wartość liczby zapisanej w systemie dwójkowym
   2.3. Wyznaczanie rozwinięcia dwójkowego liczby dziesiętnej
3. System szesnastkowy
   3.1. Wartość liczby zapisanej w systemie szesnastkowym
   3.2. Wyznaczanie rozwinięcia szesnastkowego liczby dziesiętnej
   3.3. Zależność między systemem dwójkowym a szesnastkowym

Warto powtórzyć

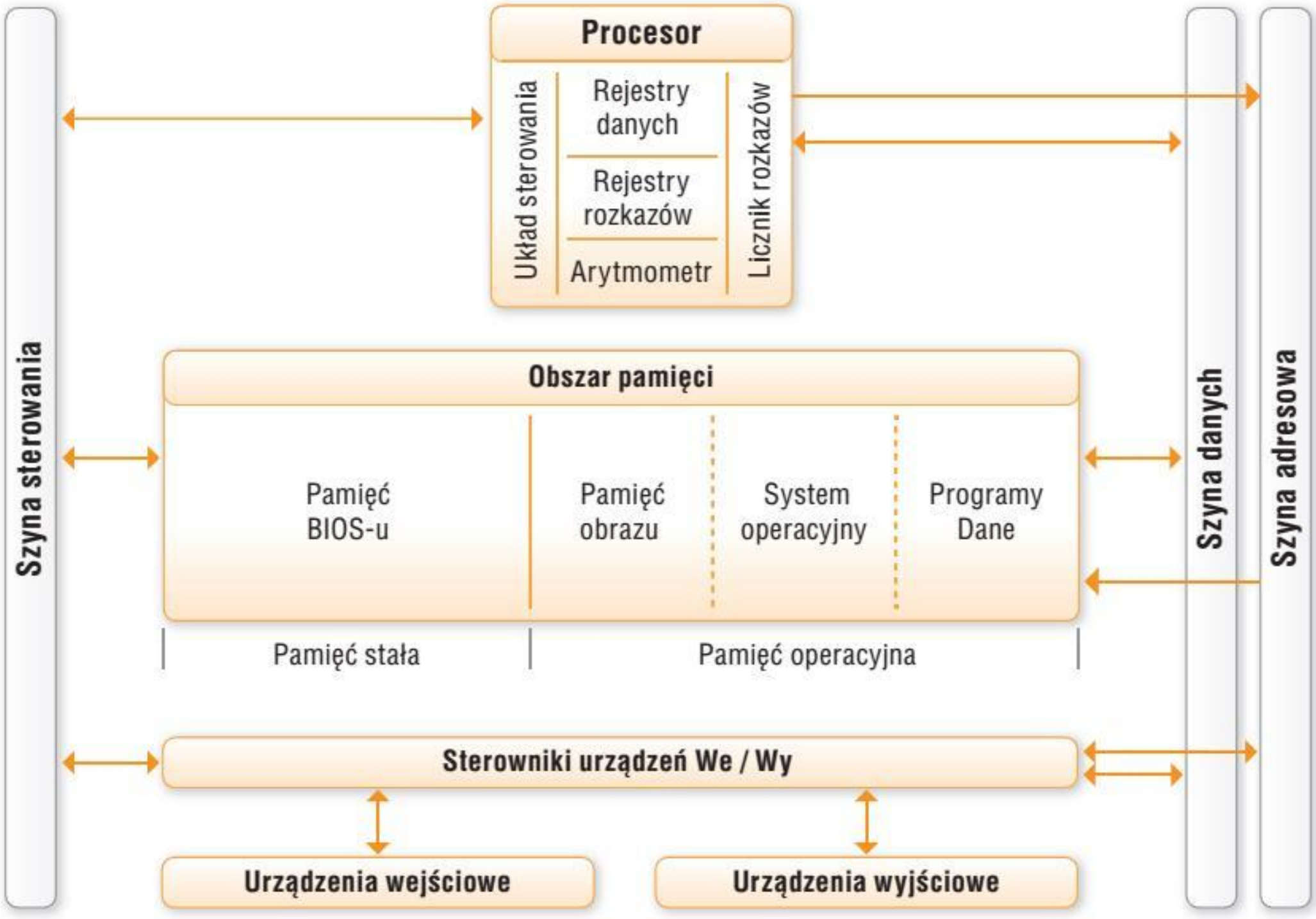
1. Wymień i krótko scharakteryzuj podstawowe elementy komputera.
2. Jaką rolę pełni w komputerze pamięć operacyjna?
3. Jaką funkcję pełni procesor? Jakie parametry wpływają na szybkość jego pracy? Czym charakteryzują się procesory wielordzeniowe?
4. W jaki sposób w matematyce zapisuje się rozwinięcie dziesiętne liczby? Wyjaśnij na przykładzie liczby 9871.

1. Logiczny model komputera

Od wprowadzenia komputerów osobistych na rynek minęło już ponad 30 lat, a od opracowania logicznego modelu komputera przez Johna von Neumanna – przeszło pół wieku. Ogólne zasady ich działania pozostały jednak niezmienne i współczesne komputery są wciąż budowane według pierwotnej koncepcji.

Idea maszyny von Neumanna:
- programy i dane posiadają taką samą postać,
- programy i dane są przechowywane w tej samej pamięci.

Zgodnie z tą koncepcją dane mogą być traktowane jako instrukcje programu komputerowego, a instrukcje programu komputerowego – jako dane.



Rys. 1. *Uproszczony model komputera zgodny z ideą von Neumanna*

1.1. Na czym polega działanie procesora?

Głównym elementem komputera jest procesor (CPU – z ang. *Central Processing Unit*). Współczesny procesor to pojedynczy układ scalony, oparty na strukturze krzemowej, zawierającej od kilkudziesięciu milionów do kilkunastu miliardów tranzystorów.

Procesor wykonuje operacje arytmetyczne i logiczne oraz odpowiada za prawidłową współpracę wszystkich modułów funkcjonalnych komputera. Szybkość taktowania współczesnego procesora wynosi kilka gigaherców, zatem w jednej sekundzie taki procesor potrafi wykonać kilka miliardów operacji (w przypadku procesorów wielordzeniowych liczbę tę należy pomnożyć przez liczbę rdzeni procesora). W ich zakres wchodzą m.in. operacje arytmetyczne (np. dodawanie, odejmowanie) i logiczne (np. suma logiczna, iloczyn, negacja, porównywanie) oraz operacje przesyłania danych z i do pamięci operacyjnej. Zbiór tych operacji nazywamy **listą rozkazów procesora**. W oparciu o możliwości listy rozkazów konstruowane są programy komputerowe.



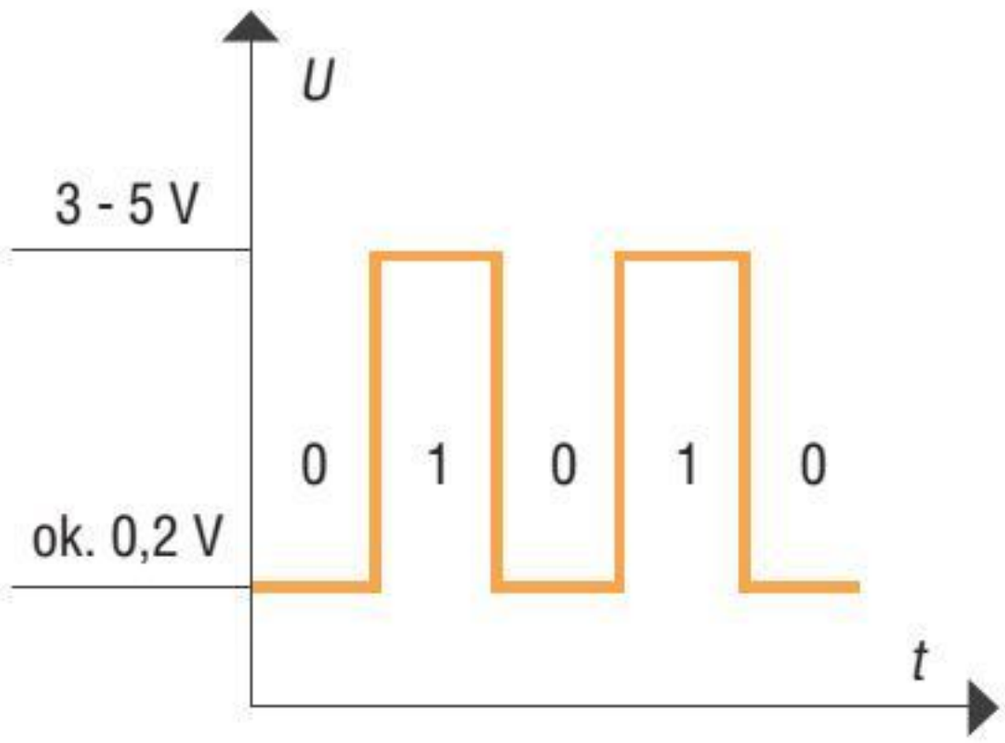
Rozkazy procesora to dane zapisane w postaci zero-jedynkowej, przy czym każdemu rozkazowi odpowiada inna sekwencja danych. Fizycznie rozkazy, tak jak i dane, istnieją w komputerze w postaci sygnałów elektrycznych.

Przyjmuje się, że obecność sygnału odpowiada cyfrze 1, a brak sygnału – cyfrze 0. Każda z tych wartości (zarówno 0, jak i 1) stanowi 1 **bit** informacji.

Kombinacja zer i jedynek pozwala na matematyczny zapis dowolnej liczby w systemie. Kombinacja na przykład ośmiu bitów stanowi 1 bajt informacji.

Uwaga

Oznaczenia:
B – bajt
b – bit
1 bajt = 8 bitów
1 Kb (kilobit) = 1024 b
1 Mb (megabit) = 1024 Kb
…
1 KB (kilobajt) = 1024 B
1 MB (megabajt) = 1024 KB
1 GB (gigabajt) = 1024 MB
1 TB (terabajt) = 1024 GB.



Rys. 2. *Reprezentacja informacji w komputerze w postaci impulsów elektrycznych*

Żeby komputer mógł wykonywać jakiekolwiek programy i operować danymi, muszą być one dla niego dostępne. Miejscem, w którym są umieszczane, jest **pamięć operacyjna**. Chociaż fizycznie pamięć operacyjna może być zrealizowana w różny sposób (jako pamięć półprzewodnikowa, wirtualna – odwzorowana za pomocą pamięci masowej) – procesor widzi ją jako jednolity obszar pamięciowy.

Adres komórki	Zawartość komórki
0	
1	Rozkaz procesora
2	Rozkaz procesora
…	…
…	…
k-1	Wartość (dane)
k	Wartość (dane)
k+1	Adres innej komórki
…	…
…	Rozkaz procesora
n	…

Rys. 3. *Dane umieszczone w komórkach pamięci operacyjnej o określonych adresach*

Pamięć operacyjną możemy sobie wyobrazić jako zbiór ponumerowanych komórek. Numer komórki jest jej adresem.

Zawartość komórki stanowią **dane**. Danymi mogą być:

- wartości wprowadzone przez użytkownika,
- wartości będące wynikiem działania programu,
- rozkazy procesora,
- adresy innych komórek,
- wartości pochodzące z urządzeń wejścia.

Procesor komunikuje się z pamięcią przez **szyny** (magistrale): adresową i danych (rys. 1). Szyna adresowa jest jednokierunkowa. Za jej pośrednictwem procesor uaktywnia odpowiednie komórki pamięci, umożliwiając tym samym zapis lub odczyt ich zawartości. Szyna danych jest dwukierunkowa. Pozwala na przesyłanie danych do i z procesora.

1.2. W jaki sposób procesor dodaje liczby?

Wykonanie programu umieszczonego w pamięci operacyjnej zaczyna się od pobrania pierwszego kodu rozkazu tego programu. Procesor żąda dostępu do niego, wstawiając na szynę adresową adres komórki, w której się znajduje. Pod żądanym adresem uaktywnia się odczyt zawartości komórki, zawartość ta jest przesyłana do procesora, gdzie zostaje umieszczona w rejestrze. Jeżeli jest to kod rozkazu, to zostanie on wykonany.

Przykład 1. Dodawanie liczb przez procesor

Dodawanie przez procesor wartości dwóch zmiennych (x, y) i zapisanie wartości w trzeciej zmiennej (z) dla: $x = 2$ i $y = 5$.

```
move.l   (ax),d0    przesłanie liczby spod adresu ax(2) w pamięci do rejestru d0
move.l   (ay),d1    przesłanie liczby spod adresu ay(5) w pamięci do rejestru d1
add.l    d0,d1      wykonanie operacji sumowania i zapisanie wyniku w rejestrze d1
move.l   d1,(az)    przesłanie liczby z rejestru d1(7) do adresu az w pamięci
```

W pamięci operacyjnej programy i dane umieszczane są tylko na czas pracy komputera. Dane, które muszą być pamiętane, gdy komputer nie pracuje, zapisujemy w **pamięci masowej** (np. na dysku komputera, na płycie DVD, na urządzeniu pendrive).

2. System dwójkowy (binarny)

Kombinacje zer i jedynek wystarczają matematycznie do zapisania dowolnej liczby, dlatego komputer posługuje się systemem, który ma do dyspozycji dwie cyfry: 0 i 1.

Komputer posługuje się systemem **binarnym**, czyli **dwójkowym**.

2.1. Wprowadzenie do systemów liczbowych

Dwa podstawowe rodzaje **systemów liczbowych** to:

- **system addytywny –** zapisaną w nim **liczbę** odczytuje się jako sumę wartości jej poszczególnych **cyfr**; przykładem jest rzymski system liczbowy;

- **system pozycyjny** – o wartości cyfry decyduje jej miejsce w zapisie liczby; obecnie najczęściej stosowanymi pozycyjnymi systemami liczbowymi są: system dziesiętny, używany w życiu codziennym, oraz system dwójkowy, stosowany w systemach komputerowych.

System liczbowy S

Sposób zapisywania i odczytywania liczb.

Podstawą systemu dziesiętnego jest liczba 10, będąca jednostką drugiego rzędu. Jednostką trzeciego rzędu jest liczba 100, czwartego rzędu – liczba 1000 itd. Jednostką pierwszego rzędu jest liczba 1. Ta sama cyfra ma w systemie dziesiętnym różne znaczenie, w zależności od miejsca, na którym stoi w zapisie liczby.

Przykład 2. Pozycyjne systemy liczbowe

W liczbie **1476** zapisanej w systemie dziesiętnym cyfra „6" ma wartość 6, ponieważ znajduje się w kolumnie jedności; cyfra „7" oznacza 70, ponieważ znajduje się w kolumnie dziesiątek; cyfra „4" oznacza 400, ponieważ znajduje się w kolumnie setek, a cyfra „1" oznacza 1000, ponieważ znajduje się w kolumnie tysięcy.
Czyli: $1476 = 1 \cdot 1000 + 4 \cdot 100 + 7 \cdot 10 + 6 \cdot 1$,
a przy użyciu zapisu potęgowego: $1476 = 1 \cdot 10^3 + 4 \cdot 10^2 + 7 \cdot 10^1 + 6 \cdot 10^0$.
Jeżeli ponumerujemy cyfry, zaczynając numerowanie od zera w kolejności od prawej strony, to zauważymy, że tak nadane numery cyfr odpowiadają kolejnym wykładnikom potęgi liczby 10 przy danej cyfrze:

3 2 1 0
1 4 7 6 $= \mathbf{1} \cdot 10^3 + \mathbf{4} \cdot 10^2 + \mathbf{7} \cdot 10^1 + \mathbf{6} \cdot 10^0$

W systemie dziesiętnym używamy potęg liczby 10, podczas gdy np. w systemie trójkowym używamy potęg liczby 3, a w siedemnastkowym – potęg liczby 17. Potęgowaną liczbę nazywamy **podstawą systemu liczbowego**.

W systemie dziesiętnym do zapisywania liczb używamy dziesięciu cyfr. W systemie trójkowym są to trzy cyfry (0, 1, 2), a w systemie siedemnastkowym – siedemnaście (problem oznaczenia cyfr o wartościach większych niż 9 pokażemy dalej na przykładzie systemu szesnastkowego).

Do zapisania liczby w systemie pozycyjnym o podstawie p konieczny jest zbiór cyfr $\{0, 1, ..., p\text{-}1\}$.

W systemie pozycyjnym o podstawie $\boldsymbol{p}$ liczba $\boldsymbol{n}$-cyfrowa: $(\boldsymbol{a}_{n-1}\boldsymbol{a}_{n-2}... \boldsymbol{a}_1\boldsymbol{a}_0)_p$
ma rozwinięcie:

$$(\boldsymbol{a}_{n-1}\boldsymbol{a}_{n-2}... \boldsymbol{a}_1\boldsymbol{a}_0)_p = \boldsymbol{a}_{n-1} \cdot \boldsymbol{p}^{n-1} + \boldsymbol{a}_{n-2} \cdot \boldsymbol{p}^{n-2} + ... + \boldsymbol{a}_1 \cdot \boldsymbol{p}^1 + \boldsymbol{a}_0 \cdot \boldsymbol{p}^0, \quad [1]$$

gdzie: $\boldsymbol{a}_{n-1}, \boldsymbol{a}_{n-2}, ..., \boldsymbol{a}_1, \boldsymbol{a}_0$ to kolejne cyfry danej liczby,
$\boldsymbol{p}$ – podstawa systemu liczbowego.

W tym temacie dowiemy się, w jaki sposób zamienia się reprezentację liczb pomiędzy systemami dwójkowym a dziesiętnym, a także pomiędzy systemami szesnastkowym a dziesiętnym, aby w temacie C3 zaprogramować algorytmy zamiany liczb między pozycyjnymi systemami liczbowymi w językach C++ i Python.

2.2. Wartość liczby zapisanej w systemie dwójkowym

Sposób reprezentacji danych stosowany w komputerach pokażemy na przykładzie liczb naturalnych.

Jeśli przykładowa liczba ma postać: 11110, to taki zapis można zinterpretować jako jedenaście tysięcy sto dziesięć w systemie dziesiętnym lub jako liczbę zapisaną w systemie dwójkowym. Z tego powodu musimy uściślić zapis, przez określenie podstawy systemu liczbowego. Podstawę zapisujemy w indeksie dolnym na końcu liczby. Na przykład:

11110_2 – liczba w systemie dwójkowym,
20122_3 – liczba w systemie trójkowym,
11110, 20122 – liczby w systemie dziesiętnym.

Ponieważ na co dzień posługujemy się systemem dziesiętnym, przyjęło się, że liczby bez podstawy zapisanej na końcu to właśnie liczby dziesiętne. Czyli np. zamiast pisać 1476_{10}, piszemy po prostu 1476.

Aby obliczyć wartość dziesiętną liczby zapisanej w systemie dwójkowym, należy dokonać rozwinięcia liczby według wzoru [1], przyjmując jako podstawę systemu liczbę 2.

Ćwiczenie 1. Obliczamy wartość liczby dwójkowej

Posługując się wzorem [1], sprawdź, czy wartość liczby 11110_2 wynosi 30.

Wskazówka: Ponumeruj cyfry, zaczynając numerowanie od zera w kolejności od prawej strony podobnie jak w przykładzie 2.:

4 3 2 1 0
1 1 1 1 0$_2$

Przykład 3. Obliczanie wartości dziesiętnej liczby zapisanej w systemie dwójkowym

Ponieważ $0 \cdot 2^n = 0$ i $1 \cdot 2^n = 2^n$, w praktyce wystarczy zsumować tylko te potęgi dwójki, które znajdują się przy cyfrach 1, np.:

11 10 9 8 7 6 5 4 3 2 1 0
101101110110$_2$ = **1 0 1 1 0 1 1 1 0 1 1 0**$_2$ =
$= 2^{11} + 2^9 + 2^8 + 2^6 + 2^5 + 2^4 + 2^2 + 2^1 =$
$= 2048 + 512 + 256 + 64 + 32 + 16 + 4 + 2 = 2934.$

W systemach komputerowych dane są zapisywane przy użyciu liczb bitów będących potęgami liczby 2. Do zapisu liczb używa się: 8 bitów – tzw. krótkie słowo (1 bajt), 16 bitów – tzw. słowo, 32 bitów – tzw. długie słowo (podwójne słowo), 64 bitów – tzw. poczwórne słowo. Liczba 2934 mogłaby więc być zapisana jako słowo:

MSB LSB
0000101101110110_2 (dopełniona do 16 cyfr zerami z lewej strony).

W zapisie binarnym skrajna lewa wartość to tzw. najbardziej znaczący bit (z ang. MSB – *Most Significant Bit*), skrajna prawa – to tzw. najmniej znaczący bit (z ang. LSB – *Least Significant Bit*).

Ćwiczenie 2. Obliczamy wartości liczb dwójkowych

Oblicz wartości dziesiętne liczb 1000000000_2 i 11111111_2.

2.3. Wyznaczanie rozwinięcia dwójkowego liczby dziesiętnej

Rozwinięcie dwójkowe liczby dziesiętnej można wyznaczyć na kilka sposobów. Przedstawimy jeden z nich.

Aby wyznaczyć postać dwójkową (zwaną rozwinięciem dwójkowym lub binarnym) liczby dziesiętnej, powtarzamy operację dzielenia z resztą tej liczby przez dwa i zapisujemy wszystkie reszty z dzieleń, aż do otrzymania ilorazu równego zero.

Rozwinięcie dwójkowe liczby to zapisane kolejno (**począwszy od ostatniej**) reszty z dzieleń.

Ćwiczenie 3. Obliczamy rozwinięcie dwójkowe liczby zapisanej w systemie dziesiętnym

Sprawdź, czy rozwinięcie dwójkowe liczby 1476 to 10111000100_2. Utwórz tabelę podobną do pokazanej na rysunku 4. i wykonaj brakujące obliczenia według wzoru.

Liczba	Reszta	Komentarz
1476	0	$1476 = 2 \cdot 738 + 0$
738	0	$738 = 2 \cdot 369 + 0$
...	...	...
5	1	$5 = 2 \cdot 2 + 1$
2	0	$2 = 2 \cdot 1 + 0$
1	1	$1 = 2 \cdot 0 + 1$
0		iloraz równy 0 – koniec (przy dzieleniu z resztą iloraz może być równy 0)

Rys. 4. *Wzór tabeli - ćwiczenie 3.*

Ćwiczenie 4. Obliczamy rozwinięcia dwójkowe liczb zapisanych w systemie dziesiętnym

1. Oblicz rozwinięcia dwójkowe liczb 255 i 1024.
2. Sprawdź, jaki rysunek utworzą cyfry 1, gdy zapiszesz rozwinięcia dwójkowe liczb: 16, 16, 56, 124, 254, 16, 16, 56, tak aby ich ostatnie cyfry znajdowały się jedna pod drugą.

Wskazówka: Polecenie z punktu 2. najlepiej wykonać na kartce w kratkę albo w tabeli. Można też w rozwinięciach dwójkowych zastąpić jedynki literami x, a zer nie wpisywać.

3. System szesnastkowy

3.1. Wartość liczby zapisanej w systemie szesnastkowym

W praktyce posługiwanie się systemem dwójkowym jest bardzo niewygodne – zapis w nim jest długi i łatwo o pomyłkę (np. o pominięcie zera lub jedynki). Często więc w informatyce, zamiast systemu dwójkowego, wykorzystuje się system szesnastkowy, w którym do dyspozycji mamy 16 cyfr. Oprócz cyfr od 0 do 9 pojawiają się następujące cyfry:

A – o wartości 10, B – 11, C – 12, D – 13, E – 14, F – 15.

Przykłady liczb w systemie szesnastkowym: 100_{16}, AB_{16}, $FFFF_{16}$, $400D_{16}$.

Przykład 4. Obliczanie wartości liczby zapisanej w systemie szesnastkowym

Aby obliczyć wartość liczby zapisanej w systemie szesnastkowym, używamy wzoru [1], przyjmując za podstawę systemu liczbę 16:

$$\overset{3\ 2\ 1\ 0}{\mathbf{A\,B\,C\,D}}_{16} = \mathbf{A}\cdot 16^3 + \mathbf{B}\cdot 16^2 + \mathbf{C}\cdot 16^1 + \mathbf{D}\cdot 16^0 =$$

$$\mathbf{10}\cdot 16^3 + \mathbf{11}\cdot 16^2 + \mathbf{12}\cdot 16^1 + \mathbf{13}\cdot 16^0 =$$

$$= \mathbf{10}\cdot 4096 + \mathbf{11}\cdot 256 + \mathbf{12}\cdot 16 + \mathbf{13}\cdot 1 = 40960 + 2816 + 192 + 13 = 43981_{10}$$

Ćwiczenie 5. Obliczanie wartości liczb zapisanych w systemie szesnastkowym

Oblicz wartość dziesiętną liczb szesnastkowych: 100_{16}, $FFFF_{16}$, $C000_{16}$.

3.2. Wyznaczanie rozwinięcia szesnastkowego liczby dziesiętnej

Do zamiany z systemu dziesiętnego na system szesnastkowy służy ten sam algorytm, z którego korzystaliśmy przy zamianie na system dwójkowy z tą tylko różnicą, że dzielimy przez podstawę systemu szesnastkowego. Podobnie jak w przypadku zamiany na system dwójkowy, wykonujemy dzielenia z resztą, aż do otrzymania ilorazu równego 0.

Ćwiczenie 6. Obliczamy rozwinięcie szesnastkowe liczby zapisanej w systemie dziesiętnym

Zamień liczbę dziesiętną 1476 na liczbę szesnastkową. Utwórz tabelę podobną do utworzonej w ćwiczeniu 3. (rys. 4.).

Wskazówka: Pamiętaj, że cyfry o wartościach powyżej 9 są reprezentowane przez litery.

Ćwiczenie 7. Obliczamy rozwinięcia szesnastkowe liczb zapisanych w systemie dziesiętnym

1. Podaj swój wiek w systemie szesnastkowym.
2. Oblicz rozwinięcia szesnastkowe liczb dziesiętnych 100 i 64.

3.3. Zależność między systemem dwójkowym a szesnastkowym

Zauważmy, że systemu szesnastkowego można używać zamiast systemu dwójkowego, ponieważ $2^4 = 16$, a więc jedna cyfra szesnastkowa odpowiada **dokładnie** czterem cyfrom dwójkowym.

W przypadku systemu dziesiętnego nie ma takiej bezpośredniej zależności pomiędzy cyframi dziesiętnymi a cyframi dwójkowymi, bowiem liczba 10 nie jest potęgą całkowitą liczby dwa.

Aby zamienić liczbę szesnastkową na dwójkową, wystarczy więc sprawdzić wartość dziesiętną każdej cyfry szesnastkowej i zamienić ją na system dwójkowy, pamiętając o ewentualnym dopełnieniu zerami z przodu, tak by każda liczba dwójkowa odpowiadająca cyfrze szesnastkowej miała dokładnie cztery cyfry. Otrzymane rozwinięcia dwójkowe zapisujemy w tej samej kolejności, co odpowiadające im cyfry szesnastkowe.



Przykład 5. Zamienianie liczby zapisanej w systemie szesnastkowym na liczbę dwójkową

Zapiszemy liczbę $5C4_{16}$ w systemie dwójkowym.

$5_{16} = 5_{10} = 101_2 = 0101_2$ (dopełniamy do 4 cyfr jednym zerem z przodu)

$C_{16} = 12_{10} = 1100_2$

$4_{16} = 4_{10} = 100_2 = 0100_2$ (dopełniamy do 4 cyfr jednym zerem z przodu)

Zatem:

5 C 4

$5C4_{16} = 0101\ 1100\ 0100_2$

Aby przyspieszyć zamianę, można posłużyć się wartościami podanymi w tabeli 3.

Wartość dziesiętna	Wartość szesnastkowa	Wartość dwójkowa
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Tabela 1. *Wartości dziesiętne, szesnastkowe i dwójkowe liczb od 0 do 15*

Zasadę, zgodnie z którą jedna cyfra szesnastkowa odpowiada czterem cyfrom dwójkowym, wykorzystujemy także przy zamianie liczb z systemu dwójkowego na system szesnastkowy. Liczbę dwójkową dzielimy na czterocyfrowe grupy, zaczynając od prawej strony. Następnie obliczamy wartość dziesiętną każdej grupy i wpisujemy odpowiadającą jej cyfrę szesnastkową. Dla ułatwienia obliczeń można posłużyć się wartościami podanymi w tabeli 1.

Przykład 6. Zamienianie liczby zapisanej w systemie dwójkowym na liczbę szesnastkową

Zapiszemy liczbę $11000111010111010111011_2$ w systemie szesnastkowym.

$11|0001|1101|0111|0101|1011_2$

11 0001 1101 0111 0101 1011 (dzielimy na grupy)
3 1 13 7 5 11 (obliczamy wartość dziesiętną każdej grupy)
3 1 D 7 5 B (wpisujemy odpowiednią cyfrę szesnastkową)

$11000111010111010111011_2 = 31D75B_{16}$

Ćwiczenie 8. Zamienianie liczby między różnymi systemami

Zapisz następujące liczby w systemach dwójkowym, dziesiętnym i szesnastkowym:

a. $ABC_{16} = ?_{10} = ?_2$
b. $2331_{10} = ?_{16} = ?_2$
c. $10011110101011011101110_2 = ?_{16} = ?_{10}$

Warto zapamiętać

- Procesor jest układem scalonym, wykonującym operacje arytmetyczne i logiczne zgodnie z programem.
- Działanie procesora polega na wykonywaniu kolejnych instrukcji programu.
- W procesorze są wykonywane proste operacje arytmetyczne i logiczne na danych pobieranych za pośrednictwem szyny adresowej i szyny danych.
- Wszelkie informacje (np. liczby, znaki alfanumeryczne) przechowywane w komputerze zapamiętywane są w postaci binarnej.
- Systemem pozycyjnym nazywamy taki sposób prezentowania liczb, w którym wartość cyfry zależy od jej pozycji w ciągu cyfr określającym liczbę.
- Rozwinięcie dwójkowe (inaczej binarne) liczby to zapisanie liczby dziesiętnej w systemie dwójkowym, czyli za pomocą zer i jedynek.

Pytania, problemy

1. Jaka jest główna idea maszyny von Neumanna?
2. Jak reprezentowane są rozkazy procesora?
3. W jaki sposób dane są umieszczane w pamięci operacyjnej?
4. Wyjaśnij, dlaczego przedstawiony w przykładzie 1. opis dodawania liczb przez procesor możemy nazwać algorytmem.
5. Przedstaw na schematycznym rysunku opis dodawania dwóch liczb przez procesor.

6. Omów pozycyjne systemy liczbowe na przykładzie systemu dziesiętnego.
7. Jakie znasz niepozycyjne systemy liczbowe?
8. Dlaczego na świecie powszechnie używanym systemem liczbowym jest system dziesiętny?
9. Czy istnieje system jedynkowy? Jak wyglądałby zapis liczby w tym systemie?
10. W jaki sposób wyznaczamy rozwinięcie dwójkowe liczby dziesiętnej? Pokaż na przykładzie.
11. Jak oblicza się wartość dziesiętną liczby binarnej?
12. Dlaczego systemu szesnastkowego można używać zamiast systemu dwójkowego? Wyjaśnij na przykładzie (innym niż podany w temacie).

Zadania

1. Oblicz wartość dziesiętną następujących liczb: 1110011010100_2, 3210_4, 444_5.
2. Oblicz rozwinięcie dwójkowe liczby określającej bieżący rok kalendarzowy.
3. Oblicz rozwinięcie dwójkowe liczby określającej twój wiek.
4. Oblicz rozwinięcie dwójkowe liczb: 2222, 500, 100.
5. Oblicz wartość następujących liczb szesnastkowych: ACE_{16}, 2000_{16}, 1111_{16}.
6. Oblicz rozwinięcie szesnastkowe następujących liczb: 640, 32678, 1024.

Dla zainteresowanych

7. Co powstanie na tym obrazku: (41702, 45705, 45801, 43657, 43657, 42633, 42633, 41606)?

Temat A2. Wybrane urządzenia cyfrowe

1. Wprowadzenie
2. Urządzenia cyfrowe w szkole
   - 2.1. Drukarki
   - 2.2. Tablice i monitory
   - 2.3. Kamery do monitoringu
3. Urządzenia cyfrowe w domu
   - 3.1. Płyta grzejna i okap kuchenny
   - 3.2. Domowy system audio
   - 3.3. Dysk sieciowy
4. Motoryzacja i turystyka
   - 4.1. System eCall
   - 4.2. Technologie Apple CarPlay i Android Auto
   - 4.3. Systemy nawigacji
   - 4.4. Lokalizatory
5. Smartwatch

Warto powtórzyć

1. Jakie znasz urządzenia peryferyjne?
2. Jak działa projektor multimedialny?
3. Jaki rodzaj drukarki najlepiej zakupić na domowy użytek? Uzasadnij odpowiedź.

1. Wprowadzenie

Coraz trudniej znaleźć urządzenie, które nie jest cyfrowe. Urządzeń cyfrowych używa się w domu, szkole, na uczelniach, w przedsiębiorstwach i w różnych instytucjach. Urządzenia cyfrowe to nie tylko drukarka, aparat cyfrowy, kamera cyfrowa czy smartfon, ale również urządzenia, które można podłączyć do systemu automatyki domowej lub którymi można sterować za pomocą smartfona, np. niektóre modele pralek, czajników, płyt grzejnych, domowych systemów audio, a także… samochód (w pewnym stopniu).

Ćwiczenie 1. Przygotowujemy prezentację na temat jednego z urządzeń cyfrowych

1. Zapoznaj się z urządzeniami, które w dalszych punktach tematu przedstawiono w umownym podziale na wykorzystywane w szkole, domu i innych miejscach.
2. Wybierz jedno urządzenie (np. drukarkę 3D, monitor interaktywny, eCall) i przygotuj krótką, 5-10 minutową prezentację na jego temat, którą przedstawisz klasie. W prezentacji podaj przykłady wybranych przez ciebie zastosowań urządzenia np. do nauki, pracy, komunikacji, rozrywki czy poprawy bezpieczeństwa.

2. Urządzenia cyfrowe w szkole

2.1. Drukarki

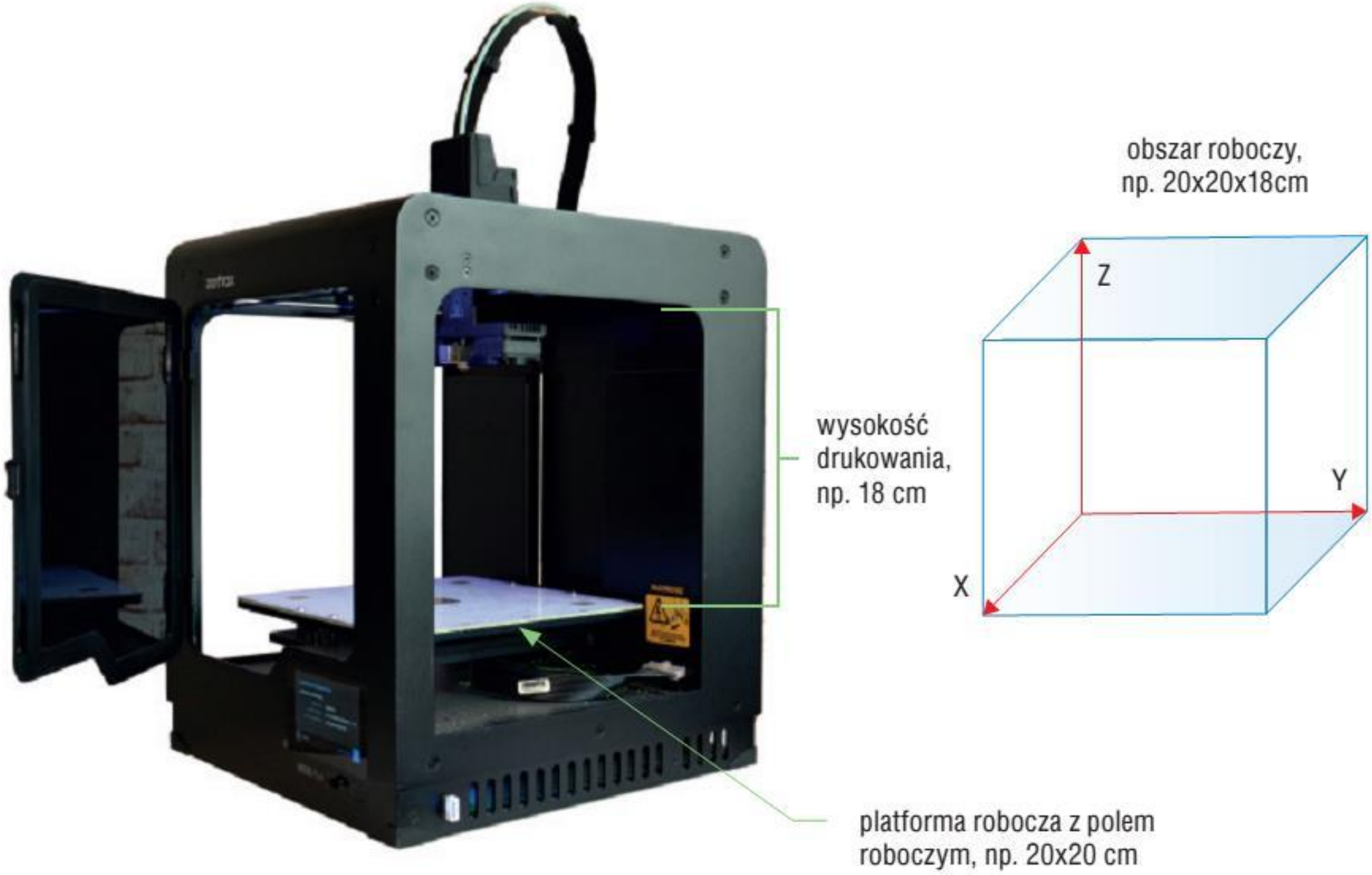
Druk 3D D

Drukowanie przestrzenne polega na wytwarzaniu fizycznego obiektu poprzez nanoszenie kolejnych warstw materiału (tzw. metoda przyrostowa lub technologia addytywna).

W szkole (a także w domu) korzystamy zazwyczaj z drukarek atramentowych i laserowych, dzięki którym uzyskujemy czarno-białe lub kolorowe wydruki dokumentów komputerowych, utworzonych np. w edytorach tekstu czy edytorach grafiki. Coraz większą popularnością cieszą się drukarki 3D. Omówimy je, podając kilka praktycznych zasad korzystania z tej technologii. Przygotujemy się do drukowania modelu 3D, który wykonamy w odpowiednim programie graficznym (temat B4).

Aby wydrukować model 3D, potrzebujemy:

- trójwymiarowego modelu komputerowego, utworzonego w edytorze grafiki 3D i zapisanego w pliku z rozszerzeniem *stl* lub *g-code*,
- programu do obróbki modelu, tzw. slicera,
- drukarki 3D,
- materiału do druku, np. plastikowej żyłki, żywicy czy proszku – zależnie od technologii.



Rys. 1. *Przykładowa drukarka 3D*

Każda drukarka ma **obszar roboczy** wyznaczający wielkość bryły, którą możemy wydrukować. Drukarka jest wyposażona w **platformę roboczą** (**stół roboczy**) o ustalonym **polu roboczym** (najczęściej 20x20 cm) oraz ma ograniczoną **wysokość drukowania** (rys. 1.).

Budowa i działanie drukarki 3D zależą od zastosowanej technologii. Najbardziej dostępnym i najpopularniejszym sposobem drukowania 3D jest FDM (z ang. *Fused Deposition Modelling*) – technologia drukowania termoplastycznym materiałem wyciskanym przez dysze. Materiał do druku (zwany **filamentem**) w technologii FDM ma postać żyłki ze sztucznego tworzywa, którą umieszcza się na specjalnej szpuli (rys. 2.).

Ogólna zasada działania drukarki 3D opiera się o **warstwy**. Każdy przedmiot, który powstał na drukarce 3D, składa się z wielu warstw układanych jedna na drugiej. Materiał ze szpuli trafia do elementu drukarki zwanego **ekstruderem**, gdzie topi się, a następnie jest przepychany do dyszy, która układa kolejne warstwy. Każda warstwa ma swoją, na ogół niewielką, wysokość. Im mniejsza wysokość warstwy, tym większa rozdzielczość wydruku. Oznacza to, że wysokość warstwy wpływa na gładkość wydrukowanych powierzchni oraz na to, czy wydruk będzie ładnie wyglądał. Jakość wydruku 3D zależy również od użytego materiału i zastosowanej technologii drukowania.





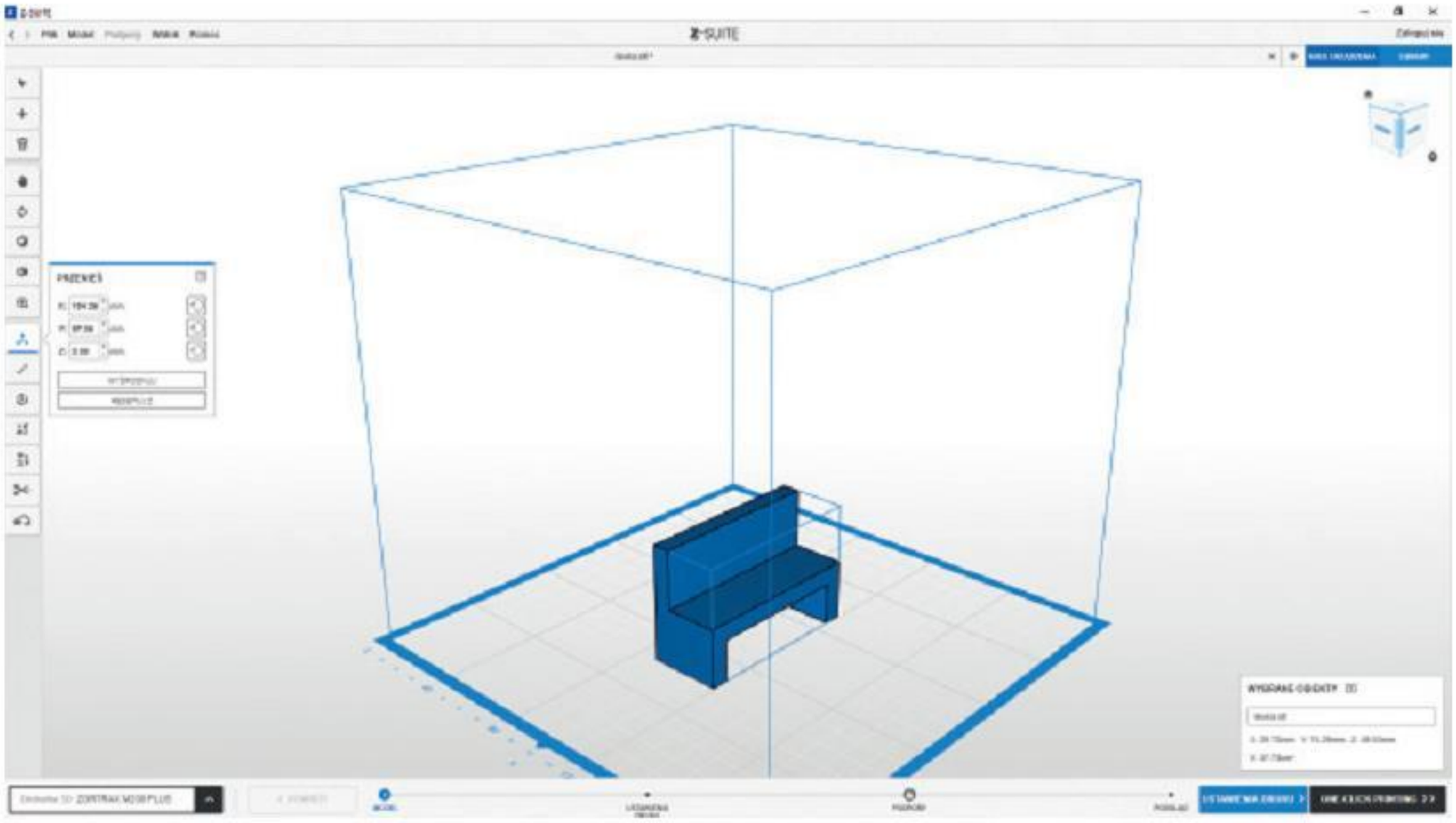
Rys. 2. *Szpula z materiałem do druku (filamentem – żyłką ze sztucznego tworzywa)*

Czy wiesz, że...

w 2006 roku Adrian Bowyer zbudował pierwszy prototyp drukarki 3D dla użytkowników domowych? Samą technikę przestrzennego druku opracował Chuck Hull w 1984 roku, a dwa lata później produkowano już pierwsze drukarki 3D, które od początku XXI wieku zdobywają coraz większą popularność. Technologia druku 3D znajduje zastosowanie m.in. w medycynie, np. do tworzenia protez, implantów.

Drukarka 3D nie wydrukuje trójwymiarowego modelu zapisanego w pliku z rozszerzeniem np. *stl* bez użycia specjalnego programu, tzw. slicera (pol. *krajalnica*). Program tego rodzaju (rys. 3.) umożliwia import modelu (np. w postaci pliku *stl*) i przygotowanie go do druku 3D, m.in. „pocięcie" modelu na warstwy. W przypadku gdy model jest większy niż obszar roboczy, slicer pozwala podzielić projektowany model na mniejsze części, które można oddzielnie wydrukować a potem skleić.

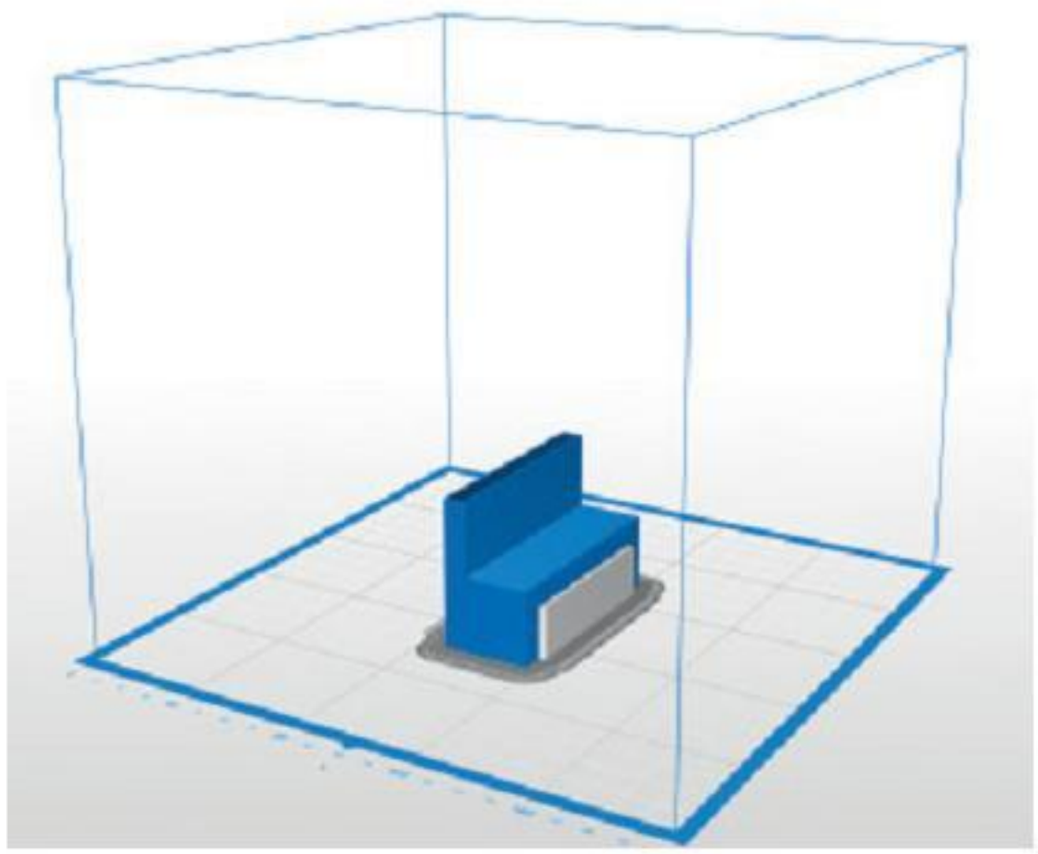
Często producenci oprogramowania dołączają slicer do danej drukarki, ale w Internecie są dostępne darmowe programy współpracujące z każdą drukarką (np. Cura, KISSlicer, Repetier-Host).



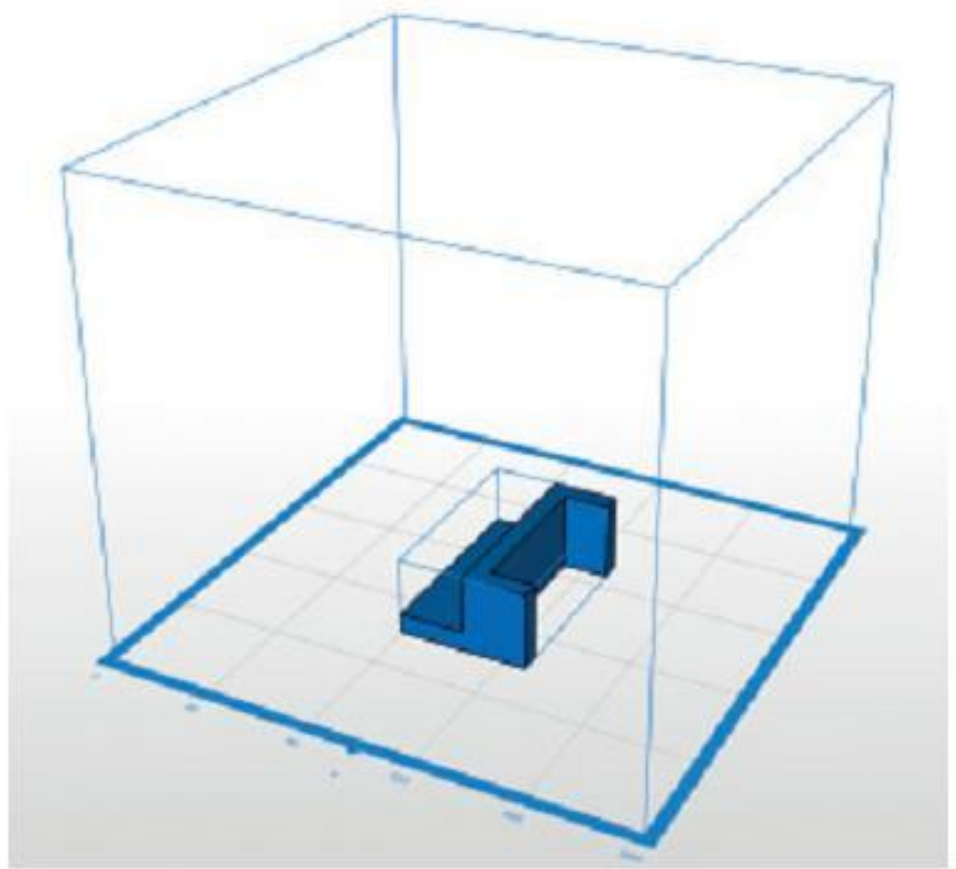
Rys. 3. *Zrzut okna przykładowego programu do obróbki modelu, tzw. slicera*

Kilka praktycznych zasad dotyczących druku 3D

1. Wydruk należy tak zaplanować, aby wygenerować jak najmniej podpór. Podpory to rusztowanie (drukowane podczas procesu) podtrzymujące powierzchnie drukowane pod kątem lub pozbawione własnej podpory. Nie można ułożyć materiału w powietrzu – powinien być podparty (rys. 4a). Podpory są odpadem, którego już nie wykorzystamy, więc podnoszą koszty wydruku oraz są nieekologiczne. Jednocześnie przez podpory zwiększa się czas potrzebny do wydrukowania przedmiotu. Żeby uniknąć stosowania podpór, można podzielić model na części. W modelu ławki prawidłowym ustawieniem (bez konieczności dodania podpór) jest ustawienie pokazane na rysunku 4b.

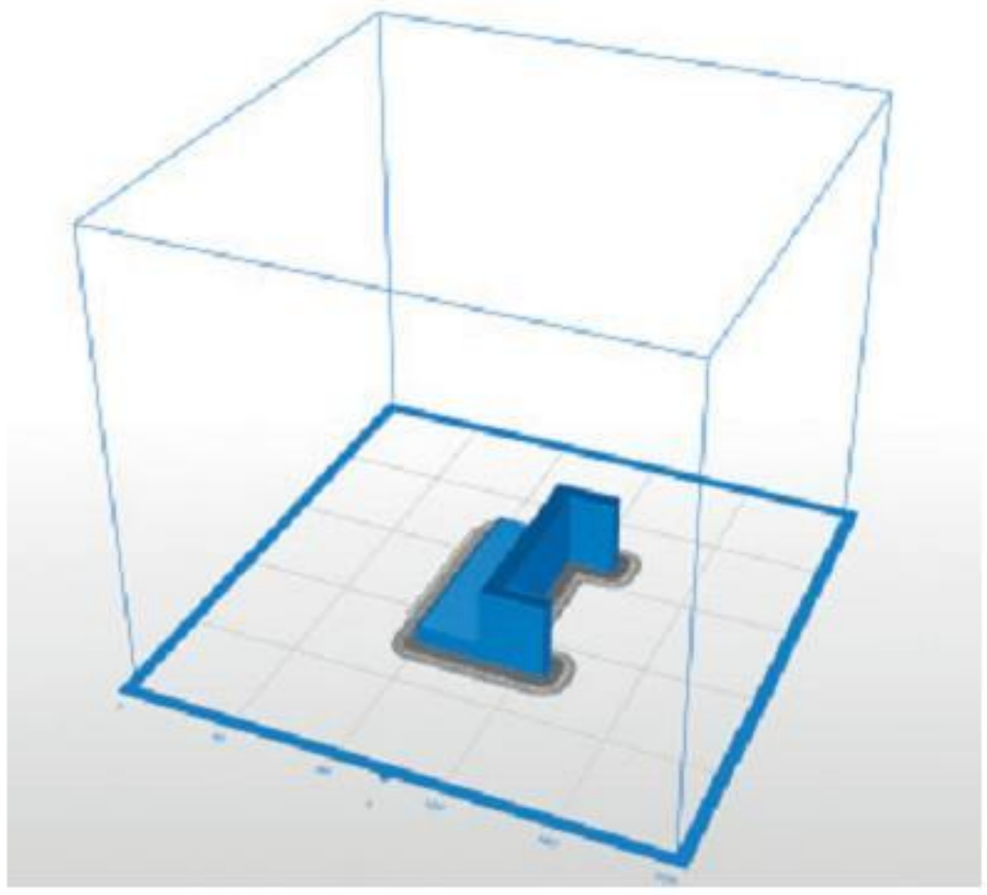


Rys. 4a. *Zastosowanie podpory do wydrukowania modelu*

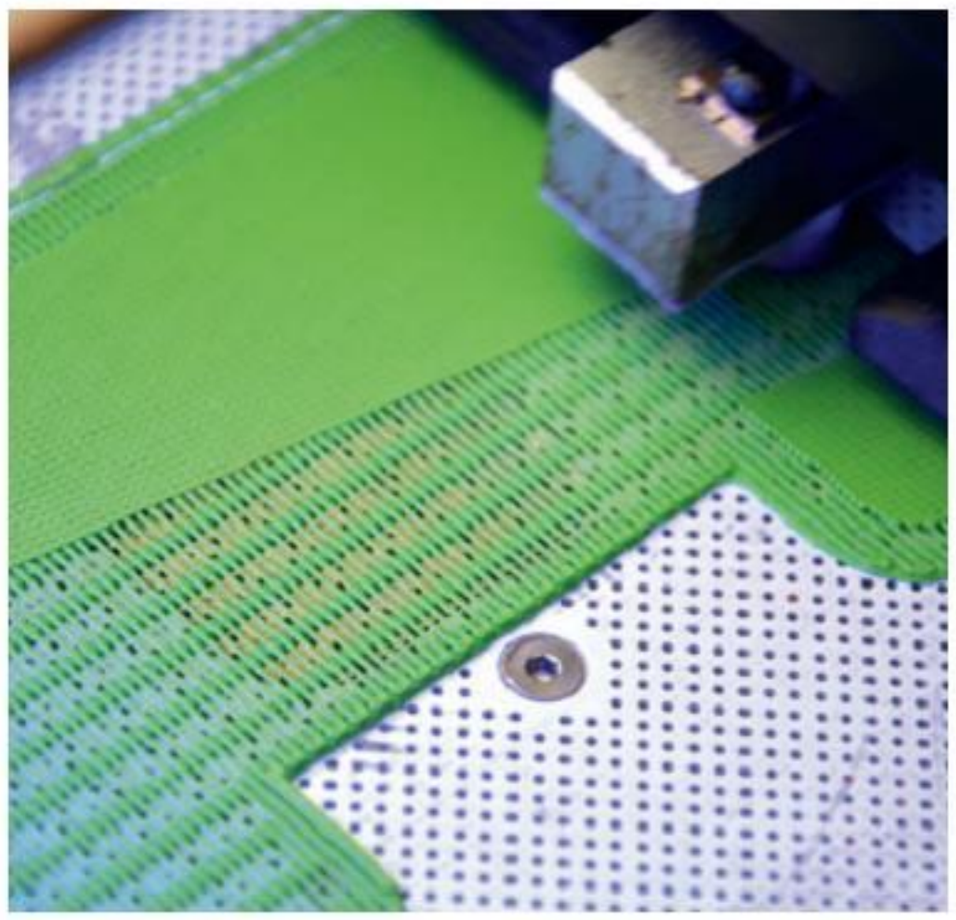


Rys. 4b. *Prawidłowo ustawiony model pokazany w oknie slicera*

2. Aby uniknąć problemów podczas druku – takich jak brak właściwej przyczepności, nierówna powierzchnia, wypaczenie wydruku – stosuje się **raft**, czyli wydrukowaną poziomą siatkę, umiejscowioną na powierzchni platformy roboczej. Raft planujemy na etapie przygotowania modelu do druku (rys. 5a). Na rafcie nanoszona jest pierwsza warstwa właściwego wydruku (rys. 5b). Jest to sposób na poprawę jakości pierwszej warstwy, a także na przytwierdzenie wydruku do platformy roboczej. Raftu używamy podczas drukowania z materiału podatnego na odkształcenia. Raft musimy zastosować zawsze wtedy, gdy drukowany element ma bardzo małą powierzchnię styku z platformą roboczą.



Rys. 5a. *Model w oknie slicera po cięciu na płaszczyzny drukowania z pokazanym raftem*



Rys. 5b. *Drukowanie raftu na drukarce 3D*

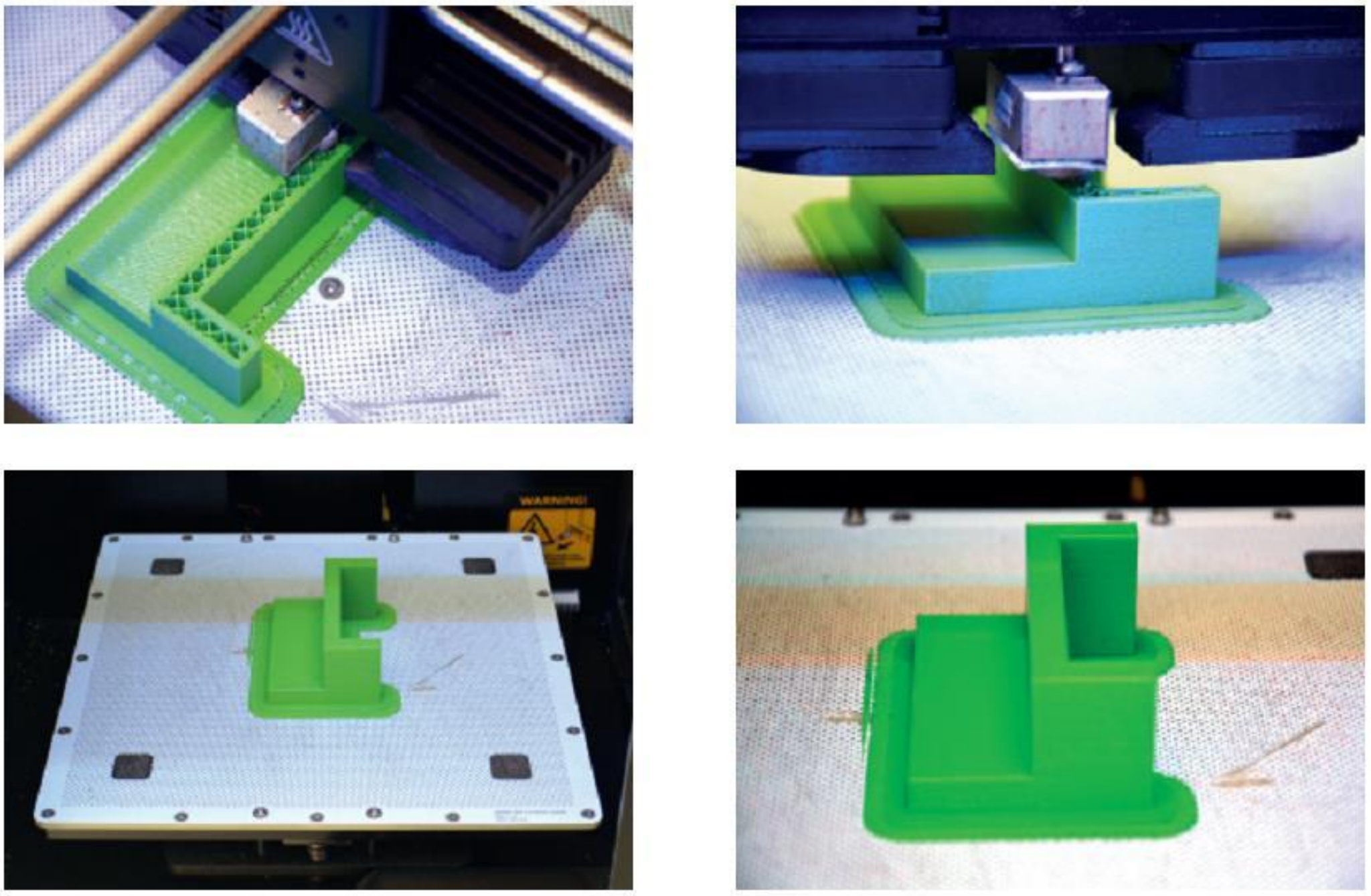
3. Przy wyborze materiału musimy wziąć pod uwagę, że niektóre materiały wymagają podgrzewanej platformy roboczej, aby zmniejszyć szybkość stygnięcia drukowanego modelu (szybkie stygnięcie może doprowadzić do deformacji).
4. Bardzo ważne jest określenie sposobu wypełnienia drukowanego modelu. Musimy ustalić, w ilu procentach model ma zostać wypełniony i jaką ma mieć strukturę. Można drukować model bez wypełnienia, ale wówczas będzie on bardzo delikatny. Standardowo przyjmuje się, że wystarczy wypełnienie 30-35%. Jeśli potrzebujemy większej wytrzymałości, zwiększamy wypełnienie, nawet do 90%. Możemy też drukować w 100% wypełnione modele, ale nie każdy materiał na to pozwala ze względu na kurczenie podczas stygnięcia.
5. Aby drukowany element dobrze przywarł do stołu roboczego, stosuje się różne techniki, np. stół perforowany lub szklany, którego powierzchnię roboczą smaruje się lub spryskuje specjalnym preparatem klejącym. Wielu użytkowników drukarek 3D ma problemy z uzyskaniem prawidłowej pierwszej warstwy wydruku, a to właśnie ona ma kluczowe znaczenie dla pomyślnego zakończenia procesu drukowania.

Czy wiesz, że...

istnieją drukarki, które potrafią w dwadzieścia cztery godziny wydrukować dom?

Ćwiczenie 2. Przygotowanie modelu 3D do druku

1. Zapoznaj się z możliwościami wybranego slicera, przygotowując model 3D do druku (przykładowy plik otrzymasz od nauczyciela).
2. Wydrukuj model, jeśli masz takie możliwości.



Rys. 6. *Wybrane fazy druku 3D*

Ćwiczenie 3. Szukamy dodatkowych informacji o druku 3D i dyskutujemy na ich temat

1. W Internecie lub innych źródłach znajdź następujące informacje:
   a. czym charakteryzuje się druk w technologiach żywicznych i proszkowych,
   b. dlaczego drukowanie modeli szklanych jest bardzo trudne,
   c. w jaki sposób drukuje się domy 3D.
2. Przeprowadźcie klasową dyskusję na temat wybranych technologii druku 3D, wykorzystując znalezione informacje.

Wskazówki: Ćwiczenie można wykonać w grupach, przydzielając każde zagadnienie innej osobie (lub kilku osobom). Wybrany uczeń wyszukuje informacje na zadany temat w określonym wcześniej czasie, a następnie w dyskusji wymienia się wiedzą z innymi.

2.2. Tablice i monitory

Tablice i monitory interaktywne

Tablica interaktywna współpracuje z komputerem i projektorem multimedialnym. Na tablicy, za pomocą projektora, wyświetlany jest obraz z komputera (rys. 7.). Korzystając z elektronicznego pióra, możemy na tablicy obsługiwać wszystkie funkcje komputera i uruchomionych aplikacji, na przykład zmienić wyświetlany rysunek i zapisać zmieniony obraz w pamięci komputera. Na monitorze komputera widzimy (w czasie rzeczywistym) to, co jest narysowane czy napisane na tablicy. Tablica interaktywna ma ponadto własne oprogramowanie, pozwalające np. na robienie notatek i zapisywanie ich w pamięci komputera.



Rys. 7. *Tablica interaktywna*



Rys. 8. *Monitor interaktywny*

Po wyłączeniu oprogramowania tablica interaktywna może pełnić funkcję zwykłej tablicy, po której można pisać pisakami suchościeralnymi.

W odróżnieniu od tablicy **monitor interaktywny** (rys. 8.) nie potrzebuje projektora, aby wyświetlić obraz.

Obraz na tym monitorze wygląda tak samo, jak na ekranie telewizora, a osoba stojąca przy nim nie rzuca cienia od światła projektora na ekran.

Monitor prezentacyjny

Monitory prezentacyjne służą do wyświetlania informacji w różnych obiektach (np. na szkolnym korytarzu). Treści (plansze) wyświetlane na monitorze prezentacyjnym aktualizuje się zdalnie przez sieć komputerową, co bardzo ułatwia zarządzanie przekazywanymi informacjami. Oprogramowanie do zarządzania monitorami prezentacyjnymi pozwala je grupować, dzięki czemu w prosty sposób możemy np. zmienić treści na wszystkich monitorach znajdujących się na danym piętrze.

Ćwiczenie 4. Porównujemy sposoby prezentacji z wykorzystaniem tablic i monitorów

Przedstaw w tabeli porównanie tablicy interaktywnej, monitora interaktywnego i monitora prezentacyjnego. Ustal samodzielnie kryteria porównania. W porównaniu możesz uwzględnić podobieństwa i różnice.

Wskazówka: Ćwiczenie można wykonać w grupach, przydzielając każde zagadnienie innej osobie (lub kilku osobom). Jeśli szkoła dysponuje odpowiednimi urządzeniami, porównanie ich działania można zaprezentować w praktyce.

2.3. Kamery do monitoringu

Coraz częściej we wszelkiego typu obiektach (w tym w szkołach, a także w domach) stosuje się monitoring wizyjny. Obecnie najczęściej wykorzystuje się kamery IP (z ang. *Internet Protocol*), czyli takie, które działają w sieci komputerowej.

Kamery dzielimy na stałe i obrotowe. Te drugie często oznaczane są symbolem PT albo PTZ (z ang. *Panning* – panoramowanie, *Tilt* – nachylanie, *Zoom* – przybliżanie).

Część kamer ma oświetlenie działające w podczerwieni, które umożliwia obserwację w nocy. Kamery często mają algorytmy analizy obrazu, aby wykrywać ruch i zapisywać lub przesyłać obrazy tylko w momencie wykrycia ruchu. Wykrywaniem ruchu może też zajmować się urządzenie rejestrujące, które cały czas odbiera i analizuje obraz z kamery. Kamery PTZ można zaprogramować w taki sposób, aby obracały się po określonej ścieżce, pokazując cyklicznie kilka wybranych punktów.

Obraz z kamery – w zależności od modelu – jest zapisywany w chmurze (usługa dostarczona przez producenta kamery) lub na dysku rejestratora znajdującego się w tej samej sieci komputerowej co kamera.



Rys. 9. *Kamera do monitoringu*

3. Urządzenia cyfrowe w domu

3.1. Płyta grzejna i okap kuchenny

Od kiedy w płytach grzejnych zaczęto stosować technologię indukcyjną, płyty te z prostych urządzeń grzewczych zmieniły się w zaawansowane technologicznie urządzenia cyfrowe. Użytkownik wybiera jedynie moc grzania i ustawia wyłącznik czasowy, ale urządzenie, poza cyfrowym generowaniem napięcia w cewce, która odpowiada za grzanie, śledzi obciążenia elementów grzewczych, temperaturę pól grzewczych oraz odpowiada za równomierne obciążenie sieci energetycznej i dopasowuje moc do jej ograniczeń.

Niektóre płyty grzejne wykorzystują dane (aktualna moc grzewcza każdego pola i jego temperatura), aby sterować okapem kuchennym (funkcja **Hob2Hood**). Sterowanie odbywa się w podobny sposób jak w zdalnym sterowaniu telewizorem – w płytę grzejną wbudowany jest nadajnik podczerwieni, a w okap kuchenny – odbiornik.



Rys. 10. *Płyta grzejna i okap*

Na przykład: płyta grzejna włącza prędkość okapu na 2, gdy zupa zaczyna się gotować, prędkość 3, gdy coś smażymy, a gdy po smażeniu wyłączamy płytę, okap jest przełączany do prędkości 1 w celu przewietrzenia pomieszczenia.

3.2. Domowy system audio

Sieciowe odtwarzacze audio

Coraz częściej słuchamy muzyki z radia internetowego czy z serwisów strumieniowych. Najczęściej wykorzystujemy do tego komputer lub smartfon. Są jednak urządzenia, które łączą cechy tradycyjnych zestawów audio z nowoczesnymi funkcjami sieciowymi. Urządzenie różni się od tradycyjnych tym, że ma gniazdo do podłączenia sieci komputerowej i/lub antenę do sieci WiFi.

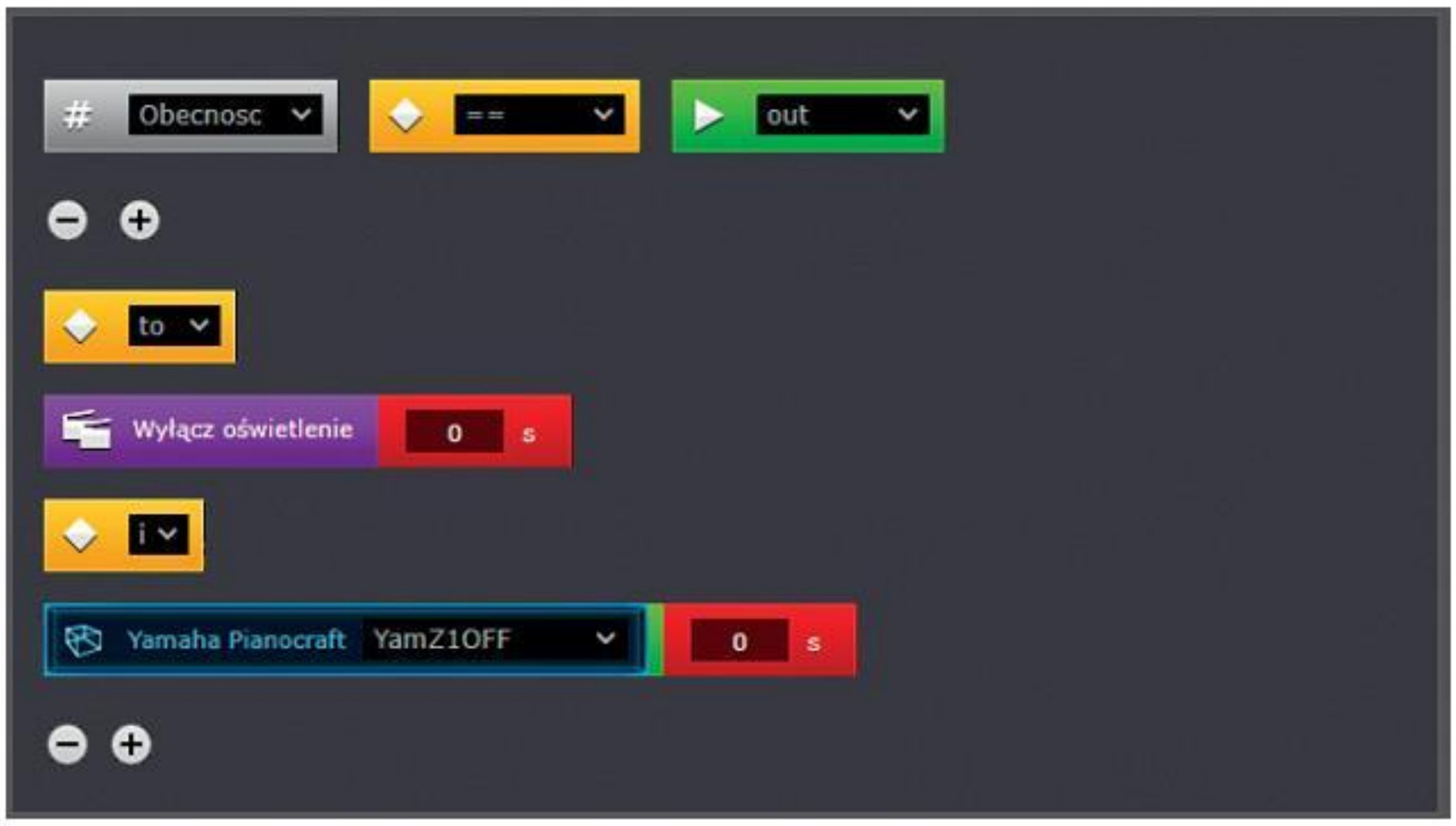
Taki system audio pozwala na słuchanie radia (analogowego i cyfrowego DAB+), płyt CD czy muzyki zapisanej na urządzeniu pendrive lub przesłanej przez Bluetooth oraz na odtwarzanie muzyki z radia internetowego, serwisów strumieniowych (takich jak Tidal, Deezer, Spotify, Qubuz) czy muzyki udostępnionej z domowego komputera lub domowego serwera w standardzie DLNA. Co istotne, tuner z funkcjami sieciowymi nie wymaga innych urządzeń do odtwarzania muzyki.

Nawet jeśli muzykę wybierzemy za pomocą smartfona, to do jej odtworzenia smartfon nie jest już potrzebny – tuner pobiera muzykę bezpośrednio z sieci, działa więc w zasadzie jak pilot zdalnego sterowania, służy do przełączania źródła dźwięku czy zmiany głośności.



Rys. 11. *Miniwieża z funkcjami sieciowymi*

Niektórzy producenci udostępniają też API (z ang. *Application Programming Interface* – interfejs programistyczny aplikacji) pozwalający zintegrować urządzenia z innymi systemami – np. automatyki budynkowej (**Smart Home**).



Rys. 12. *„Scena" w systemie automatyki budynkowej (Smart Home) Fibaro uruchamiana automatycznie przy wyjściu z mieszkania. W ostatnim wierszu widać polecenie wyłączenia systemu audio*

Na rysunku 13. pokazano fragment kodu w języku LUA odpowiadający za wyłączenie miniwieży. Wywołuje on adres urządzenia w lokalnej sieci z odpowiednimi parametrami:
http://192.168.1.200/YamahaExtendedControl/v1/main/setPower?power=standby.

```
local ip _ module = fibaro:get(fibaro:getSelfId(), "IPAddress")
local port = fibaro:get(fibaro:getSelfId(), "TCPPort")
YAM = Net.FHttp(ip _ module, port)
response = YAM:GET('/YamahaExtendedControl/v1/main/
setPower?power=standby')
```

Rys. 13. *Fragment kodu w języku LUA odpowiadający za wyłączenie miniwieży*

System multiroom

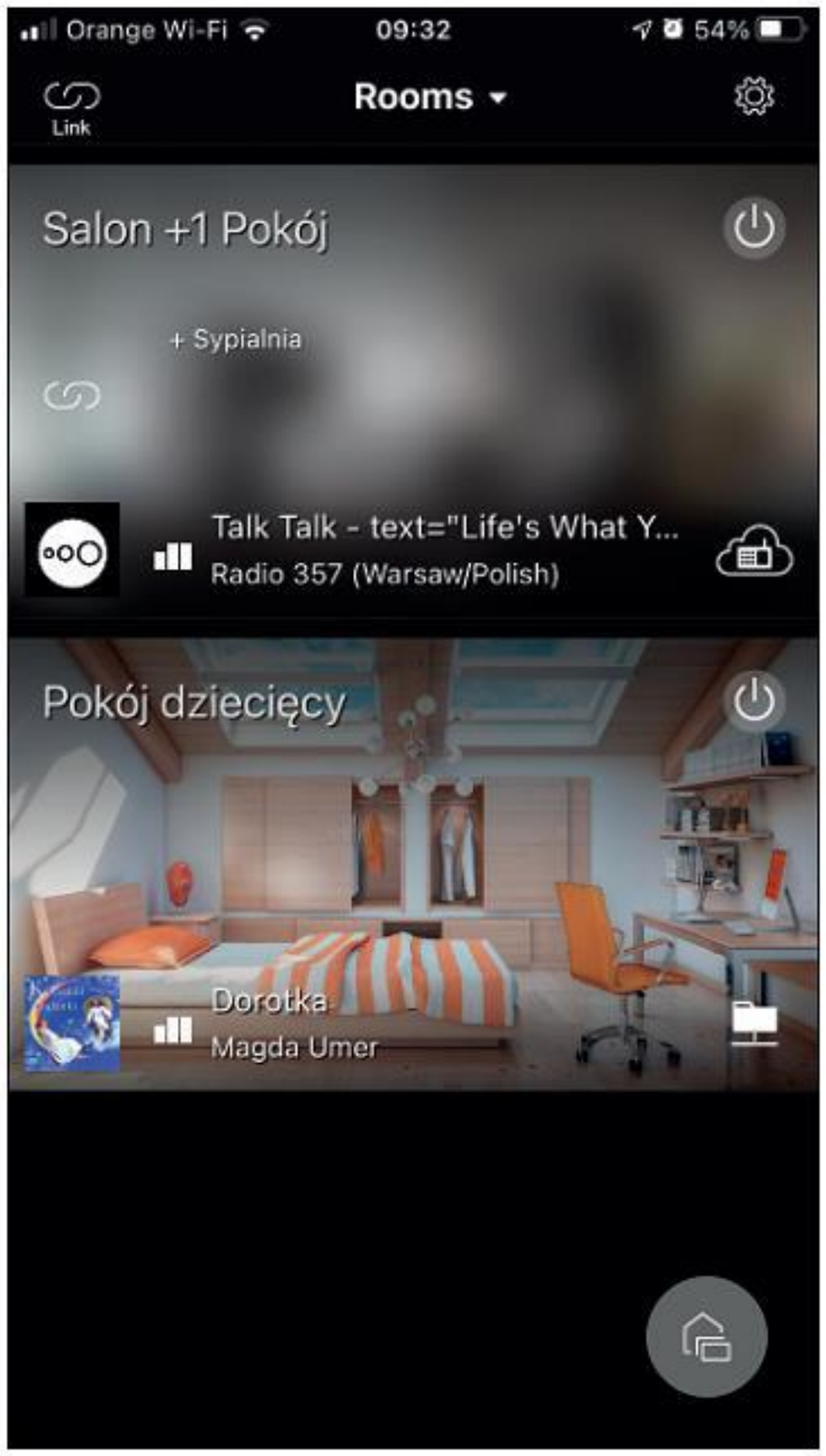
Cyfrowe odtwarzacze audio pozwalają na łączenie wielu urządzeń w jeden system nagłaśniający cały dom czy mieszkanie. Poza dużymi urządzeniami, do których podłączamy zestawy głośnikowe, producenci oferują też „głośniki sieciowe", które łączą w niewielkiej obudowie głośnik i odtwarzacz sieciowy. Występują one też w formie soundbarów.



Rys. 14. *Schemat systemu audio*



Rys. 15. *Głośnik sieciowy*



Rys. 16. *Ekran smartfona z uruchomioną aplikacją do sterowania systemem audio – salon i sypialnię połączono w jedną grupę, dzięki czemu odtwarzana jest tam ta sama muzyka (radio internetowe), w pokoju dziecięcym odtwarzana jest muzyka zapisana na domowym serwerze*

Multiroom (system nagłośnienia wielostrefowego) pozwala na ustalenie, czy w każdym pomieszczeniu słuchamy innej muzyki czy dźwięku z tego samego źródła. Urządzenia możemy dowolnie grupować, korzystając z aplikacji w smartfonie. Możemy też tak skonfigurować dwa pojedyncze głośniki sieciowe, aby współpracowały jako zestaw stereo (jeden z głośników będzie odpowiedzialny za kanał lewy, a drugi za kanał prawy).

Ćwiczenie 5. Szukamy w Internecie więcej informacji na temat wybranego urządzenia nagłaśniającego

Znajdź w Internecie dokumentację API wybranego urządzenia (np. Yamaha Musiccast) i odnajdź polecenie odpowiadające za zwiększenie głośności poprzez sieć lokalną.

3.3. Dysk sieciowy

Dysk sieciowy (NAS – *Network Attached Storage*) to wyspecjalizowany niewielki komputer, w którym instalujemy możliwie duże dyski twarde, aby współdzielić zasoby w domowej czy firmowej sieci. Przestrzeń dyskową można też wykorzystać do przechowywania kopii zapasowej danych z komputera osobistego.

Część zasobów dysku sieciowego możemy udostępnić w Internecie.

Dyski sieciowe oferują takie usługi, jak:

- współdzielenie plików,
- serwer WWW lub serwer baz danych,
- tworzenie albumów zdjęć (rozpoznawanie twarzy, lokalizacji i inne automatyczne albumy),
- serwer multimediów (udostępniający np. za pomocą DLNA zasoby dla telewizora czy amplitunera z funkcjami sieciowymi),
- rejestrator nagrań z kamer monitoringu.



Rys. 17. *Serwer NAS*

Dyski sieciowe mają mechanizmy zabezpieczające dane przed uszkodzeniem – np. RAID (pozwala zachować dane przy uszkodzeniu jednego z dysków) czy zaawansowane narzędzia tworzenia kopii zapasowych. Umożliwiają też utworzenie własnej chmury danych, pozwalającej na synchronizowanie plików na wielu urządzeniach.

Każdy użytkownik NAS ma osobne konto z odpowiednio przydzielonymi uprawnieniami.

4. Motoryzacja i turystyka

4.1. System eCall

W razie wypadku (aktywacja poduszek powietrznych) lub po naciśnięciu przycisku alarmowego urządzenie **eCall** łączy się przy użyciu sieci telefonii komórkowej ze służbami ratunkowymi, przekazując dane o lokalizacji, dane pojazdu (w tym numer VIN, rodzaj paliwa), szacowaną liczbę pasażerów (na podstawie zapiętych pasów), informację, czy zgłoszenie uruchomiono automatycznie czy ręcznie, oraz nawiązuje połączenie głosowe, aby osoby znajdujące się w pojeździe mogły udzielić dodatkowych informacji. Zgłoszenia są odbierane przez centra powiadamiania ratunkowego (te same, które obsługują telefon alarmowy 112).

E **eCall**

Ogólnoeuropejski system szybkiego powiadamiania o wypadkach drogowych.

System eCall nie jest odpowiednikiem „czarnych skrzynek" instalowanych w samolotach i nie rejestruje parametrów podczas jazdy. System jest uruchamiany dopiero w momencie wypadku, nie można więc wykorzystać go do odnalezienia skradzionego pojazdu, nie narusza też naszej prywatności, gdyż nie umożliwia śledzenia pojazdu przez operatora usługi czy służby państwowe.

Urządzenia eCall są obowiązkowo montowane we wszystkich nowych pojazdach osobowych i dostawczych.

4.2. Technologie Apple CarPlay i Android Auto

Apple CarPlay i Android Auto to technologie, które umożliwiają przeniesienie pewnych funkcji telefonu na ekran wbudowany w samochodzie. Dzięki temu mamy dostęp do podstawowych funkcji telefonu w sposób uproszczony, który nie rozprasza uwagi kierowcy.

Umożliwia to wygodne korzystanie np. z map. Dostępne są również połączenia telefoniczne (wykorzystuje się zestaw głośnomówiący wbudowany w samochód), wiadomości oraz serwisy z muzyką i podcastami.

Niektóre samochody wyświetlają informacje na ekranie głównym w środkowej części samochodu, a także na ekranie znajdującym się nad kierownicą.



Rys. 18. *Apple CarPlay*



Rys. 19. *Android Auto*

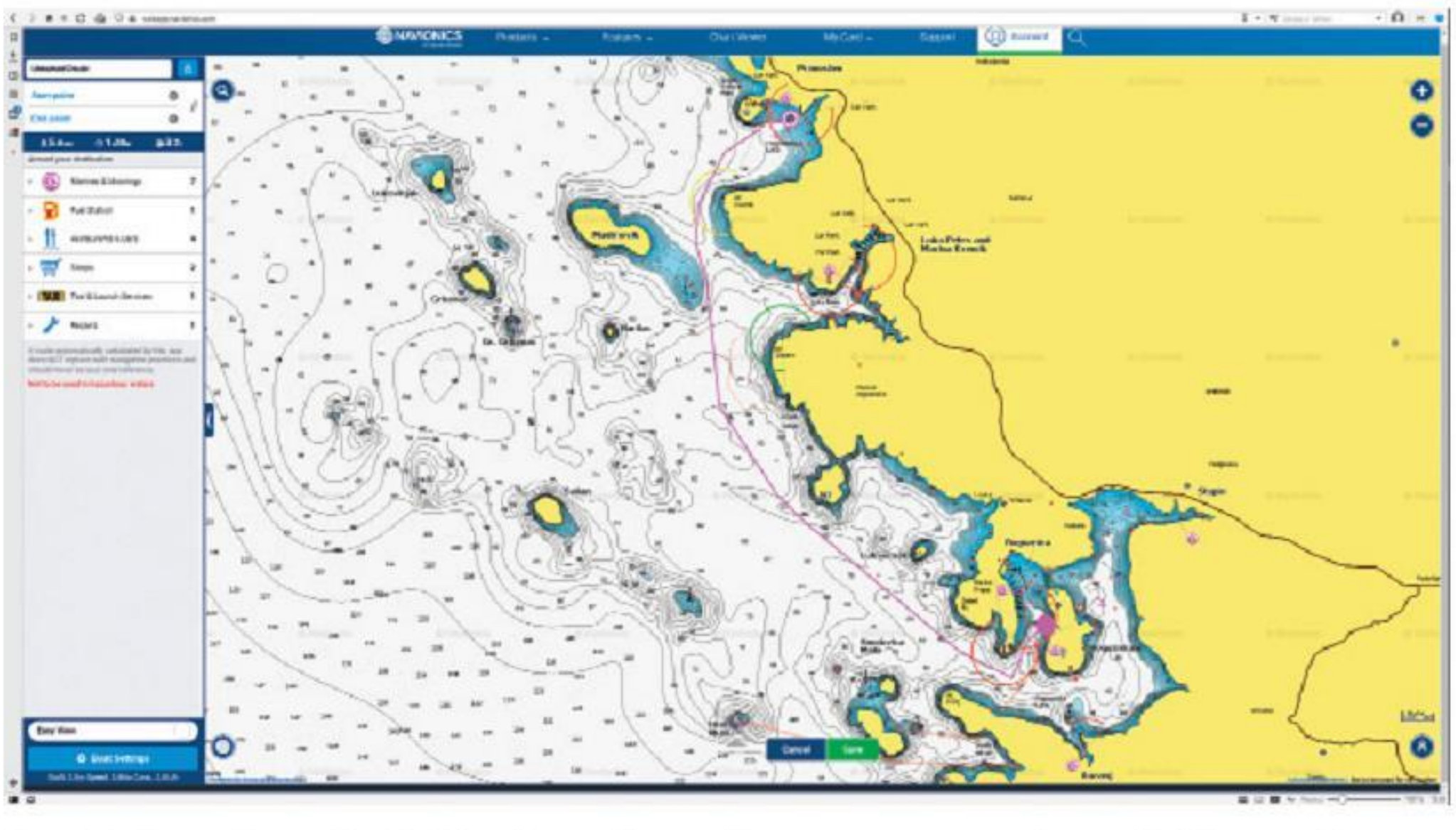
4.3. Systemy nawigacji

Obecnie najczęściej wykorzystywanymi systemami określania pozycji geograficznej są systemy satelitarne takie jak GPS, GLONASS czy Galileo. Warto jednak wspomnieć, że w lotnictwie, gdzie nawigacja musi być niezależna od urządzeń zewnętrznych (jak satelity), stosuje się nawigację bezwładnościową – do komputera nawigacyjnego wprowadza się pozycję startową, a komputer na podstawie czujników przeciążeń (np. siła odśrodkowa przy pokonywaniu zakrętu) określa ruch samolotu i wylicza aktualną pozycję z dokładnością pozwalającą dotrzeć do celu.

Proste urządzenia do nawigacji turystycznej wyświetlają jedynie naszą pozycję na mapie, nie umożliwiając planowania tras. Bardziej zaawansowane urządzenia (czy aplikacje) pozwalają nam zaplanować najbardziej optymalną drogę do celu z uwzględnieniem specyfiki podróży (inna trasa zostanie wybrana dla pieszego turysty, inna dla samochodu osobowego, a inna dla ciężarówki).

W nawigacji samochodowej przy planowaniu trasy mogą być brane pod uwagę informacje o korkach (odebrane przez Internet), a trasa może być aktualizowana w czasie podróży na podstawie przesyłanych na bieżąco nowych danych. Informacje o naszej podróży są przesyłane do serwera, w celu analizy natężenia ruchu na poszczególnych drogach i powiadamiania innych użytkowników o korkach. W bardziej zaawansowanych urządzeniach nawigację satelitarną uzupełnia nawigacja bezwładnościowa, aby nasza aktualna pozycja wyświetlała się także wtedy, gdy jedziemy przez długi tunel i nie dociera do nas sygnał z satelitów.

W przypadku nawigacji morskiej, gdzie nie ma dróg, urządzenia nawigacyjne wyznaczają trasy po liniach prostych z uwzględnieniem głębokości akwenu i zanurzenia jednostki (aby ominąć płycizny). Nawigacja omija obszary, gdzie żegluga jest zakazana. W przypadku dłuższych odcinków (pokonywanie oceanu) kurs (kierunek względem bieguna) jest zmienny, ponieważ najkrótsza trasa po powierzchni kuli to ortodroma. Na płaskiej mapie taką najkrótszą trasę zobaczymy w postaci krzywej wygiętej w kierunku bliższego bieguna ziemi.



Rys. 20. *Trasa z Rogoznicy do Primošten w Chorwacji wyznaczona na mapie morskiej. Zanurzenie jednostki ustalono na 2 metry*

Smartfony i nowoczesne aparaty fotograficzne wykorzystują dane lokalizacyjne do oznaczenia miejsca wykonania zdjęcia – informacje te zapisują się w metadanych EXIF.

Ćwiczenie 6. Szukamy informacji na temat innych systemów nawigacji

Znajdź w Internecie informacje na temat systemów nawigacji lotniczej.
Przedstaw porównanie poszczególnych systemów nawigacji w tabeli. Samodzielnie ustal kryteria porównania.

Wskazówka: Ćwiczenie można wykonać w grupach, przydzielając każde zagadnienie innej osobie (lub osobom).

4.4. Lokalizatory

Coraz bardziej popularnymi urządzeniami są lokalizatory, które za pomocą technologii GPS i GSM lokalizują interesujące nas obiekty. Używa się ich m.in. w przedsiębiorstwach transportowych do lokalizacji pojazdów – często są to urządzenia podłączone do złącza diagnostycznego ODB2, co pozwala im pobierać informacje dotyczące pojazdu (np. poziom paliwa). Zarządzający flotą widzi wszystkie informacje o pojazdach naniesione na mapę, może też ustawić powiadomienia, gdy pojazd przekroczy prędkość albo wyjedzie poza wyznaczoną strefę. Lokalizator możemy też umieścić we własnym pojeździe, aby mieć szansę odnaleźć pojazd w przypadku kradzieży.

Lokalizatory wysyłają dane o pozycji do systemu informatycznego. To, co różni oferty poszczególnych firm, to głównie sposób wizualizacji odbieranych danych, możliwość ustawiania powiadomień i tworzenia zestawień.



Rys. 21. *Pies z lokalizatorem GPS*

5. Smartwatch

Smartwatch to elektroniczne urządzenie noszone na nadgarstku, przypominające zegarek. Urządzenie komunikuje się z naszym smartfonem. Dzięki temu, że mamy je zawsze przy sobie, nie musimy sięgać po telefon, aby zobaczyć, kto dzwoni czy odczytać otrzymaną wiadomość. Smartwatche monitorują też stale naszą aktywność, podczas gdy telefon czasami odkładamy w innym pomieszczeniu.

Zakres funkcji oferowanych przez smartwatche jest bardzo różny. Możliwości smartwatcha zależą od ceny, a różne modele odpowiadają odmiennym potrzebom użytkowników. Niektóre urządzenia służą głównie do komunikacji i pracy (informacja o połączeniach, wiadomości, kalendarz), inne wspomagają uprawianie sportu (statystyki aktywności, możliwość podłączenia zewnętrznych urządzeń jak np. zewnętrzny czujnik tętna), jeszcze inne mają ograniczoną funkcjonalność, za to są podobne do tradycyjnych zegarków. Modele przeznaczone dla turystów mogą mieć wbudowany odbiornik GPS czy czujnik ciśnienia atmosferycznego.

Smartwatche na podstawie naszych ruchów mogą określać liczbę wykonanych kroków czy np. przypominać o potrzebie ruchu, gdy zbyt długo siedzimy przy komputerze. Mogą też monitorować jakość snu. Wiele urządzeń stale mierzy nasze tętno i saturację krwi oraz szacuje ciśnienie krwi lub analizuje pracę serca (podobnie do EKG) i informuje, gdy dzieje się coś niepokojącego, a nawet sugeruje wizytę u lekarza.

Zestawienia zebranych danych możemy oglądać w odpowiednich aplikacjach w telefonie.

Urządzenia komunikują się ze smartfonem głównie za pomocą Bluetooth (jeśli były poza zasięgiem, to aktualizują dane o naszej aktywności w telefonie po przywróceniu połączenia). Niektóre smartwatche wykorzystują komunikację WiFi lub GSM (możemy wtedy odbierać połączenia, nie mając przy sobie telefonu).



Rys. 22. *Smartwatch*

Warto zapamiętać

- Drukarki 3D pozwalają drukować modele przestrzenne poprzez nanoszenie kolejnych warstw specjalnego materiału metodą przyrostową.
- Aby wydrukować model 3D, należy go wcześniej zaprojektować w edytorze grafiki 3D i zapisać w pliku z rozszerzeniem *stl* lub *g-code*.
- Tablica interaktywna współpracuje z komputerem i projektorem multimedialnym. Monitor interaktywny nie potrzebuje projektora, aby wyświetlić obraz.
- Monitory prezentacyjne są umieszczane w różnych miejscach i służą do wyświetlania informacji, które aktualizuje się zdalnie przez sieć komputerową.
- Kamery do monitoringu wizyjnego instaluje się głównie w celu kontroli i zapewnienia bezpieczeństwa na określonym obszarze.
- Do urządzeń cyfrowych stosowanych w gospodarstwach domowych zaliczamy m.in. pralkę, czajnik, płytę grzejną, okap kuchenny, domowy system audio.
- System eCall w razie wypadku umożliwia przekazanie służbom ratunkowym danych o lokalizacji, danych pojazdu, danych o liczbie pasażerów itp.
- Smartwatch to elektroniczne urządzenie noszone na nadgarstku, przypominające zegarek, które komunikuje się z naszym smartfonem.

Zadania

Uwaga: Zadania można wykonać w grupach, przydzielając każde zagadnienie innej osobie (lub osobom). Następnie można w dyskusji podzielić się wiedzą.

1. W temacie podzieliliśmy urządzenia na te, których używamy w szkole, domu, w motoryzacji i turystyce. Zastanów się nad innym podziałem urządzeń wykorzystujących technologie cyfrowe. Podaj przykłady.
   Wskazówka: Podział może dotyczyć m.in. sposobów zastosowania urządzeń.
2. Wyjaśnij szerzej następujące zagadnienia (znajdź informacje w Internecie lub innych źródłach).
   a. możliwość drukowania w polu roboczym kilku modeli naraz tym samym materiałem.
   b. drukarki 3D, które mają dwie dysze (dwa ekstrudery) i mogą drukować model dwoma różnymi materiałami.
   c. udostępnianie przez producentów urządzeń plików *stl* ze wzorami części zamiennych, które można samemu wydrukować w razie awarii urządzenia.
3. Podaj nieomówione w tym temacie zastosowania urządzeń cyfrowych w domu.
4. Przedstaw w wybranej formie genezę cyfrowych systemów nawigacji.

Dla zainteresowanych

5. W 1999 roku po raz pierwszy użyto terminu „Internet rzeczy" (ang. *Internet of Things*, *IoT*). Określono tak przedmioty (ang. „*things*"), które mają wbudowane czujniki, mikrokontrolery, oprogramowanie i inne technologie służące do łączności i wymiany danych z urządzeniami lub systemami poprzez sieć Internet. Internet rzeczy jest stosowany w sprzętach AGD, w medycynie (zdalna diagnostyka), automatyce budynkowej i wielu urządzeniach. Przygotuj w wybranej formie opracowanie na temat Internetu rzeczy.

Wokół dokumentów komputerowych

B

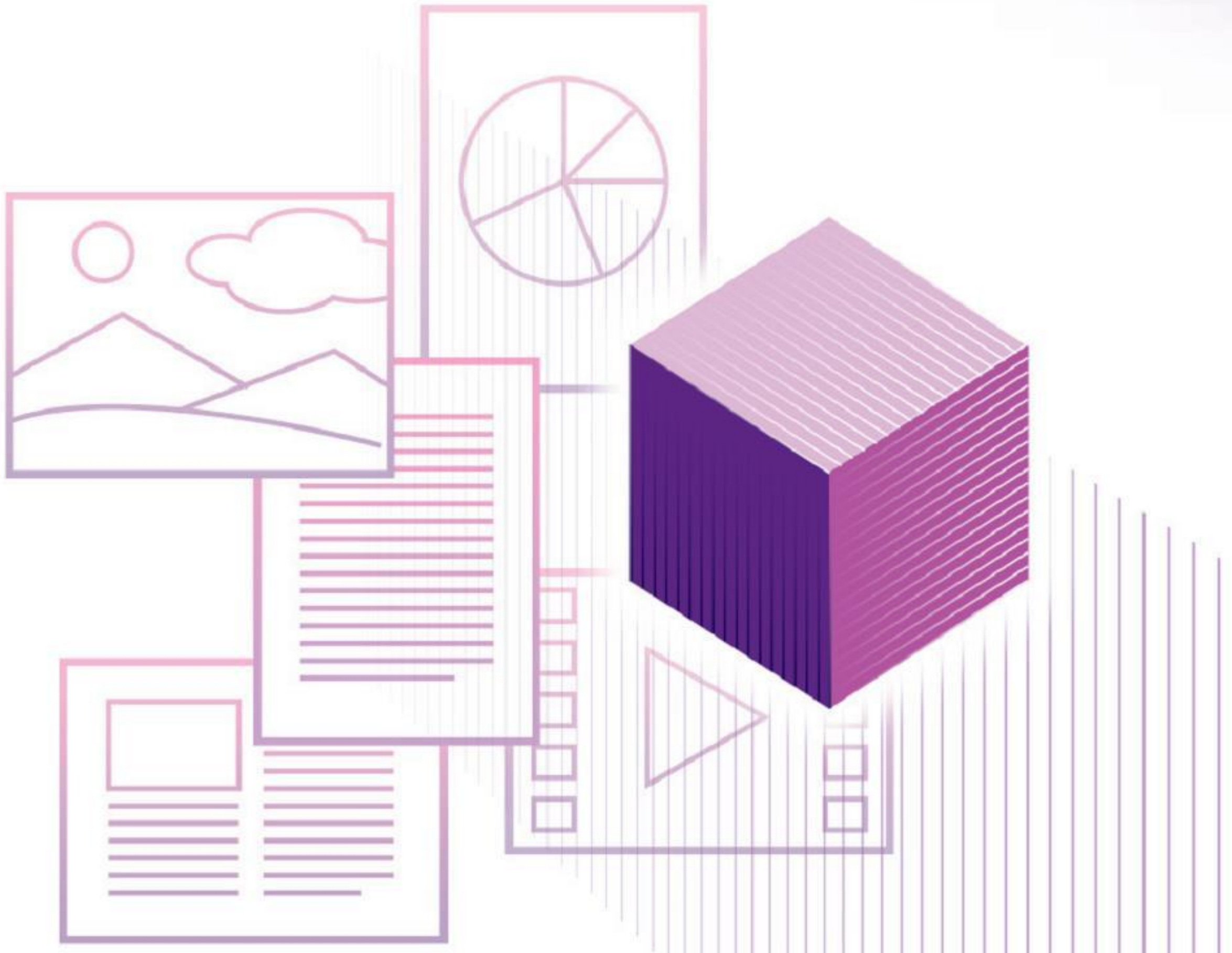
Moduł

Temat B1. Tworzenie bazy danych

Temat B2. Tworzenie i stosowanie formularzy w relacyjnej bazie danych

Temat B3. Wykonywanie podstawowych operacji na relacyjnej bazie danych

Temat B4. Projektowanie modeli dwuwymiarowych i trójwymiarowych



Temat B1. Tworzenie baz danych

1. W gąszczu informacji…
2. Podstawowe pojęcia związane z bazami danych
3. Etapy tworzenia bazy danych
4. Przykład bazy danych – obsługa szkolnych zawodów pływackich
   - 4.1. Określenie wymagań bazy danych *Zawody*
   - 4.2. Wstępne projektowanie formularzy i raportów w bazie danych *Zawody*
   - 4.3. Przygotowanie tabel w bazie danych *Zawody*
   - 4.4. Definiowanie relacji w bazie danych *Zawody*
5. Tworzenie bazy danych – obsługa gabinetu stomatologicznego
   - 5.1. Określenie wymagań bazy danych *Stomatolog*
   - 5.2. Wstępne projektowanie formularzy i raportów w bazie danych *Stomatolog*
   - 5.3. Tworzenie tabel w bazie danych *Stomatolog*
   - 5.4. Definiowanie relacji w bazie danych *Stomatolog*

Warto powtórzyć

1. Jak zbudowana jest tabela arkusza kalkulacyjnego?
2. Podaj przykładowe operacje, które wykonywaliśmy na danych zgromadzonych w tabeli arkusza kalkulacyjnego.

1. W gąszczu informacji…

Ludzie od wieków stosowali różne formy przechowywania informacji, np. księgi, rejestry i katalogi. Średniowieczni kupcy (rys. 1a) zapisywali w księgach handlowych należności i zobowiązania, z czasem również ewidencjonowali gotówkę, weksle, towary itp. Jeszcze na przełomie XX i XXI wieku numery telefonów umieszczano w papierowych książkach telefonicznych (rys. 1b). Szukano w nich numeru telefonu do apteki, sklepu, fryzjera, kina, koleżanki lub kolegi z wakacji. W bibliotekach do dziś korzysta się z papierowych katalogów książek (rys. 1c), ułatwiających dotarcie do poszukiwanego tytułu.

Informacji ciągle przybywa, a nieuporządkowane stają się nieprzydatne, niespójne i przypadkowe. Aby nie pogubić się w ich gąszczu, wymyślano różne sposoby porządkowania, np. według kolejności alfabetycznej, dat, cen artykułów, wartości dostaw. Poza tym wybrane informacje wyróżniano i łączono te, które mają ze sobą coś wspólnego, np. nazwisko i imię klienta z kwotą należności i datą zapłaty, nazwę zakładu fryzjerskiego z numerem telefonu i adresem, nazwisko i imię autora z tytułem książki.

a.



b.



c.



Rys. 1. *Przykłady sposobów gromadzenia informacji na przestrzeni wieków*

Po pojawieniu się komputerów zauważono, że zawartość rejestrów, ksiąg i katalogów można przenieść do pamięci komputera, dzięki czemu korzystanie z informacji stało się wygodniejsze, efektywniejsze i szybsze.

Miejsca i formy przechowywania informacji zmieniły się. Nie zmieniły się jednak główne cele, dla których niegdyś tworzono tradycyjne spisy, czyli **przetwarzanie informacji**.

2. Podstawowe pojęcia związane z bazami danych

Aby informacje mogły być **przetwarzane** w systemie informatycznym, czyli stały się **danymi** dla komputera, muszą zostać zakodowane w formie zapisu elektronicznego, w określonej przez dany system postaci – czyli **odpowiednio zorganizowane**.



Przetwarzanie danych to uporządkowane wykonywanie operacji na danych, m.in.:

- wprowadzenie,
- modyfikowanie (poprawianie, uzupełnianie, aktualizowanie),
- wyszukiwanie,
- porządkowanie (sortowanie),
- prezentowanie (wyświetlanie, drukowanie).

Baza danych B

Zbiór danych, czyli odpowiednio zorganizowanych informacji, które można przetwarzać za pomocą programów.

Uwaga !

Pojęcia **baza danych** używamy często także w odniesieniu do oprogramowania umożliwiającego tworzenie baz danych i przetwarzanie danych.

Bazy danych znalazły zastosowanie w prawie każdej dziedzinie życia. Są wykorzystywane na przykład w takich miejscach jak: sekretariat, sklep, apteka, magazyn, hurtownia, kino, teatr, bank, poczta, wypożyczalnia filmów, gabinet lekarski, obiekty sportowe.

Korzystając z wybranej bazy danych, użytkownik komputera może wprowadzać do systemu komputerowego różne informacje (np. o uczniach, zawodnikach, lekach, książkach, produktach), które po wprowadzeniu do komputera (czyli zapisaniu w postaci cyfrowej) stają się danymi, a następnie mogą być przetwarzane.

Przetwarzanie danych stało się również podstawą funkcjonowania wyszukiwarek internetowych. Wyświetlana przez wyszukiwarkę lista łączy do stron WWW zawierających poszukiwane treści jest efektem przeszukiwania gigantycznej bazy danych, zgromadzonej na serwerach wyszukiwarki.

Ćwiczenie 1. Przykład bazy danych z naszego otoczenia

Podaj przykład bazy danych ze swojego otoczenia, m.in. ze szkoły. Dowiedz się, jakie informacje są wprowadzane do takiej bazy i jakie otrzymuje się wyniki przetwarzania danych.

Dane w relacyjnej bazie danych **są zorganizowane w tabelach** reprezentujących logiczne powiązanie danych (**relacje**).
Tabela składa się z **wierszy** (zwanych **rekordami**) i **kolumn** (zwanych **polami**).
W jednym wierszu (rekordzie) znajdują się informacje o jednym obiekcie bazy danych.

3. Etapy tworzenia bazy danych

Istotnym etapem tworzenia bazy danych jest określenie celu bazy danych i wymagań, jakie powinna spełniać. Podczas tworzenia profesjonalnych systemów analitycy systemowi zbierają informacje o środowisku, dla którego będą projektowali system informatyczny, m.in. przeprowadzają wywiady, zapoznają się z zasadami funkcjonowania wybranego działu przedsiębiorstwa, analizują charakter i typ produkcji, przeglądają istniejące formularze, raporty i inne dokumenty.

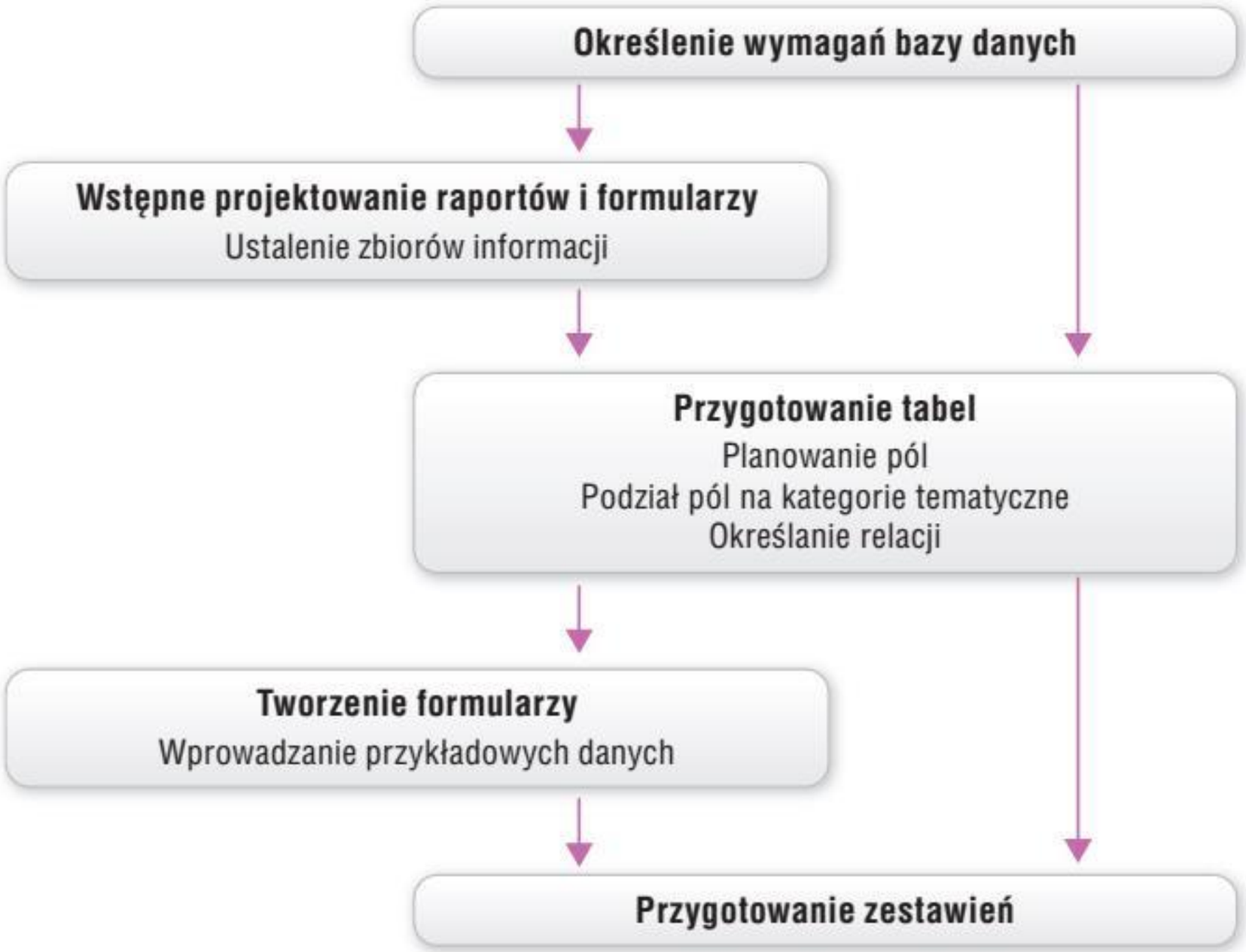
Aby prawidłowo (zgodnie z oczekiwaniami zleceniodawcy) zaprojektować bazę danych, należy wiedzieć nie tylko, jakie dane mają być gromadzone w bazie, ale również, jak mają być przetwarzane i co mamy otrzymać w wyniku ich przetwarzania.

Przykładową relacyjną bazę danych i sposób tworzenia nowej bazy pokażemy, wykorzystując program Microsoft Access. Profesjonalne bazy danych są najczęściej tworzone z wykorzystaniem innego oprogramowania, np. Microsoft SQL Server lub MySQL, jednak podstawowe zasady tworzenia baz danych są uniwersalne.

Baza danych tworzona w programie Microsoft Access może zawierać następujące obiekty:

- tabele – do przechowywania danych,
- formularze – do wprowadzania i przeglądania danych,
- zapytania (kwerendy) – do wyszukiwania danych spełniających określone kryterium,
- raporty – do przedstawiania (wyświetlania na ekranie, drukowania) danych zawartych w tabelach i otrzymanych w wyniku działania kwerendy.

Wszystkie obiekty i dane umieszczone w bazie programu Microsoft Access są zapisywane w jednym pliku.



Rys. 2. *Schemat przygotowania prostej relacyjnej bazy danych*

4. Przykład bazy danych – obsługa szkolnych zawodów pływackich

Zawody pływackie odbywały się także w czasach, gdy nie używano jeszcze komputerów. Stosowano wtedy tradycyjne sposoby tworzenia list zawodników, zapisywania czasów przepłynięcia danego dystansu, a także układania listy zawodników (np. według najlepszych czasów w danej konkurencji).

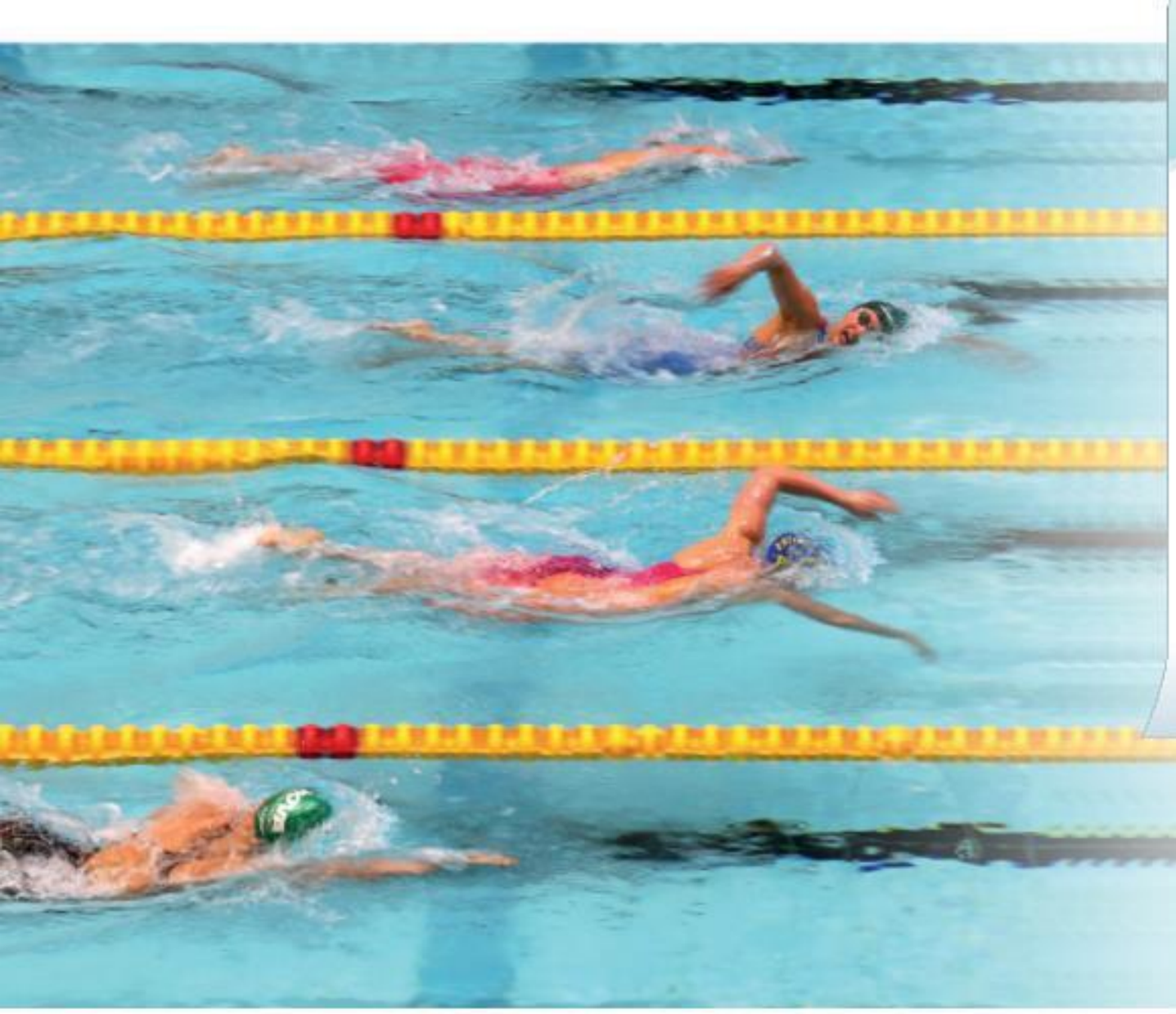
Pojawienie się komputerów i odpowiednich programów umożliwiło m.in. gromadzenie w wersji elektronicznej danych o zawodnikach i ich startach (np. w tabelach bazy danych) oraz przetwarzanie danych i otrzymywanie różnych zestawień.

Poszczególne etapy przygotowania bazy danych pokażemy na przykładzie tworzenia prostej bazy *Zawody*, której celem będzie przetwarzanie danych o zawodnikach i ich startach w szkolnych zawodach pływackich.

Pokażemy, jak umożliwić obsługę zawodów pływackich z wykorzystaniem metod informatyki. Zadanie (tu: wymagania bazy danych) musimy więc tak sformułować, aby (zgodnie z myśleniem komputacyjnym) do jego rozwiązania można było wykorzystać komputer.

4.1. Określenie wymagań bazy danych *Zawody*

Przed utworzeniem komputerowej bazy danych dotyczącej zawodów pływackich musimy poznać zasady ich przeprowadzania i dopiero na tej podstawie określić wymagania bazy danych. Należy ustalić, jakie informacje będą wprowadzane do bazy i w niej gromadzone (czyli „wejście") oraz jakie zestawienia chcemy uzyskać z bazy danych (czyli „wyjście").



KARTA STARTOWA ZAWODNIKA

MIKOŁAJKOWE ZAWODY PŁYWACKIE

Data: .....

Nazwisko i imię zawodnika:

.....

Klasa: .....

Konkurencja: .....

.....

Wynik

Rys. 3. *Przykładowa karta startowa uczestnika szkolnych zawodów pływackich*

Podczas zawodów każdemu zawodnikowi mierzy się czas przepłynięcia danego dystansu. Sędzia pływacki po zmierzeniu czasu stoperem zapisuje wynik na karcie startowej zawodnika. Gdy wszyscy pływacy przepłyną dany dystans, wyniki trzeba posortować według czasów, aby uzyskać informację o lokatach zawodników.

Na rysunku 3. pokazujemy przykładową kartę zawodnika, którą można przygotować na szkolne zawody pływackie. Często są na niej również umieszczone takie informacje, jak wiek zawodnika, nazwa klubu sportowego, nazwa szkoły (w przypadku uczestników z różnych klubów lub szkół).

Na profesjonalnych zawodach czas mierzy się w sposób automatyczny za pomocą tablic dotykowych zainstalowanych na torach. Czasy zawodników są wtedy automatycznie wprowadzane do komputera.

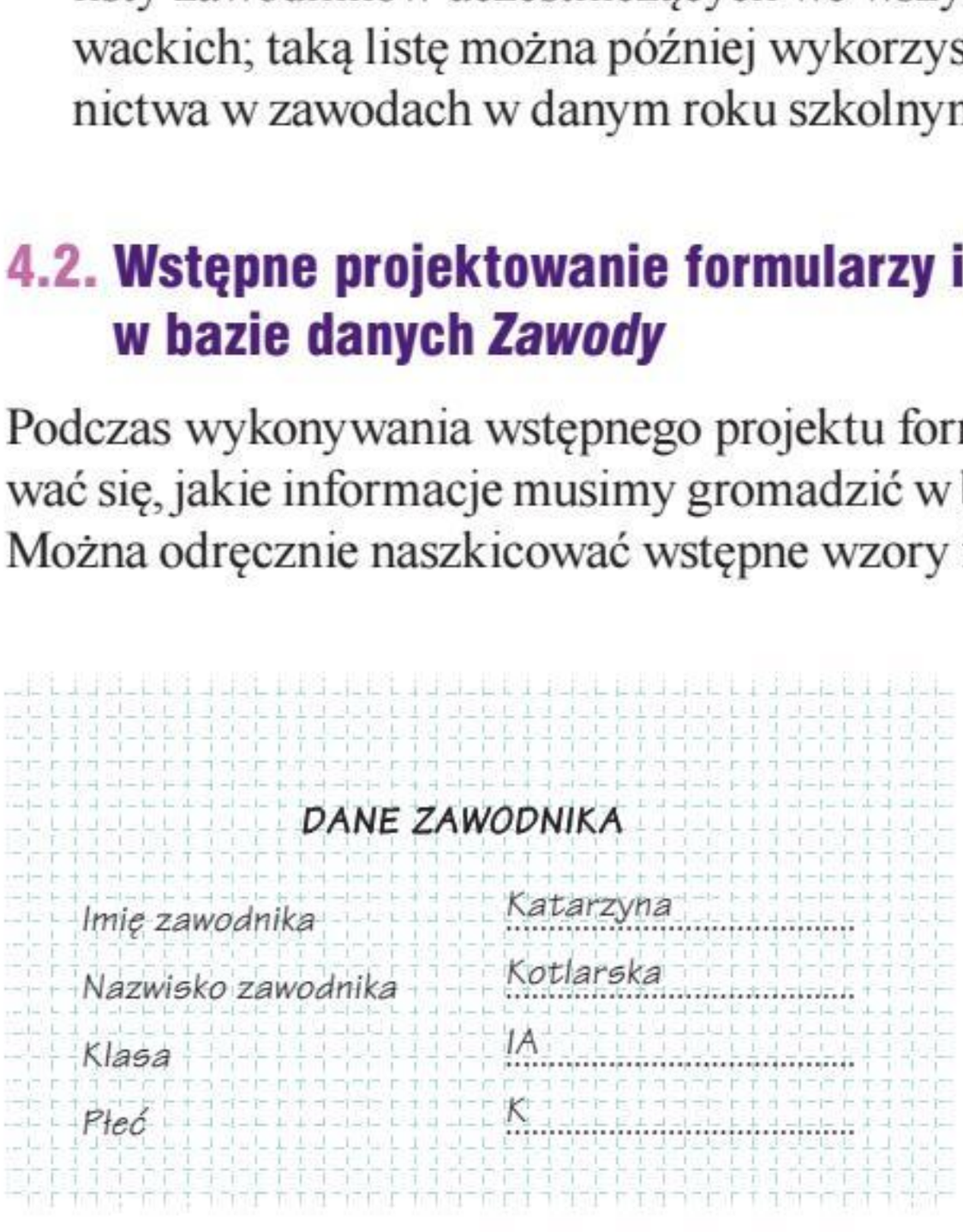
Gdy na zawodach, na przykład szkolnych, nie używa się tablic dotykowych, ktoś musi najpierw wprowadzić informacje o czasach z kart startowych zawodników do komputera. Dopiero wtedy program będzie mógł je przetwarzać, dzięki czemu zostaną wyłonieni zwycięzcy.

W bazie danych *Zawody* chcemy gromadzić dane o zawodnikach i ich startach oraz otrzymać m.in.:

- kolejność zawodników (od najlepszego czasu) w każdej konkurencji (oddzielnie dla dziewcząt i chłopców) wraz z czasami, m.in. w celu wyłonienia najlepszych trzech pływaków, którzy staną na podium;
- listę zawodników z danych zawodów wraz ze wszystkimi konkurencjami, w których startowali i czasami, które osiągnęli; takie zestawienie tworzy się, aby zawodnicy mogli w jednym miejscu zobaczyć wszystkie swoje wyniki; może też ono służyć do wybrania najlepszego zawodnika/zawodniczki danych zawodów;
- listy zawodników uczestniczących we wszystkich organizowanych zawodach pływackich; taką listę można później wykorzystać do wysłania zaświadczenia uczestnictwa w zawodach w danym roku szkolnym.

4.2. Wstępne projektowanie formularzy i raportów w bazie danych *Zawody*

Podczas wykonywania wstępnego projektu formularzy i raportów, możemy zorientować się, jakie informacje musimy gromadzić w bazie i jakie umieścimy na wydrukach. Można odręcznie naszkicować wstępne wzory formularza i przykładowych raportów.


DANE ZAWODNIKA

Imię zawodnika ..... Katarzyna .....

Nazwisko zawodnika ..... Kotlarska .....

Klasa ..... IA .....

Płeć ..... K .....

Rys. 4. *Wstępny „odręczny" projekt formularza*

MIKOŁAJKOWE ZAWODOWE PŁYWACKIE

Wrocław, 6 grudnia 2021

Nazwisko i imię	Klasa	Styl	Dystans	Czas
Katarzyna Kotlarska	IA			
		dowolny	50	30.21
		klasyczny	50	36.45
		motylkowy	50	45.25
Wojciech Kalarus	IB			
		grzbietowy	50	52.44
		klasyczny	50	36.45

Rys. 5. *Wstępny „odręczny" projekt raportu – lista zawodników z danych zawodów ze wszystkimi konkurencjami i czasami, które osiągnęli na zawodach*

Uwzględniając wymagania systemu oraz zaplanowane formularze i raporty, dzielimy informacje, które powinny być wprowadzane bądź wyprowadzane, na dwie kategorie.

Informacje o zawodnikach	Informacje o startach
imię	data zawodów
nazwisko	dystans
klasa	styl
płeć	uzyskany czas

Tabela 1. *Przykładowy podział informacji na kategorie*

4.3. Przygotowanie tabel w bazie danych *Zawody*

Na podstawie informacji z tabeli 1. (uwzględniających wymagania systemu) planujemy pola, które są potrzebne do przechowywania odpowiednich danych. Dla poszczególnych pól określamy typy danych i ewentualnie inne właściwości. Liczba tabel wynika z kategorii informacji, określonych w tabeli 1.



Szczegółowe informacje odpowiadają **polom** w bazie danych (**kolumnom** w tabeli), a grupy tematyczne, których dotyczą informacje – **tabelom**. Każda kolumna (każde pole) ma unikalną w ramach tabeli nazwę.

Pola łączymy w grupy tematyczne, odpowiadające kategoriom informacji określonym w tabeli 1. Każda grupa będzie stanowiła oddzielną tabelę. Określimy tym samym, do których tabel należą poszczególne pola.

W bazie *Zawody* powinniśmy utworzyć dwie tabele:

- *Zawodnicy* – do zapisywania informacji o zawodnikach biorących udział w szkolnych zawodach pływackich,
- *Starty* – do zapisywania informacji o startach zawodników.

Pola tabeli *Zawodnicy*	Pola tabeli *Starty*
Imie	*DataZawodow*
Nazwisko	*Dystans*
Klasa	*Styl*
Plec	*Czas*

Tabela 2. *Przykładowy przydział pól do tabel bazy danych*

Zasada	Opis	Przykład
Należy unikać powtarzania tych samych informacji w kilku tabelach.	Jeśli umieścimy tę samą informację w kilku tabelach, każdą zmianę trzeba będzie nanosić w kilku miejscach.	Jeśli umieścimy informacje o adresie danej osoby (np. zawodnika, pacjenta, klienta) w dwóch tabelach, każdą zmianę adresu trzeba będzie nanosić w obydwu miejscach.
W każdej kolumnie powinno się umieszczać pojedynczą informację.	Gdy umieścimy imię i nazwisko w oddzielnych kolumnach, ułatwione będzie na przykład porządkowanie tych danych według nazwiska.	W przypadku umieszczania nazwiska i imienia zawodnika w jednym polu, mogłoby dochodzić do sytuacji, w której użytkownik wpisze odwrotnie: najpierw imię, a potem nazwisko.
Nazwy pól i tabel powinny odzwierciedlać ich zawartość. Warto przyjąć dla całej bazy jednolitą konwencję nazewnictwa wszystkich obiektów bazy.	Możemy przyjąć, że nazwy pól i obiektów bazy zaczynamy wielką literą, nie używamy polskich znaków diakrytycznych i spacji, a każdy człon nazwy będzie zaczynał się wielką literą.	Nazwy pól: *DataZawodow, DataUrodzenia*, *Nazwisko*. Tabele: *Zawodnicy, Klienci, Pacjenci, Adresy*, *Zamowienia*.

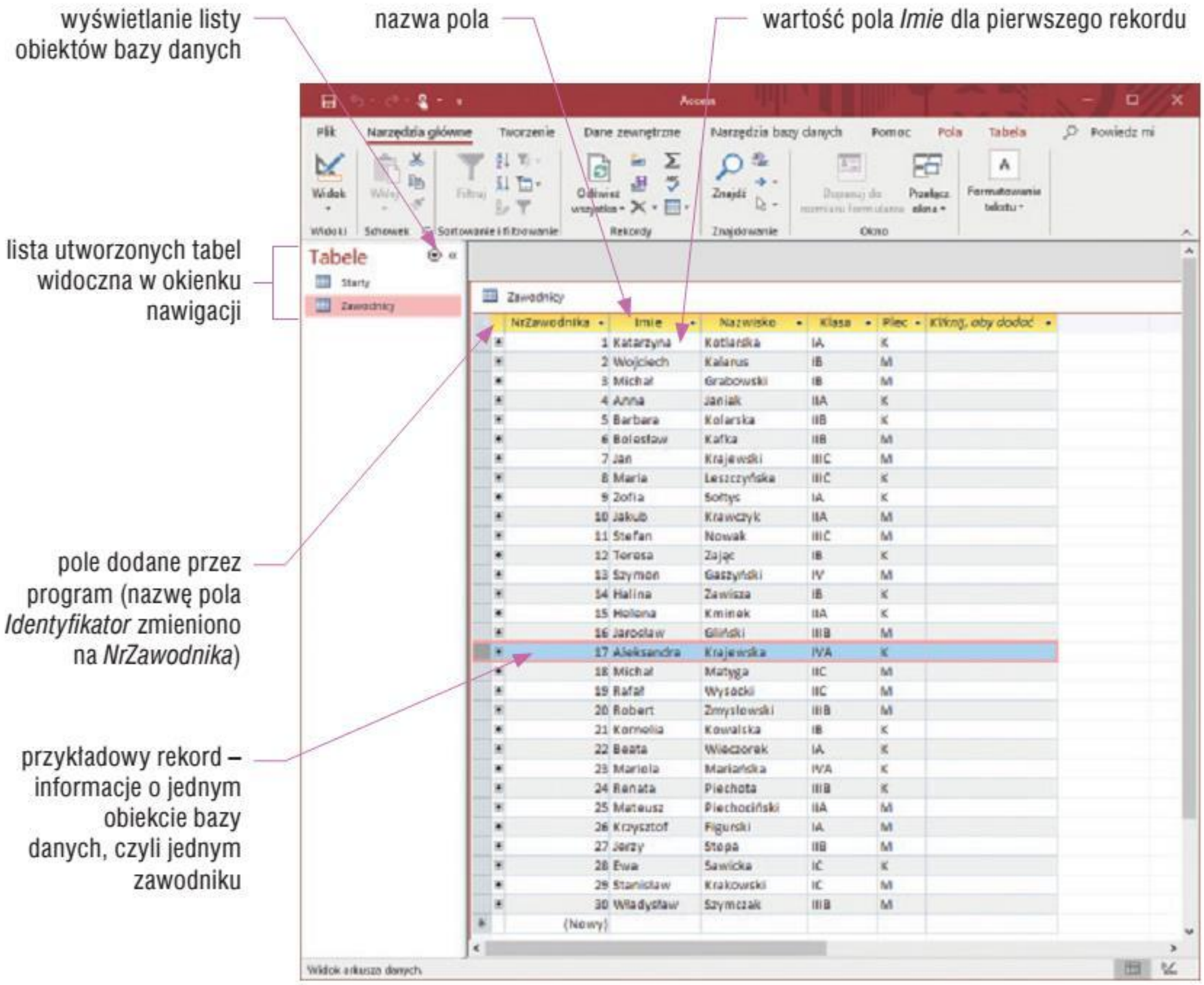
Tabela 3. *Zasady tworzenia tabel*

Ćwiczenie 2. Przeglądamy tabele przykładowej bazy danych

1. W pliku, który otrzymasz od nauczyciela, utworzono bazę danych *Zawody*, zawierającą dwie tabele: *Zawodnicy* i *Starty*. W okienku nawigacji wyświetl nazwy tabel. Opowiedz, jakie informacje zawarto w tych tabelach.
2. Podaj nazwy pól i przykładowe wartości zapisane w rekordach wyróżnionych na rysunkach 6. i 7.
3. Odpowiedz ustnie na pytania:
   a. Czy zostały spełnione zasady tworzenia tabel podane w tabeli 3.? Uzasadnij odpowiedź.
   b. Jaką konwencję nazywania obiektów przyjęto w tej bazie?

Uwaga

Ze względu na ograniczoną liczbę torów na basenie starty w danej konkurencji odbywają się zazwyczaj w tzw. seriach – w jednej serii maksymalnie tylu zawodników, ile jest torów. Dla uproszczenia w naszej bazie nie uwzględniamy serii.



Rys. 6. *Tabela* Zawodnicy *otwarta w programie Microsoft Access w* ***Widoku arkusza danych*** *– tabela zawiera fikcyjne dane*

Rekordy zawierające dane o startach zawodnika nr 1

Starty

NrZawodnika	DataZawodow	Styl	Dystans	Czas
1	2020-12-06	dowolny	50	30,21
1	2021-03-21	klasyczny	50	36,45
1	2021-06-01	motylkowy	50	45,24
2	2020-12-06	grzbietowy	50	32,44
2	2020-12-06	klasyczny	50	36,45
9	2021-12-06	grzbietowy	50	35,15
12	2021-12-06	motylkowy	50	50,01
0			0	0,00

Rekord: 1 z 7 Bez filtru Wyszukaj

Rys. 7. *Tabela* Starty *otwarta w programie Microsoft Access w* ***Widoku arkusza danych***

4.4. Definiowanie relacji w bazie danych *Zawody*

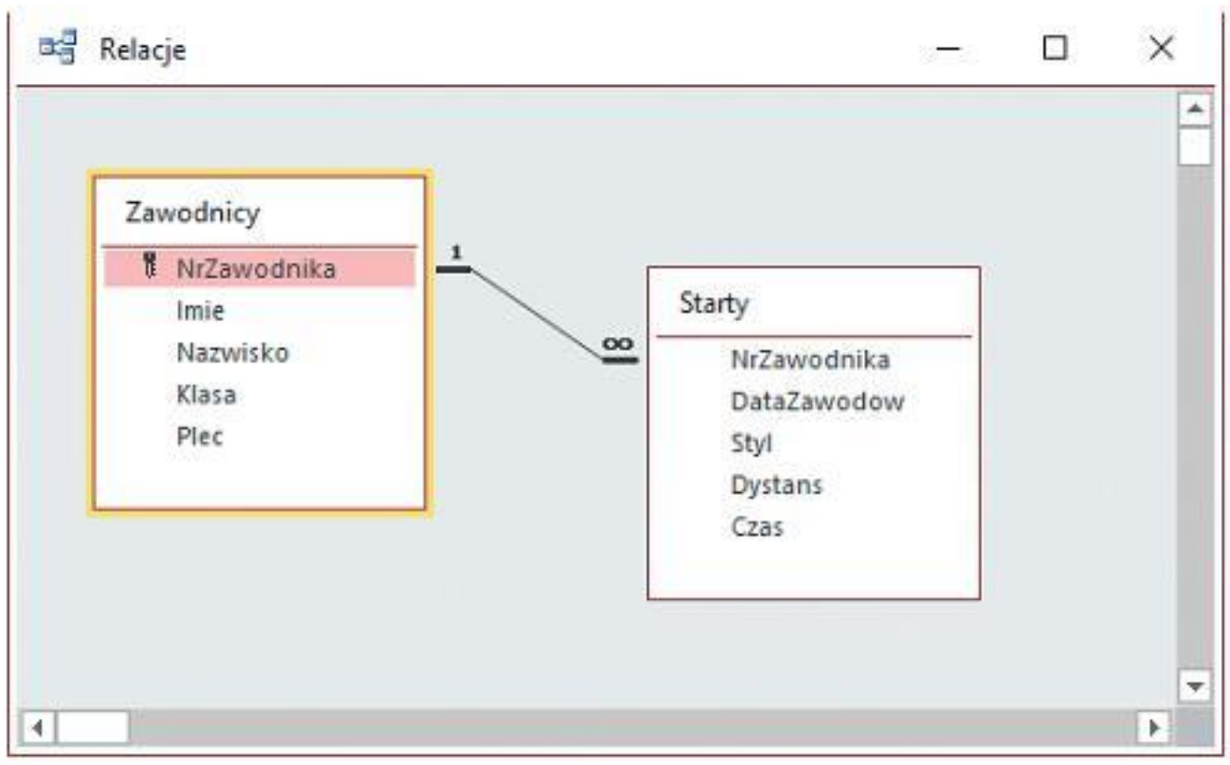
Wiele grup informacji jest ze sobą powiązanych, np. wielu uczniów chodzi do tej samej klasy, jeden człowiek może mieć kilka kont bankowych i kilka adresów e-mail, wielu dostawców może dostarczać ten sam rodzaj artykułu do magazynu, a jeden klient może składać wiele zamówień w hurtowni.

W bazie *Zawody* jeden zawodnik może wziąć udział w wielu startach, a informacja o jednym starcie dotyczy tylko jednego zawodnika.

W bazach danych tego rodzaju powiązania powinny zostać odtworzone. W relacyjnej bazie danych określamy je mianem **związków** (w terminologii programu Microsoft Access zwanych **relacjami**, ponieważ termin *relationship*, po angielsku oznaczający „związek", został przetłumaczony jako „relacja").

Przypuśćmy, że wszystkie informacje w bazie *Zawody* zapisano w jednej tabeli. Jeden rekord tej tabeli zawierałby wówczas dane zawodnika i informacje o jego starcie. Po każdym starcie do tabeli trzeba by było dopisać rekord z wszystkim danymi. Wielokrotnie powtarzałyby się dane zawodnika w każdym rekordzie. Aby uniknąć takiej sytuacji, rozdzielono dane na dwie tabele – oddzielnie informacje o zawodniku i oddzielnie o jego startach.

W programie Microsoft Access dostępne jest narzędzie umożliwiające definiowanie związków (relacji) między tabelami w postaci graficznej – jako linii łączącej poszczególne tabele (rys. 8.).



Rys. 8. *Przykład relacji określonej między dwiema tabelami*



Funkcję **klucza podstawowego** może pełnić tylko takie pole, którego wartości są **unikalne** (niepowtarzalne) w ramach tabeli. Klucz podstawowy jednoznacznie identyfikuje rekord.



Aby w programie Microsoft Access zdefiniować relację między daną tabelą a inną, jedno pole tej tabeli musi pełnić funkcję **klucza podstawowego**, natomiast w drugiej tabeli musi być wyznaczone pole do zapisywania wartości tego klucza.

Wartości pola klucza podstawowego mogą być wprowadzane automatycznie przez program. W tym celu należy zdefiniować pole typu **Autonumerowanie**. Każdy rekord będzie miał wówczas jednoznacznie przydzielony numer – kolejną liczbę naturalną. W programie Microsoft Access w nowej tabeli od razu jest wprowadzane pierwsze pole o nazwie *Identyfikator* typu **Autonumerowanie**.

Wartości klucza podstawowego mogą być również wprowadzane przez użytkownika systemu, ale wówczas typ pola musi być inny niż **Autonumerowanie**.

W tabeli *Zawodnicy* funkcję klucza podstawowego pełni pole *NrZawodnika*, które służy także do definiowania relacji między tabelą *Zawodnicy* a tabelą *Starty* (rys. 8.). W tabeli *Starty* wyznaczono pole *NrZawodnika*, w którym zapisywana jest wartość identyfikatora danego zawodnika (pobrana z pola *NrZawodnika* w tabeli *Zawodnicy*). Takie pole nazywane jest **kluczem obcym**, ponieważ identyfikuje rekordy z obcej tabeli (tu: z tabeli *Zawodnicy*).

W naszym przykładzie pola klucza podstawowego i klucza obcego mają taką samą nazwę (*NrZawodnika*), ale można je nazwać odmiennie.

Aby pole klucza podstawowego i odpowiadające mu pole klucza obcego zostały powiązane relacją, muszą być tego samego typu.

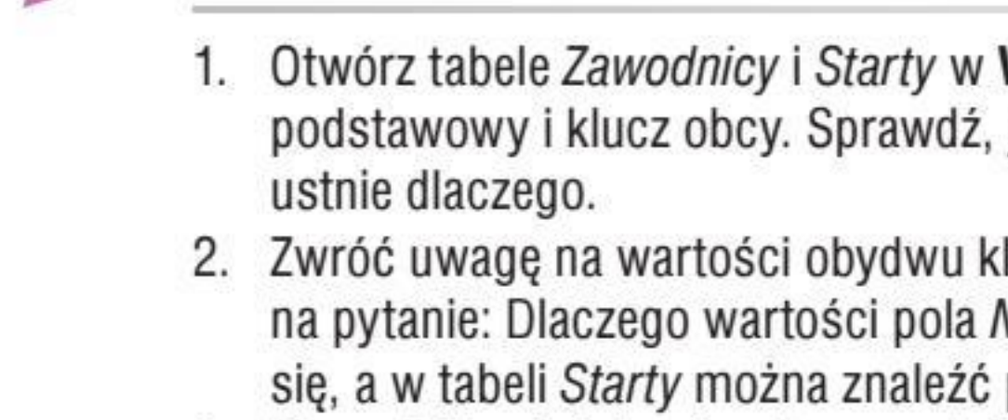
W naszym przykładzie pole *NrZawodnika* tabeli *Zawodnicy* jest typu **Autonumerowanie**, a więc zawiera liczby. Z tego powodu powiązane relacją pole *NrZawodnika* w tabeli *Zawodnicy* również powinno być typu liczbowego. Nie można jednak nadać polu *NrZawodnika* w tabeli *Starty* typu **Autonumerowanie**, ponieważ podczas dopisywania nowych rekordów pola w obydwu tabelach numerowałyby się niezależnie.

Dla każdego pola można określić typ danych, które będą w nim przechowywane, np. tekstowy (**Krótki tekst** lub **Długi tekst**), **Liczba**, **Data/Godzina**. Program Microsoft Access domyślnie przypisuje polom tekstowy typ danych (w praktyce większość gromadzonych danych jest typu tekstowego).

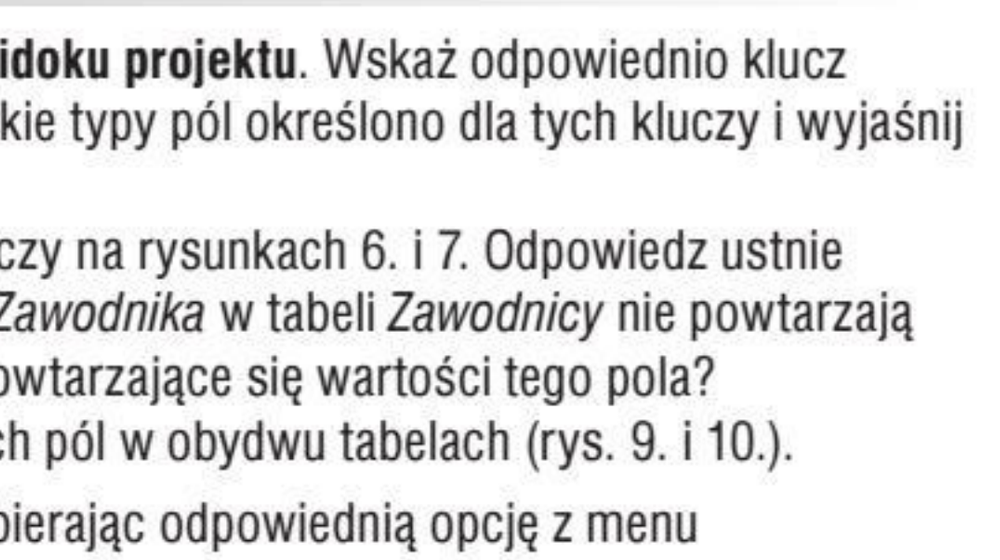
Ćwiczenie 3. Sprawdzamy typy pól w tabelach

1. Otwórz tabele *Zawodnicy* i *Starty* w **Widoku projektu**. Wskaż odpowiednio klucz podstawowy i klucz obcy. Sprawdź, jakie typy pól określono dla tych kluczy i wyjaśnij ustnie dlaczego.
2. Zwróć uwagę na wartości obydwu kluczy na rysunkach 6. i 7. Odpowiedz ustnie na pytanie: Dlaczego wartości pola *NrZawodnika* w tabeli *Zawodnicy* nie powtarzają się, a w tabeli *Starty* można znaleźć powtarzające się wartości tego pola?
3. Uzasadnij wybór typów dla pozostałych pól w obydwu tabelach (rys. 9. i 10.).

Wskazówka: Widok tabeli zmieniamy, wybierając odpowiednią opcję z menu kontekstowego nagłówka tabeli lub nazwy w okienku nawigacji.



Rys. 9.** Tabela Zawodnicy w **Widoku projektu

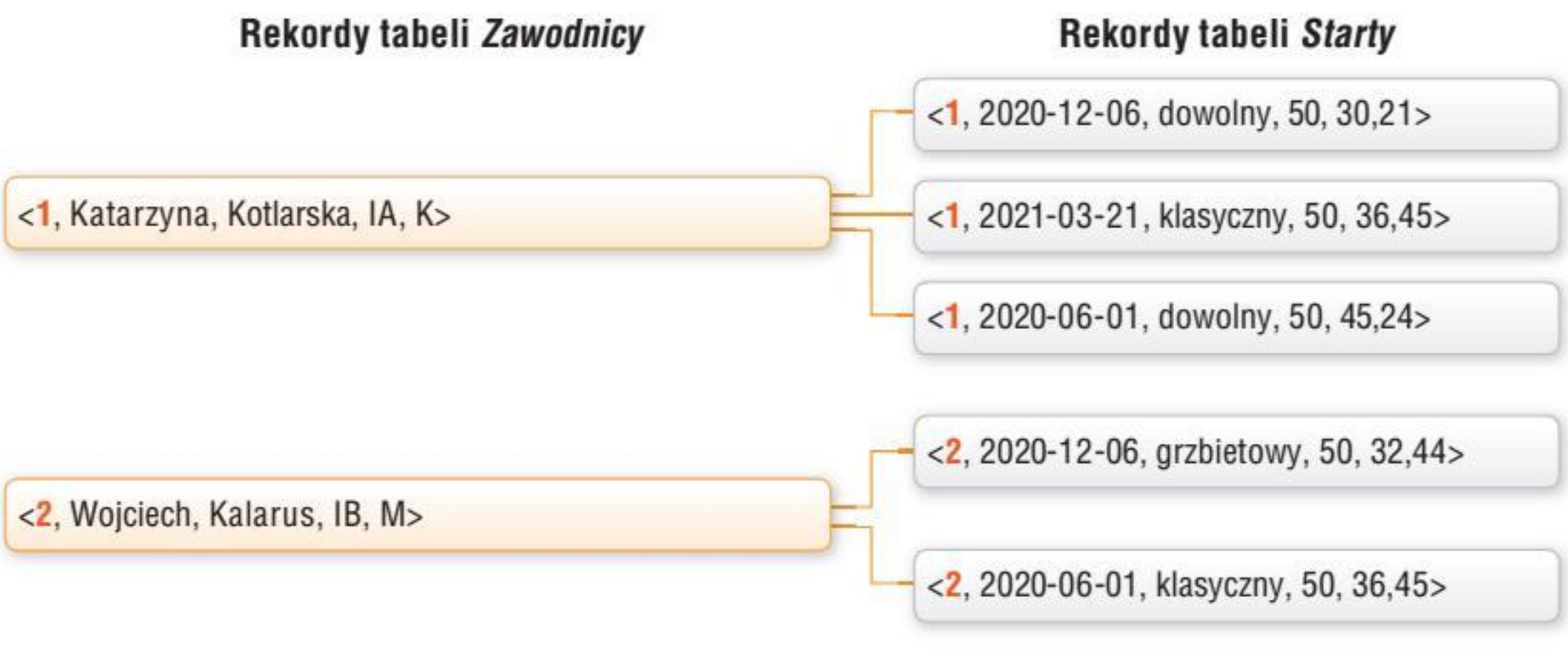


Rys. 10.** Tabela Starty w **Widoku projektu

W relacyjnej bazie danych można określić różne typy relacji. W omawianej bazie określono relacje typu „jeden do wielu", ponieważ jeden zawodnik może wziąć udział w kilku startach, a informacja o danym starcie dotyczy tylko jednego zawodnika. Jest to najczęściej występujący typ relacji.

Relacja **„jeden do wielu"** występuje, gdy rekord z pierwszej tabeli (tu: *Zawodnicy*) jest powiązany z wieloma rekordami drugiej tabeli (tu: *Starty*), natomiast rekord z drugiej tabeli jest powiązany tylko z jednym rekordem z pierwszej (rys. 11. i 12.).

Inne typy relacji to: „jeden do jednego" i „wiele do wielu".



Rys. 11. *Schemat relacji „jeden do wielu". Każdy prostokąt oznacza oddzielny rekord, np. rekord tabeli* Zawodnik *o numerze 1 jest powiązany z trzema rekordami tabeli* Starty *(zawodnik wziął udział w trzech startach)*



Rys. 12. *Fragment tabeli* Zawodnicy *z rozwiniętą listą startów zawodnika nr 1. (rekord nr 1 z tabeli* Zawodnicy *jest powiązany z trzema rekordami z tabeli* Starty*)*

Ćwiczenie 4. Sprawdzamy relację między tabelami

1. W bazie *Zawody* otwórz okno **Relacje** (karta **Narzędzia bazy danych**). Zwróć uwagę na graficzne przedstawienie relacji między tabelami (rys. 8.).
2. Sprawdź starty zawodników o numerach identyfikacyjnych 7. i 28.

Wskazówka: Aby wyświetlić rekordy powiązane z danym rekordem, kliknij kwadracik z lewej strony rekordu (rys. 12.).

5. Tworzenie bazy danych – obsługa gabinetu stomatologicznego

Stomatolog gromadzi informacje o pacjentach i ich kolejnych wizytach na kartach, które wypełnia ręcznie. Chcemy umożliwić mu gromadzenie i przetwarzanie danych z wykorzystaniem komputera. W jaki sposób możemy na przykładzie bazy *Zawody* utworzyć bazę danych, której celem będzie przetwarzanie danych o pacjentach gabinetu stomatologicznego?

5.1. Określenie wymagań bazy danych *Stomatolog*

Mimo że baza *Zawody* dotyczy zawodów pływackich, czyli dziedziny odległej od stomatologii, może posłużyć jako wzór do przygotowania bazy danych *Stomatolog*.

Celem bazy *Stomatolog* jest przetwarzanie danych o pacjentach i ich wizytach (w bazie *Zawody* odpowiednio – o zawodnikach i ich startach).

Każdy z was zapewne był kiedyś u stomatologa, dlatego sytuacja taka nie jest wam obca. Gdy przychodzi nowy pacjent, stomatolog na karcie pacjenta zapisuje jego dane osobowe m.in.: imię, nazwisko, datę urodzenia, PESEL, adres, numer telefonu. Umieszcza również na karcie informacje dotyczące kolejnych wizyt, np.: datę i godzinę wizyty, rodzaj zabiegu, koszt (w naszej bazie przyjmiemy okrojony zakres usług).

Załóżmy przykładowo, że stomatolog na koniec każdego dnia potrzebuje listę pacjentów, których przyjął, wraz z wykazem wykonanych zabiegów i ich kosztem (tzw. raport dzienny).

Naszym zadaniem będzie umożliwienie przetwarzania danych z wykorzystaniem komputera.



Podpis świadczeniobiorcy

Nr karty .....

Data rejestracji

KARTA STOMATOLOGICZNA

UBEZPIECZONY w .....

Nazwisko ..... Imię .....

Data urodzenia ..... PESEL

Adres .....

Data	Ząb	Rozpoznanie	Wykonany zabieg	Punkty	% odpłatności

Rys. 13. *Fragment przykładowej karty pacjenta*

Ćwiczenie 5. Określamy wymagania bazy *Stomatolog*

1. Jesteś informatykiem, który ma wykonać bazę danych dla gabinetu stomatologicznego. Dowiedz się, jakie informacje o pacjentach lekarz chce wprowadzać do bazy oraz jakie wyniki z ich przetwarzania chce uzyskiwać, m.in. co powinno być umieszczane na wydrukach.
2. Wypisz w punktach wymagania bazy *Stomatolog*.

Wskazówki:

- Ćwiczenie możesz wykonać w grupie z podziałem na role – z koleżanką lub kolegą.
- Możesz wzorować się na wymaganiach określonych dla bazy *Zawody*.

5.2. Wstępne projektowanie formularzy i raportów w bazie danych *Stomatolog*

Ćwiczenie 6. Przygotowujemy odręczne projekty formularza pacjenta i raportu

Na podstawie wymagań określonych w ćwiczeniu 5., przygotuj:

a. odręczny projekt formularza wprowadzania danych pacjenta,
b. odręczny projekt wybranego raportu.

Wskazówka: Możesz wzorować się na projektach pokazanych na rysunkach 4. i 5.

Ćwiczenie 7. Ustalamy zbiory informacji

Podziel informacje, które mają być przetwarzane w bazie *Stomatolog*, na grupy – według wzoru pokazanego w tabeli 1. dla szkolnych zawodów pływackich.

5.3. Tworzenie tabel w bazie danych *Stomatolog*

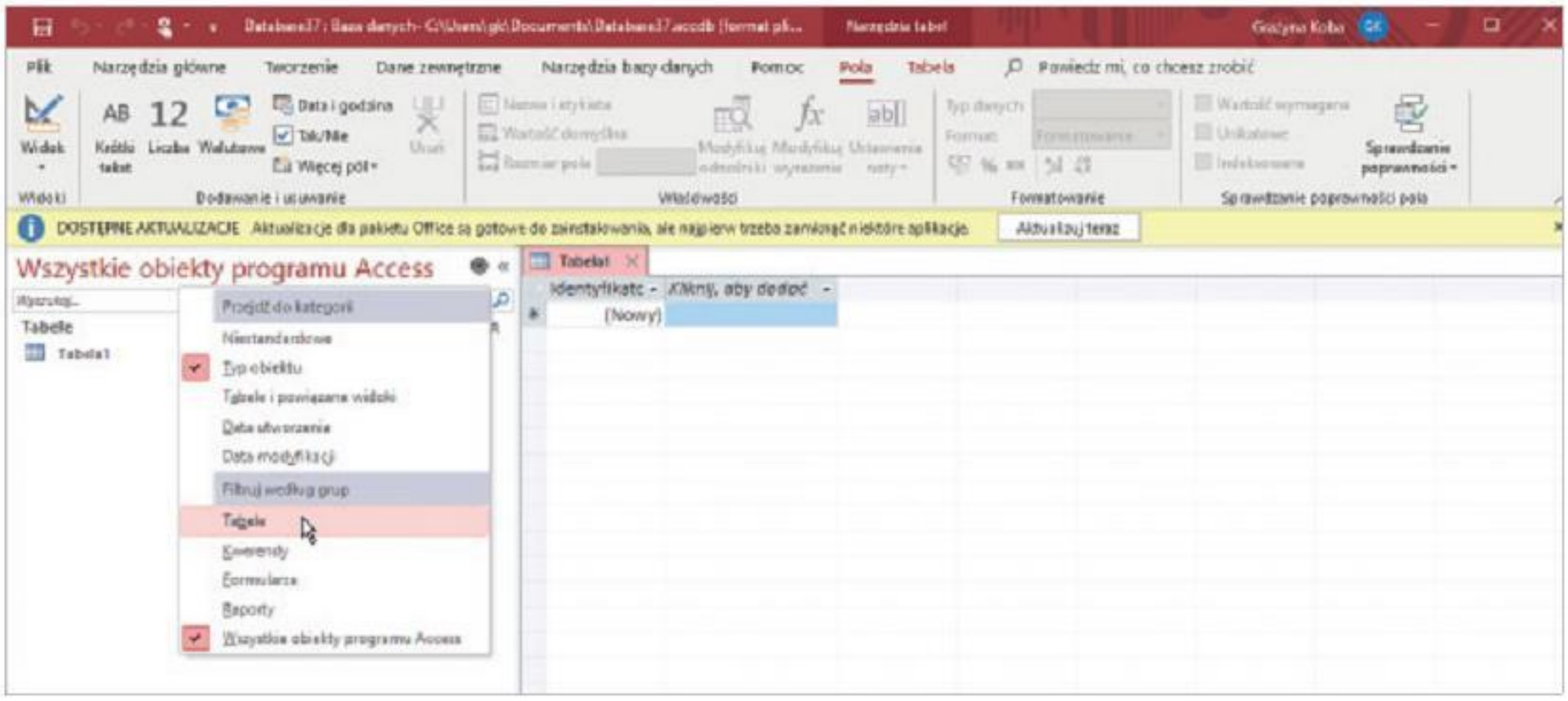
Ćwiczenie 8. Planujemy pola i grupujemy je w tabele

1. Ustal, ile tabel potrzebujesz, aby móc zapisywać informacje określone w ćwiczeniu 7. Określ nazwy tabel, np. *Pacjenci* i *Wizyty*.
2. Zaplanuj odpowiednie pola dla bazy danych *Stomatolog*, przydzielając je odpowiednio do tabel według wzoru pokazanego w tabeli 2. dla szkolnych zawodów pływackich.

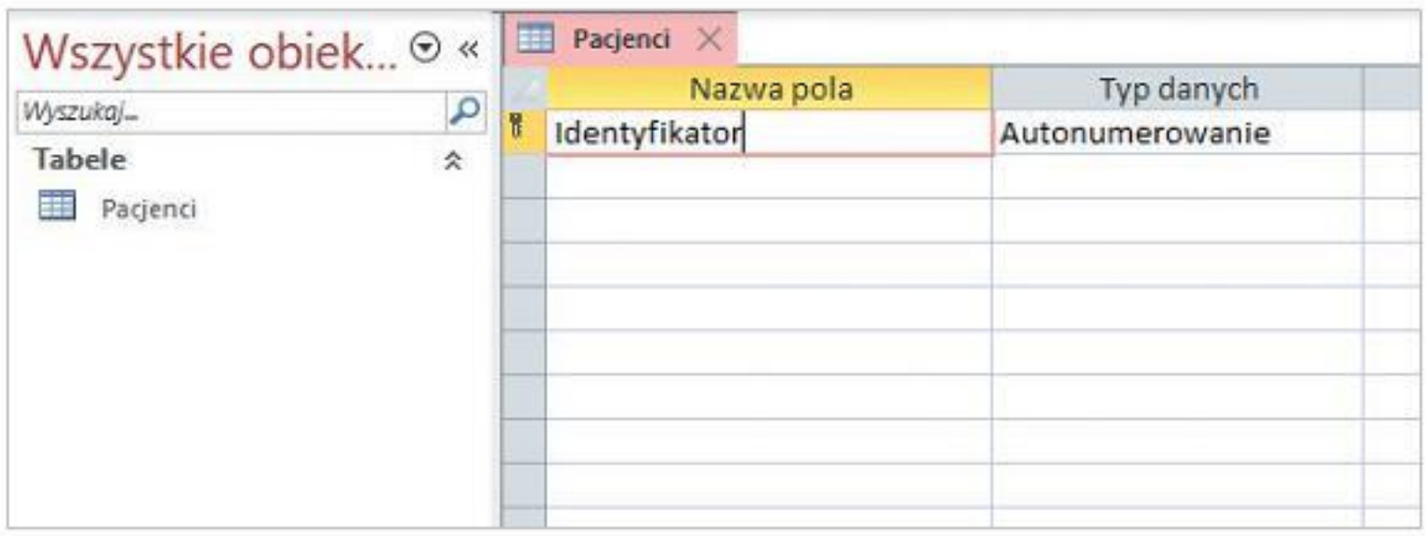
Wskazówka: Pamiętaj, że do jednej tabeli należy przypisać jedną kategorię informacji.

Po zaplanowaniu i zaprojektowaniu bazy możemy przystąpić do jej tworzenia. Zaczniemy od tabel. Po uruchomieniu programu i wybraniu opcji tworzenia pustej bazy danych otworzy się okno jak na rysunku 14. Domyślnie w okienku nawigacji (w którym możemy wyświetlać również inne obiekty bazy danych) pojawia się nazwa tabeli *Tabela1* i otwiera okno tabeli w **Widoku arkusza danych**.

Aby wprowadzać pola bazy danych i określać ich typy, należy zmienić widok bazy danych na **Widok projektu**. Przed zmianą widoku tabelę trzeba nazwać.



Rys. 14. *Tworzenie nowej tabeli w programie Microsoft Access*



Rys. 15. *Tabela* Pacjenci *w* ***Widoku projektu***

Program domyślnie wprowadza pole klucza podstawowego o nazwie *Identyfikator*. Przy nazwie pola klucza powinien być umieszczony kluczyk (rys. 15.). W tabeli *Pacjenci* klucz podstawowy jest potrzebny do jednoznacznej identyfikacji rekordów oraz do zdefiniowania relacji między tą tabelą a tabelą *Wizyty*.

Program Microsoft Access 2016 zapisuje bazę danych w pliku pod domyślną nazwą *Database* i z domyślnym rozszerzeniem *accdb*. Pliku zapisanego w tym formacie nie można otworzyć w wersjach 2003 i wcześniejszych. Jeśli chcemy mieć możliwość otwierania pliku również w starszych wersjach programu, możemy podczas zapisu wybrać inne rozszerzenie, np. *mdb* (zgodne z wersją programu Access 2002-2003). W wersji 2016 można otwierać pliki z rozszerzeniem *mdb*.



Poszczególne obiekty bazy danych (tabele, formularze, zapytania, raporty) nazywamy oddzielnie, ale zapisywane są one w jednym pliku.

Ćwiczenie 9. Tworzymy tabelę *Pacjenci*

1. Utwórz nowy plik bazy danych. Zapisz plik bazy danych pod nazwą *Stomatolog*.
2. Zmień widok tabeli na **Widok projektu** i nadaj jej nazwę *Pacjenci*.
3. Zmień nazwę klucza podstawowego *Identyfikator* na nazwę *NrPacjenta*.
4. Wpisz pozostałe nazwy pól, korzystając z listy pól utworzonej w ćwiczeniu 8. Określ typy pól, wybierając typ pola z listy wyboru (rys. 10.). Zapisz tabelę, gdy program tego zażąda.

Uwaga: W starszych wersjach programu podczas próby zapisania tabeli, w której nie zdefiniowano klucza podstawowego, program zleci jego zdefiniowanie. Jeśli klucz nie został wstawiony automatycznie, można go wstawić, klikając na pasku narzędzi programu ikonę klucza podstawowego (,).

Ćwiczenie 10. Tworzymy tabelę *Wizyty*

1. W bazie *Stomatolog* utwórz tabelę o nazwie *Wizyty*. Jeśli klucz podstawowy został zdefiniowany automatycznie – usuń „kluczyk" przy nazwie pola, zmień nazwę pola *Identyfikator* na *NrPacjenta*, a typ danych – na **Liczba**. Pole to będzie pełnić funkcję klucza obcego, podobnie jak pole *NrZawodnika* w tabeli *Starty* w bazie *Zawody*.
2. Wpisz pozostałe nazwy pól, korzystając z listy pól utworzonej w ćwiczeniu 8. Określ typy pól. Zapisz tabelę.
3. Odpowiedz na pytania, uzasadniając odpowiedzi:
   a. Dlaczego funkcję klucza podstawowego może pełnić tylko pole *NrPacjenta* w tabeli *Pacjenci*?
   b. Dlaczego pole *NrPacjenta* w tabeli Wizyty nie może być typu **Autonumerowanie**?
   c. Dlaczego pole *NrPacjenta* w tabeli *Pacjenci* nazywamy kluczem podstawowym, a w tabeli *Wizyty* – kluczem obcym?

Wskazówki:

- Opcję dodawania nowej tabeli i projektowania tabeli znajdziesz na karcie **Tworzenie**.
- Aby usunąć „kluczyk" przy nazwie pola, kliknij wybrane pole, a następnie przycisk z kluczykiem.

5.4. Definiowanie relacji w bazie danych *Stomatolog*

Przed określeniem typu relacji między tabelami, musimy odpowiedzieć na pytania:

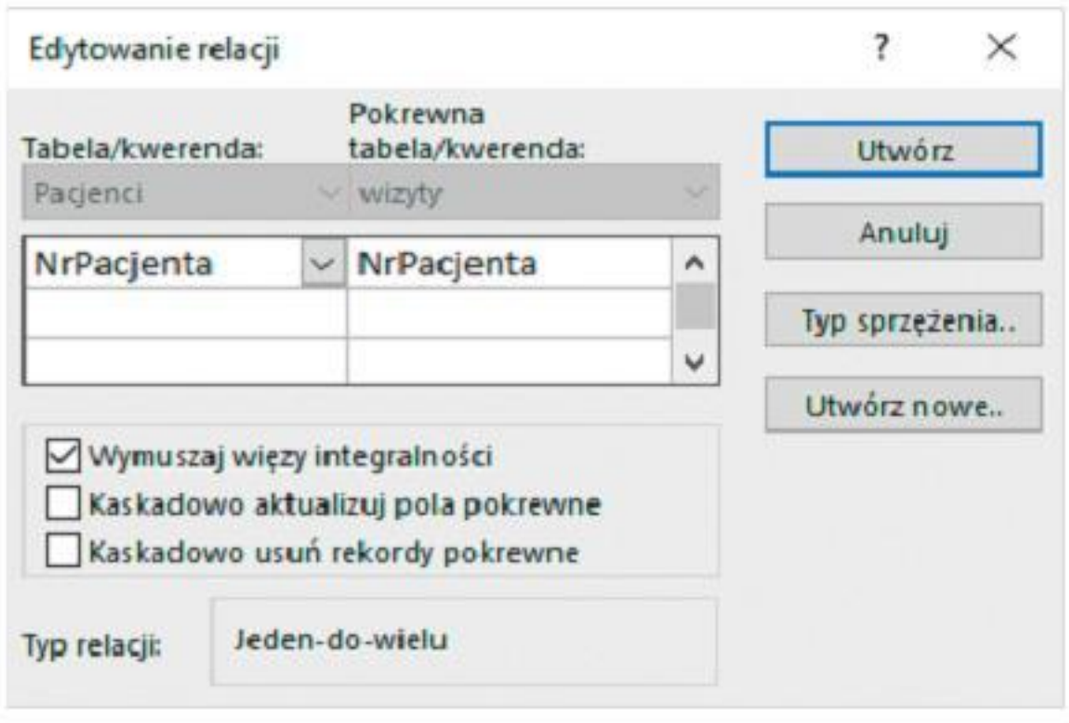
- Czy ten sam pacjent może przyjść do stomatologa wiele razy?
- Czy jedna wizyta dotyczy jednego pacjenta?

Jeśli odpowiedzi na te pytania są twierdzące, należy zdefiniować relację „jeden do wielu".

Przykład 1. Definiowanie relacji między tabelami

Chcemy zdefiniować relację między tabelami *Pacjenci* i *Wizyty*.

1. Wybierz opcję **Relacje** (na karcie **Narzędzia bazy danych**) – otworzy się okno obiektu **Relacje**.
2. Umieść w oknie **Relacje** tabele, między którymi ma być określona relacja (tu: *Pacjenci* i *Wizyty*), przeciągając nazwy tabel z okienka nawigacji lub panelu **Dodawanie tabel**.
3. Przeciągnij pole klucza podstawowego na odpowiadające mu pole klucza obcego w drugiej tabeli – otworzy się okno **Edytowanie relacji** (rys. 16.).
4. Zaznacz opcję **Wymuszaj więzy integralności** i kliknij przycisk **Utwórz**.



Rys. 16.** Okno **Edytowanie relacji** w widoku **Relacji

Uwagi:

- Panel **Dodawanie tabel** można otworzyć, wybierając z menu kontekstowego okna **Edytowanie relacji** polecenie **Pokaż tabele**.
- Wymuszanie więzów integralności oznacza, że nie ma możliwości wprowadzenia do pola klucza obcego wartości nieistniejących w polu klucza podstawowego, czyli w tabeli *Wizyty* nie można dopisać rekordu dla pacjenta, którego nie ma w tabeli *Pacjenci*.

Ćwiczenie 11. Definiujemy relacje

1. Korzystając z przykładu 1., zdefiniuj w bazie *Stomatolog* relację między tabelami *Pacjenci* i *Wizyty*. Zapisz obiekt **Relacje**.
2. Korzystając z informacji opisanych w punkcie 4.4., odpowiedz na pytania:
   a. Dlaczego zdefiniowanie klucza podstawowego w tabeli *Pacjenci* wystarczy, aby określić relację między tabelami *Pacjenci* i *Wizyty*? Uzasadnij odpowiedź.
   b. Dlaczego w bazie *Stomatolog* należy zdefiniować relację „jeden do wielu"?

Warto zapamiętać

- Baza danych to zbiór danych, czyli odpowiednio zorganizowanych informacji, które można przetwarzać za pomocą programów komputerowych.
- Wykonywanie operacji na danych, czyli przetwarzanie, to m.in. ich wprowadzanie, redagowanie, wyszukiwanie, prezentowanie.
- Dane w relacyjnych bazach danych przechowujemy w tabelach. Tabele składają się z wierszy (rekordów) i kolumn (pól). W obrębie tabeli nazwy pól nie mogą się powtarzać.
- W programie Microsoft Access definiuje się klucz podstawowy, który jednoznacznie identyfikuje rekordy w bazie.
- Aby w programie Microsoft Access zdefiniować relację między dwiema tabelami, jedno pole pierwszej tabeli musi pełnić funkcję klucza podstawowego, natomiast w drugiej tabeli musi być wyznaczone pole do zapisywania wartości tego klucza. Pola tych kluczy muszą być tego samego typu.
- Nie należy powtarzać tych samych informacji w kilku tabelach.

Pytania i polecenia

1. Czym jest baza danych?
2. Na czym polega przetwarzanie danych w bazie danych?
3. Jak są zorganizowane dane w relacyjnej bazie danych?
4. Czym jest rekord? Podaj przykład.
5. Podaj etapy przygotowania w programie Microsoft Access prostej relacyjnej bazy danych.
6. Dlaczego istotne jest wstępne określenie wymagań dotyczących rodzaju informacji wprowadzanych do bazy danych?
7. Omów na przykładach zasady tworzenia tabel.
8. Czym jest relacja?
9. Jakie warunki muszą być spełnione, aby zdefiniować relację między dwiema tabelami?
10. Jaką funkcję pełni klucz podstawowy, a jaką klucz obcy?
11. Dlaczego pole *NrZawodnika* (w bazie *Zawody*) jednoznacznie identyfikuje rekordy w tabeli *Zawodnicy*?
12. Kiedy występuje relacja typu „jeden do wielu”?

Zadania

1. W magazynie dżemów klienci składają zamówienia na różne rodzaje dżemów. Jeśli klient chce zamówić kilka rodzajów dżemów, na każdy z nich składa oddzielne zamówienie. Przygotuj bazę danych składającą się z dwóch tabel: *Klienci* i *Zamowienia*. Tabela *Klienci* powinna zawierać m.in.: nazwisko, imię i adres klienta, a tabela *Zamowienia* – informacje o zamówieniu, np. numer zamówienia, rodzaj dżemu, liczbę sztuk, datę złożenia zamówienia, datę realizacji. Zdefiniuj relację między tabelami, odwzorowując rzeczywiste powiązania. Zapisz bazę w pliku pod nazwą *Magazyn dżemów*.
Uwaga: Profesjonalny system obsługi magazynu zawiera więcej powiązanych tabel i danych.

2. Przygotuj bazę danych umożliwiającą przetwarzanie danych z innych zawodów sportowych niż pływanie. Baza ma składać się z dwóch tabel: jednej do przechowywania danych o zawodnikach i drugiej do zapisywania informacji o startach, w których uczestniczy dany zawodnik. Określ odpowiednie zbiory informacji, które będą przechowywane w obydwu tabelach. Następnie ustal odpowiednio pola i określ ich typy. Zdefiniuj relacje, biorąc pod uwagę, że jeden zawodnik może wziąć udział w kilku startach, a informacje o danym starcie dotyczą konkretnego zawodnika.
3. Zaprojektuj bazę danych: ustal zbiory informacji, zastanów się nad rzeczywistymi powiązaniami między grupami informacji, określ wymagania bazy, zaplanuj pola i tabele. Zdefiniuj odpowiednie klucze i relacje.
   a. *Wypożyczalnia samochodów* (np. informacje o samochodach, klientach, transakcjach wypożyczenia).
   b. *Moja elektroniczna płytoteka* (np. informacje o albumach, wykonawcach, piosenkach).
   c. *Obsługa górskiego schroniska* (np. informacje o gościach, pokojach, rezerwacjach).
   d. *Serwis sprzętu komputerowego* (np. informacje o klientach, sprzęcie, reklamacjach).

Dla zainteresowanych

4. Zaprojektuj bazę danych *Klinika zwierząt*. Ustal zbiory informacji i wymagania systemu, zastanów się nad rzeczywistymi powiązaniami między grupami informacji, zaplanuj pola i tabele. Zdefiniuj odpowiednie klucze i relacje. Baza powinna składać się z trzech tabel powiązanych relacjami (*Wlasciciele*, *Zwierzaki*, *Wizyty*). Zakładamy, że jeden właściciel może mieć kilka zwierząt, z którymi przyjdzie na wizytę tego samego dnia, ale informacja o wizycie zapisywana jest oddzielnie dla każdego ze zwierząt.

Temat B2.

Tworzenie i stosowanie formularzy w relacyjnej bazie danych

1. Korzystanie z formularzy w bazie *Zawody*
2. Zasady tworzenia formularzy na przykładzie bazy *Zawody*
   2.1. Formularz standardowy
   2.2. Formularz z podformularzem
3. Przygotowanie formularzy w bazie *Stomatolog*
4. Importowanie danych z innych dokumentów do tabeli bazy danych

Warto powtórzyć

1. Jakie etapy tworzenia bazy danych już zrealizowaliśmy?
2. Jak zbudowana jest tabela w relacyjnej bazie danych?
3. Jakie poznaliśmy zasady projektowania tabel?
4. Jakie pola zawarliśmy w tabelach w bazie *Zawody*?
5. Jaka jest rola klucza podstawowego i klucza obcego w tabelach bazy danych?
6. Jakie wymagania określiliśmy dla bazy *Stomatolog*?

1. Korzystanie z formularzy w bazie *Zawody*

Formularz służy do wprowadzania i przeglądania danych, dlatego powinien być przejrzyście zaprojektowany i starannie przygotowany.

Różne formularze (tradycyjne i elektroniczne) wypełniamy często w codziennym życiu, np. przy przyjęciu do szkoły, pracy, szpitala, przy składaniu PIT-u, zakładaniu konta e-mailowego. Niektóre formularze nadal wypełnia się ręcznie, ale wiele z nich ma też formę elektroniczną (np. PIT).

Gdy wpisujemy swoje nazwisko, imię i inne informacje do pól tekstowych, np. w formularzu zakładania konta pocztowego wybranego portalu internetowego, to są one zapamiętywane w bazie danych na serwerze tego portalu.

Uwaga

Aby rozróżniać poszczególne nazwy obiektów bazy, na początku nazwy formularza będziemy dodawać literę F, nazwę kwerendy będziemy zaczynać od litery K, a raportu – od litery R.

Ćwiczenie 1. Podajemy przykłady formularzy

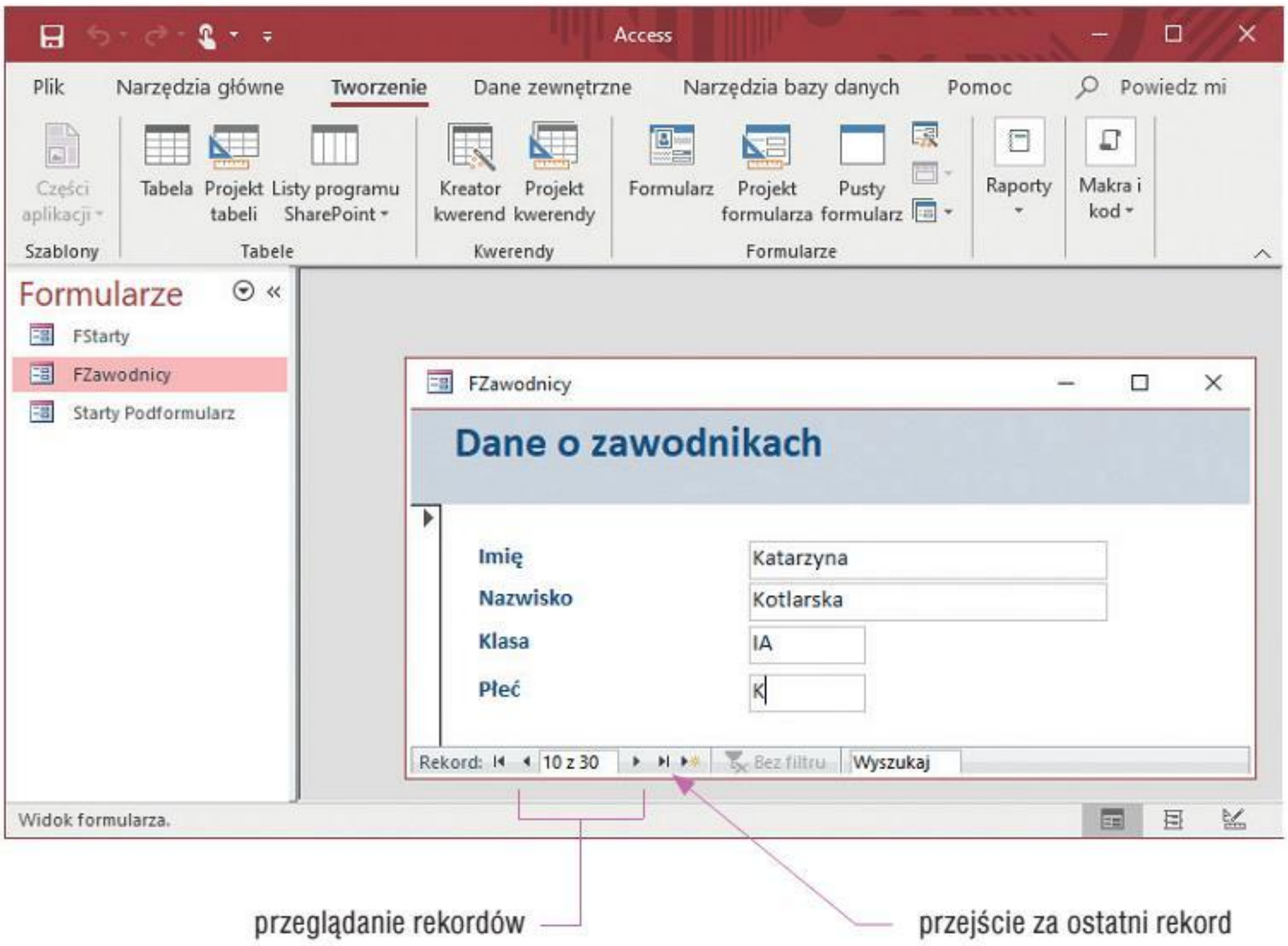
Podaj przykłady formularzy z różnych dziedzin życia. Znajdź w Internecie przykładowy formularz. Wymień informacje, jakie trzeba do niego wpisać. Oceń czytelność i przejrzystość formularza.

W bazie danych tworzonej w programie Microsoft Access dane wprowadzane poprzez formularze są przechowywane w tabelach. Dane wyświetlane lub wprowadzane za pomocą formularza mogą pochodzić z jednej tabeli lub kilku tabel.

W związku z wcześniej zaplanowanymi wymaganiami i wykonanymi odręcznie wzorami formularzy (temat B1) utworzymy w bazie *Zawody* dwa formularze:

- *FZawodnicy* – do wprowadzania danych o pływakach (**formularz standardowy**, który tworzy się na podstawie jednej tabeli),
- *FStarty* – do wprowadzania informacji o startach (**formularz z podformularzem**, który umożliwia wyświetlenie danych powiązanych; można go utworzyć na podstawie tabel, dla których określono relację „jeden do wielu")

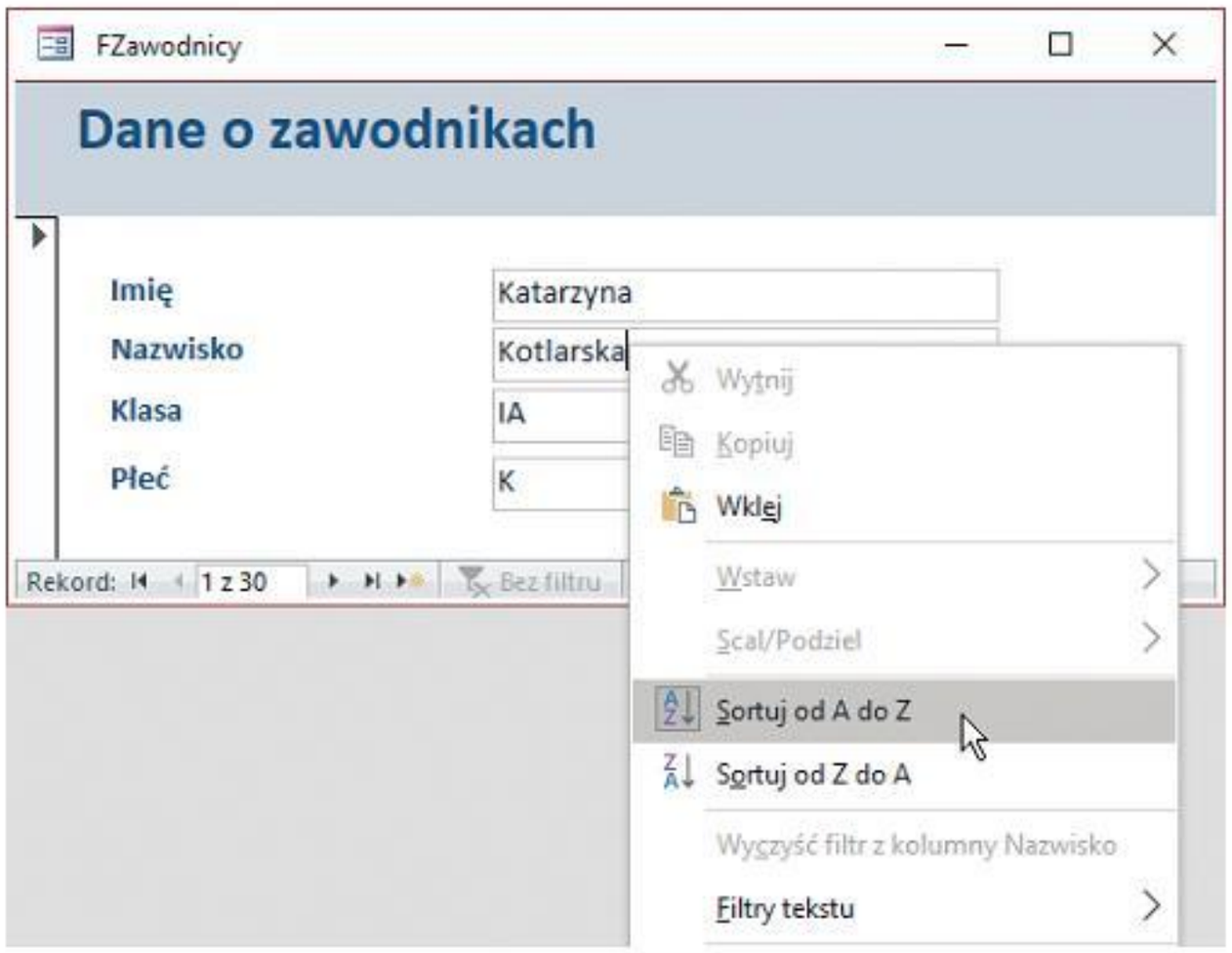
Dane wprowadzane z wykorzystaniem formularzy są zapisywane w konkretnych polach odpowiednich tabel.



Rys. 1. *Przykład formularza standardowego do wprowadzania danych zawodnika w* ***Widoku formularza***

W celu ułatwienia przeglądania danych w formularzu *FZawodnicy* posortowano rekordy alfabetycznie według pola *Nazwisko*.

Aby w formularzu wyświetlać rekordy posortowane według wybranego pola, wystarczy umieścić kursor tekstowy w polu tekstowym danego pola (np. pola *Nazwisko*) i wybrać opcję sortowania, np. z menu kontekstowego (rys. 2.). Kolejność rekordów w tabeli, z której wybrano pole, pozostaje bez zmian.



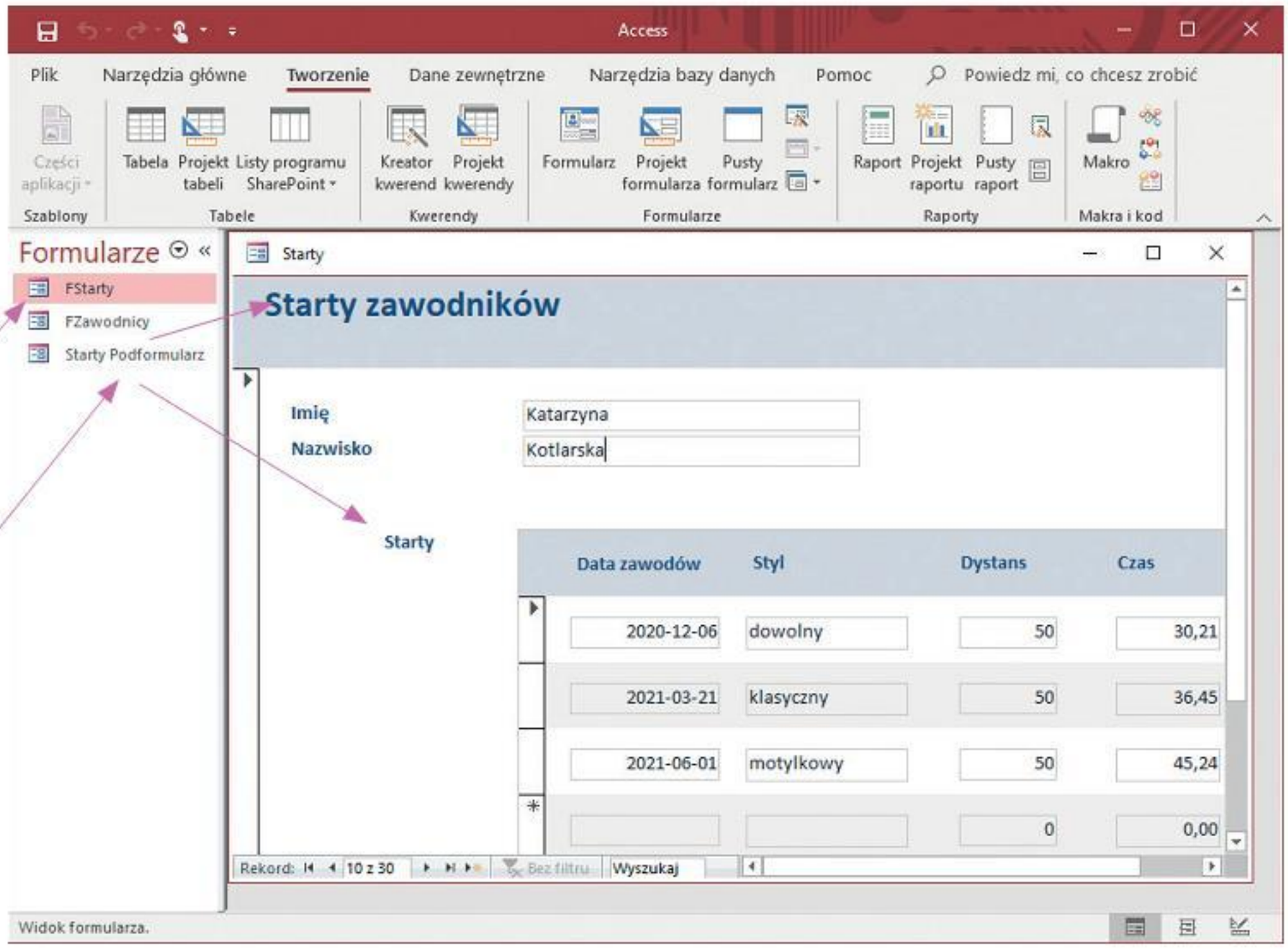
Rys. 2. *Wyświetlanie rekordów uporządkowanych według pola* Nazwisko *w formularzu* FZawodnicy

Ćwiczenie 2. Korzystamy z formularzy w bazie *Zawody*

1. W bazie *Zawody* (plik otrzymasz od nauczyciela) utworzono dwa formularze. Przejrzyj dane kilku zawodników, korzystając z formularza *FZawodnicy*.
2. Przed Mikołajkowymi zawodami pływackimi okazało się, że pojawiło się dwoje nowych zawodników. Korzystając z formularza *FZawodnicy*, dopisz nowych pływaków do bazy *Zawody*.
   Przykładowe, fikcyjne dane:
   <Małgorzata, Gerula, IA, K>,
   <Adam, Konieczny, IA, M>.
   Zapisz formularz pod tą samą nazwą.
3. Otwórz tabelę *Zawodnicy* i sprawdź, czy dane nowych zawodników są umieszczone w bazie. Jakie są ich identyfikatory?

Wskazówka: Aby dopisać dane nowych zawodników za pomocą formularza, należy najpierw przejść za ostatni rekord.

Wprowadzenie danych nowego zawodnika jest równoznaczne z dopisaniem nowego rekordu do tabeli *Zawodnicy*. W ten sposób **przetworzyliśmy dane**.



formularz *FStarty*

podformularz formularza *FStarty* – oddzielny obiekt bazy danych

Rys. 3. *Przykład formularza z podformularzem do wprowadzania startów konkretnego zawodnika w* ***Widoku formularza***

Ćwiczenie 3. Dodajemy informacje do bazy *Zawody* poprzez formularze

1. Otwórz formularz *FStarty* i przejrzyj starty kilku zawodników.
2. Trwają „Mikołajkowe zawody pływackie". Sześcioro zawodników (trzy dziewczęta i trzech chłopców, w tym dwoje nowych dopisanych w ćwiczeniu 2.) przepłynęło dystans 50 m stylem dowolnym. Korzystając z formularza *FStarty*, dopisz informacje o startach tych zawodników. Przyjmij, że zawody odbywały się 6 grudnia 2021 (pole *DataZawodow*). Zapisz formularz pod tą samą nazwą.
3. Otwórz tabelę *Starty* i sprawdź, czy starty tych sześciorga zawodników są umieszczone w bazie. Sprawdź, jakie są identyfikatory tych zawodników.
4. Dopisz jeszcze po jednym starcie na dystansie 50 m stylem klasycznym tym samym zawodnikom. Zapisz formularz *FStarty* pod tą samą nazwą. Otwórz tabelę *Starty*. Co zauważasz?

Zapisanie w formularzu nowych danych dotyczących startu zawodnika jest równoznaczne z dopisaniem nowego rekordu do tabeli *Starty*.
W ten sposób **przetworzyliśmy dane**.

2. Zasady tworzenia formularzy na przykładzie bazy *Zawody*

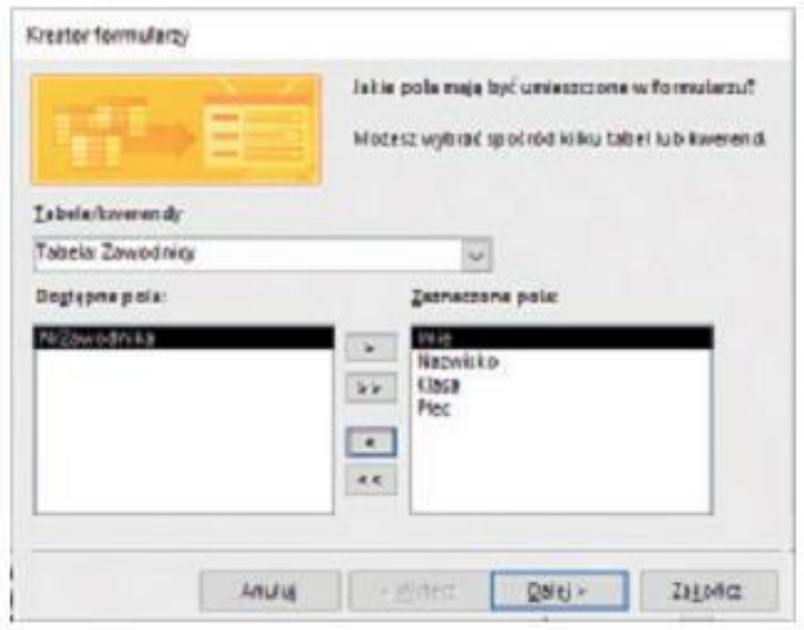
Każdy obiekt bazy danych programu Microsoft Access (w tym formularze) możemy utworzyć, korzystając z kreatora, lub zaprojektować samodzielnie (potrzebne opcje znajdują się na karcie **Tworzenie**). Możemy również najpierw skorzystać z kreatora, a potem zmodyfikować formularz w widoku projektu, w którym są dostępne dodatkowe opcje.

2.1. Formularz standardowy

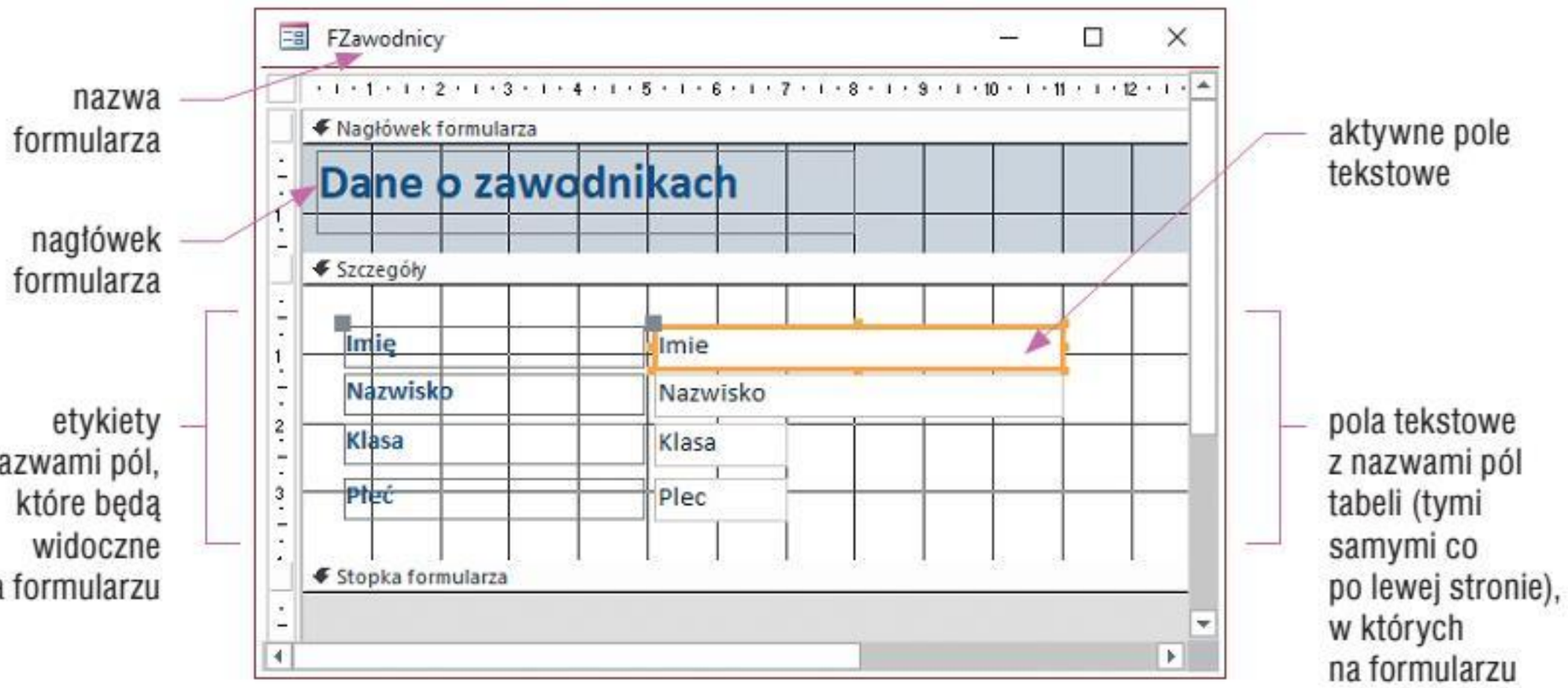
Aby utworzyć formularz standardowy za pomocą kreatora, należy w oknie **Kreatora formularzy**:

- wybrać tabelę, na podstawie której chcemy wykonać formularz,
- wybrać pola tabeli, które powinny znaleźć się w formularzu,
- wybrać układ formularza. np. kolumnowy,
- nazwać i zapisać formularz.

Formularz do wprowadzania danych utworzono na podstawie tabeli *Zawodnicy*. Wybrano z niej wszystkie pola, poza polem *NrZawodnika*, ponieważ jego wartości są wpisywane automatycznie. Dla formularza widocznego na rysunku 5. wybrano styl **Kolumnowy**.



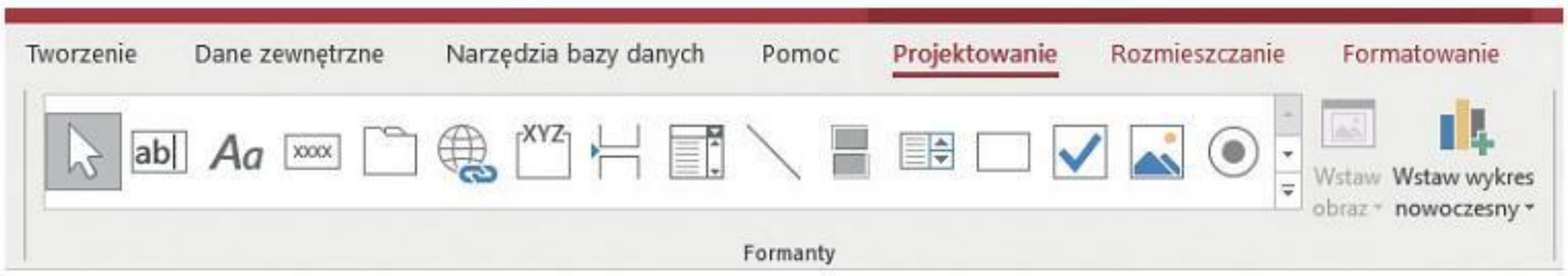
Rys. 4. *Okno* ***Kreatora formularzy***



Rys. 5. *Formularz standardowy* FZawodnicy *przygotowany na podstawie tabeli* Zawodnicy *w* ***Widoku projektu*** *– efekt działania kreatora*

Kreator pomaga w zaprojektowaniu podstawowego wyglądu formularza, który można następnie modyfikować w widoku projektu. Możemy m.in.: poprawić etykiety widoczne po lewej stronie formularza (rys. 5.), zmienić położenie pól, w razie potrzeby zwiększyć ich rozmiar, ustalić inne parametry czcionki, umieścić lub poprawić tytuł w nagłówku formularza czy wstawić dodatkowe opisy zwiększające czytelność formularza.

Na formularzu możemy umieszczać różne obiekty: etykiety, pola, obrazy, wykresy. Aby umieścić na formularzu np. dodatkowy opis (etykietę), można skorzystać z narzędzi projektowania formularzy (rys. 6.).



Rys. 6. *Narzędzia projektowania formularzy*

2.2. Formularz z podformularzem

Formularz *FStarty* w bazie *Zawody* został przygotowany na podstawie dwóch tabel: *Zawodnicy* i *Starty*, dla których została zdefiniowana relacja „jeden do wielu". Utworzono więc formularz, w którym możemy przeglądać wszystkie starty danego zawodnika oraz dopisywać informacje o nowym starcie.

Aby utworzyć formularz z podformularzem za pomocą kreatora, należy w oknie **Kreatora formularzy**:

- wybrać tabele, na podstawie których chcemy wykonać formularz, oraz pola, które powinny znaleźć się w formularzu,
- ustalić inne elementy, m.in. sposób wyświetlania danych i układ formularza, np. dla formularza głównego kolumnowy, a dla podformularza – tabelaryczny.
- nazwać oraz zapisać formularz i podformularz.

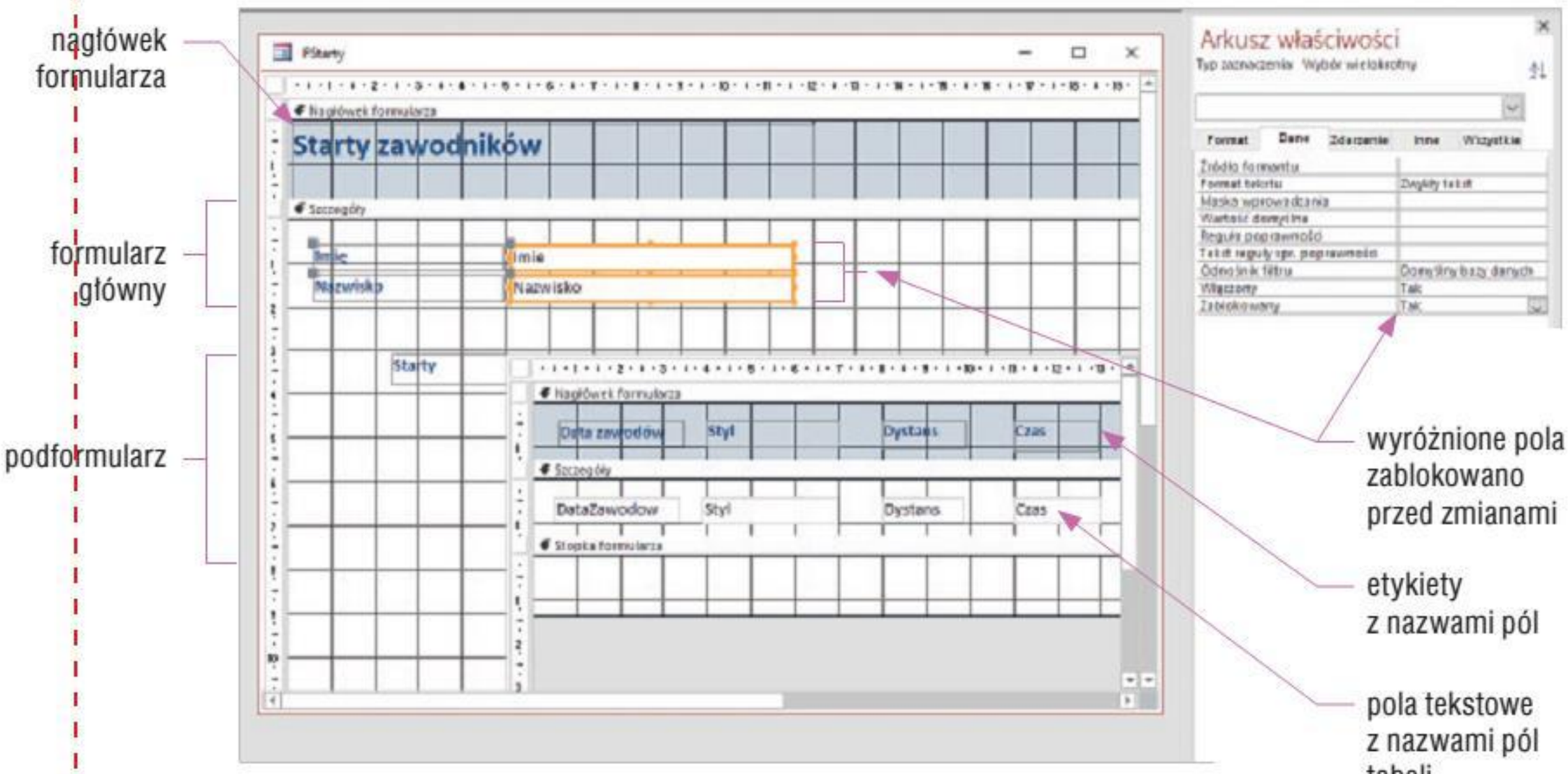
Mimo że podformularz jest umieszczony na formularzu, to zapisuje się go pod własną nazwą.

Z tabeli *Zawodnicy* wybrano pola *Imie* i *Nazwisko*. Wartości tych pól będą widoczne w formularzu głównym. Nie wybrano pola *NrZawodnika*. Wartości tego pola są wprowadzane automatycznie przez program i służą do identyfikacji rekordów w bazie. Założono, że użytkownik posługuje się imieniem i nazwiskiem zawodnika, a nie jego numerem, gdyż nazwisko i imię jest łatwiej zapamiętać. Dla uproszczenia przyjęto, że w omawianej bazie nie ma osób o takim samym nazwisku i imieniu.

Z tabeli *Starty* wybrano pola informujące o starcie zawodnika, czyli: *DataZawodow*, *Styl*, *Dystans*, *Czas*.

Dla danego zawodnika mają być wyświetlane dane dotyczące jego startów, dlatego wybrano sposób wyświetlania danych **przez Zawodnicy.**

Gdy wybierzemy z menu kontekstowego danego elementu polecenie **Właściwości**, otworzy się okno, w którym możemy zmieniać wiele różnych właściwości formatowania pola, np. styl obramowania i zacieniowania, rozmiar i kolor zastosowanej czcionki. We właściwościach pola można także zablokować dane pole przed zmianami. W formularzu *FStarty* zablokowano możliwość wpisania nowego imienia i nowego nazwiska, ponieważ te dane wprowadza się w innym formularzu. Wartości pól *Imie* i *Nazwisko* pochodzą z tabeli *Zawodnicy*. Wartości wprowadzane do pozostałych pól są zapisywane w tabeli *Starty*.



Rys. 7. *Formularz* FStarty *z podformularzem* Starty *– podformularz przygotowany na podstawie tabel* Zawodnicy *i* Starty *w* ***Widoku projektu***

Ćwiczenie 4. Przeglądamy gotowe projekty formularzy w bazie *Zawody*

1. W pliku *Zawody*, który otrzymasz od nauczyciela, otwórz formularze *FZawodnicy* i *FStarty* w **Widoku projektu**.
2. Wskaż pola pobrane z poszczególnych tabel. Z jakiej tabeli wybrano pola *Imie* i *Nazwisko*?
3. Zmodyfikuj formularze, zmieniając kolor i rozmiar czcionki w nagłówkach formularzy i etykietach według własnego pomysłu.
4. Zobacz formularze w **Widoku formularza** po modyfikacji. Spróbuj zmienić nazwisko lub imię wybranego zawodnika w formularzu *FStarty*. Czy jest to możliwe? Uzasadnij odpowiedź.

3. Przygotowanie formularzy w bazie *Stomatolog*

Stomatolog, dla którego przygotowujemy bazę danych, chce za pomocą formularzy wprowadzać informacje o pacjentach i ich wizytach.
W jaki sposób przygotować formularze, aby spełniły określone wcześniej wymagania?

Aby zaplanować formularze w bazie *Stomatolog*, musimy wrócić do wymagań bazy danych i wstępnego projektu formularza wykonanego w temacie B1. Na podstawie tego projektu i formularza *FZawodnicy* z bazy *Zawody* przygotujemy formularz wprowadzania danych pacjenta.

Ćwiczenie 5. Przygotowujemy formularz do wprowadzania danych pacjenta

1. W bazie *Stomatolog* utworzonej w temacie B1 utwórz formularz wprowadzania danych pacjenta. Skorzystaj z opisu sposobu przygotowania formularza *FZawody* w bazie *Zawody*.
2. Zmodyfikuj według uznania wygląd formularza utworzonego za pomocą kreatora.
3. Formularz nazwij *FPacjenci*.
4. Zapisz plik pod tą samą nazwą.

Wskazówka: Jeśli podczas tworzenia formularza z użyciem kreatora nie wstawiła się etykieta z tytułem formularza, należy powiększyć obszar nagłówka formularza, wstawić etykietę i wpisać do niej tekst.

Zanim zaczniemy wprowadzać informacje o wizytach, dane pacjentów muszą zostać wprowadzone do tabeli *Pacjenci*.

Nazwisko	Imię	Kod pocztowy	Ulica i nr	Miejscowość	Telefon	PESEL
Kowalski	Piotr	50-022	Wiosenna 12/5	Wrocław	717654321	61052389012
Nowak	Ewa	50-020	Radosna 10/2	Wrocław	717654321	65061532167
Kozińska	Jolanta	45-098	Słoneczna 1	Wałbrzych	746543278	95080598743

Tabela 1. *Przykładowe fikcyjne dane pacjentów – ćwiczenie 6.*

Ćwiczenie 6. Wprowadzamy dane pacjenta, korzystając z formularza

1. Otwórz plik *Stomatolog* zapisany w ćwiczeniu 5. W tabeli 1. przedstawiono fikcyjne dane trojga pacjentów. Wprowadź te dane do bazy *Stomatolog*, korzystając z formularza *FPacjenci*. Sprawdź zawartość tabeli *Pacjenci*.
2. Zapisz plik pod tą samą nazwą.

Stomatolog, dla którego tworzymy bazę, chce mieć możliwość wprowadzania informacji o każdej wizycie danego pacjenta. W bazie *Stomatolog* utworzymy odpowiedni formularz. Możemy go przygotować podobne jak formularz do wprowadzania startów zawodników *FStarty* w bazie *Zawody*. W przypadku bazy *Stomatolog* chcemy móc przeglądać wizyty danego pacjenta i wprowadzać informacje o nowej wizycie. Jeśli tabele *Pacjenci* i *Wizyty* połączyliśmy relacją „jeden do wielu", to najlepiej będzie utworzyć formularz z podformularzem.

Założymy, że do stomatologa mogą przyjść osoby o tym samym nazwisku i imieniu, więc na formularzu wizyt umieścimy dodatkowo (poza imieniem i nazwiskiem) numer PESEL do identyfikacji pacjenta.

Ćwiczenie 7. Przygotowujemy formularz do wprowadzania danych pacjenta

1. Otwórz plik *Stomatolog* zapisany w ćwiczeniu 6. Utwórz formularz wprowadzania danych o wizycie. Skorzystaj z opisu przygotowania formularza *FZawody* w bazie *Zawody*.
2. Formularz nazwij *FWizyty*, a podformularz – *FWizytyPacjenta*.
3. Zmodyfikuj wygląd formularza utworzonego za pomocą kreatora według uznania.
4. Zapisz plik pod tą samą nazwą.

Wskazówki:

- Możesz przygotować najpierw odręczny projekt formularza.
- Nie trzeba umieszczać na formularzu numeru identyfikacyjnego z tabeli *Pacjenci*. Natomiast warto umieścić numer PESEL, który służy do identyfikacji pacjentów.

Ćwiczenie 8. Wprowadzamy informacje o wizytach pacjentów z wykorzystaniem formularza

1. Otwórz plik *Stomatolog* zapisany w ćwiczeniu 7. Korzystając z formularza *FWizyty*, wprowadź fikcyjne dane o wizytach pacjentów dodanych do bazy w ćwiczeniu 6. Zapisz tabelę.
2. Uporządkuj rekordy w formularzu *FWizyty* według nazwiska pacjenta.
3. Sprawdź zawartość obydwu tabel bazy.
   a. Czy mamy tu do czynienia z przetwarzaniem danych? Uzasadnij odpowiedź i wyjaśnij, z czym jest związane wprowadzenie danych nowego pacjenta, a z czym wprowadzenie informacji o wizycie pacjenta.
   b. Czy po uporządkowaniu rekordów w formularzu zmienił się układ rekordów w tabeli *Pacjenci*? Uzasadnij odpowiedź.
4. Zapisz plik pod tą samą nazwą.

Wskazówka: Przykładowe rodzaje wizyt: ekstrakcja, leczenie kanałowe, kontrola uzębienia, plombowanie, konsultacja.

Porządkując (sortując) rekordy według wybranego pola, **przetwarzamy dane**.

4. Importowanie danych z innych dokumentów do tabeli bazy danych

Dane są często przechowywane w dokumentach różnego typu, np. w tabelach arkusza kalkulacyjnego lub w dokumentach tekstowych. Po odpowiednim przygotowaniu takiego dokumentu dane można **importować** do nowej tabeli bazy danych lub dołączać do istniejącej tabeli. Dane można również **eksportować** z tabeli bazy danych do tabeli arkusza kalkulacyjnego lub do pliku tekstowego.

Jeśli importujemy dane z tabeli arkusza kalkulacyjnego lub z dokumentu tekstowego do istniejącej tabeli bazy danych, należy pierwszy wiersz tabeli arkusza lub dokumentu tekstowego przeznaczyć na nazwy pól, które muszą być takie same jak nazwy pól w tabeli bazy.

A1 | Nazwisko

	A	B	C	D	E	F	G
1	Nazwisko	Imie	KodPocztowy	UlicaNr	Miejscowosc	NrTelefonu	Pesel
2	Zając	Tomasz	53-123	Powstańców Śl 132/7	Wrocław	60 0112223	61021298712
3	Nowak	Agnieszka	43-234	Budziszyńska 12/1	Wrocław	897 897 324	60021298741
4	Wrona	Bolesław	11-234	Paczkowa 2/9	Wrocław	11 9900199	77041298712
5	Kalarus	Krzysztof	45-080	M.Buczka 111	Wałbrzych	70 0118223	80051208712
6	Kminek	Franciszka	34-124	Pabianicka 32/9	Oleśnica	066 897 324	88042298727
7	Krajewski	Kacper	52-300	Zielona 98	Wrocław	11 2340011	86021298712
8	Szymczak	Barbara	54-345	Kamienna 98	Wrocław	70 9004423	89021448725

Rys. 8. *Przykładowe fikcyjne dane pacjentów zapisane w tabeli arkusza kalkulacyjnego – ćwiczenie 9.*

Przykład 1. Importowanie danych z arkusza kalkulacyjnego do istniejącej tabeli bazy danych

Stomatolog, dla którego tworzymy system, część danych o pacjentach zapisał w tabeli arkusza kalkulacyjnego. W tabeli *Pacjenci* wprowadziliśmy już trzy rekordy (w ćwiczeniu 6.). Teraz dodamy kolejne rekordy do istniejącej tabeli.

1. Otwieramy plik bazy danych, natomiast nie otwieramy tabeli, do której będziemy importować dane.
2. Wybieramy opcję pobierania danych zewnętrznych (importowania) i w otwartym oknie **Pobieranie danych zewnętrznych** wybieramy:
   a. plik, w którym są zapisane dane,
   b. sposób i miejsce wstawienia danych (do istniejącej lub do nowej tabeli) – w przypadku istniejącej tabeli wybieramy nazwę tabeli (tu: *Pacjenci*) z listy wyboru.
3. Otworzy się okno **Kreatora importu arkuszy**. Program poprowadzi nas przez kolejne kroki – wystarczy czytać polecenia i je wykonywać, m.in.: kliknąć pole **Pierwszy wiersz zawiera nagłówki kolumn**, jeśli program sam go nie zaznaczył.

Uwaga: Polecenie importowania danych zewnętrznych można znaleźć na karcie **Dane zewnętrzne** (grupa **Importowanie i łączenie**), opcja **Nowe źródło danych/z pliku/Excel** – gdy importujemy dane z pliku arkusza kalkulacyjnego.

W starszych wersjach programu polecenia importowania danych można znaleźć w menu **Plik** (opcja **Pobierz dane zewnętrzne/Importuj**).

Ćwiczenie 9. Importujemy dane z arkusza kalkulacyjnego do istniejącej tabeli bazy danych

1. Do tabeli *Pacjenci* w bazie *Stomatolog* zaimportuj dane z arkusza kalkulacyjnego zapisane w pliku, który otrzymasz od nauczyciela. Sprawdź zawartość tabeli *Pacjenci*.
2. Przejrzyj rekordy, korzystając z formularza *FPacjenci*. Odszukaj opcję sortowania i posortuj rekordy według nazwiska pacjenta.
3. Zapisz plik pod tą samą nazwą.

Uwaga: Zauważ, że w tabeli arkusza kalkulacyjnego (rys. 8.) nie ma pola *NrPacjenta* – wartości tego pola zostaną wprowadzone automatycznie przez program Microsoft Access.

W przypadku importowania danych do nowej tabeli, nazwy i typy pól można określić w oknie kreatora. W oknie kreatora możemy również zlecić, by program sam dodał pole klucza podstawowego do tabeli, wypełniając je odpowiednimi wartościami.

Warto zapamiętać

- W programie Microsoft Access wszystkie obiekty bazy danych (tabele, formularze, zapytania, raporty) oraz wszystkie dane wprowadzone do tabel są zapisywane w jednym pliku.
- Formularze służą do wprowadzania danych i przeglądania danych.
- W programie Microsoft Access możemy tworzyć formularze standardowe i formularze z podformularzem.
- Do tabel baz danych wykonanych w programie Microsoft Access można importować (pobierać) dane z innych dokumentów, przygotowanych np. w arkuszu kalkulacyjnym.
- Wprowadzając nowe dane (nowe rekordy) do tabel baz danych i porządkując (sortując) rekordy według wybranego pola, przetwarzamy dane.

Pytania i polecenia

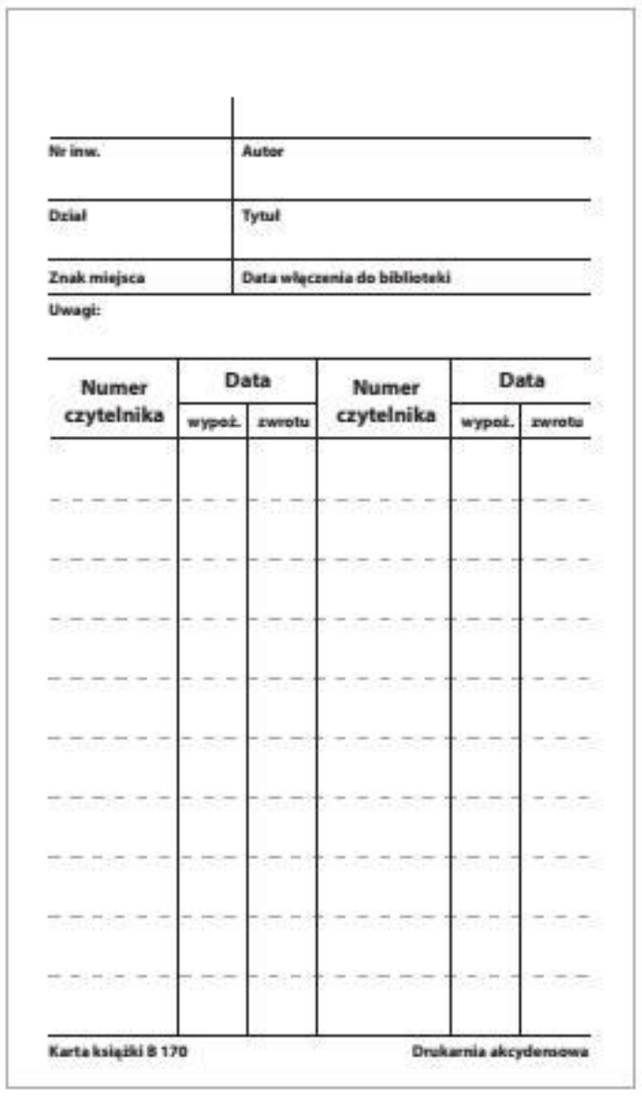
1. Do czego służą formularze?
2. Czym różni się formularz standardowy od formularza z podformularzem?
3. W jaki sposób należy przygotować formularz z podformularzem?
4. Na czym polega przetwarzanie danych w przykładowej bazie danych *Zawody*?
5. Otrzymałeś listę z danymi nowych pacjentów, zapisanymi w tabeli arkusza kalkulacyjnego. Omów, w jaki sposób należy umieścić te informacje w tabeli *Pacjenci* w bazie *Stomatolog*.

Zadania

1. Otwórz plik *Zawody* zapisany w ćwiczeniu 3. Korzystając z odpowiedniego formularza, wprowadź informacje dla wybranych zawodników o dwudziestu startach stylem dowolnym na dystansie 50 m w zawodach z okazji Dnia Dziecka z dnia 1 czerwca bieżącego roku. Zapisz plik pod tą samą nazwą.
2. Otwórz plik *Magazyn dżemów*, zapisany w zadaniu 1. z tematu B1. Zaprojektuj formularze do wprowadzania danych klientów *(FKlienci)* i realizacji zamówień *(FZamowienie*). Zapisz plik pod tą samą nazwą.
3. Otwórz plik bazy danych zapisany w zadaniu 3. z tematu B1. Zaprojektuj formularze do wprowadzania odpowiednich danych. Zapisz utworzone formularze pod odpowiednimi nazwami. Zapisz plik bazy danych pod tą samą nazwą.

Dla zainteresowanych

4. Otwórz plik *Klinika zwierząt* zapisany w zadaniu 4. z tematu B1. Zaprojektuj formularze do wprowadzania danych. Zapisz utworzone formularze pod odpowiednimi nazwami. Zapisz plik bazy danych pod tą samą nazwą.
5. Zaprojektuj bazę danych: ustal zbiory informacji, zastanów się nad rzeczywistymi powiązaniami między grupami informacji, określ wymagania bazy, zaplanuj pola i tabele. Zdefiniuj odpowiednie klucze i relacje. Utwórz potrzebne formularze. Wybierz temat:



Nr inw.	Autor
Dział	Tytuł
Znak miejsca	Data włączenia do biblioteki

Uwagi:

Numer czytelnika	Data wypoż.	Data zwrotu	Numer czytelnika	Data wypoż.	Data zwrotu

Karta książki B 170 — Drukarnia akcydensowa

a. sekretariat szkolny (np. informacje o klasach, przedmiotach, nauczycielach, wychowawcach),
b. biblioteka szkolna (na rysunku 9. pokazano wzór tradycyjnej karty bibliotecznej),
c. hurtownia kosmetyków (np. informacje o dostawcach, kosmetykach, realizacji dostaw),
d. magazyn książek (np. informacje o książkach, odbiorcach, wydaniu z magazynu).

Rys. 9. *Tradycyjna karta biblioteczna – zadanie 5b*

Temat B3. Wykonywanie podstawowych operacji na relacyjnej bazie danych

1. Wprowadzenie
2. Zastosowanie filtrów do wyszukiwania danych
3. Zasady tworzenia kwerend na przykładzie bazy *Zawody*
   - 3.1. Tworzenie kwerend wybierających
   - 3.2. Tworzenie kwerend parametrycznych
4. Zasady tworzenia raportów na przykładzie bazy *Zawody*
5. Przygotowywanie kwerend i raportów w bazie *Stomatolog*
6. Przygotowywanie korespondencji seryjnej

Warto powtórzyć

1. Do czego wykorzystywaliśmy filtry w arkuszu kalkulacyjnym?
2. W jaki sposób wyszukujemy informacje w internetowych zasobach danych?
3. Jakie operacje wykonujemy, przetwarzając dane?
4. Na jakim etapie tworzenia relacyjnej bazy danych przygotowujemy zestawienia (kwerendy i raporty)?
5. Co stanowi podstawę utworzenia relacji między dwiema tabelami bazy danych?
6. Na czym polega relacja „jeden do wielu"?

1. Wprowadzenie

Wśród operacji wykonywanych na relacyjnej bazie danych ważną rolę odgrywają zestawienia, które umożliwiają wyszukanie potrzebnych danych i ich wyświetlenie lub wydrukowanie.

Z możliwości wyszukiwania informacji w bazach danych (głównie w internetowych) korzystamy często w życiu codziennym. Gdy wpisujemy hasło w wyszukiwarce internetowej, wyświetla się lista adresów stron, na których zostało ono znalezione. Jeśli bibliotekarz szuka książek konkretnego autora, wpisuje nazwisko w bazie danych do obsługi biblioteki i otrzymuje wykaz wszystkich książek tego pisarza zgromadzonych w bazie.

W tym temacie pokażemy, w jaki sposób wyszukać i wyświetlić dane w bazie danych tworzonej w programie Microsoft Access, np. wyniki „Mikołajkowych zawodów pływackich".

2. Zastosowanie filtrów do wyszukiwania danych

Do prostego wyszukiwania danych, np. do znalezienia osób o konkretnym nazwisku lub samochodu konkretnej marki, można stosować **filtry**. Z filtrów korzystamy w podobny sposób, jak w arkuszu kalkulacyjnym. Kursor należy ustawić w polu, dla którego chcemy zastosować dany filtr.

Ćwiczenie 1. Korzystamy z filtrów

1. Otwórz plik zmodyfikowany w ćwiczeniu 9. z tematu B2, a w nim – formularz *FWizyty*.
2. Korzystając z filtrów, sprawdź wizyty Piotra Kowalskiego. Następnie wprowadź dla tego pacjenta informacje o dwóch nowych wizytach.
3. Sprawdź zawartość tabeli *Wizyty* (powinny pojawić się dwa nowe rekordy). Zwróć uwagę na wartości pola klucza obcego dla wprowadzonych rekordów – wartości te zostały dopisane automatycznie.
4. Zapisz plik pod tą samą nazwą.

3. Zasady tworzenia kwerend na przykładzie bazy *Zawody*

Za pomocą filtrów możemy wykonać tylko proste wyszukiwanie (według jednego pola, w jednej tabeli), natomiast **kwerendy** umożliwiają złożone wyszukiwanie według wielu pól z jednej tabeli lub kilku tabel.

SQL S
(z ang. *Structured Query Language*)

Strukturalny język zapytań wykorzystywany do obsługi baz danych.

Kwerenda (zapytanie) pobiera z bazy danych dane spełniające określone kryterium.
Kwerendy zapisywane są jako instrukcje języka **SQL**.

Program Microsoft Access ma narzędzia umożliwiające tworzenie kwerend. Nie musimy znać instrukcji języka SQL, aby tworzyć zapytania i wyszukiwać potrzebne dane.

3.1. Tworzenie kwerend wybierających

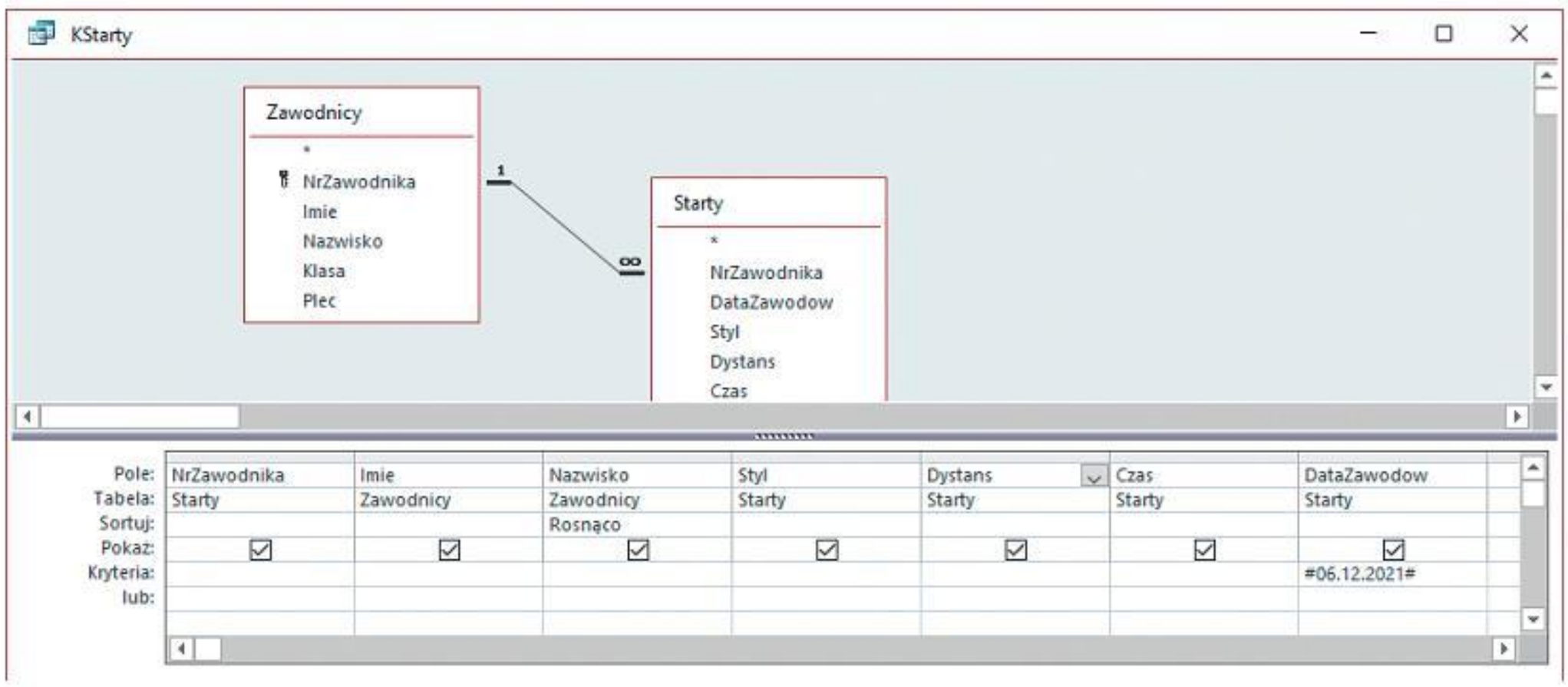
Najczęściej używanym rodzajem kwerend jest **kwerenda wybierająca**, która służy do wybierania danych z jednej tabeli lub z kilku tabel bazy danych i wyświetlania wyników w postaci arkusza danych. W wyniku zapytania powstaje tzw. **dynamiczny zestaw wyników**. Mimo że na ekranie widzimy tabelę z wynikami (arkusz danych), nie jest ona zapisywana w bazie danych. Przechowywany jest jedynie ciąg instrukcji języka SQL, pozwalający na utworzenie tej tabeli. Kwerendę możemy zapisać i wielokrotnie z niej korzystać.

Tabela, która powstaje w wyniku działania kwerendy, może być też źródłem danych dla formularza.

Kwerendę wybierającą możemy utworzyć na podstawie jednej tabeli lub kilku tabel. Źródłem danych dla kwerendy może być również inna kwerenda (lub kwerendy). Wybór konkretnych pól z każdej tabeli zależy od tego, jakie informacje chcemy uzyskać w wynikach kwerendy. Kryterium można określić dla jednego pola lub dla kilku pól.

Aby utworzyć kwerendę wybierającą, należy:

- wybrać opcję tworzenia kwerendy lub skorzystać z **Kreatora kwerend**,
- wybrać tabele lub kwerendy, na podstawie których będzie tworzona kwerenda,
- wybrać pola, które będą uwzględnione w kwerendzie,
- wpisać kryterium dla danego pola lub kryteria dla kilku pól,
- nazwać i zapisać kwerendę.



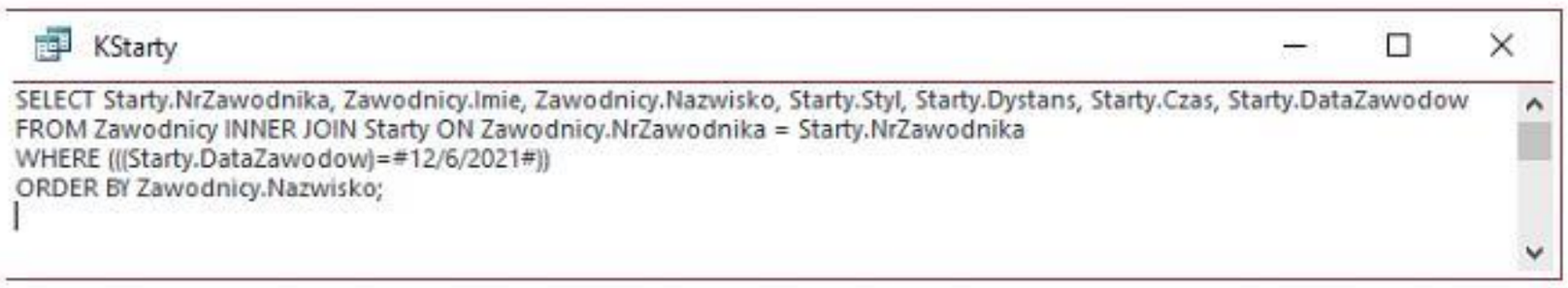
Rys. 1a. *Przykładowa kwerenda utworzona w bazie* Zawody *w* ***Widoku projektu***

KStarty

NrZawodnika	Imie	Nazwisko	Styl	Dystans	Czas	DataZawodow
26	Krzysztof	Figurski	klasyczny	50	38,11	06.12.2021
26	Krzysztof	Figurski	grzbietowy	50	32,44	06.12.2021
4	Anna	Janiak	dowolny	50	40,58	06.12.2021
6	Bolesław	Kafka	klasyczny	50	36,45	06.12.2021
6	Bolesław	Kafka	grzbietowy	50	35,12	06.12.2021
21	Kornelia	Kowalska	klasyczny	50	40,82	06.12.2021
21	Kornelia	Kowalska	dowolny	50	40,58	06.12.2021
18	Michał	Matyga	grzbietowy	50	35,12	06.12.2021
9	Zofia	Sołtys	grzbietowy	50	35,15	06.12.2021
12	Teresa	Zając	motylkowy	50	50,01	06.12.2021

Rekord: 1 z 10 Bez filtru Wyszukaj

Rys. 1b. *Wynik działania kwerendy wybierającej*



Rys. 1c. *Instrukcja języka SQL wykonująca kwerendę*

Przykład 1. Tworzenie kwerendy wybierającej na podstawie dwóch tabel

Zgodnie z wymaganiami bazy *Zawody*, chcemy m.in. znać zawodników uczestniczących w „Mikołajowych zawodach pływackich" wraz ze wszystkimi konkurencjami, w których startowali, i czasami, które osiągnęli.
Do tak postawionego pytania potrzebne są dane z obydwu tabel: *Zawodnicy* i *Starty*.
Kwerendę utworzymy w **Widoku projektu**.

1. Po wybraniu tworzenia kwerendy w **Widoku projektu** otworzy się okno podobne do okna pokazanego na rys. 1a. Jako źródło danych wybieramy tabele: *Zawodnicy* i *Starty*.
2. Wybór konkretnych pól z każdej tabeli zależy od tego, jakie informacje chcemy uzyskać w wyniku działania kwerendy. Przede wszystkim mają to być informacje o startach, więc z tabeli *Starty* wybieramy pola: *NrZawodnika*, *DataZawodow*, *Styl*, *Dystans*, *Czas*.
3. Brakujące dane (czyli pola *Nazwisko*, *Imie*) dobierzemy z tabeli *Zawodnicy*.
4. Chcemy otrzymać starty z 6 grudnia 2021 roku, dlatego jako kryterium dla pola *DataZawodow* wpiszemy: 2021-12-06.
5. Nazywamy kwerendę zgodnie z przyjętą konwencją *KStarty*.

Uwaga: Jeśli korzystamy z **Kreatora kwerend**, kryteria dla pól wpisujemy w **Widoku projektu**.

Ćwiczenie 2. Przeglądamy projekt i wyniki działania kwerendy

1. W pliku *Zawody*, który otrzymasz od nauczyciela, wyświetl wyniki kwerendy *KStarty*.
2. Przejdź do **Widoku projektu** i wskaż pola pobrane z poszczególnych tabel. Z jakiej tabeli wybrano pola *Imie* i *Nazwisko*?
3. Zapisz plik pod tą samą nazwą.

3.2. Tworzenie kwerend parametrycznych

Zgodnie z wymaganiami bazy *Zawody* chcemy znać kolejność zawodników (od najlepszego czasu) w każdej konkurencji wraz z czasami – oddzielnie dla dziewcząt i chłopców. Jak utworzyć kwerendę, która umożliwi wyszukanie takich danych?

Aby wyszukać zawodników (np. chłopców), którzy startowali w konkretnej konkurencji (stylem klasycznym 50 m) w danych zawodach, możemy utworzyć prostą kwerendę wybierającą, której sposób tworzenia już poznaliśmy. Aby wyszukać dziewczęta, które startowały stylem dowolnym 50 m w tych samych zawodach, możemy zmodyfikować utworzoną kwerendę, zmieniając wartości kryteriów.

Ćwiczenie 3. Tworzymy kwerendę wybierającą

1. W pliku *Zawody*, który otrzymasz od nauczyciela, utwórz kwerendę wyświetlającą wyniki uporządkowane od najlepszego czasu dla chłopców startujących na 50 m stylem klasycznym w dniu 6 grudnia 2021 r. Zapisz kwerendę pod nazwą *KMikołajkowe_2021*. Wyświetl wyniki kwerendy.
2. Zmodyfikuj kwerendę *KMikołajkowe_2021*, aby wyświetlała wyniki dziewcząt startujących na 50 m stylem dowolnym. Zapisz kwerendę pod tą samą nazwą. Wyświetl wyniki kwerendy.
3. Zapisz plik pod nazwą *Zawody _1*.

Wskazówka: Aby posortować rekordy według konkretnego pola, należy wybrać w projekcie kwerendy odpowiedni sposób porządkowania w polu **Sortuj** dla tego pola.

Zastanówmy się teraz, czy wygodne jest wielokrotne modyfikowanie tej samej kwerendy (lub tworzenie nowej) dla każdej konkurencji (w dodatku oddzielnie dla dziewcząt i chłopców).

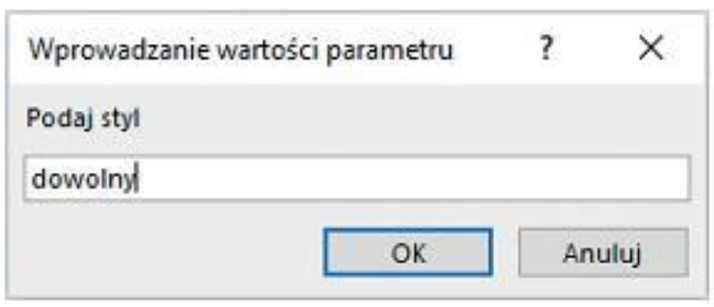
Skoro utworzona kwerenda wybiera wartości z określonych pól (tu: *Styl*, *Dystans*, *Plec*), to lepiej by było, gdyby umożliwiała wybór wartości danego pola przy każdym uruchomieniu.

Jeśli określona kwerenda ma być uruchamiana dla różnych wartości danego pola, można zastosować kwerendę z parametrami, czyli **kwerendę parametryczną**.

Aby zastosować w kwerendzie parametry, należy w polu **Kategoria** zamiast konkretnej wartości (np. „dowolny” dla pola *Styl*) wprowadzić komunikat (np. [*Podaj styl*]) – rys. 2a.
Po uruchomieniu kwerenda wyświetli okienko dialogowe z pytaniem o wartość danego pola (rys. 2b), a następnie utworzy kryteria kwerendy na podstawie podanej wartości.

Pole:	Imie	Nazwisko	DataZawodow	Styl	Dystans	Czas	Plec
Tabela:	Zawodnicy	Zawodnicy	Starty	Starty	Starty	Starty	Zawodnicy
Sortuj:						Rosnąco	
Pokaż:	☑	☑	☑	☑	☑	☑	☑
Kryteria:			#2021-12-06#	[Podaj styl]	[Podaj dystans]		[Wpisz K lub M]
lub:							

Rys. 2a. *Przykład kwerendy parametrycznej – parametry określono dla trzech pól*



Rys. 2b. *Okno dialogowe, które pojawi się po uruchomieniu kwerendy parametrycznej*

Ćwiczenie 4. Tworzymy kwerendę parametryczną

1. W pliku *Zawody_1* zapisanym w ćwiczeniu 3. zmodyfikuj kwerendę *KMikołajkowe_2021*, aby umożliwiała wyświetlanie zawodników w kolejności od najlepszego czasu w każdej konkurencji wraz z czasami na danych zawodach – oddzielnie dziewcząt i chłopców. Ustal nazwę stylu, dystans i płeć jako parametry.
2. Zapisz kwerendę pod tą samą nazwą. Wyświetl wyniki kwerendy dla różnych wartości parametrów.
3. Zapisz plik pod tą samą nazwą.

Wyszukując dane według wybranego pola (wybranych pól), **przetwarzamy dane**.

4. Zasady tworzenia raportów na przykładzie bazy *Zawody*

Raporty służą do przedstawiania danych zawartych w tabelach i otrzymywanych w wyniku działania kwerendy. Podobnie jak w przypadku formularza, bardzo ważne jest przygotowanie czytelnej formy raportu.

Jeśli chcemy przedstawić (wyświetlić, wydrukować) zestawienie wszystkich rekordów, przygotowujemy raport na podstawie tabeli (tabel). Natomiast gdy chodzi o zestawienie wybranych rekordów, przygotowujemy raport na podstawie kwerendy (kwerend).

Raporty możemy tworzyć samodzielnie, wykorzystując odpowiednie opcje programu, lub korzystać z kreatora.

Aby utworzyć raport za pomocą kreatora, należy w oknie **Kreatora raportów**:
- wybrać tabele lub kwerendy, na podstawie których będzie tworzony raport oraz pola, które mają być umieszczone w raporcie,
- ustalić inne elementy, m.in.: sposób wyświetlania danych, porządek sortowania, układ raportu,
- nazwać i zapisać raport.

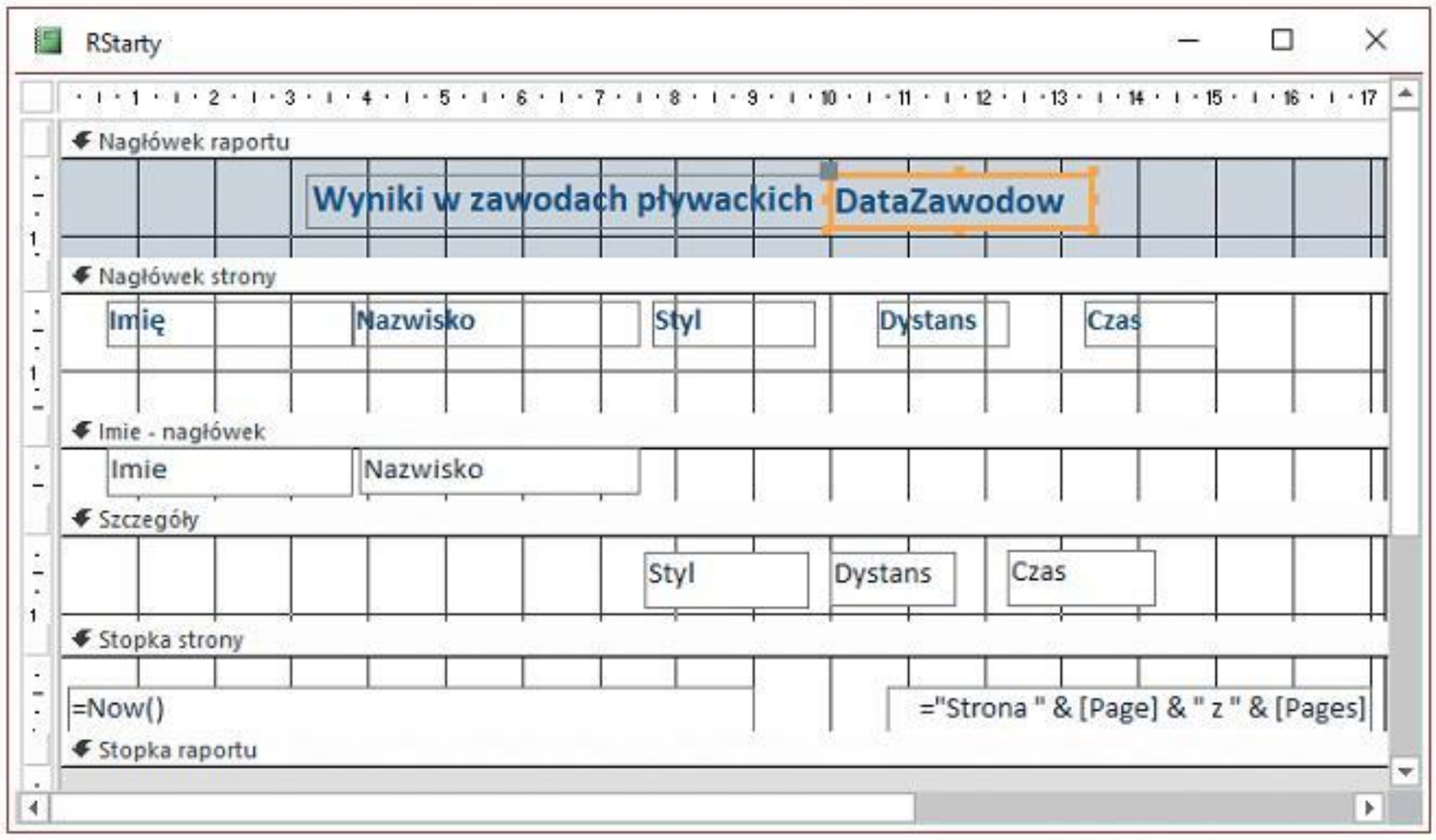
Przykład 2. Przygotowanie raportu na podstawie kwerendy

Chcemy wyświetlać listę zawodników uczestniczących w Mikołajkowych zawodach pływackich wraz ze wszystkimi konkurencjami, w których startowali, i czasami, które osiągnęli. Ponieważ w bazie *Zawody* utworzono kwerendę wyświetlającą takie wyniki, możemy na jej podstawie przygotować odpowiedni raport.

1. Skorzystamy z **Kreatora raportów**, jako źródło danych wybierając kwerendę *KStarty*.
2. Wybieramy pola, które mają być umieszczone w raporcie (nie wyświetlamy w raporcie numeru identyfikacyjnego *NrZawodnika*).
3. Aby dla danego zawodnika wyświetlić dane o jego startach, wybieramy sposób wyświetlania danych: **przez *Zawodnicy*** (podobnie jak było w przypadku formularza z podformularzem *FStarty*).
4. Czytamy i wykonujemy kolejne polecenia kreatora, m.in. ustalamy porządek sortowania (np. według czasu) oraz układ raportu.
5. Wpisujemy tytuł raportu zgodnie z przyjętą konwencją: *RStarty*. Raport będzie widoczny pod tą nazwą w wykazie obiektów bazy.

Uwagi:

- Po przygotowaniu raportu z wykorzystaniem kreatora należy go dopracować w **Widoku projektu** (raport można przygotowywać od razu w tym widoku, wybierając odpowiednią opcję).
- W raporcie pokazanym na rysunku 3a pole *DataZawodów* umieszczono w nagłówku i usunięto etykietę pola. W ten sposób data wyświetla się tylko w tym miejscu, a nie przy danych każdego zawodnika (rys. 3b). Dodatkowo we właściwościach pola zmieniono format wyświetlania daty oraz rozmiar i kolor czcionki.



Rys. 3a. *Przykładowy raport w* ***Widoku projektu***

RStarty

Wyniki w zawodach pływackich 6 grudnia 2021

Imię	Nazwisko	Styl	Dystans	Czas
Krzysztof	Figurski			
		grzbietowy	50	32,44
		klasyczny	50	38,11
Anna	Janiak			
		dowolny	50	40,58
Bolesław	Kafka			
		grzbietowy	50	35,12
		klasyczny	50	36,45
Kornelia	Kowalska			
		dowolny	50	40,58
		klasyczny	50	40,82
Maria	Leszczyńska			
		klasyczny	50	47,88
Michał	Matyga			
		grzbietowy	50	35,12

Rys. 3b. *Przykładowy raport (fragment) w* ***Podglądzie wydruku***

Raport *RStarty*, który utworzyliśmy na podstawie kwerendy wyświetlającej informacje o startach z 6 grudnia 2021 r. (*KStarty*), może służyć do wyświetlenia również innych wartości pól umieszczonych w projekcie raportu, np. z innego dnia.

Ćwiczenie 5. Wyświetlamy raport przygotowany na podstawie kwerendy

1. W pliku *Zawody*, który otrzymasz od nauczyciela, wyświetl raport *RStarty*.
2. Przejdź do **Widoku projektu** i wskaż pola pobrane z poszczególnych tabel. Z której tabeli wybrano wszystkie pola?
3. Zmodyfikuj kwerendę *KStarty*, aby wyświetlała rekordy z danymi zawodników uczestniczących w zawodach 21 marca 2021 roku. Wyświetl wyniki działania kwerendy oraz raport. Co zauważasz?
4. Nie zapisuj zmian.

Ćwiczenie 6. Tworzymy raport na podstawie kwerendy parametrycznej

1. Otwórz plik *Zawody_1* zapisany w ćwiczeniu 4. i utwórz raport na podstawie kwerendy parametrycznej *K_Mikołajkowe_2021*. Zapisz raport pod nazwą *R_ Mikołajkowe_2021*.
2. Wyświetl raport dla różnych parametrów.
3. Zapisz plik pod tą samą nazwą.

Prezentując dane (wyświetlając, drukując) w postaci raportów, **przetwarzamy dane**.

5. Przygotowywanie kwerend i raportów w bazie *Stomatolog*

W bazie *Stomatolog* chcemy przygotować odpowiednie zestawienia, które umożliwią lekarzowi wyszukiwanie i przedstawianie informacji o pacjentach i ich wizytach. W jaki sposób to zrobić?

Aby zaplanować zestawienia w bazie *Stomatolog*, musimy, podobnie jak w przypadku formularzy, wrócić do wymagań tej bazy danych i wstępnego projektu raportu wykonanego w temacie B1. Na tej podstawie należy przygotować odpowiednią kwerendę, tak aby można ją było wykorzystać do przygotowania potrzebnego raportu.

Jeśli stomatolog chce np. otrzymać informacje, którzy pacjenci mieli wizytę danego dnia, jakie zabiegi przeprowadzono i jaki był ich koszt (tzw. raport dzienny), możemy przygotować kwerendę i raport, wzorując się na kwerendzie *KStarty* i raporcie *RStarty* z bazy *Zawody*.

Ćwiczenie 7. Tworzymy kwerendę w bazie *Stomatolog*

1. Na podstawie określonych wcześniej wymagań i wzoru raportu, przygotuj kwerendę w bazie *Stomatolog*:
   a. wybierz odpowiednie tabele jako źródło danych, a z nich pola,
   b. określ odpowiednie kryterium (kryteria) dla pola (pól).
2. Zapisz kwerendę pod nazwą *KWizyty*.
3. Przejrzyj zestaw wyników zapytania w arkuszu danych.
4. Zapisz plik pod tą samą nazwą.

Wskazówka: Jeśli chcemy otrzymać wizyty z bieżącego dnia, do określenia kryterium dla pola z datami wizyt należy zastosować funkcję `Date()`, której wartością jest bieżąca data. Korzystając z formularza *FWizyty*, wprowadź kilka nowych rekordów z danymi o wizytach z bieżącego dnia, aby tabela z wynikami kwerendy nie była pusta.

Ćwiczenie 8. Tworzymy raport na podstawie kwerendy

1. W bazie *Stomatolog* utwórz raport przedstawiający wyniki działania kwerendy utworzonej w ćwiczeniu 7.
2. Zapisz raport pod nazwą *RWizyty* i obejrzyj go w **Podglądzie wydruku**.
3. Zapisz plik pod tą samą nazwą.

Wskazówka: Jeśli raport przedstawia wizyty pacjentów z konkretnego dnia, można w nagłówku raportu umieścić w tytule pole daty, podobnie jak w raporcie *RStarty* w bazie *Zawody*.

Jeśli utworzyliśmy raport na podstawie kwerendy wyświetlającej informacje o wizytach z bieżącego dnia, możemy wykorzystać go do wyświetlenia również innych wartości pól umieszczonych w projekcie raportu. Na przykład możemy wyświetlić wszystkie wizyty wszystkich pacjentów, wizyty pacjentów z danego dnia, wizyty danego pacjenta czy wizyty, podczas których wykonywano konkretny zabieg, np. plombowanie. Wystarczy tylko zmienić lub usunąć kryterium dla określonych pól w kwerendzie.

Ćwiczenie 9. Modyfikujemy istniejące kwerendy i raporty lub tworzymy nowe

1. Stomatolog chciałby otrzymać wykaz wszystkich wizyt wszystkich pacjentów z danego dnia. Zastanów się, czy możesz zmodyfikować kwerendę utworzoną w ćwiczeniu 7., aby spełnić te wymagania. Jeśli tak – odpowiednio ją zmodyfikuj. Jeśli nie – utwórz nową.
2. Korzystając z przygotowanej w punkcie 1. kwerendy, wyświetl wszystkie wizyty wszystkich pacjentów z danego dnia w postaci raportu. Jeśli to konieczne, zmodyfikuj projekt raportu lub utwórz nowy.
3. Zapisz plik pod tą samą nazwą.

Można również tworzyć kwerendy, w których nie określamy konkretnych kryteriów dla pól, tylko wybieramy określone pola z bazy.

Ćwiczenie 10. Tworzymy kwerendę bez określania kryteriów dla pól

1. W bazie *Stomatolog* utwórz kwerendę, w wyniku której otrzymamy następujące informacje o pacjentach: imię, nazwisko, numer PESEL. Zapisz kwerendę pod nazwą *KPesele*.
2. Zapisz plik pod tą samą nazwą.

Jeśli chcemy wyświetlić listę wszystkich pacjentów w postaci raportu, możemy przygotować go na podstawie kwerendy utworzonej w ćwiczeniu 10. lub na podstawie tabeli *Pacjenci* z danymi pacjentów.

Ćwiczenie 11. Tworzymy raport wyświetlający dane pacjentów

1. W bazie *Stomatolog* utwórz raport wyświetlający imię, nazwisko i numer PESEL wszystkich pacjentów. Wybierz odpowiednie źródło danych. Zapisz raport pod nazwą *RPesele*.
2. Zapisz plik pod tą samą nazwą.

6. Przygotowywanie korespondencji seryjnej

Stomatolog, dla którego tworzymy bazę danych, chce wysłać do swoich pacjentów listy z informacją o zmianie adresu gabinetu. Jak to zrobić z wykorzystaniem narzędzi informatycznych?

W różnych instytucjach czasem wysyła się tę samą informację (powiadomienie, przypomnienie, zarządzenie, zaświadczenie) do wielu osób. Dawniej listy pisano odręcznie, potem używano maszyny do pisania, aż wreszcie – komputera i edytora tekstu.

Nawet jeśli treść listu była taka sama, pozostawał problem z wpisaniem danych, które w każdym liście powinny być inne, np. imię i nazwisko. Zazwyczaj te dane dopisywano na każdym liście – ręcznie lub z wykorzystaniem edytora tekstu (co często nadal się robi). Jeśli listy wysyła się do dziesięciu osób, problem nie jest duży, ale gdy trzeba je skierować do tysiąca lub kilku tysięcy osób…

Zaczęto się zastanawiać, jak usprawnić te czynności. Skoro dane osobowe mogą być zapisane w komputerze (m.in. w tabelach baz danych), może wystarczy je połączyć z listem? Wymyślono więc sposób, który to umożliwia. Tym „łącznikiem" stało się pole tabeli bazy danych.

Program Microsoft Access nie ma własnych mechanizmów służących do przygotowywania korespondencji seryjnej (m.in. listów, etykiet adresowych, kopert), ale stanowi dobre narzędzie w połączeniu z edytorem tekstu. List piszemy w edytorze tekstu, a źródłem danych może być np. tabela bazy danych.



W przypadku korespondencji seryjnej dane z bazy nie są eksportowane do edytora tekstu – edytor tekstu pobiera dane z tabeli lub kwerendy i dołącza je do opracowanego przez nas szablonu listu lub etykiety adresowej.



Aby utworzyć korespondencję seryjną, należy:

- w edytorze tekstu wybrać opcję tworzenia korespondencji seryjnej (można skorzystać z kreatora),
- wybrać rodzaj korespondencji, np. listy, etykiety, koperty,
- wybrać źródło danych, np. bazę danych, a w niej tabelę,
- wstawić do dokumentu tekstowego odpowiednie pola z tabeli, np. nazwisko, imię, adres odbiorcy,
- wygenerować korespondencję (listy, etykiety, koperty) z wykorzystaniem danych pobranych z bazy danych.

Treść listu lub etykiety tworzymy w edytorze tekstu, umieszczając w niej odwołania do odpowiednich pól bazy danych, np. *Imie* i *Nazwisko* (rys. 4a). Po zakończeniu tworzenia korespondencji seryjnej (połączeniu dokumentu tekstowego ze źródłem danych), w miejscu wstawienia pól (rys. 4b) będą wyświetlały się konkretne imiona i nazwiska (rys. 4c).

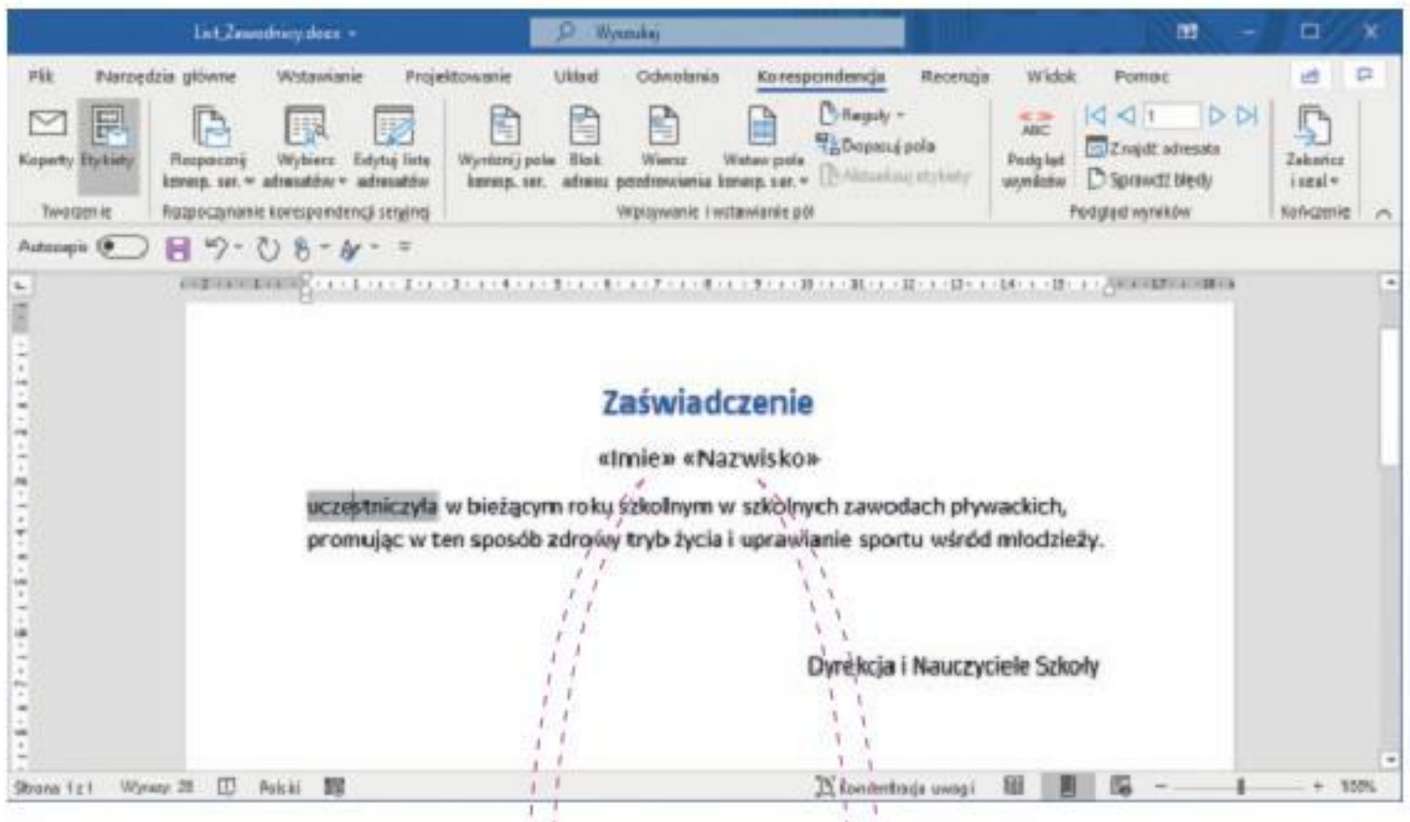
Poza bazą danych utworzoną w programie Microsoft Access źródłem danych dla korespondencji seryjnej mogą być np. kontakty z książki adresowej, utworzonej w programie Microsoft Outlook lub dane zapisane w tabeli arkusza kalkulacyjnego.

Opcje przygotowania korespondencji seryjnej oraz podglądu wyników znajdują się na karcie **Korespondencja**.

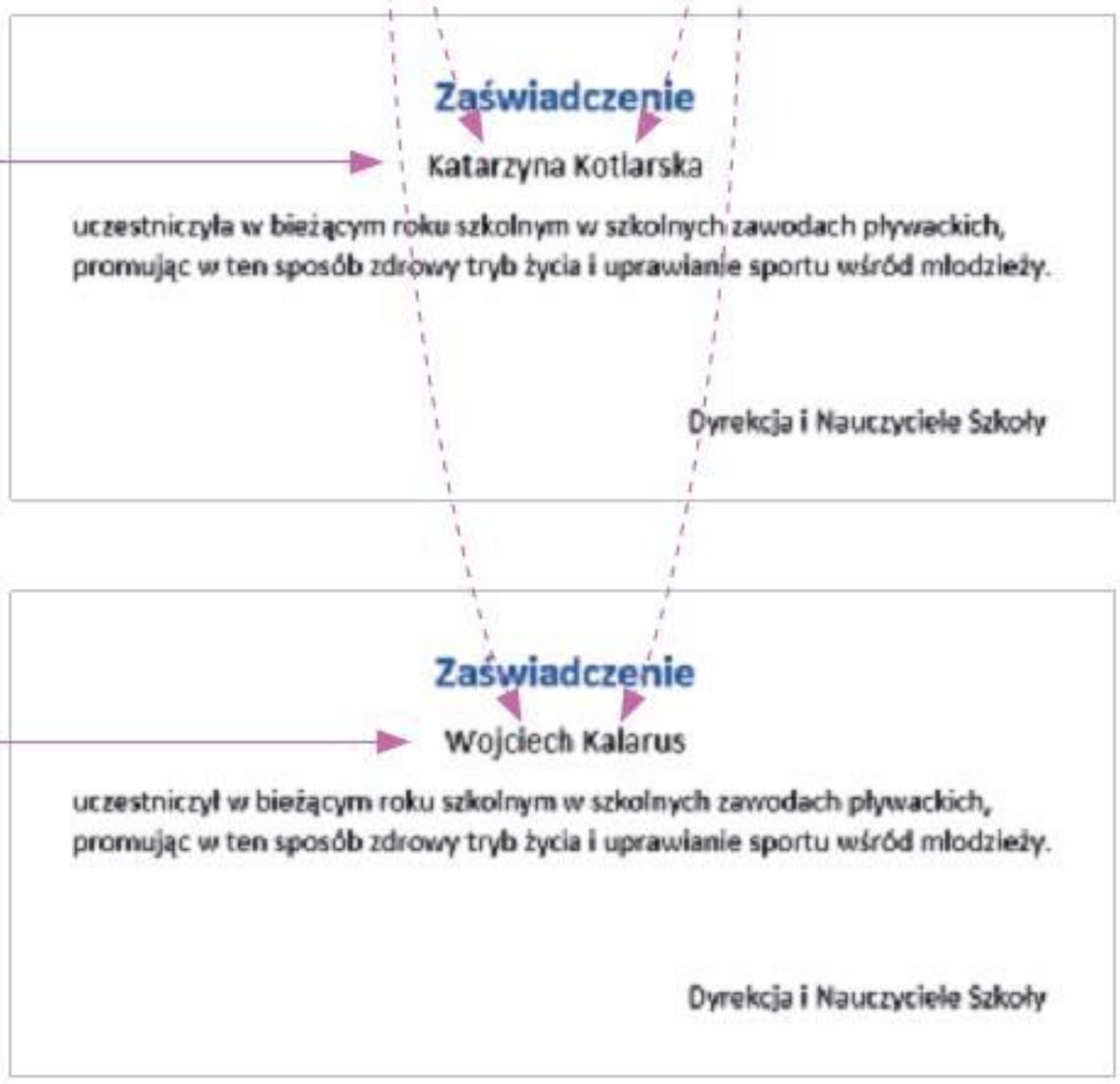
Zawodnicy

NrZawodnika	Imie	Nazwisko	Klasa	Plec	Kliknij, aby dodać
1	Katarzyna	Kotlarska	IA	K	
2	Wojciech	Kalarus	IB	M	
3	Michał	Grabowski	IB	M	
4	Anna	Janiak	IIA	K	
5	Barbara	Kolarska	IIB	K	
6	Bolesław	Kafka	IIB	M	
7	Jan	Krajewski	IIIC	M	
8	Maria	Leszczyńska	IIIC	K	
9	Zofia	Sołtys	IA	K	
10	Jakub	Krawczyk	IIA	M	
11	Stefan	Nowak	IIIC	M	

Rys. 4a. *Tabela* Zawodnicy, *z której pobierane są dane*



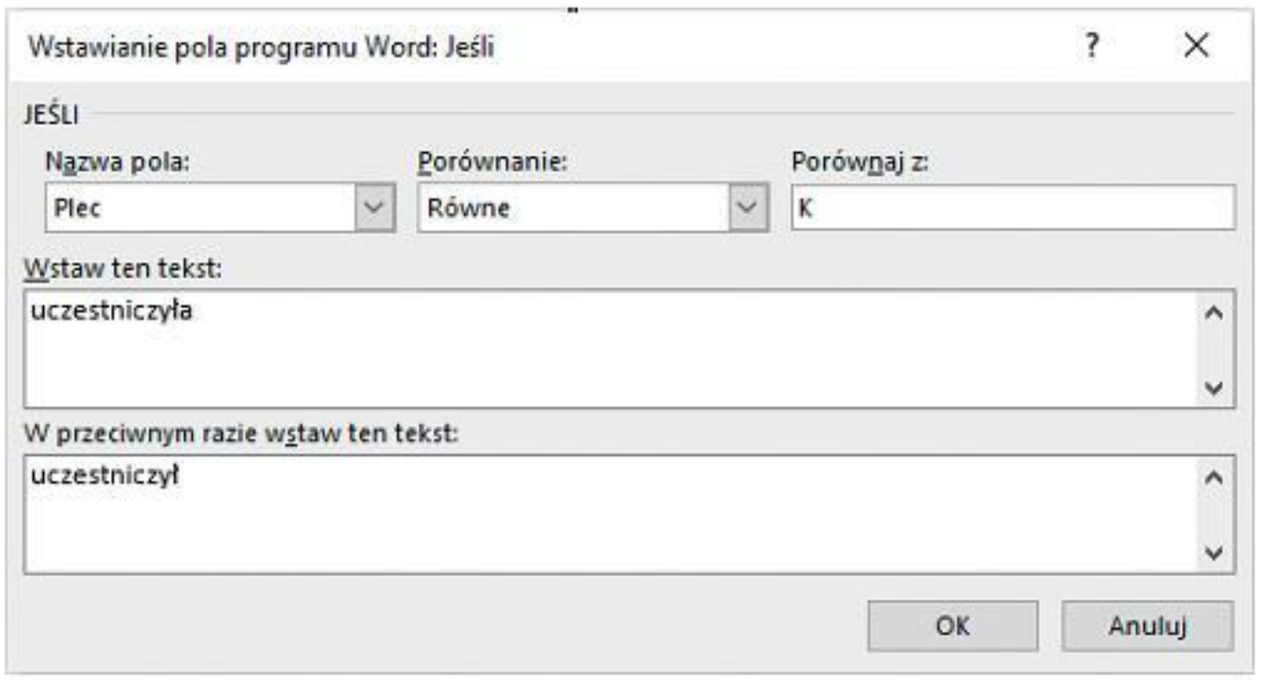
Rys. 4b. *List seryjny z widocznymi polami z tabeli bazy danych*



Rys. 4c. *Przykładowe listy seryjne (spersonalizowane) wypełnione danymi pobranymi z tabeli bazy danych*

Możemy skorzystać z gotowych szablonów listów seryjnych lub przygotować własny list.

Jeśli w liście seryjnym niektóre sformułowania chcemy spersonalizować (rozróżnić ze względu na płeć), możemy to zrobić, wybierając opcję **Reguły** (otworzy się okno jak na rysunku 5.). W utworzonym liście seryjnym zastosowano regułę do pola *Plec* i zależnie od wartości pola w liście pojawia się słowo „uczestniczyła" lub „uczestniczył".



Rys. 5. *Okno ustalania reguł (Microsoft Word)*

Ćwiczenie 12. Przygotowujemy list seryjny

1. W pliku *List_zawodnicy*, który otrzymasz od nauczyciela, zapisano przykładowy list seryjny. Otwórz go i przejrzyj w podglądzie wyników (lub podglądzie wydruku).
2. Przygotuj list seryjny do wszystkich pacjentów z bazy *Stomatolog* z informacją o zmianie adresu gabinetu stomatologicznego. Do treści listu wstaw w odpowiednie miejsce dane z tabeli *Pacjenci* z pól, w których zapamiętano imiona i nazwiska pacjentów.
3. Dodaj do tabeli pole płeć (jeśli wcześniej go nie było) i umieść przed imieniem i nazwiskiem pacjenta zwrot „Pani" lub „Pan" – zależnie od wartości tego pola.
4. Wydrukuj listy dla czterech pierwszych osób.
5. Zapisz dokument w pliku pod nazwą *Listy*.

Warto zapamiętać

- Kwerendy umożliwiają wyszukiwanie danych w bazie danych – kwerenda pobiera z bazy dane spełniające określone kryterium.
- Korzystając z kwerendy wybierającej, możemy pobrać dane z jednej tabeli lub z kilku tabel. Możemy ustalić kryterium dla jednego pola lub dla kilku pól. Jeśli określona kwerenda ma być uruchamiana dla różnych wartości danego pola, można zastosować kwerendę parametryczną.
- Za pomocą raportu prezentujemy dane zawarte w tabelach i otrzymane w wyniku działania kwerendy.

- W edytorze tekstu (np. Microsoft Word) można przygotowywać listy seryjne i etykiety adresowe, pobierając dane np. z tabel opracowanych w programie Microsoft Access.

Pytania i polecenia

1. Czym różni się wyszukiwanie danych za pomocą filtrów od zastosowania do tego celu kwerend?
2. W jaki sposób należy przygotować kwerendę wybierającą?
3. Podaj przykład zastosowania kwerendy parametrycznej.
4. Jakie jest zastosowanie raportów? W jaki sposób można szybko przygotować raport?
5. Wyjaśnij, w jaki sposób można przygotować list seryjny do osób, których dane zapisano w tabeli bazy danych.

Zadania

1. Utwórz w bazie *Zawody*, zapisanej w zadaniu 1. z tematu B2, kwerendę wybierającą z bazy danych wszystkich tych zawodników, którzy w stylu dowolnym na dystansie 50 m osiągnęli czas krótszy niż 37 sekund. Przygotuj na podstawie tej kwerendy odpowiedni raport. Zapisz plik pod tą samą nazwą.
2. Korzystając z raportu *RWizyty*, utworzonego w ćwiczeniu 8., wyświetl wszystkie wizyty pacjenta o nazwisku Kowalski.
3. W bazie *Stomatolog* wyszukaj:
   a. pacjentów, którym wykonano leczenie kanałowe lub plombowanie,
   b. pacjentów, którzy za wizytę zapłacili mniej niż 200 zł i mieszkają we Wrocławiu.

   W wyniku działania kwerendy wyświetl: nazwisko i imię pacjenta, datę wizyty, rodzaj wizyty i jej koszt.

   Wskazówka: Wprowadź kilka dodatkowych rekordów z odpowiednimi danymi.
4. W bazie *Stomatolog* utwórz kwerendę z parametrami i raport na jej podstawie.
5. Przygotuj list seryjny z życzeniami świątecznymi dla swoich koleżanek i kolegów, wykorzystując adresy z kontaktów e-mailowych. W treści listu umieść imię koleżanki lub kolegi. Wyślij list do kilku osób.

Dla zainteresowanych

6. Utwórz kwerendy i raporty w bazie danych utworzonej w zadaniu 5. z tematu B2.
7. Przygotuj etykiety adresowe dla pacjentów z bazy *Stomatolog*. W razie potrzeby skorzystaj z **Pomocy** programu Microsoft Word.
8. Zaprojektuj i utwórz bazę danych (zbiory informacji, tabele, pola, relacje, formularze, kwerendy i raporty), określając samodzielnie jej cel i wymagania.

Temat B4. Projektowanie modeli dwuwymiarowych i trójwymiarowych

1. Rodzaje grafiki – przypomnienie
2. Określenie problemu – zaprojektowanie ogrodu
3. Wybrane możliwości programu SketchUp przydatne do projektowania ogrodu
   - 3.1. Program SketchUp
   - 3.2. Przekształcanie modelu 2D w 3D
   - 3.3. Projektowanie trójwymiarowego modelu ławki
4. Tworzenie projektu ogrodu
   - 4.1. Model 2D ogrodu
   - 4.2. Model 3D ogrodu
5. Projektowanie własnego pokoju

Warto powtórzyć

1. Jakie poznaliśmy programy graficzne? Jaki rodzaj grafiki można w nich tworzyć?
2. Który z poznanych programów umożliwia rysowanie przestrzenne? W jaki sposób rysowaliśmy bryłę?
3. Wymień przykłady narzędzi do rysowania z poznanych programów graficznych, które mimo różnych nazw mają podobne działanie.

1. Rodzaje grafiki – przypomnienie

Ze względu na sposób reprezentowania obrazu możemy wyróżnić:

- grafikę dwuwymiarową (2D),
- grafikę trójwymiarową (3D).

Grafika dwuwymiarowa służy do przedstawiania obiektów w dwóch wymiarach, a trójwymiarowa – do przedstawiania obiektów w przestrzeni (trójwymiarowych).

Przykładowe programy do tworzenia grafiki trójwymiarowej to: Paint 3D, Blender, 3D studio Max, POV-Ray, SketchUp.

2. Określenie problemu – zaprojektowanie ogrodu

Jesteś projektantem, który ma wykonać projekt ogrodu wokół domu jednorodzinnego. Projekt ma uwzględnić nie tylko rozmieszczenie roślin, ale również alejki i elementy architektury ogrodowej. W jaki sposób przygotować taki projekt?

Przed pojawieniem się komputerów projekty architektoniczne wykonywano na papierze. Wraz z rozwojem technologii zaczęto wykorzystywać komputer i odpowiednie oprogramowanie.

Niezależnie od tego, czy projektant pracuje bez użycia komputera czy na komputerze, pomysł pojawia się jego w głowie. Dopiero potem przedstawia wizję na rysunkach – na kartce lub na ekranie (korzystając z programu graficznego). Zazwyczaj wcześniej ogląda miejsce, w którym ma powstać ogród i wykonuje niezbędne pomiary.

Obrazy dwuwymiarowe tworzyliśmy w takich programach, jak Paint, GIMP i Inkscape. Programy graficzne do tworzenia modeli 3D umożliwiają również wykonanie obrazu 2D. Zawierają narzędzia będące odpowiednikami tradycyjnych narzędzi projektanta (ołówka, cyrkla, gumki, miary), ale także gotowe elementy i mechanizmy, które pozwalają na tworzenie obiektów 3D.



Rys. 1. *Przykład projektu działki (z ogrodem i budynkami) - rysunki 2D i model 3D*

Pokażemy, jak wykonać projekt ogrodu z wykorzystaniem komputera. Zaczniemy od ustalenia wymiarów działki i naniesienia na jej plan wszystkich istniejących obiektów, np. budynku (budynków), ścieżek, tarasu, studni oraz roślin, które chcemy zachować. Następnie (po rozmowie ze zleceniodawcą) zaplanujemy nowe elementy, np. drzewa, krzewy, klomby, warzywniak, sadzawkę, altankę, ławkę, murek przy tarasie, stałe skrzynki na rośliny, obudowę starej studni lub nową dekoracyjną studnię.

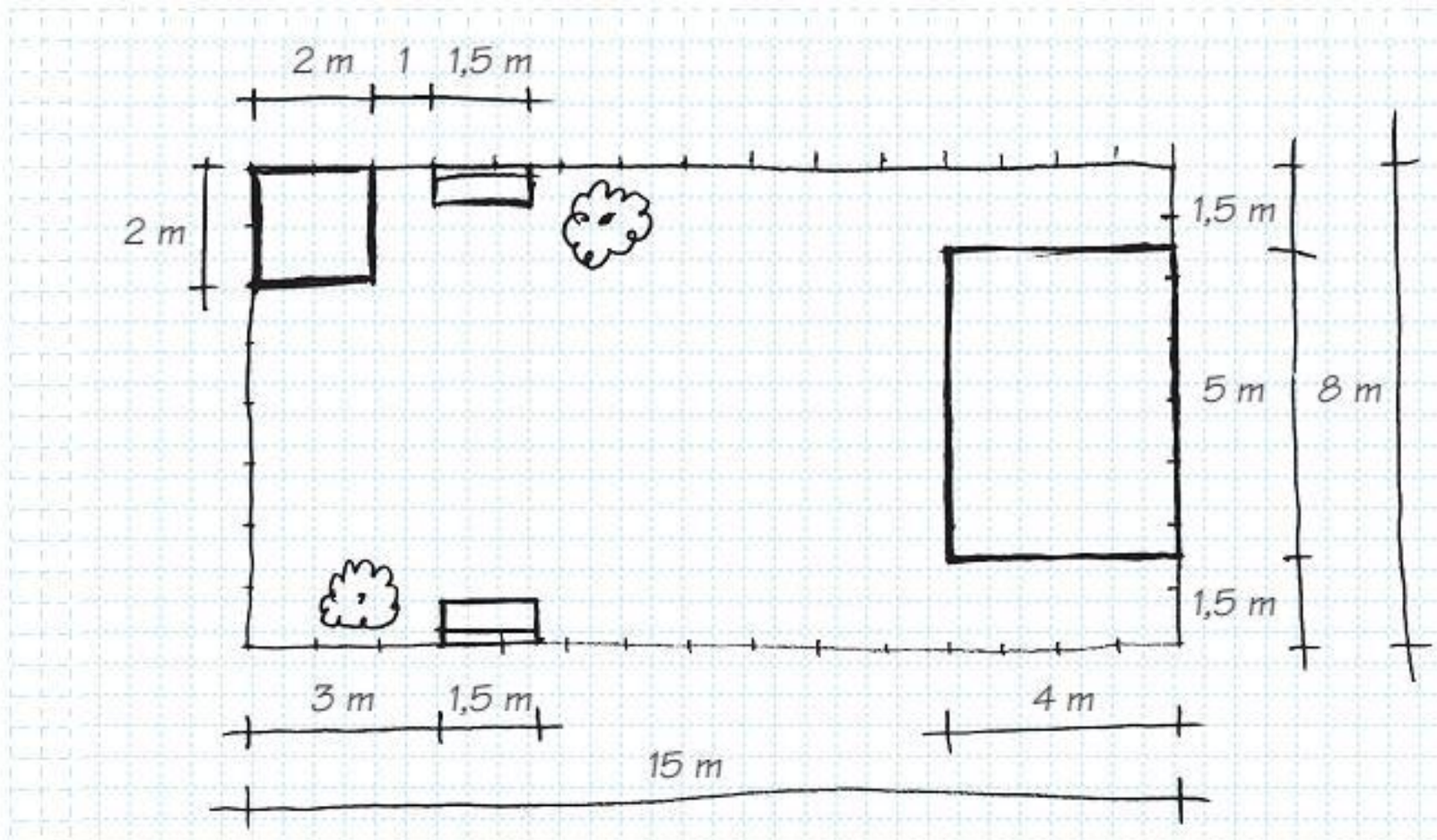
Ćwiczenie 1. Określamy wymagania projektu ogrodu

Ustal wymagania dotyczące ogrodu:

a. wymiary działki, na której masz zaprojektować ogród,
b. wymiary i położenie istniejących elementów na działce,
c. nowe elementy (w tym ławkę) – ich rozmieszczenie, ewentualnie wymiary.

Wskazówki:

- Ćwiczenie możesz wykonać w grupie z podziałem na role (zleceniodawca i projektant) – z koleżanką lub kolegą.
- Możesz stworzyć odręczny rysunek z wykonanymi pomiarami (rys. 2.).



***Rys. 2.** Przykładowy odręczny szkic ogrodu z naniesionymi pomiarami*

Przy projektowaniu ogrodu wykorzystamy możliwości tworzenia dwuwymiarowych i trójwymiarowych modeli w programie graficznym.

Gdy zaczynamy pracę w nowym programie użytkowym (np. graficznym), warto najpierw ustalić, co chcemy zrobić (czego oczekujemy), a potem zastanowić się, jak to wykonać. Zaczynamy od określenia problemu (zadania), a potem szukamy rozwiązania (opcji, narzędzi i mechanizmów programu). Należy również sprawdzić, czy w nowym programie możemy użyć metod znanych z innych programów.

3. Wybrane możliwości programu SketchUp przydatne do projektowania ogrodu

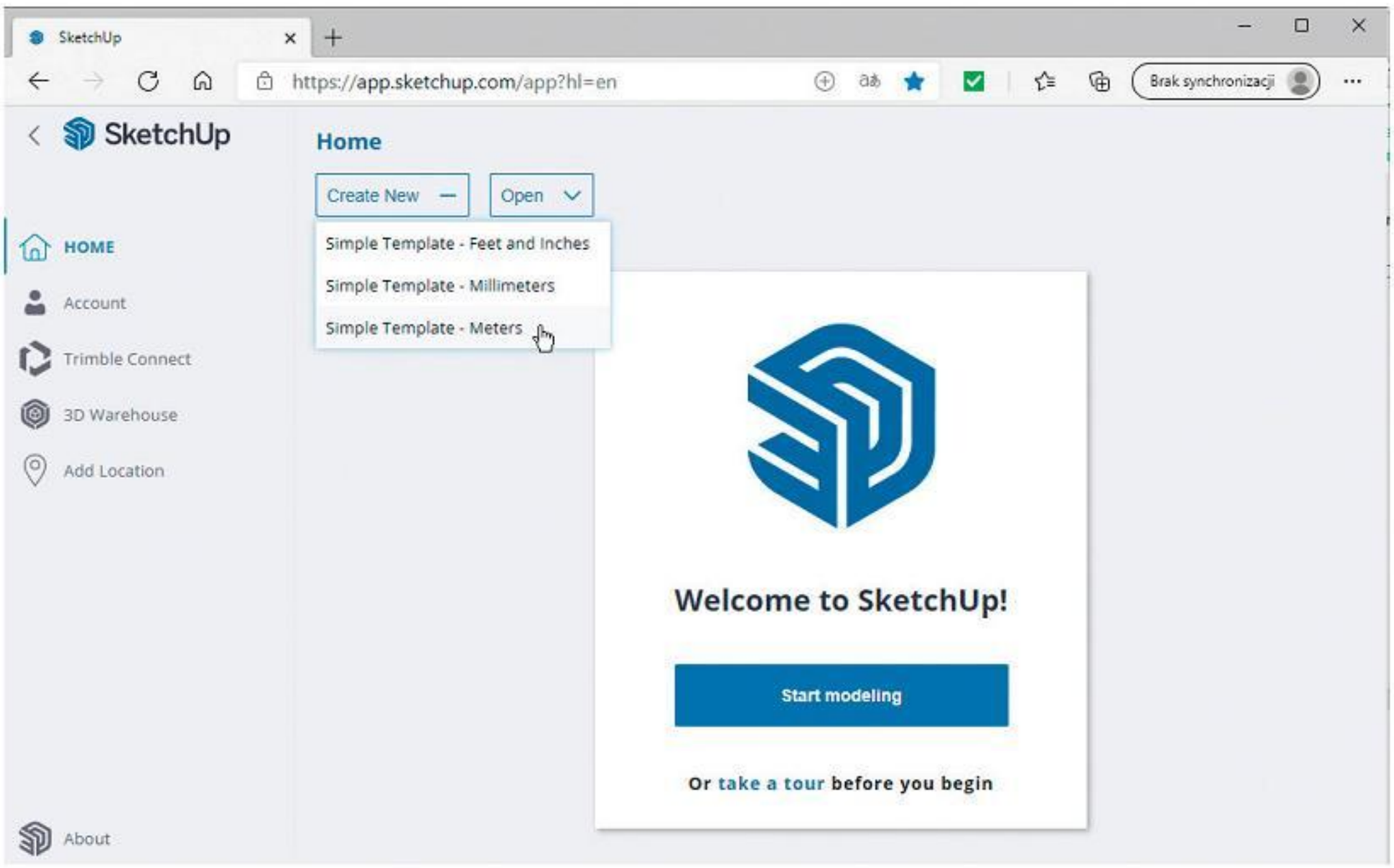
3.1. Program SketchUp

Projekt ogrodu wykonamy w programie graficznym SketchUp, który jest dostępny online poprzez przeglądarkę internetową (*https://www.sketchup.com/products/sketchup-free*) po założeniu bezpłatnego konta użytkownika.

Najpierw zapoznamy się z wybranymi możliwościami programu, potem wykonamy projekt 2D ogrodu, który następnie przekształcimy w projekt 3D.

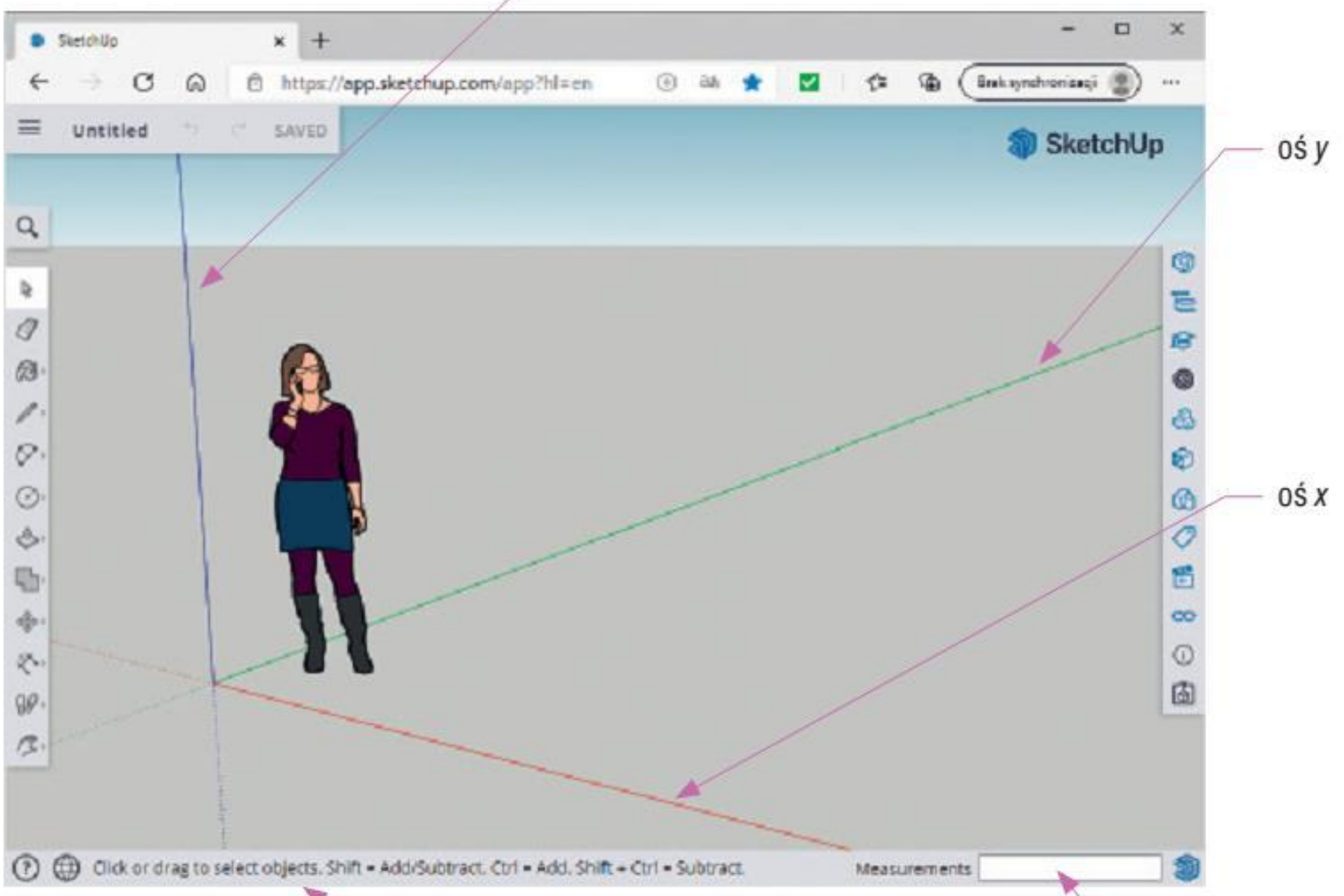
Jeśli posługiwaliśmy się programami graficznymi takimi jak Paint, GIMP i Inkscape, poznanie programu SketchUp nie będzie trudne. Występują w nim podobne narzędzia i sposoby korzystania z nich jak w innych programach, np. rysowanie linii, prostokąta, okręgu, wypełnianie obszaru kolorem, rysowanie odręczne ołówkiem, używanie gumki czy narzędzia zaznaczania. Wskażemy kilka wybranych narzędzi i możliwości programu SketchUp, przede wszystkim przydatnych do wykonania naszego projektu, pozostawiając pozostałe do samodzielnego odkrywania.

Wymiary ogrodu i jego elementów będziemy podawać w metrach, dlatego po uruchomieniu programu warto od razu ustawić odpowiedni system mierniczy (domyślnie program ma ustawione cale) – rys. 3.



Rys. 3. *Zmienianie jednostek systemu mierniczego na metry*

Po rozpoczęciu pracy (**Start modeling**) w oknie programu zobaczymy trójwymiarową przestrzeń wskazywaną przez odpowiednie osie oraz postać człowieka, której zadaniem jest ułatwienie użytkownikowi orientacji przestrzennej. Po lewej stronie okna programu umieszczono narzędzia do rysowania i tworzenia modeli 2D oraz 3D (rys. 4a). Większość z nich (po kliknięciu) udostępnia dodatkowe narzędzia (rys. 4b). Po wybraniu danego narzędzia w dolnej części okna (na pasku stanu) pojawiają się podpowiedzi dotyczące tego narzędzia (rys. 4a). Po prawej stronie okna umieszczono narzędzia, umożliwiające m.in. zmianę widoku, kolorystyki, dodawanie gotowych obiektów pobranych z Internetu.



Rys. 4a. *Okno programu SketchUp*



Rys. 4b. *Przykłady wybierania narzędzi programu SketchUp*

Projektując ogród i jego elementy, będziemy poznawać wybrane funkcje programu SketchUp, m.in.:

- tworzenie obiektów 2D i 3D,
- wycinanie otworu w obiekcie 3D,
- rysowanie obiektu o określonych wymiarach i umieszczanie wymiarów na rysunku,
- dodawanie linii pomocniczych,
- zaznaczanie obiektu, usuwanie, kopiowanie, zmianę rozmiaru (skalowanie),
- przeglądanie projektu w różnych rzutach.

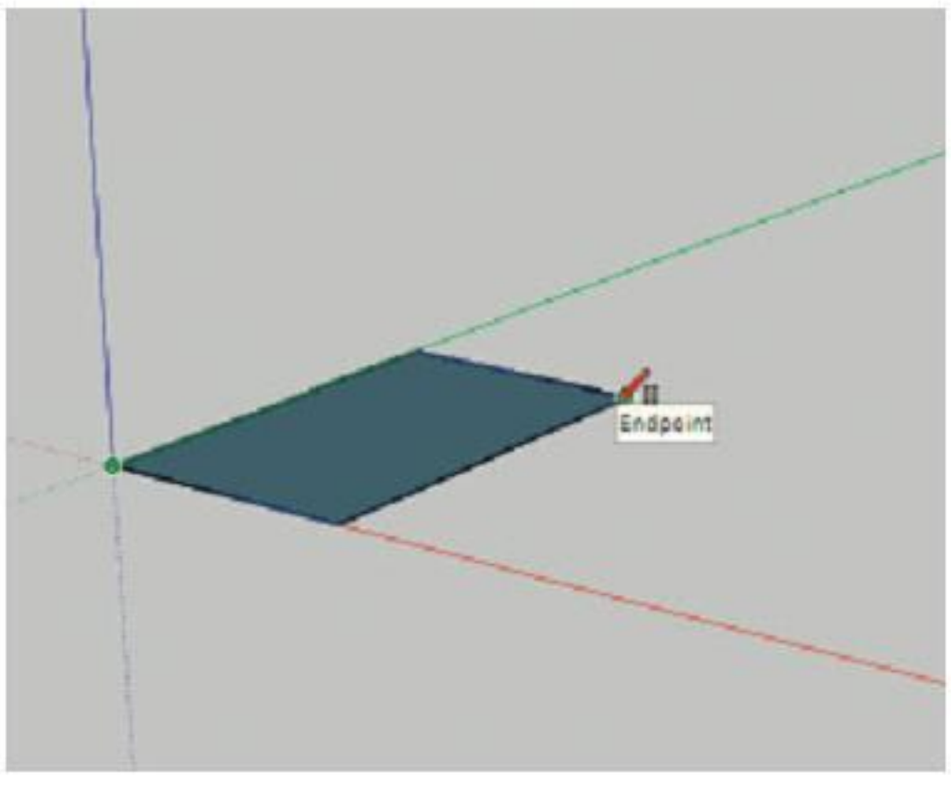
3.2. Przekształcanie modelu 2D w 3D

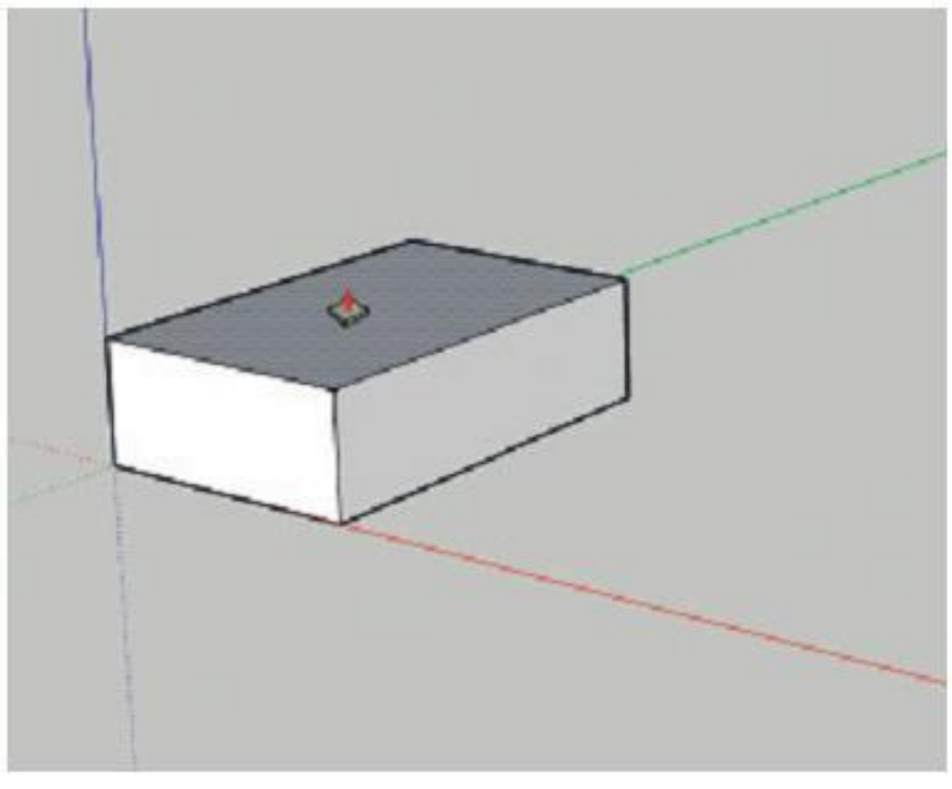
W programie SketchUp możemy łatwo przekształcić figurę płaską w bryłę, czyli model 2D w 3D.



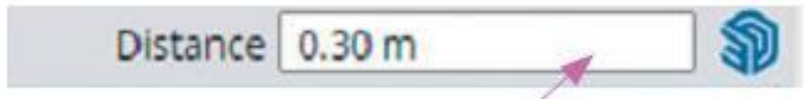
Aby w programie SketchUp przekształcić dwuwymiarowy kształt w trójwymiarowy, można:

- narysować dowolny dwuwymiarowy kształt (rys. 5a),
- wybrać narzędzie **Push/Pull**, umieścić kursor na powierzchni figury i „wyciągnąć" trzeci wymiar figury w górę lub w dół (rys. 5b).





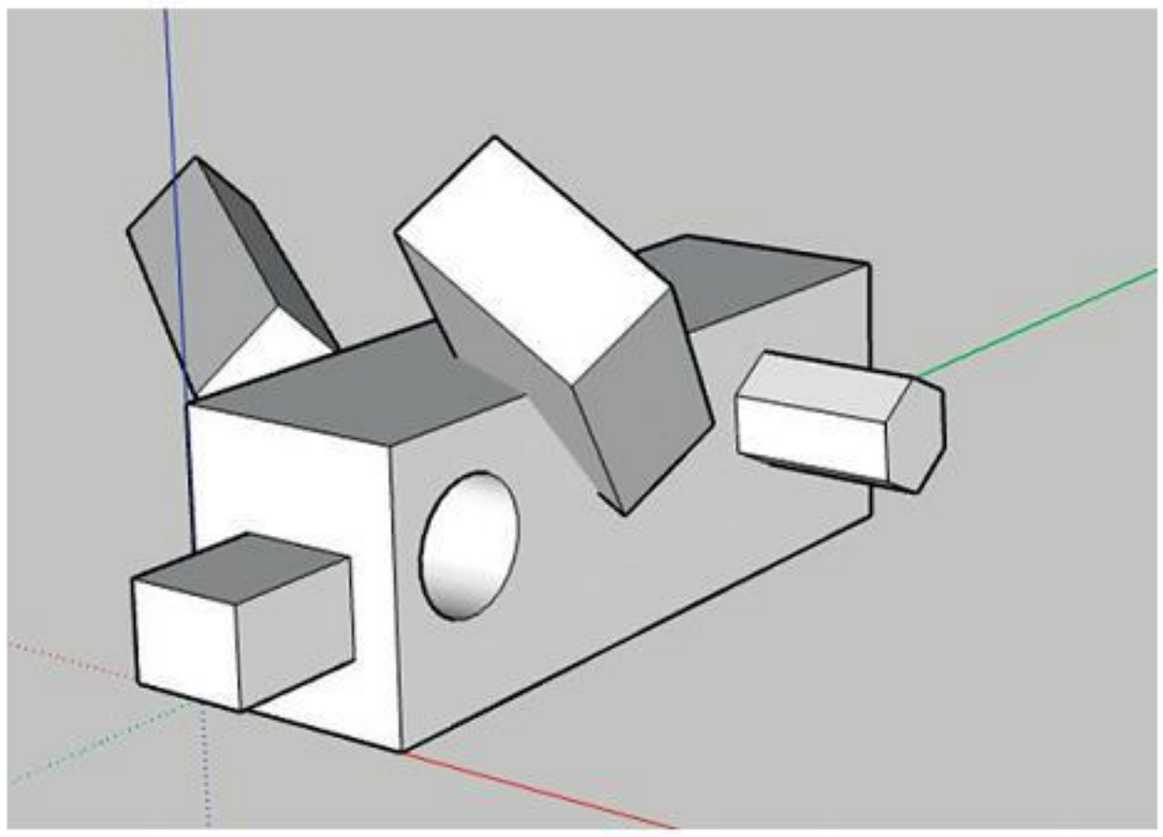




Jeśli sami nie wpiszemy wymiarów, wyświetlą się automatycznie.

Rys. 5a. *Stosowanie narzędzia* ***Rectangle***

Rys. 5b. *Stosowanie narzędzia* ***Push/Pull***



Uwaga

Po wybraniu narzędzia do rysowania albo rozpoczęciu rysowania można wpisać wymiary obiektu (i zatwierdzić je klawiszem **Enter**), np. 2.00 m, 0.50 m. W programie SketchUp część dziesiętną liczby zapisujemy po kropce, a nie po przecinku. Przed rozpoczęciem wpisywania wymiarów nie trzeba uaktywniać pola tekstowego w prawym dolnym rogu okna programu.

Rys. 6. *Przykład kompozycji 3D utworzonej za pomocą wybranych narzędzi, m.in.:* ***Rectangle***, ***Polygon***, ***Circle***, ***Push/Pull***

Ćwiczenie 2. Tworzymy przykładowy model 3D

1. W programie SketchUp utwórz prostą kompozycję 3D według własnego pomysłu. Eksperymentuj z różnymi narzędziami i sprawdź ich działanie.
2. Zapisz plik pod nazwą *Kompozycja 3D*.

Wskazówka: Jeśli postać człowieka ci przeszkadza, można ją zaznaczyć (narzędzie **Select**) i przesunąć (narzędzie **Move**) lub usunąć (klawisz **Delete**).

3.3. Projektowanie trójwymiarowego modelu ławki

W ogrodzie chcemy umieścić elementy małej architektury ogrodowej według własnego projektu. Zanim nauczymy się projektować cały ogród, utworzymy jego jeden element, np. ławkę.

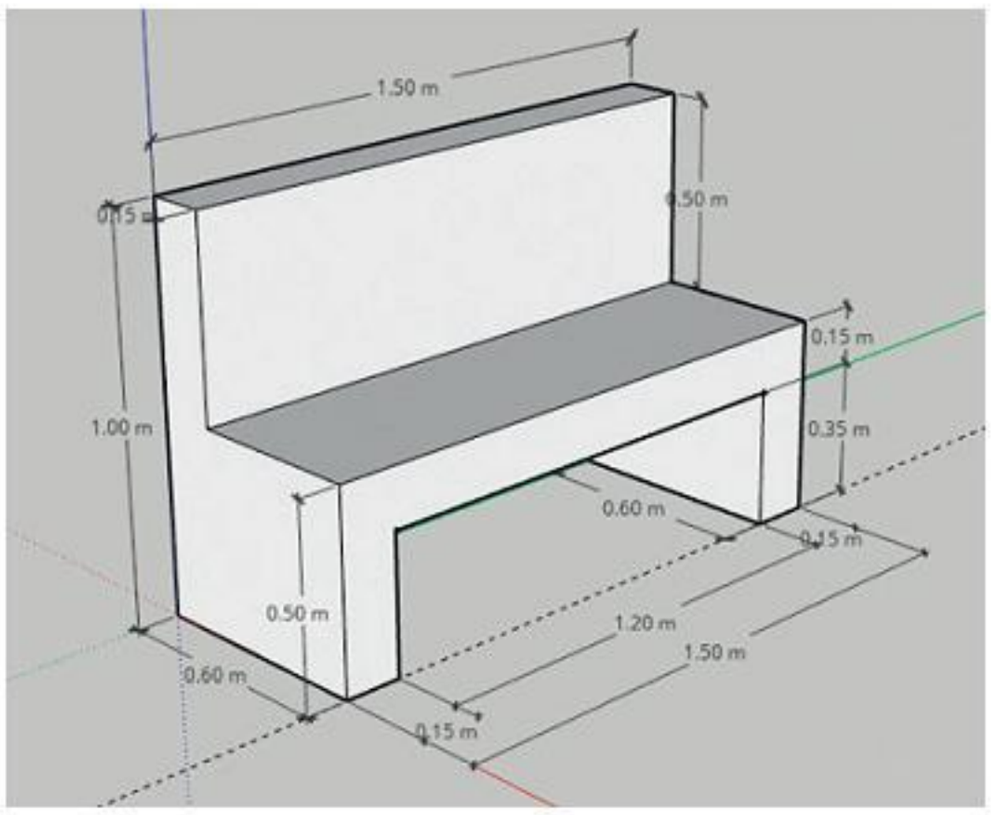
Jeśli będziemy chcieli zlecić wykonanie ławki na podstawie projektu, powinniśmy umieścić na rysunku potrzebne wymiary – inaczej projekt będzie bezużyteczny. Moglibyśmy wpisać wymiary za pomocą narzędzia do umieszczania tekstu na rysunku i rysowania linii, ale SketchUp jest programem graficznym do projektowania, dlatego warto sprawdzić, czy nie znajdziemy w nim potrzebnych narzędzi.

Na rysunku 7a pokazano przykładowy projekt prostej ławki z dodanymi tzw. liniami wymiarowymi.

Aby w programie SketchUp dodać do modelu linie wymiarowe, należy użyć narzędzia **Dimensions** , które umożliwia wyświetlenie takich linii łącznie z wyliczeniem odległości pomiędzy wskazanymi punktami modelu (rys. 7a).

Chcielibyśmy, aby wybrane elementy modelu były umieszczone precyzyjnie w tej samej linii. Przydałaby się tzw. linia pomocnicza. Narysowanie takiej linii umożliwia narzędzie **Tape Measure** .

Jeśli zauważymy jakieś niedociągnięcia, możemy zmodyfikować utworzony model, np. zmienić rozmiary (skalować), przesunąć, usunąć niepotrzebne linie.



Rys. 7a. *Przykładowy projekt 3D ławki ogrodowej wykonany w programie SketchUp*



Rys. 7b. *Trójwymiarowy model ławki wydrukowany na drukarce 3D*

Ćwiczenie 3. Projektujemy ławkę do ogrodu

1. W programie SketchUp utwórz projekt 3D ławki według wcześniej określonych wymagań.
2. Umieść na rysunku wszystkie wymiary niezbędne dla wykonawcy (określone w ćwiczeniu 1.).
3. Odszukaj narzędzie **Tape Measure** i zastosuj je do tych elementów ławki, które powinny być umieszczone w jednej linii.
4. Wprowadź do projektu poprawki, jeśli są konieczne.
5. Zapisz plik pod nazwą *Ławka*.
6. Wydrukuj wykonany model ławki na drukarce 3D, jeśli masz takie możliwości.

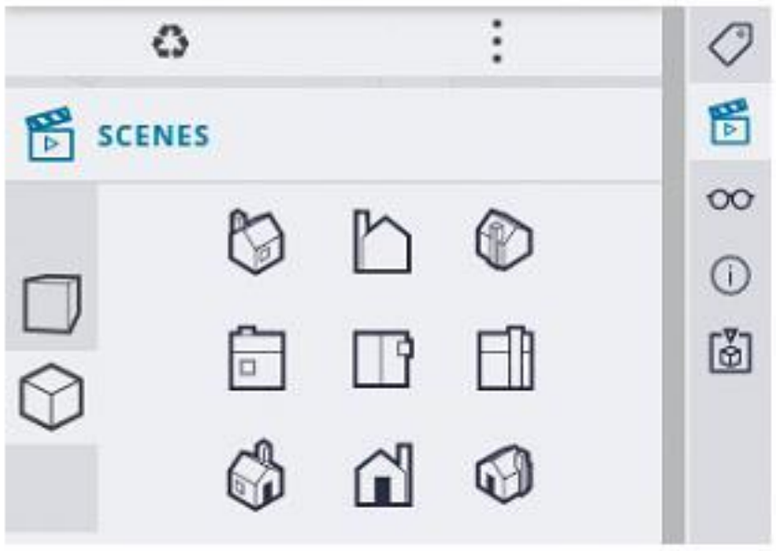


Uwaga: Możesz utworzyć bardziej złożony model niż na rysunku 7a, np. dodać podłokietniki, cztery nogi, fantazyjne oparcie.

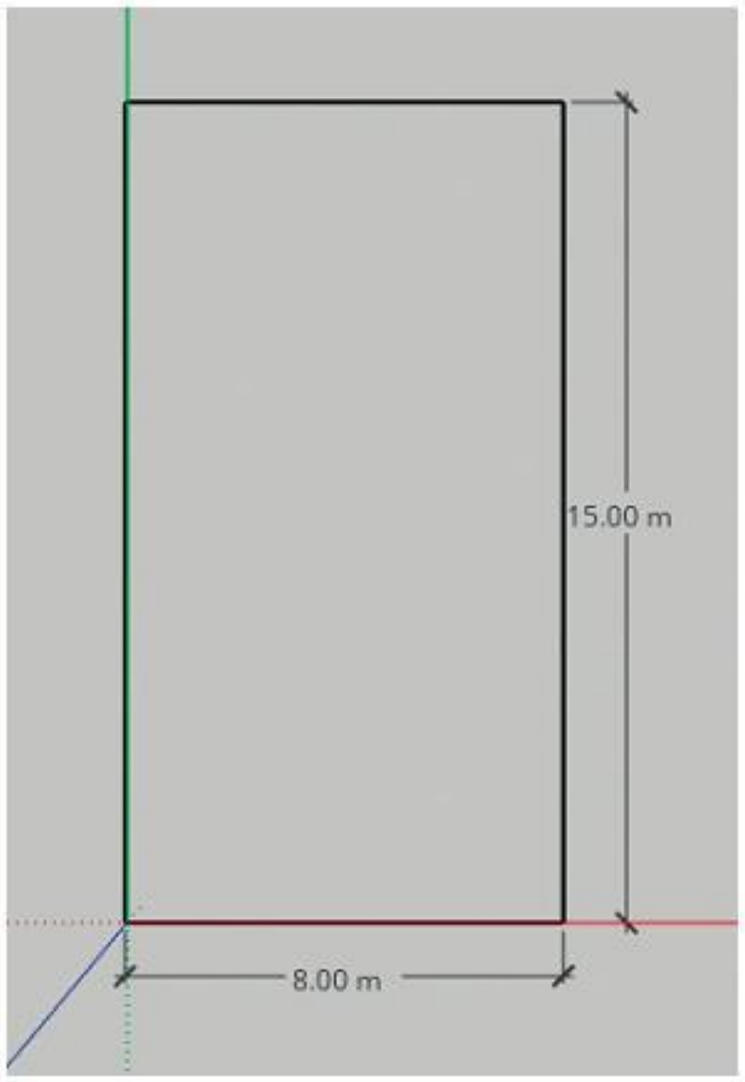
4. Tworzenie projektu ogrodu

4.1. Model 2D ogrodu

Tworzenie projektu ogrodu zaczniemy od narysowania rzutu (modelu 2D), a potem dodamy trzeci wymiar. Możliwości przełączania się pomiędzy różnymi rodzajami widoku znajdziemy pod przyciskiem **Scenes** .



Rys. 8. *Przełączanie się pomiędzy różnymi widokami*



Rys. 9. *Przykładowa działka – rzut z góry*

Ćwiczenie 4. Tworzymy początkową wersję modelu 2D ogrodu

1. W nowym dokumencie programu SketchUp wybierz widok rzutu z góry i narysuj działkę o wymiarach ustalonych w ćwiczeniu 1.
2. Zgodnie z wymaganiami projektu określonymi w ćwiczeniu 1.:
   a. umieść na działce istniejące elementy (np. budynek mieszkalny, studnię, chodniki) jako obiekty 2D,
   b. dodaj przynajmniej jeden nowy element, zaplanowany w ćwiczeniu 1.
3. Zapisz plik pod nazwą *Ogród*.

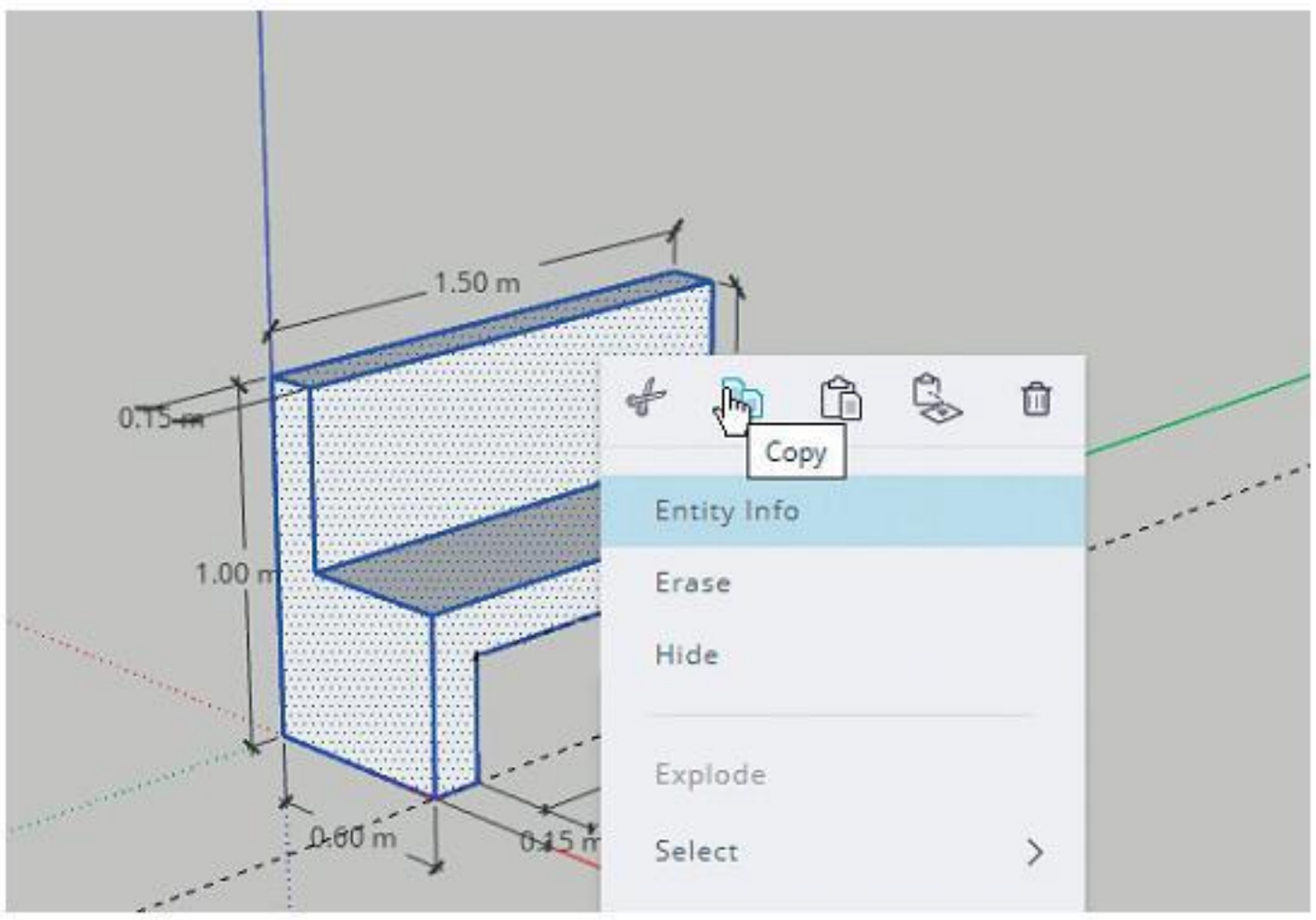
Wskazówki:

- Widok możemy przybliżyć lub oddalić, kręcąc kółkiem myszy.
- Jeśli podczas projektowania przeszkadza ci kolorowa powierzchnia działki, możesz ją usunąć i wypełnić powierzchnię kolorem na końcu pracy.

4.2. Model 3D ogrodu

W pracy nad projektem możemy przełączyć się na widok 3D i przekształcić wybrane elementy w trójwymiarowe modele. Do tworzonego projektu możemy również dodać modele utworzone wcześniej i zapisane w innym pliku. Na przykład do projektu pokazanego na rysunkach 11. i 12. wklejono dwukrotnie ławkę skopiowaną z innego pliku (rys. 10.). Następnie jedna z ławek została obrócona tak, aby ławki stały naprzeciw siebie. Do obracania modelu użyto narzędzia **Rotate** . Aby umieścić ławki dokładnie w tej samej linii, wykorzystano linie pomocnicze (rys. 11.).

Obiekty możemy nie tylko przesuwać i obracać, ale również skalować, czyli zmieniać ich rozmiary, korzystając z narzędzia **Scale**.



Rys. 10. *Kopiowanie zaznaczonej wcześniej ławki*

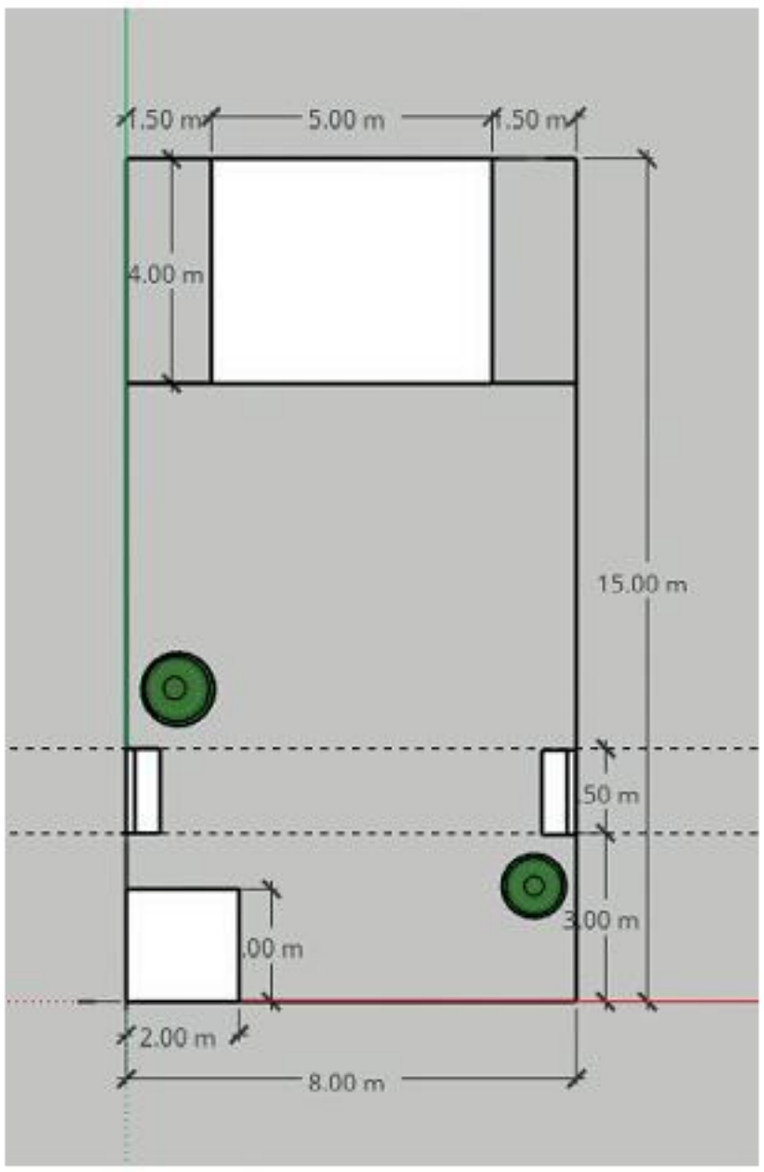
Ćwiczenie 5. Kopiujemy element z innego pliku i wklejamy do projektu ogrodu

1. Umieść we wcześniej określonym miejscu na terenie ogrodu ławkę zaprojektowaną w ćwiczeniu 3. Jeśli w twoim ogrodzie ma być kilka ławek, skopiuj je i wklej w odpowiednie miejsca projektu.
2. Zapisz plik pod tą samą nazwą.

Wskazówka: Możesz skopiować model ławki z wszystkimi liniami wymiarowymi lub bez – zależnie, czy wykonawca dostanie oddzielnie projekt samej ławki.

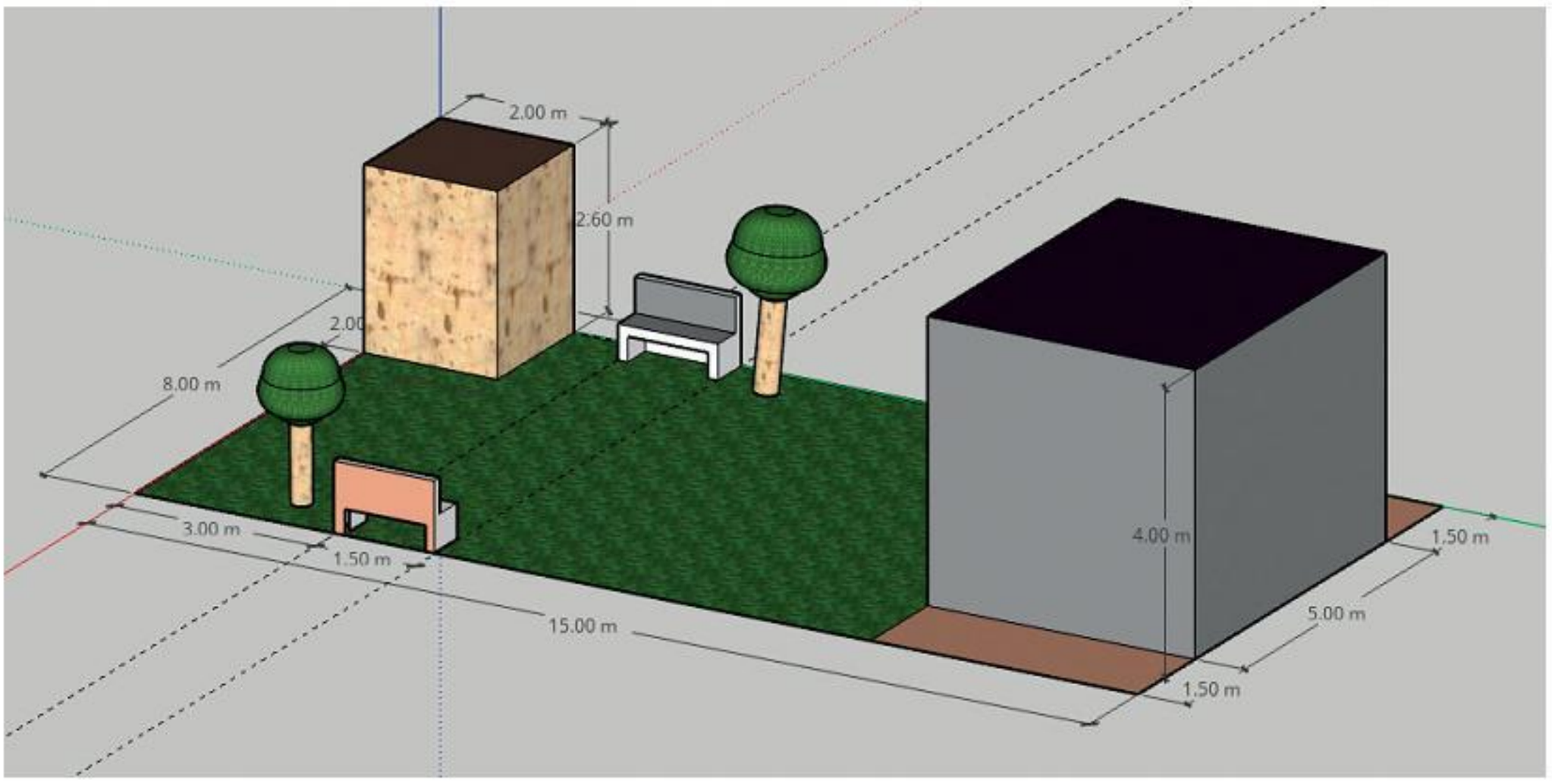
Do projektu można dodać gotowe elementy z magazynu online (narzędzie **3D Warehouse**). Na rysunkach 11. i 12. umieszczono drzewa z tego magazynu.

Programy do projektowania grafiki 3D zawierają zwykle tzw. edytor materiałów, które możemy przypisywać wybranym obiektom na scenie, np. trawę – trawnikom, drewno – podłodze, szkło – szybom. W ten sposób decydujemy o wyglądzie powierzchni tych obiektów. Możemy korzystać z gotowych materiałów, modyfikować je lub tworzyć nowe.



W programie SketchUp edytor materiałów znajdziemy pod przyciskiem **Materials** . Okno z materiałami otworzy się również, gdy klikniemy narzędzie **Paint** . Można w nim wybrać m.in. kolor lub teksturę i ewentualnie zmienić stopień przezroczystości.

Rys. 11. *Przykładowy wstępny projekt ogrodu – rzut z góry (model 2D)*



Rys. 12. *Przykładowy wstępny projekt ogrodu (model 3D)*

Ćwiczenie 6. Modyfikujemy projekt

1. Uzupełnij projekt o brakujące elementy zgodnie z wymaganiami określonymi w ćwiczeniu 1. Wykorzystaj m.in. gotowe obiekty z magazynu online (np. drzewka), dodaj nawierzchnię.
2. Odszukaj narzędzie skalowania obiektów i wykorzystaj je przy modyfikowaniu projektu.
3. Obejrzyj projekt w różnych widokach.
4. Zapisz projekt pod tą samą nazwą.
5. Wydrukuj projekt.

5. Projektowanie własnego pokoju

Chcesz na nowo zaprojektować swój pokój. Niektóre meble zostaną, niektóre przestawisz lub usuniesz, wymienisz okna i drzwi, dodasz (zaprojektujesz) nowe wyposażenie. Czy możemy wykonać taki projekt, wykorzystując doświadczenia z projektowania ogrodu?

Doświadczenia nabyte przy rozwiązywaniu jednego problemu mogą zostać wykorzystane przy rozwiązywaniu innych problemów.

Zastanówmy się teraz, co może się przydać z doświadczeń zdobytych przy projektowaniu ogrodu. Wiemy już m.in.: jak wykonać rzut z góry (czyli model 2D), narysować odcinek i figurę o określonych wymiarach, zastosować linie wymiarowe i pomocnicze oraz jak przekształcić model 2D w 3D.

Ćwiczenie 7. Określamy wymagania projektu pokoju

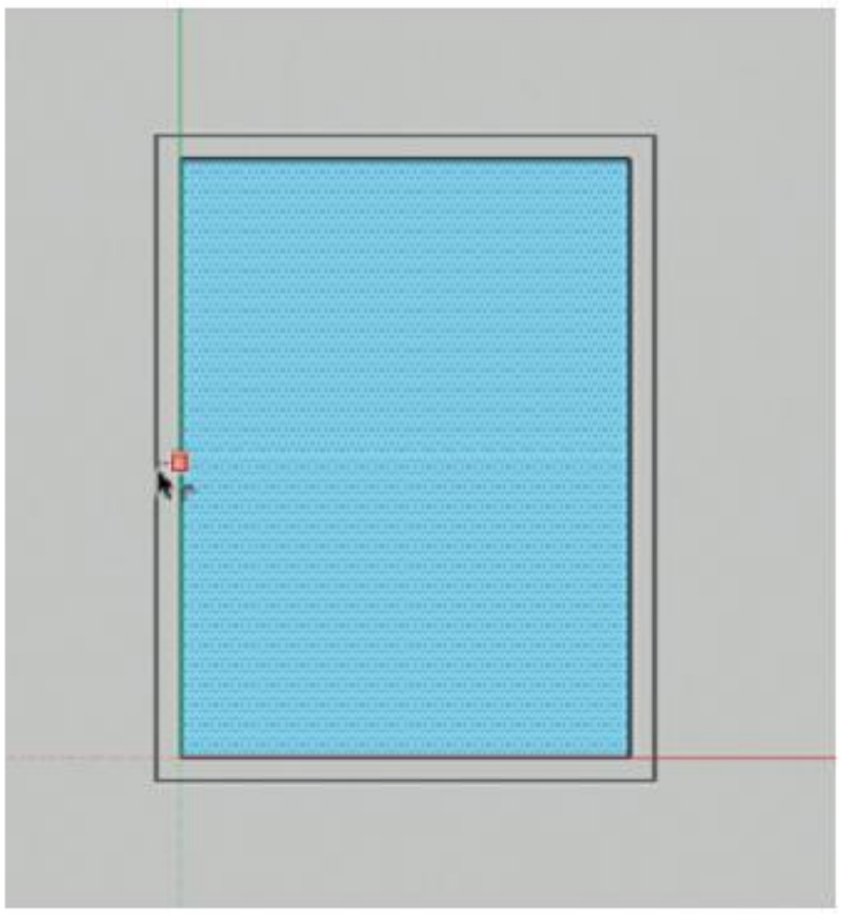
1. Określ wymagania dotyczące projektu twojego pokoju:
   a. wymiary pokoju,
   b. wymiary i położenie istniejących elementów pokoju, w tym okna (okien) i drzwi,
   c. nowe elementy – ich rozmieszczenie i wymiary.
2. Narysuj odręczny szkic projektu pokoju.

Zanim zaczniemy projektować wnętrze pokoju, warto wiedzieć m.in. jak utworzyć ściany i wyciąć potrzebne otwory, np. na okna, drzwi. Czy do ich wykonania wystarczą poznane dotychczas narzędzia? Praktycznie tak. Ale w przypadku ścian o jednakowej szerokości można wykorzystać narzędzie **Offset** , za pomocą którego tworzymy kontur zewnętrzny lub wewnętrzny dowolnego kształtu, z zachowaniem takiej samej odległości pomiędzy oryginalnymi krawędziami a nowym konturem (rysunki 13a i 13b). W ten sposób powstaną ściany o określonej grubości, które następnie wystarczy wyciągnąć w górę za pomocą narzędzia **Pull/Push** (rys. 13c). Korzystając z tych dwóch narzędzi, można utworzyć inne modele 3D, np. pudełko, pojemnik, kwietnik, skrzynię na narzędzia.

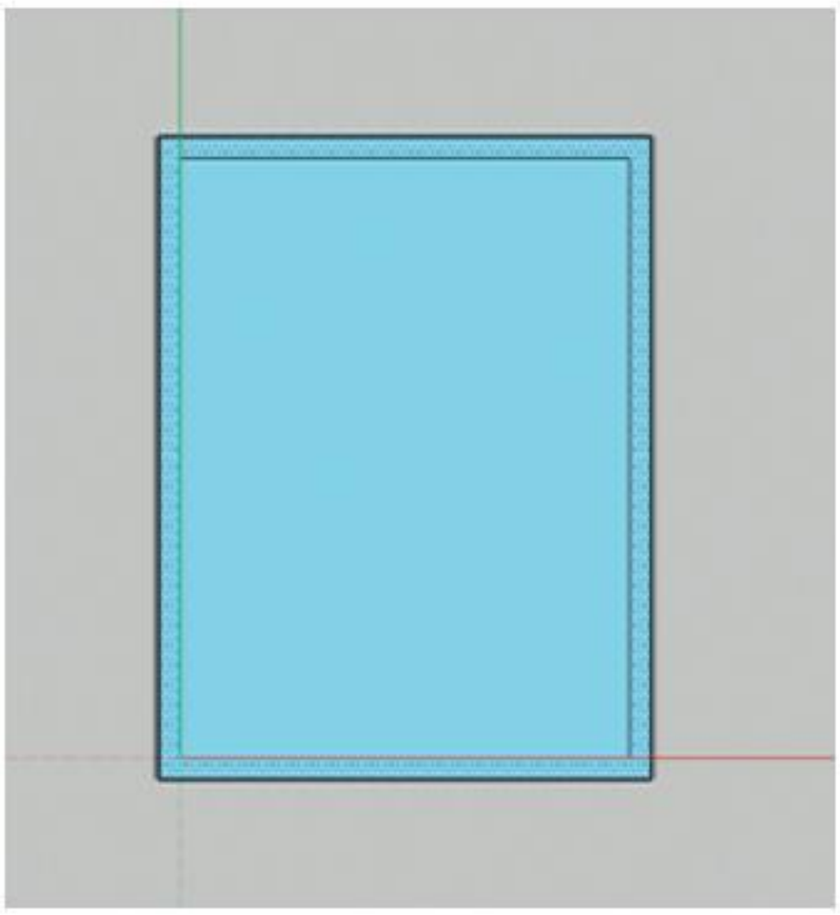
Ćwiczenie 8. Tworzymy ściany pokoju

1. W nowym dokumencie programu SketchUp wybierz widok rzutu z góry, narysuj podłogę pokoju i utwórz ściany o wymiarach określonych w ćwiczeniu 7.
2. Zapisz plik pod nazwą *Pokój*.

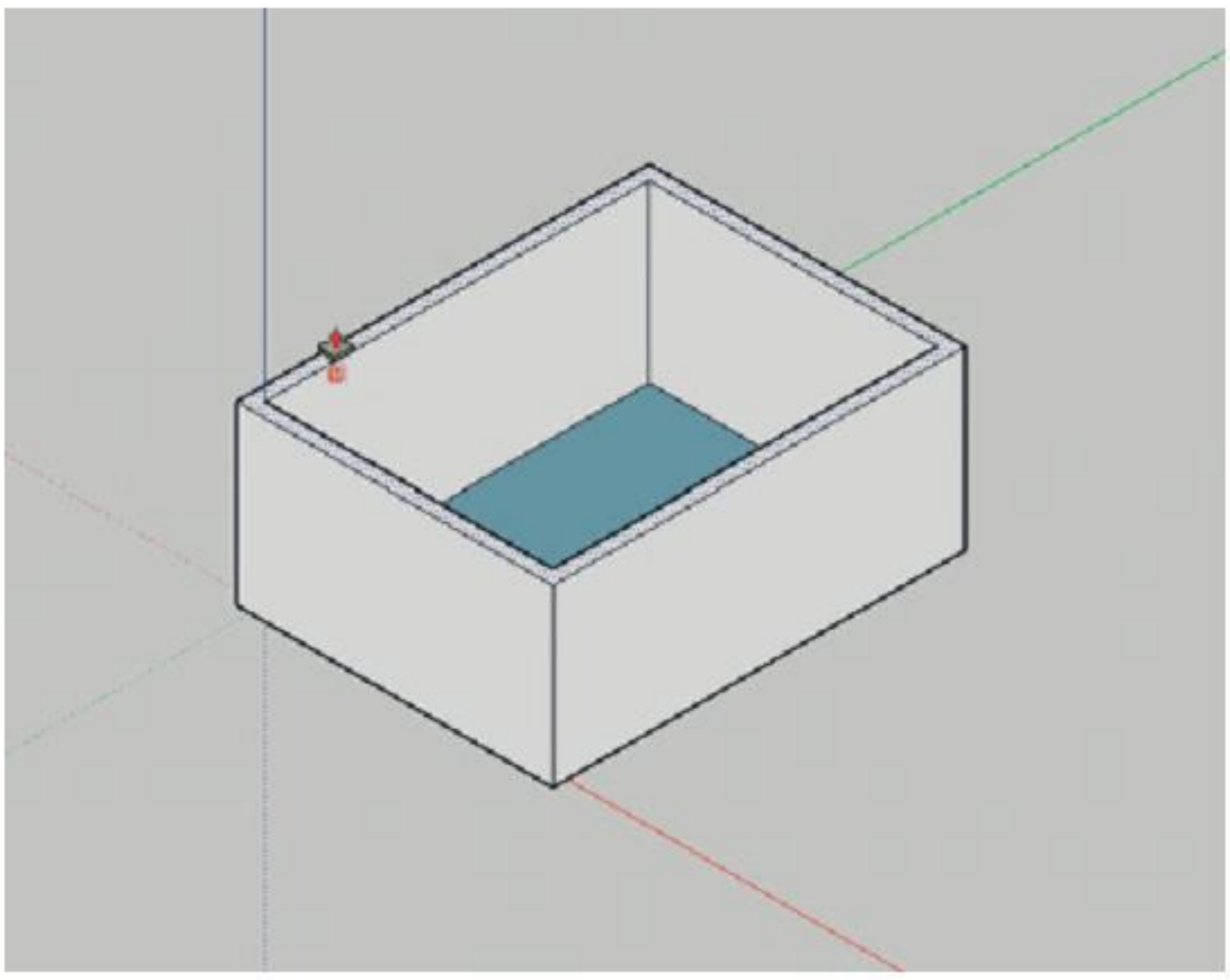
Wskazówka: Po wybraniu narzędzia **Offset** wystarczy umieścić kursor w dowolnym miejscu figury tak, aby została zaznaczona. Po ustaleniu grubości ściany (rys. 13a) zakończ pracę z narzędziem **Offset**, zaznacz powierzchnię obrysu (rys. 13b), a następnie wyciągnij ściany w znany sposób (rys. 13c).



Rys. 13a**. Tworzenie obrysu prostokąta za pomocą narzędzia **Offset



***Rys. 13b**. Utworzony obrys o szerokości 0,15 m*



Rys. 13c**. Wyciąganie ścian pokoju za pomocą narzędzia **Pull/Push

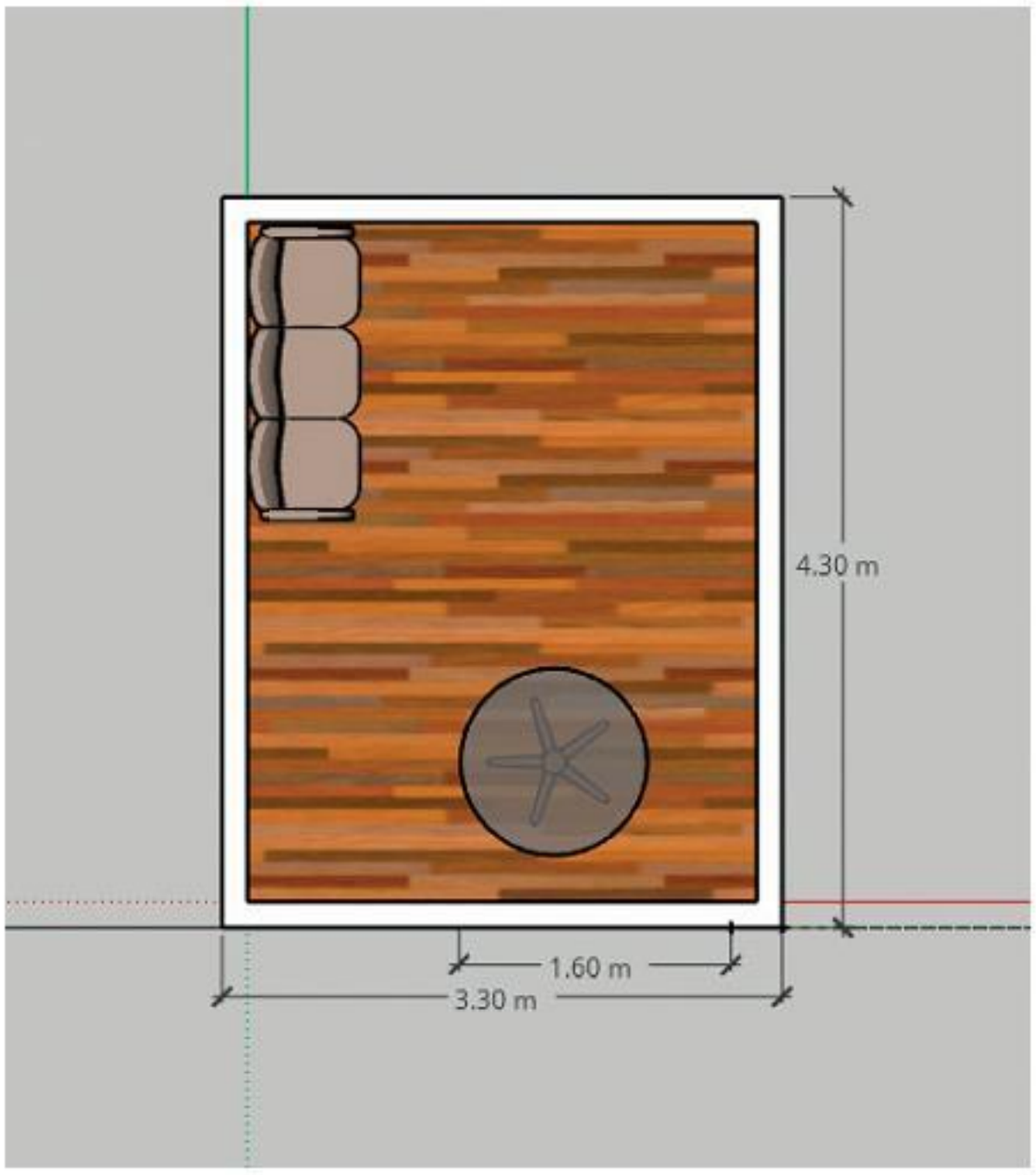
Załóżmy, że w pokoju chcemy wymienić okno na inne, dlatego na jednej ze ścian trzeba wyciąć otwór. Aby w projekcie umieścić precyzyjnie (zgodnie z określonymi wcześniej wymiarami) otwór na okno, warto użyć linii pomocniczych (jak pokazano na rysunku 15.).



Aby wyciąć otwór w ścianie (np. na okno lub drzwi), rysujemy obrys figury i, używając wewnątrz tego obrysu narzędzia **Push/Pull**, wytłaczamy otwór do momentu aż wyrównamy do płaszczyzny znajdującej się z drugiej strony ściany (pojawi się podpowiedź *On Face*). Wartość wyświetlana w polu tekstowym pomiarów (**Measurements**) powinna być równa grubości ściany.

Uwaga

Jeśli chcemy zmienić punkt widzenia utworzonego projektu 3D, czyli obrócić kamerę wokół modelu 3D, możemy zastosować narzędzie **Orbit** . To narzędzie zastosowano na rysunkach 12. i 15.



Rys. 14. *Przykładowy wstępny projekt pokoju – rzut z góry (model 2D)*

Ćwiczenie 9. Wycinamy otwory na okna i drzwi

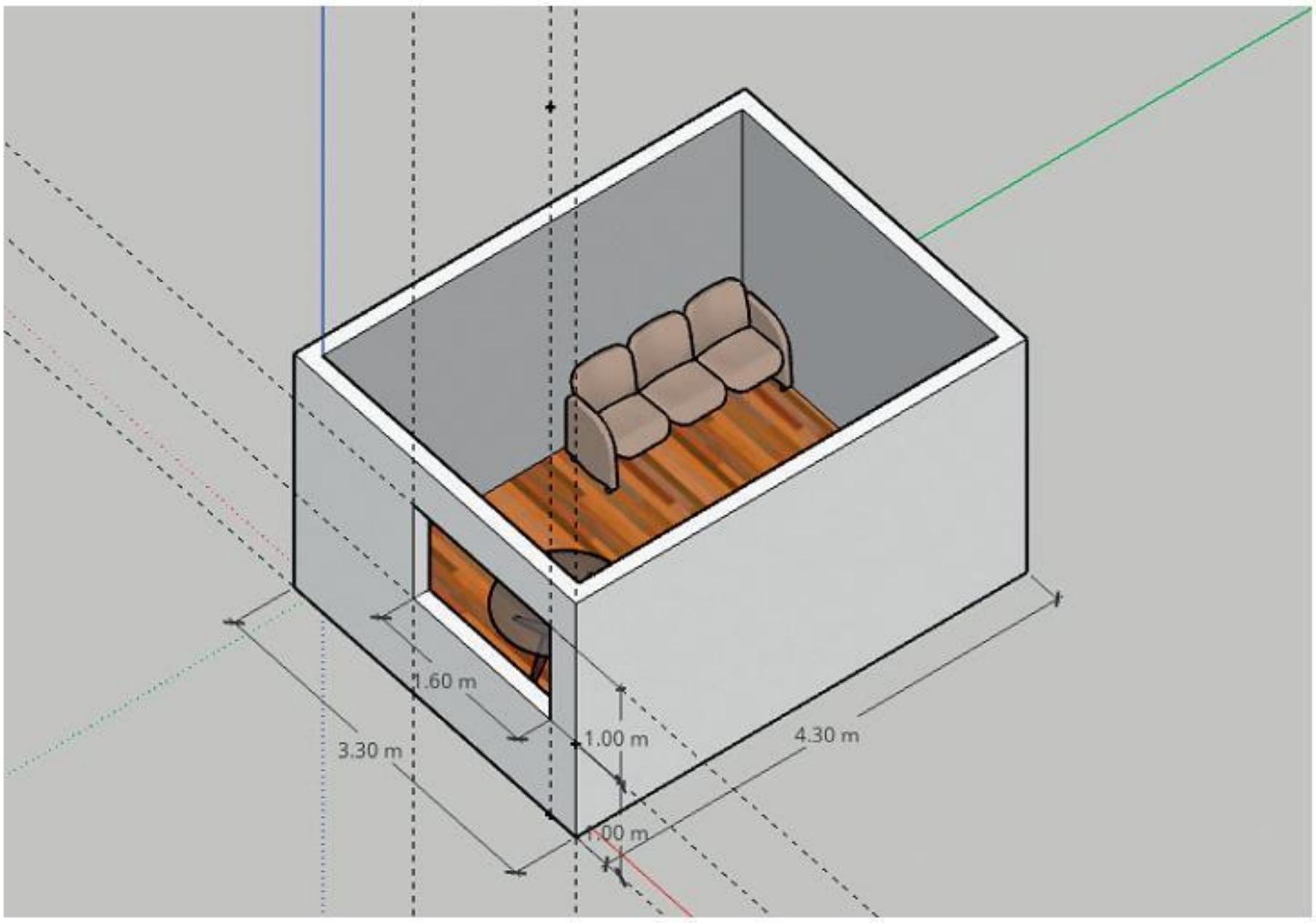
1. Odczytaj wymiary pokoju i okna z przykładowego projektu pokazanego na rysunku 15. Na rysunku nie podano grubości ścian, ale wiemy, że wszystkie mają jednakową grubość 0,15 m. Oblicz wymiary podłogi.
2. W projekcie swojego pokoju wytnij otwory na okno (okna), drzwi (w tym ewentualnie drzwi balkonowe), zgodnie z ustaleniami określonymi w ćwiczeniu 7.
3. Zapisz plik pod tą samą nazwą.

Wskazówki:

- Zmień widok na najwygodniejszy do wycinania okna (okien) i drzwi na danej ścianie.
- Jeśli okno lub drzwi składają się z kilku prostokątów (np. okno balkonowe), przed wytłaczaniem należy usunąć zbędne krawędzie, aby prostokąty utworzyły jeden kształt.

Ćwiczenie 10. Projektujemy wnętrze pokoju

1. W projekcie swojego pokoju umieść meble i inne elementy – zgodnie z wymaganiami określonymi w ćwiczeniu 7. Meble możesz zaprojektować samodzielnie i zapisać w oddzielnych plikach lub skorzystać z gotowych zbiorów.
2. Zmień wygląd powierzchni (materiał) podłogi i ścian według uznania.
3. Zapisz plik pod tą samą nazwą.
4. Wydrukuj projekt.



Rys. 15. *Przykładowy wstępny projekt pokoju (model 3D)*

Warto zapamiętać

- Ze względu na sposób reprezentowania danych obrazu wyróżniamy grafikę dwuwymiarową 2D i trójwymiarową 3D.
- Program SketchUp umożliwia tworzenie modeli 2D i 3D oraz projektowanie większych obiektów, np. ogrodu, pokoju lub całego mieszkania.
- Podczas projektowania ogrodu i pokoju zastosowaliśmy m.in.:
  - przekształcanie modelu 2D w model 3D,
  - wycinanie otworów w obiekcie 3D,
  - rysowanie obiektu o określonych wymiarach i umieszczanie wymiarów na rysunku,
  - dodawanie linii pomocniczych, zaznaczanie obiektu, usuwanie, kopiowanie, zmianę rozmiaru (skalowanie),
  - tworzenie konturu zewnętrznego lub wewnętrznego figury,
  - przeglądanie projektu w różnych rzutach.

Pytania i polecenia

1. Czym się różni grafika 2D od grafiki 3D?
2. W jaki sposób można utworzyć graniastosłup w programie SketchUp?
3. W jaki sposób dodać do projektu linie wymiarowe i linie pomocnicze?
4. Jakie są zastosowania narzędzi **Scale** i **Offset**?
5. Jak wykonać ściany w projekcie pokoju, jeśli każda ma taką samą grubość, a jak – gdy są różnej grubości?
6. W jaki sposób „przykryć” podłogę w projekcie pokoju wybranym materiałem?

Zadania

1. Zaprojektuj trójwymiarową skrzynkę na kwiaty według własnego pomysłu. Skrzynka pokazana na rysunku 16. może stanowić inspirację. Zapisz projekt w pliku pod nazwą *Skrzynka*.



Rys. 16. *Przykładowy wzór skrzynki ogrodowej – zadanie 1.*

2. Do projektu zapisanego w ćwiczeniu 6. dodaj ogrodzenie. Wzoruj się na sposobie tworzenia ścian pokoju. Zapisz plik pod tą samą nazwą.
3. Wykonaj model 3D stolika, którego powierzchnia jest pięciokątem foremnym. Zapisz plik pod nazwą *Stolik*.
   Wskazówka: Po wybraniu narzędzia rysowanie wielokątów (**Polygon**) wpisz liczbę boków z literą „s" do pola tekstowego **Sides**, np. wpisanie „6s" umożliwi narysowanie sześciokąta foremnego.
4. Wykonaj model 3D według własnego pomysłu. Postaraj się wykorzystać także nieomówione w temacie narzędzia.

Dla zainteresowanych

5. Do projektu pokoju zapisanego w ćwiczeniu 10. dodaj okna i drzwi, wybierając odpowiednie materiały, m.in. szyby do okien, drewno do drzwi. Zapisz plik pod tą samą nazwą.
6. W programie SketchUp zapoznaj się z narzędziem **Follow me** . Wykonaj model 3D według własnego pomysłu, wykorzystując m.in. to narzędzie. Opisz w punktach sposób wykonania swojego modelu.
7. Zaprojektuj mieszkanie o powierzchni 30 m^2 (5 m x 6 m), które ma się składać z przedpokoju, pokoju z aneksem kuchennym i łazienki. Zaprojektuj również wyposażenie. Zapisz projekt w pliku pod nazwą *Mieszkanie*.

Wokół algorytmiki i programowania

C

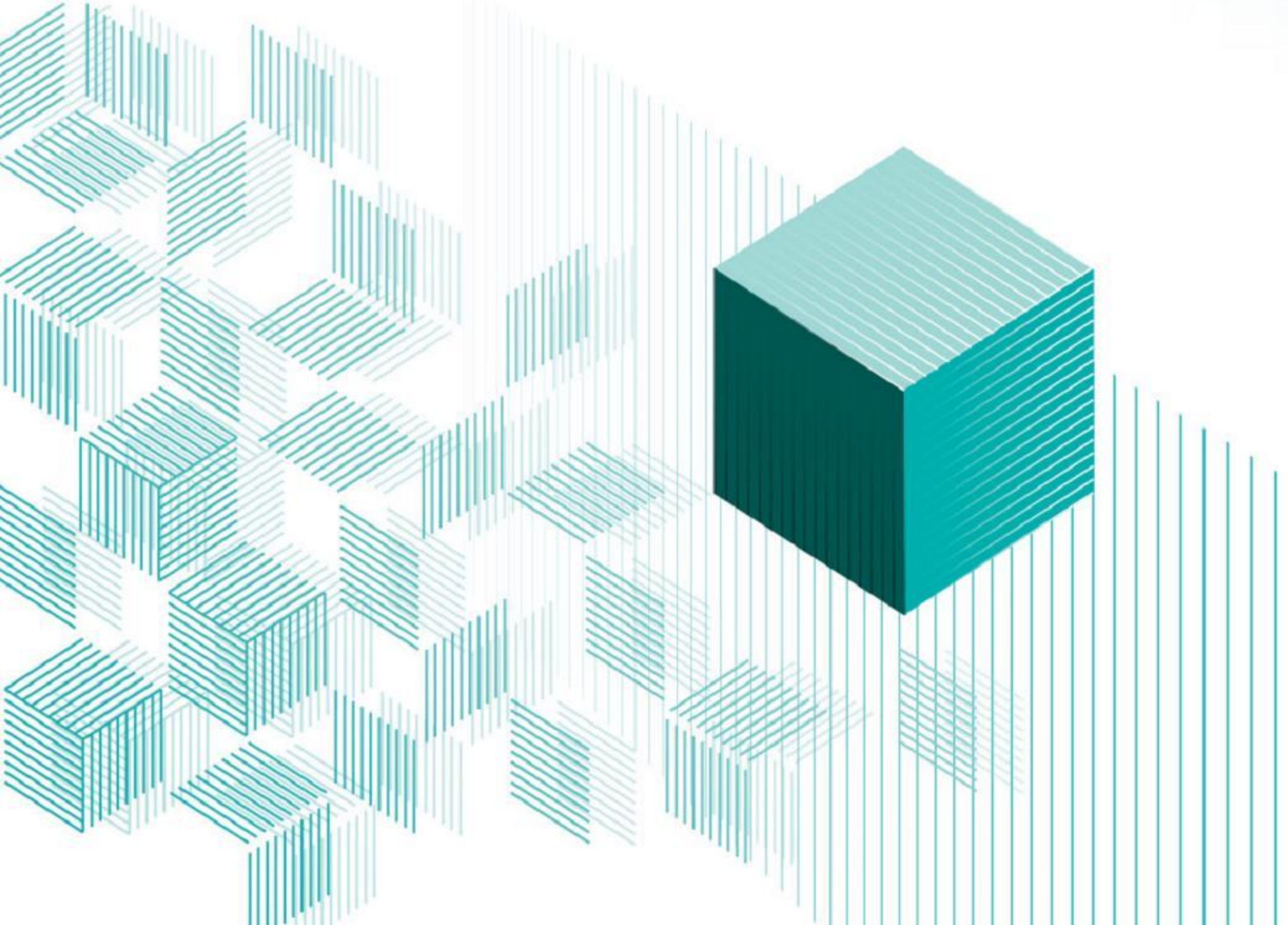
Moduł

Temat C1. Algorytmy na tekstach w językach C++ i Python

Temat C2. Algorytmy szyfrowania w językach C++ i Python

Temat C3. Metoda połowienia, podejście zachłanne i rekurencja

Temat C4. Algorytmy na liczbach w językach C++ i Python



Temat C1. Algorytmy na tekstach w językach C++ i Python

1. Wybrane algorytmy na tekstach
   - 1.1. Porównywanie tekstów
   - 1.2. Szukanie i zliczanie wystąpień wybranego znaku w tekście
   - 1.3. Wyszukiwanie wzorca w tekście metodą naiwną
2. Praca z danymi tekstowymi w językach C++ i Python
   - 2.1. Dane tekstowe w języku C++
   - 2.2. Dane tekstowe w języku Python
   - 2.3. Używanie danych tekstowych w językach C++ i Python
3. Programowanie algorytmów na tekstach w językach C++ i Python
   - 3.1. Zapisywanie algorytmu porównywania tekstów w językach C++ i Python
   - 3.2. Zapisywanie algorytmu szukania i zliczania wystąpień znaku w tekście w językach C++ i Python
   - 3.3. Zapisywanie algorytmu wyszukiwania wzorca w tekście metodą naiwną w językach C++ i Python

Warto powtórzyć

1. W jaki sposób deklarujemy zmienne w języku C++?
2. Dlaczego w języku Python nie trzeba deklarować zmiennych?
3. Jak działa instrukcja warunkowa w języku programowania C++ lub Python?
4. Jak można zapisać algorytm iteracyjny w języku programowania C++ lub Python?
5. W jaki sposób definiujemy funkcje w języku programowania C++ lub Python?
6. W jaki sposób możemy odwołać się do elementu tablicy w języku C++ lub do elementu listy w języku Python?

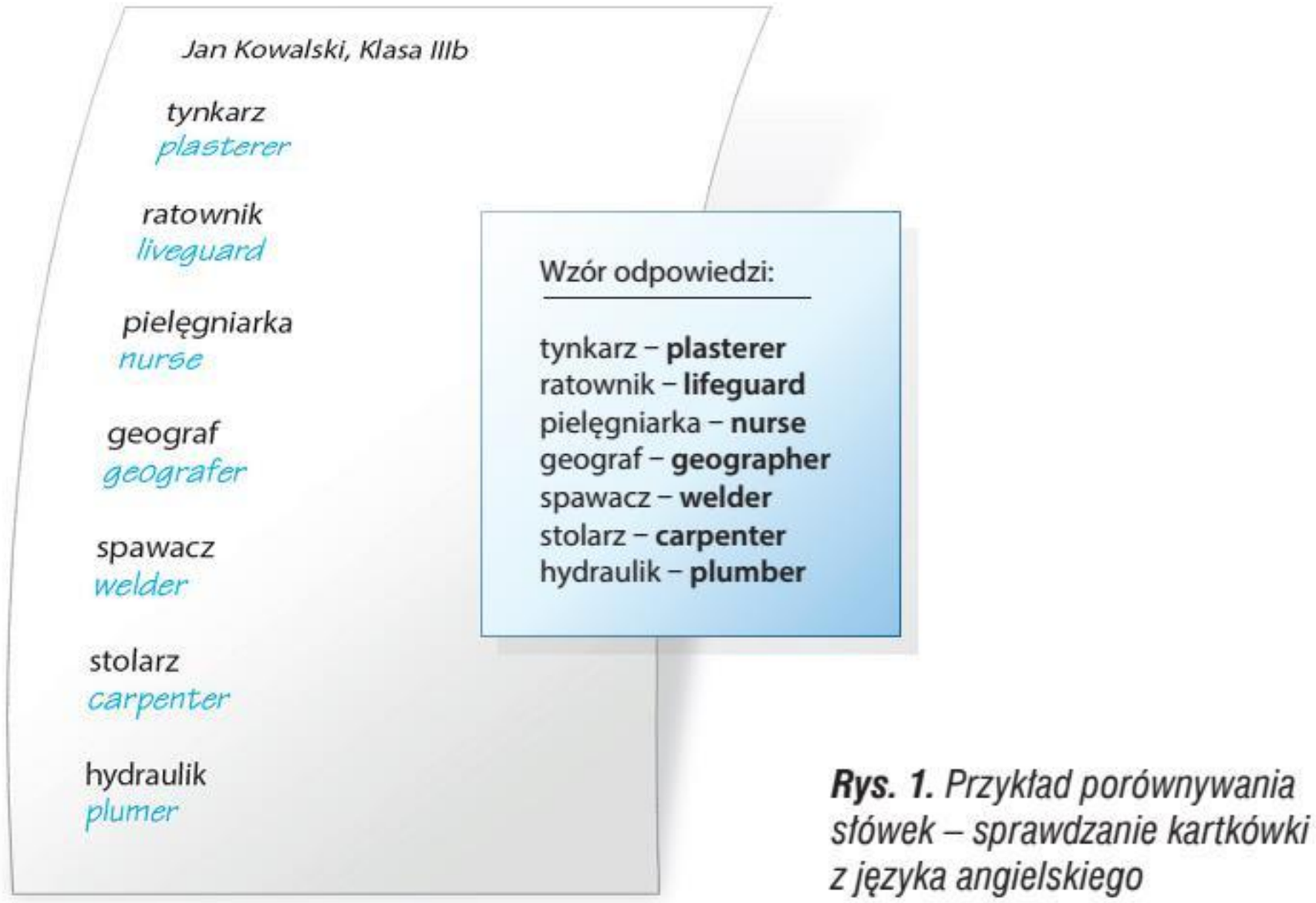
1. Wybrane algorytmy na tekstach

Porównywanie tekstów i wyszukiwanie określonego znaku lub fragmentu tekstu to czynności, które wykonujemy zarówno podczas pracy nad tekstem pisanym na kartce (bez użycia komputera), jak i tworzonym w edytorze tekstu (z wykorzystaniem komputera). Najpierw wskażemy kilka przykładów tych operacji i poszukamy sposobów ich wykonywania – sformułujemy algorytmy. Następnie zapiszemy te algorytmy w wybranym języku programowania.

1.1. Porównywanie tekstów

Nauczyciel języka angielskiego przygotował kartkówkę ze słówek. W jaki sposób może usprawnić sprawdzanie prac?

Nauczyciel przygotował wzorcowe rozwiązanie, z którym porównywał odpowiedzi uczniów – przykład pokazano na rysunku 1.



Rys. 1. *Przykład porównywania słówek – sprawdzanie kartkówki z języka angielskiego*

Słówko w języku polskim	Odpowiedź ucznia	Odpowiedź poprawna
tynkarz	plasterer	plasterer
ratownik	livegard	lifeguard
pielęgniarka	nurse	nurse
geograf	geografer	geographer
spawacz	welder	welder
stolarz	carpenter	carpenter
hydraulik	plumer	plumber

Tabela 1. *Zestawienie do ćwiczenia 1.*

Ćwiczenie 1. Szukamy rozwiązania problemu porównywania słówek

W tabeli 1. przedstawiono odpowiedzi ucznia z kartkówki ze słówek oraz poprawne odpowiedzi. Wskaż błędy w odpowiedziach ucznia. Wyjaśnij sposób porównywania słowa napisanego przez ucznia ze wzorem odpowiedzi. Czy musisz zawsze porównać wszystkie litery z odpowiedzi ucznia ze wzorem? Kiedy kończymy porównywanie?

Ćwiczenie 2. Korzystamy z rozwiązania jednego problemu do rozwiązania innego problemu

Pewien uczeń chce wysłać zaproszenia do zagranicznych gości. Zaadresował je odręcznie. Chcemy pomóc mu sprawdzić, czy nie popełnił błędu w zapisie nazwisk. W tabeli 2. pokazano poprawną formę zapisu oraz nazwiska, które widnieją na zaproszeniach.
Jak sprawdzić, czy dane nazwisko zostało poprawnie zapisane? Czy można zastosować sposób z ćwiczenia 1.? Uzasadnij odpowiedź.

Zapis nazwiska na zaproszeniu	Poprawny zapis nazwiska
Kobayashi	Kobayashi
Rodrígez	Rodríguez
Yorck won Wartenburg	Yorck von Wartenburg
Bhaktwedanta	Bhaktiwedanta
Aberystwyth	Aberystwyth
von Kauniz-Rietberg	von Kaunitz-Rietberg

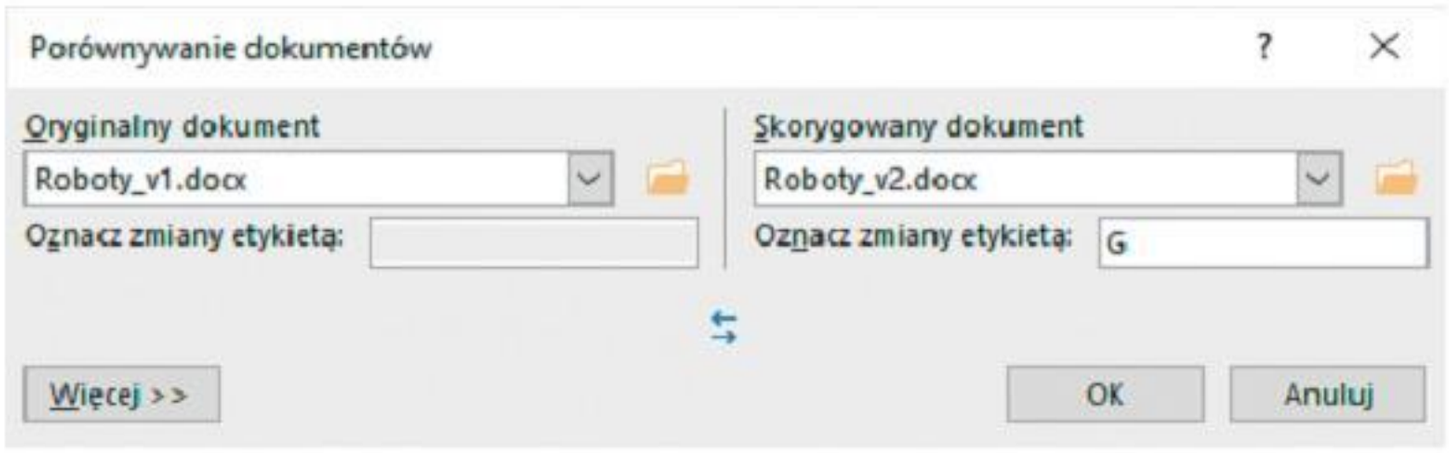
Tabela 2. *Zestawienie do ćwiczenia 2.*

Ćwiczenie 3. Sprawdzamy stosowanie tego samego algorytmu do rozwiązania innych problemów

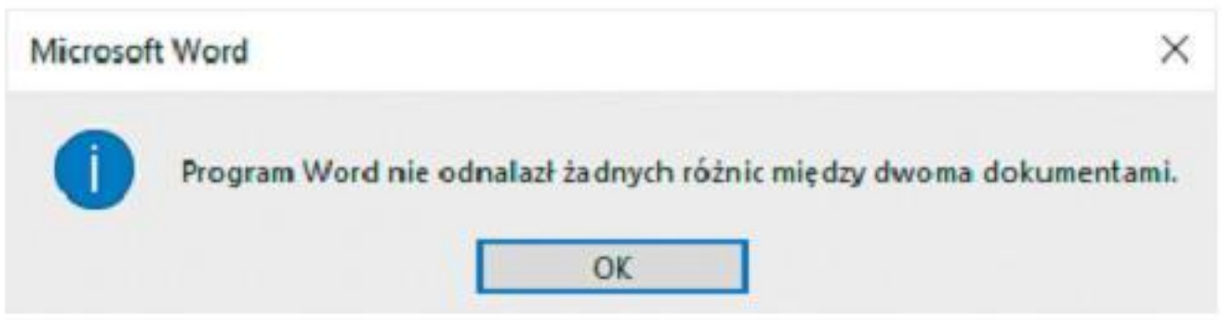
1. Czy algorytm porównywania tekstów z ćwiczeń 1. i 2. można zastosować w następujących sytuacjach:
   a. sprawdzanie numeru konta, które wpisaliśmy do pola formularza bankowego,
   b. sprawdzanie adresu e-mail, który wpisaliśmy do pola adresu.
2. Podaj dwa inne przykłady sytuacji, w których porównuje się teksty.

Napisaliśmy tekst o robotach. Pracę sprawdziła inna osoba, nie wiemy jednak, w których miejscach wprowadziła poprawki, ponieważ nie zaznaczyła ich w tekście. W jaki sposób można porównać oba teksty?

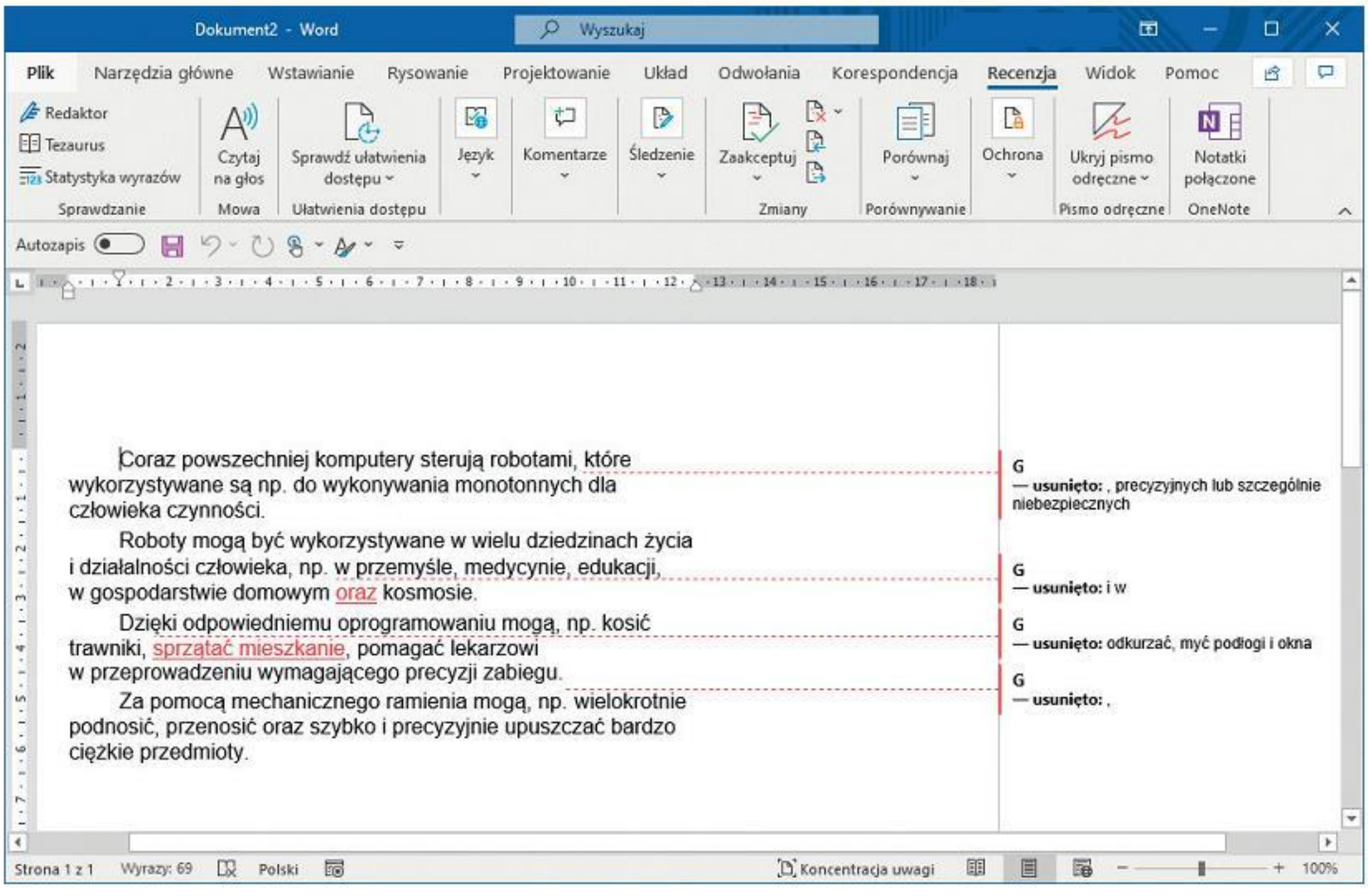
W przypadku dokumentów pisanych ręcznie musielibyśmy czytać wiersz po wierszu, aby znaleźć różnice. Edytory tekstu (m.in. Microsoft Word) mają odpowiednią funkcję do porównywania dokumentów.



Rys. 2a. *Porównywanie dokumentów w programie Microsoft Word*



Rys. 2b. *Komunikat programu Microsoft Word w przypadku nieróżniących się dokumentów*



Rys. 2c. *Nowy dokument (wynik porównania) utworzony w przypadku różniących się dokumentów*

Ćwiczenie 4. Porównujemy dwa teksty w edytorze tekstu

1. Odszukaj w edytorze tekstu Microsoft Word opcję porównywania tekstów.
2. Porównaj dwa dokumenty tekstowe (zapisane w plikach, które otrzymasz od nauczyciela).

Uwaga !

Do profesjonalnych zastosowań stosuje się bardziej złożone algorytmy porównywania tekstów.

Porównywanie tekstów to sprawdzenie (znak po znaku), czy teksty są takie same. Algorytmy porównywania tekstów mogą uwzględniać lub ignorować wielkość liter, np. „iteracja" i „Iteracja" – mogą być jednakowymi tekstami (przy nieuwzględnianiu wielkości liter) lub różnymi (przy uwzględnianiu wielkości liter).



Algorytm porównywania dwóch tekstów (np. *tekst1* i *tekst2*) polega na porównywaniu kolejnych znaków, zaczynając od pierwszego znaku każdego z tekstów.

Sprawdzanie powtarzamy do momentu, gdy *i*-te znaki obydwu tekstów będą różne lub gdy osiągniemy koniec jednego tekstu lub obu tekstów.

1.2. Szukanie i zliczanie wystąpień wybranego znaku w tekście

{ W pracy na temat robotów chcemy wyszukać znaki średnika i zamienić je na przecinki. W jaki sposób można to zrobić? }

Jeśli w tekście mielibyśmy wyszukać znaki średnika tradycyjnie, przeglądalibyśmy tekst znak po znaku. Jeśli tekst wprowadziliśmy w edytorze tekstu, to znajdowanie znaków, ich zliczanie oraz zamianę jednego znaku na inny umożliwiają odpowiednie opcje programu.

Edytor Microsoft Word wyróżnia kolorem żółtym wszystkie znalezione znaki i podaje liczbę ich wystąpień.



Ćwiczenie 5. Szukamy wystąpień znaku i zastępujemy go innym w dokumencie tekstowym

W pliku otrzymanym od nauczyciela wyszukaj wystąpienia średnika (znaku ;). Sprawdź liczbę tych znaków w tekście. Zamień wszystkie średniki na przecinki, stosując odpowiednie opcje edytora.



Ćwiczenie 6. Szukamy wystąpień znaku w tekście

1. Wyszukaj i zlicz litery „a" w słowie „matematyka". Przeprowadź analizę rozwiązania, tworząc i uzupełniając tabelę podobną do tabeli 3. Odpowiedz na pytania:
   a. Ile razy powtarza się operacja porównania, a ile zliczania?
   b. Czy trzeba sprawdzać wszystkie znaki tekstu? Uzasadnij odpowiedź.
2. Sformułuj algorytm szukania wystąpień wybranego znaku.

Wskazówka: Przyjmij na początku *licznik* = 0.

Lp.	Pytanie	Odpowiedź	Działanie
1.	Czy pierwszy znak tekstu jest taki sam jak szukany znak	NIE	Przejdź do sprawdzenia następnego znaku tekstu.
2.	Czy drugi znak tekstu jest taki sam jak szukany znak?	TAK	Zwiększ *licznik* o 1: *licznik* = *licznik* + 1. Przejdź do sprawdzenia następnego znaku tekstu.

Tabela 3. *Przykładowa analiza rozwiązania – wzór tabeli do ćwiczenia 6.*

1.3. Wyszukiwanie wzorca w tekście metodą naiwną

Napisaliśmy pracę domową z informatyki na temat robotów. Okazało się jednak, że w niektórych miejscach w skrócie „np." wstawiliśmy na końcu przecinek zamiast kropki. Jak szybko wyszukać i poprawić błędne zapisy?

Gdyby praca była zapisana ręcznie na kartce, podobnie jak podczas szukania pojedynczego znaku czytalibyśmy tekst wiersz po wierszu, aby znaleźć konkretne zapisy. Jeśli pracę wykonaliśmy w edytorze tekstu, to znajdowanie ciągów znaków, ich zliczanie oraz zamianę na inne umożliwia ta sama opcja, która służy do wyszukania pojedynczego znaku.

Ćwiczenie 7. Szukamy ciągu znaków i zastępujemy go innym w dokumencie tekstowym

Uruchom program Microsoft Word. Korzystając z pliku otrzymanego od nauczyciela, wyszukaj w tekście wystąpienia ciągu znaków „np,". Popraw tekst, stosując odpowiednią opcję edytora tekstów.

Spróbujmy znaleźć sposób szukania ciągu znaków, czyli **wzorca** w tekście.

Przykład 1. Szukanie wystąpień wzorca w tekście

Chcemy znaleźć wystąpienia wzorca „mat" w słowie „matematyka".
Szukanie zaczynamy od początku tekstu. Sprawdzamy, czy pierwszy znak tekstu jest taki sam jak pierwszy znak wzorca. Jeśli znaki są różne, przesuwamy wzorzec o jedną pozycję w prawo względem tekstu. Jeśli znaki są takie same, sprawdzamy drugi znak tekstu i drugi znak wzorca, trzeci znak itd. – aż do sprawdzenia wszystkich znaków wzorca. Jeśli są różne, przesuwamy wzorzec o jedną pozycję w prawo.
W przypadku znalezienia wzorca, kolejne wyszukiwanie wzorca zaczynamy od następnego znaku tekstu. Może się zdarzyć, że wzorzec pojawia się w tekście wielokrotnie.

Wzorzec

Spójny ciąg znaków.



Rys. 3. *Szukanie wystąpień wzorca w tekście – znaleziono pierwsze wystąpienie wzorca*

Ćwiczenie 8. Szukamy wystąpień wzorca w tekście

1. Korzystając z przykładu 1., dokończ wyszukiwanie wystąpień wzorca „mat” w słowie „matematyka”. Wykonaj ćwiczenie, przygotowując odpowiednie pomoce dydaktyczne (kartoniki z literami). Przeprowadź analizę rozwiązania, tworząc i uzupełniając tabelę podobną do tabeli 4.
2. Odpowiedz na pytania:
   a. *Ile jest wystąpień szukanego wzorca w danym tekście i na jakiej pozycji je znaleziono?*
   b. *Kiedy kończymy przeglądanie tekstu? Czy musimy sprawdzać ostatnie dwa znaki tekstu? Uzasadnij odpowiedź.*
   c. *Jakie powtarzające się operacje zauważasz w rozwiązaniu?*

Lp.	Pytanie	Odpowiedź	Działanie
1.	Czy pierwszy znak tekstu jest taki sam jak pierwszy znak wzorca?	TAK	Przejdź do sprawdzenia następnego znaku tekstu i wzorca.
2.	Czy drugi znak tekstu jest taki sam jak drugi znak wzorca?	TAK	Przejdź do sprawdzenia następnego znaku tekstu i wzorca.
3.	Czy trzeci znak tekstu jest taki sam jak trzeci znak wzorca?	TAK	Wzorzec znaleziono na pozycji 1. Przesuwamy wzorzec o jedną pozycję w prawo.
4.	Czy drugi znak tekstu jest taki sam jak pierwszy znak wzorca?	NIE	Przesuwamy wzorzec o jedną pozycję w prawo.

Tabela 4. *Przykładowa analiza rozwiązania – wzór tabeli do ćwiczenia 8.*

2. Praca z danymi tekstowymi w językach C++ i Python

Zapisując wybrany algorytm w języku programowania (C++ lub Python), m.in. dobieramy odpowiednie struktury danych. Omówione algorytmy dotyczą pracy na tekstach, dlatego będziemy stosować tekstowe typy danych. Powinniśmy wiedzieć, jak używać zmiennych typu tekstowego, m.in. jak je deklarować (jeśli to konieczne), jak się do nich odwoływać i wykonać na nich inne działania w programie. Dlatego najpierw poznamy kilka zasad pracy z danymi tekstowymi.

2.1. Dane tekstowe w języku C++

Jeśli w języku C++ danymi do zadania są pojedyncze znaki, wówczas deklarujemy zmienne typu znakowego.

Deklaracja zmiennych typu znakowego	`char zmienna_znakowa;`	C++

W języku C++ stałe znakowe umieszcza się w apostrofach (w odróżnieniu od tekstów, które umieszcza się w górnych cudzysłowach):

```
zmienna_znakowa = 'x';
```

W języku C++ zmienna typu **char** ma również wartość liczbową, odpowiadającą kodowi ASCII w zmiennej zapamiętanego znaku. Poprawna jest zatem np. instrukcja wyprowadzająca na ekran znak „y” (w przypadku uprzedniego wykonania powyższej instrukcji):

```
cout << zmienna_znakowa + 1;
```

Jeśli danymi w algorytmie są teksty lub inne ciągi znaków, to zapisując algorytm w postaci programu, należy zmienne tekstowe zadeklarować w języku C++ jako typ łańcuchowy (ciąg znaków w języku C++ nazywamy łańcuchem).

W języku C++ w celu łatwiejszego korzystania z łańcuchów można wykorzystać klasę standardową `std::string`. Aby ją wykorzystać, należy umieścić na początku programu polecenie `#include <string>`.

W języku C++ łańcuch to również tablica znaków o dowolnym, wcześniej ustalonym rozmiarze. W języku tym łańcuch zakończony jest znakiem `NULL`, zapisywanym jako `'\0'`. Przy deklarowaniu zmiennej łańcuchowej należy więc dodać 1 do przewidywanego rozmiaru tekstu, by zmieścić znak `NULL`.

Deklaracja zmiennych typu tekstowego	`string lancuch;`	C++
	`char lancuch[n + 1];`	C++



Aby w języku C++ wyznaczyć długość łańcucha, możemy użyć standardowej funkcji `length()`.
Na przykład `napis.length()` oznacza długość łańcucha pamiętanego w zmiennej `napis` typu `string`.
Do wyznaczenia długości tekstu zapisanego jako tablica znaków można użyć funkcji standardowej `strlen()`.

Do pojedynczego znaku łańcucha odwołujemy się tak samo jak do elementu tablicy, czyli używając indeksu: `lancuch[i]`. Znaki łańcucha są numerowane od zera. Indeks ostatniego znaku obliczamy, odejmując 1 od długości napisu, np. `napis.length() - 1`.

Deklaracja łańcucha o długości do 15 znaków	`char slowo[16];`	W zmiennej `slowo` można zapamiętać napis o długości do 15 znaków.
Deklaracja łańcucha o długości do 19 znaków	`#define N 20` `char tekst[N];`	W zmiennej `tekst` można zapamiętać napis o długości do 19 znaków.
Deklaracja łańcucha z wykorzystaniem klasy `string`	`#include <string>` ... `string napis;`	Liczba znaków pamiętanych w zmiennej typu `string` jest ograniczona tylko przez pamięć RAM, czyli teoretycznie w systemie 32-bitowym można w zmiennej `napis` zapamiętać 2^{32} znaków.

***Tabela 5.** Przykłady deklarowania zmiennych tekstowych w języku C++*

W języku C++ deklaracja łańcucha danej długości to określenie maksymalnej długości łańcucha. Oznacza to, że zawarty w nim napis może zawierać mniejszą liczbę znaków. Jeśli wprowadzony łańcuch jest krótszy niż zadeklarowany, nie wolno odwoływać się do nieistniejących znaków, np. w przypadku wykonania instrukcji: `slowo = "drzewo"` odwołanie `slowo[8]` będzie błędne, ponieważ słowo „drzewo" składa się z sześciu liter.

Przykład 2. Wprowadzanie danych tekstowych w języku C++

W języku C++ dane tekstowe możemy:

- wprowadzać z klawiatury:
  C++
  ```
  cin >> napis;
  ```
- zastosować instrukcję przypisania:
  C++
  ```
  napis = "Akumulator";
  ```
- przypisać wartość zmiennej łańcuchowej w momencie jej deklarowania:
  C++
  ```
  string napis = "Akumulator";
  ```
  lub
  ```
  char napis[11] = "Akumulator";
  ```
  lub
  ```
  char napis[11] = {'A', 'k', 'u', 'm', 'u', 'l', 'a', 't', 'o',
  'r', '\0'};
  ```
  lub
  ```
  char napis[] = "Akumulator";
  ```
  (wielkość tablicy zostanie dopasowana do napisu, w tym przypadku będzie to 10 + 1 znaków).

Aby wyprowadzić napis na ekran, używamy (tak jak w przypadku liczb i komunikatów) polecenia `cout <<` (rys. 4a).

C++ Wyprowadzanie_napisu_v1.cpp

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

int main()
{
    string wyraz;

    cout << "Podaj wyraz: ";
    cin >> wyraz;
    cout << "Wprowadzony wyraz: " << wyraz;
    return 0;
}

```

Rys. 4a. *Wyprowadzenie napisu na ekran (C++)*

Możemy też wyprowadzić na ekran litery wprowadzonego napisu w kolumnie, używając instrukcji iteracyjnej **`for`** (rys. 4b).

C++

```
    cout << "Litery wyrazu: " << wyraz << " w kolumnie" << endl;
    for(int i = 0; i < wyraz.length(); i++)
        cout << wyraz[i] << endl;

```

Rys. 4b. *Wyprowadzenie w kolumnie liter wprowadzonego napisu (C++)*

Napisy możemy ze sobą łączyć, używając operatora +. Na przykład instrukcja

```
cout << "Stosujemy dane" + " " + "tekstowe w C++";
```

wyprowadzi na ekran napis: „Stosujemy dane tekstowe w C++” . Łączenie łańcuchów nazywamy **konkatenacją**.

Możemy również łączyć łańcuchy, które wcześniej przypiszemy do zmiennych (rys. 5.).

C++

```
string s1, s2;
s1 = "Stosujemy dane";
s2 = "tekstowe w C++";
cout << s1 + " " + s2;
```

Rys. 5. *Przykład łączenia napisów (C++)*

2.2. Dane tekstowe w języku Python

Jak wiemy, w języku Python nie deklarujemy zmiennych – zmienna przyjmuje typ wartości, która zostanie do niej przypisana. Na przykład instrukcja:

```
nazwisko = "Kowalski"
```

zmiennej `nazwisko` przypisze wartość, która jest łańcuchem tekstowym, czyli zmienna `nazwisko` będzie typu łańcuchowego `str`.

Typ tekstowy (`str`) reprezentuje ciągi danych znakowych (napisy, łańcuchy znaków).

Przykład 3. Wprowadzanie danych tekstowych w języku Python

Dane tekstowe można wprowadzać z klawiatury:

Python
```
napis = input("Wprowadź tekst: ")
```

Można również zastosować instrukcję przypisania:

Python
```
napis = "akumulator"
```

Aby wyprowadzić napis na ekran, użyjemy (tak jak w przypadku liczb i komunikatów) funkcji `print()` – rys. 6a.

Python

"Wyprowadzanie_napisu_v1.py - C:\Python\Wyprowadzanie_napisu_v1.py (3.9.0)

File Edit Format Run Options Window Help

```
wyraz = input("Podaj wyraz: ")
print(wyraz)
```

Rys. 6a. *Wyprowadzenie napisu na ekran (Python)*

Możemy też litery wyprowadzić na ekran w kolumnie, używając instrukcji iteracyjnej **`for`** (w języku Pyton łańcuchy znaków są traktowane podobnie jak listy zawierające znaki).

Python

Wyprowadzanie_napisu_v2.py - C:\Python\Wyprowadzanie_napisu_v2.py (3.9.0)

File Edit Format Run Options Window Help

```
wyraz = input("Podaj wyraz: ")
for litera in wyraz:
    print(litera)
```

Rys. 6b. *Wyprowadzenie liter napisu w kolumnie (Python)*

Aby w języku Python wyznaczyć długość łańcucha, możemy użyć standardowej funkcji `len()`. Na przykład `len(napis)` oznacza długość łańcucha pamiętanego w zmiennej `napis`.

W języku Python możemy odwoływać się do pojedynczego znaku łańcucha tak samo jak do elementu listy, czyli używając indeksu, np. `napis[i]`. Znaki łańcucha są numerowane od zera. Indeks ostatniego znaku obliczamy, odejmując 1 od długości napisu, np. `len(napis) - 1`.

Programy pokazane na rysunkach 6b i 6c umożliwiają wprowadzenie napisu z klawiatury i wyprowadzenie na ekran w kolumnie wszystkich jego liter.

Python

Wyprowadzanie_napisu_v3.py - C:\Python\Wyprowadzanie_napisu_v3.py (3.9.0)

File Edit Format Run Options Window Help

```
wyraz = input("Podaj wyraz: ")
for i in range(len(wyraz)):
    print(wyraz[i])
```

Rys. 6c. *Wyprowadzenie liter napisu w kolumnie z wykorzystaniem zmiennych indeksowanych (Python)*

Napisy możemy ze sobą łączyć, używając operatora +. Na przykład instrukcja: `print("sa" + "mo" + "chód")` wyprowadzi na ekran napis „samochód". Łączenie łańcuchów nazywamy **konkatenacją**.

Możemy również łączyć łańcuchy, które wcześniej przypiszemy do zmiennych.

Python

Dodawanie_napisow.py - C:\Python\Dodawanie_napisow.py (3.9.0)

File Edit Format Run Options Window Help

```
w1 = "oto"
w2 = "ma"
w3 = "na"
print(w1 + w2 + w3)
```

Rys. 7. *Przykład łączenia napisów (Python)*

W języku Python możemy **wyodrębnić** fragment napisu, wskazując indeksy pierwszego i ostatniego znaku fragmentu i rozdzielając je dwukropkiem.



Aby w języku Python wyodrębnić fragment tekstu z łańcucha `napis`:

- od znaku o indeksie *`indeks1`* do znaku o indeksie *`indeks2`* – 1 (włącznie) stosujemy zapis: *`napis`*`[`*`indeks1`*`:`*`indeks2`*`]`,
- od pierwszego znaku tekstu do znaku o indeksie *`indeks2`* – 1 (włącznie) stosujemy zapis: *`napis`*`[:`*`indeks2`*`]`,
- od znaku o indeksie *`indeks1`* do końca tekstu stosujemy zapis: *`napis`*`[`*`indeks1`*`:]`.

Program pokazany na rysunku 8. wyświetla pierwszy znak tekstu (o indeksie 0) i dwa ostatnie znaki (fragment tekstu od znaku o indeksie 3 do znaku o indeksie 4).

Python

Wycinanie.py - C:\Python\Wycinanie.py (3.9.0)

File Edit Format Run Options Window Help

```
wyraz = "abecd"
print(wyraz[0:1])
print(wyraz[3:5])
```

```
==========
a
cd
>>>
```

Rys. 8. *Przykład wyodrębniania fragmentów tekstu (Python)*

Załóżmy, że w wyrazie „aaba” chcemy zamienić pierwszą literę „a” na „b”. W języku Python nie możemy zamienić znaków w zmiennej typu tekstowego, używając instrukcji przypisania w taki sposób jak w języku C++ (rys. 11a). W programie pokazanym na rysunku 9. interpreter wskaże błąd.

Python

Zamiana_niepoprawnie.py - C:\Python\Zamiana_niepoprawnie.py (3.9.0)

File Edit Format Run Options Window Help

```
wyraz =  "aaba"
wyraz[0] = "b"
print(wyraz)
```

```
================== RESTART: C:\Python\Zamiana_niepoprawnie.py =====
Traceback (most recent call last):
  File "C:\Python\Zamiana_niepoprawnie.py", line 2, in <module>
    wyraz[0] = "b"
TypeError: 'str' object does not support item assignment
>>>
```

Rys. 9. *Błędny przykład wyodrębniania fragmentów tekstu (Python)*

Możemy poradzić sobie z tym problemem w następujący sposób:
zamiast instrukcji: `wyraz[0] = "b"`
możemy napisać: `wyraz = "b" + wyraz[1:]`,
co oznacza: zmiennej `wyraz` przypisz konkatencję łańcucha składającą się z litery „b”
i fragmentu łańcucha `wyraz`, począwszy od drugiej litery do ostatniej.

Python

Zamiana_poprawnie.py - C:\Python\Zamiana_poprawnie.py (3.9.0)

File Edit Format Run Options Window Help

```
wyraz =  "aaba"
wyraz = "b"  + wyraz[1:]
print(wyraz)
```

```
============
baba
>>> |
```

Rys. 10. *Przykład wyodrębniania fragmentów tekstu i zamiany liter (Python)*

2.3. Używanie danych tekstowych w językach C++ i Python

Ćwiczenie 9. Poznajemy pracę z tekstami w wybranym języku programowania

Zapoznaj się z zasadami pracy z danymi tekstowymi podanymi w punktach 2.1. lub 2.2. Sprawdź je, pisząc krótkie programy pokazane w przykładach i na rysunkach.

Wskazówka: (Python) W języku Python działanie instrukcji lub krótkich programów możesz sprawdzić w trybie interaktywnym, wpisując polecenia po znaku zachęty >>> w oknie powłoki Pythona.

Ćwiczenie 10. Analizujemy gotowy program w wybranym języku programowania

1. Przeanalizuj program pokazany na rysunku 11a lub 11b. Sformułuj treść zadania realizowanego przez ten program i napisz specyfikację zadania (określ dane i wyniki). Wyjaśnij działanie poszczególnych instrukcji.
2. Przepisz program, uruchom go i przetestuj dla kilku różnych danych.

C++

[*] Wyraz.cpp

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

int main()
{
    string wyraz;

    cout << "Podaj wyraz: ";
    cin >> wyraz;
    for(int i = 0; i < wyraz.length(); i++)
    if( wyraz[i] == 'a' )
        wyraz[i] = 'b';
    cout << "Wyraz po przetworzeniu: " << wyraz << endl;
    return 0;
}
```

Rys. 11a*. Przykładowy program z wykorzystaniem danych tekstowych (C++)*

Python

Wyraz.py - C:\Python\Wyraz.py (3.9.0)

File Edit Format Run Options Window Help

```
wyraz = input("Podaj wyraz: ")

for i in range(len(wyraz)):
    if wyraz[i] == "a":
        wyraz = wyraz[:i] + "b" + wyraz[i+1:]

print("Wyraz po przetworzeniu:", wyraz)


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

***Rys. 11b**. Przykładowy program z wykorzystaniem danych tekstowych (Python)*

Ćwiczenie 11. Stosujemy dane tekstowe w wybranym języku programowania

1. W języku C++ lub Python napisz program wyprowadzający na ekran pierwszą i ostatnią literę wyrazu wprowadzonego z klawiatury.
2. Zapisz program w pliku pod nazwą *Wyraz_pierwsza_ostatnia*.
3. Przetestuj program dla kilku różnych danych.

Ćwiczenie 12. Modyfikujemy program w wybranym języku programowania

1. Zmodyfikuj program zapisany w ćwiczeniu 11. Program powinien zamieniać miejscami pierwszą literę z ostatnią, gdy są różne, a następnie wyprowadzić na ekran zmieniony wyraz. W przypadku jednakowych liter powinien się wyświetlić odpowiedni komunikat.
2. Zapisz program w pliku pod nazwą *Wyraz_zamiana_pierwsza_ostatnia*.
3. Przetestuj działanie programu dla kilku różnych danych.

Ćwiczenie 13. Stosujemy dane tekstowe w wybranym języku programowania

1. W języku C++ lub Python napisz program zamieniający co drugą literę wyrazu wprowadzonego z klawiatury na literę „x” i wyprowadzający zmieniony wyraz na ekran.
2. Zapisz program w pliku pod nazwą *Wyraz_zamiana_na_x*.
3. Przetestuj działanie programu dla kilku różnych danych.

Wskazówka: Zastanów się, którą instrukcję pętli najlepiej tu zastosować.

```
Podaj wyraz: qwerty
Wyraz po zamianie co drugiej litery na x: qxextx
```

***Rys. 12.** Przykładowy wynik programu – ćwiczenie 13.*

3. Programowanie algorytmów na tekstach w językach C++ i Python

W algorytmach, które poznaliśmy w punkcie 1. tego tematu, pracowaliśmy na tekstach i mieliśmy do czynienia z sytuacjami warunkowymi oraz powtarzaniem operacji. Dlatego w programach będziemy stosować zmienne typu tekstowego, instrukcję warunkową oraz odpowiednie instrukcje iteracyjne.

3.1. Zapisywanie algorytmu porównywania tekstów w językach C++ i Python

Zauważmy, że wynikiem działania algorytmu porównywania tekstów jest prawda albo fałsz (teksty są takie same lub różne). Aby zapisać ten algorytm w wybranym języku programowania, zdefiniujemy funkcję logiczną zwracającą wartość z dwoma parametrami (będącymi tymi tekstami).

Ćwiczenie 14. Analizujemy funkcję porównującą teksty

Objaśnij działanie funkcji pokazanej na rysunku 13a lub 13b, odpowiadając na pytania:

a. Co określają parametry funkcji?
b. Dlaczego w funkcji *teksty_sa_identyczne()* zastosowano instrukcję iteracyjną **`while`**, a nie **`for`**? Uzasadnij odpowiedź.
c. Dlaczego w instrukcji **`while`** zastosowano złożony warunek logiczny (koniunkcję)? Objaśnij poszczególne elementy tego warunku (omów, co sprawdzają).
d. Jaką wartość zwraca funkcja?
e. Jakie będzie działanie funkcji dla tekstów różniących się tylko wielkością liter, np. „abc" i „ABC"?

Uwaga

Języki programowania często udostępniają funkcje i metody związane z operacjami na tekstach, np. w języku Python metoda `count()` szuka i zlicza wystąpienia znaków lub ciągów znaków w tekście; w języku C++ funkcja `strcmp()` porównuje dwa łańcuchy znaków.

C++

```
 5  bool teksty_sa_identyczne(string t1, string t2)
 6  {
 7      int i, dl1, dl2;
 8      i = 0;
 9      dl1 = t1.length();
10      dl2 = t2.length();
11      while(i < dl1 && i < dl2 && t1[i] == t2[i])
12          i++;
13      return i == dl1 && i == dl2;
14  }
```

***Rys. 13a**. Funkcja porównująca teksty (C++) – ćwiczenia 14. i 15.*

Python

```
def teksty_sa_identyczne(t1, t2):
    i = 0
    dl1 = len(t1)
    dl2 = len(t2)
    while(i < dl1 and i < dl2 and t1[i] == t2[i]):
        i = i + 1
    return i == dl1 and i == dl2
```

***Rys. 13b**. Funkcja porównująca teksty (Python) – ćwiczenia 14. i 15.*

Ćwiczenie 15. Piszemy program realizujący algorytm porównywania tekstów

1. Korzystając z opisu algorytmu porównywania tekstów z punktu 1.1. tematu, napisz specyfikację zadania. Na tej podstawie utwórz program realizujący algorytm porównywania dwóch tekstów. Zdefiniuj funkcję *teksty_sa_identyczne()* pokazaną na rysunku 13a lub 13b. Wywołaj ją w programie głównym z parametrami aktualnymi (argumentami) *tekst1* i *tekst2*. Wyprowadzaj na ekran odpowiednie komunikaty.
2. Zapisz program w pliku pod nazwą *Porównywanie*.
3. Przetestuj działanie programu dla kilku różnych danych.

Wskazówka: W funkcji *teksty_sa_identyczne()* przed instrukcją pętli możesz sprawdzać, czy teksty są takiej samej długości: jeśli nie, to wartością funkcji będzie *fałsz*.

3.2. Zapisywanie algorytmu szukania i zliczania wystąpień znaku w tekście w językach C++ i Python

Zliczanie wystąpień jednego znaku

W algorytmie szukania i zliczania wystąpień znaku w tekście powtarzamy sprawdzenie, czy kolejny znak tekstu jest równy podanemu znakowi – jeśli tak, zliczamy go. Mamy tu do czynienia z sytuacją warunkową i iteracją. Aby zaprogramować ten algorytm, zastosujemy instrukcję warunkową (do sprawdzenia, czy kolejny znak tekstu jest równy podanemu znakowi), instrukcję przypisania (do zwiększania wartości licznika) i iteracyjną (do powtarzania tych obydwu operacji).

Do zliczenia liczby wystąpień konkretnego znaku (np. „a") możemy zdefiniować funkcję z jednym parametrem (*tekst*) zwracającą do programu głównego liczbę znalezionych znaków „a" w tekście *tekst* (wartość zmiennej *licznik*).

Jeśli funkcja miałaby zliczać dowolny znak tekstu, który wprowadzalibyśmy (tak jak tekst) z klawiatury, należy dodać do definicji funkcji drugi parametr – określający ten znak.

Ćwiczenie 16. Piszemy program zliczający wystąpienia znaku „a" w tekście

1. Napisz program wyszukujący i zliczający literę „a" w tekście wprowadzanym z klawiatury. Zapisz specyfikację tego zadania. Zdefiniuj funkcję *zlicz_znak* z jednym parametrem *tekst*. Wywołaj funkcję w programie głównym z parametrem aktualnym (argumentem) *napis*. Dla przejrzystości wykonywania programu wyprowadzaj odpowiednie komunikaty.
2. Zapisz program w pliku pod nazwą *Zliczanie_znaku_a*.
3. Przetestuj działanie programu dla kilku różnych danych.

Wskazówki:

Instrukcja **`for`** zastosowana w funkcji *zlicz_znak* może mieć postać:

C++

```
for(int i = 0; i < tekst.length(); i++)
    if(tekst[i] == 'a')
        liczba_znakow++;
```

Python

```
for i in range(len(tekst)):
    if(tekst[i] == 'a'):
        liczba_znakow = liczba_znakow + 1
return liczba_znakow
```

lub

```
for znak in tekst:
    if(znak == 'a'):
        liczba_znakow = liczba_znakow + 1
return liczba_znakow
```

W języku C++ wprowadzanie danych za pomocą polecenia `cin >>` powoduje wczytanie danych do zmiennej, aż do napotkania tzw. białego znaku (np. spacji, tabulatora, znaku nowego wiersza). Jeśli wprowadzimy dwa wyrazy oddzielone spacją, to drugi wyraz nie zostanie wczytany. Podajemy dwa przykładowe sposoby poradzenia sobie z tym problemem:

- Jeśli łańcuch deklarujemy jako tablicę znaków, możemy zastosować instrukcję `cin.get()`, która czeka na naciśnięcie dowolnego klawisza klawiatury i natychmiast zwraca jego kod ASCII. Wykorzystywana jest więc do odczytania „znak po znaku" całego napisu, aż do odczytania kodu znaku nowego wiersza.
- Jeśli zastosujemy klasę standardową `string` (łańcuch zadeklarujemy jako `string`), możemy wykorzystać funkcję `getline()`, umożliwiającą wczytywanie całego wiersza. Naciśnięcie klawisza **Enter** kończy wprowadzanie danych. Na przykład zamiast instrukcji `cin >> napis;` należy użyć `getline(cin, napis)`.

W języku Python do zliczania znaków w tekście możemy zastosować metodę `count()`, na przykład dla tekstu zapisanego w zmiennej `tekst`, w którym chcemy zliczyć litery „a", zastosujemy ją w następujący sposób: `tekst.count("a")`.



Ćwiczenie 17. Modyfikujemy program zliczający znaki

1. Zmodyfikuj program zapisany w ćwiczeniu 16.:
   a. w języku C++ zastosuj funkcję `getline()` do wprowadzania napisów składających się z wyrazów oddzielonych spacjami,
   b. w języku Python zastosuj metodę `count()`.
2. Zapisz program w pliku pod nazwą *Zliczanie_znaku_a_zmod*.
3. Przetestuj działanie programu dla różnych danych.

Zliczanie wystąpień kilku znaków

Jeśli chcielibyśmy zliczać w jednym programie więcej niż jeden znak, możemy zastosować kilkakrotnie instrukcję warunkową (tyle razy, ile różnych znaków chcemy zliczać). Można również użyć **instrukcji wyboru** (w języku C++) lub instrukcji **`if`** z klauzulą **`else`** (w języku Python). Instrukcje te są również przydatne np. do tworzenia menu ekranowego, w którym użytkownik musi wybrać jedną z opcji.

C++

Instrukcja wyboru	**switch**(wyrażenie) { **case** *etykieta1:* *instrukcja1;* *break;* **case** *etykieta2:* *instrukcja2;* **break;** … **default:** *instrukcjan;* **break;** } *kolejna_instrukcja;*	Na przykład: **switch**(tekst[i]) { **case** 'a': liczba_a++; **break;** **case** 'b': liczba_b++; **break;** **case** ',': liczba_przec++; **break;** }

W języku C++ *wyrażenie* musi mieć typ całkowity lub znakowy, a wszystkie *etykiety* muszą być stałymi: liczbami całkowitymi lub znakami. Najpierw obliczana jest wartość *wyrażenia*, po czym, zależnie od tego, której *etykiecie* będzie równa, zostanie wykonana dana *instrukcja*. Jeśli wynikiem wyrażenia nie będzie żadna z wartości podanych w etykietach, zostanie wykonana *instrukcja* znajdująca się po opcjonalnym słowie **default**. Jeśli nie ma tych słów, program przejdzie do *kolejnej_instrukcji*.

Konieczne jest zwykle zastosowanie instrukcji **break**, w przeciwnym przypadku program przejdzie do wykonania instrukcji znajdującej się po następnym słowie **case**.

Python

Instrukcja if z klauzulą elif	**if** *warunek1*: *instrukcja1* **elif** *warunek2*: *instrukcja2* **elif** *warunek3*: *instrukcja3* … **else**: *instrukcjan* *kolejna_instrukcja*	Na przykład: **if** tekst[i] == 'a': liczba_a = liczba_a + 1 **elif** tekst[i] == 'b': liczba_b = liczba_b + 1 **elif** tekst[i] == ';': liczba_sred = liczba_sred + 1

Gdy wykonanie operacji zależy od różnych wartości zmiennej (tu od wartości znaku łańcucha), w języku Python można zastosować kilka instrukcji warunkowych **if** lub instrukcję warunkową **if** z klauzulą **elif**. Najpierw jest sprawdzany *warunek1*. Jeśli *warunek1* jest prawdziwy, wykonywana jest *instrukcja1* i następuje wyjście z instrukcji. Jeśli *warunek1* nie jest prawdziwy, sprawdzany jest *warunek2* itd.

W instrukcji **if** można umieścić dowolną liczbę klauzul **elif**.

Ćwiczenie 18. Zliczamy wystąpienia kilku znaków w tekście

1. Napisz program wyszukujący i zliczający litery „a", „b" i „c" w tekście wprowadzanym z klawiatury. Zapisz specyfikację tego zadania.
2. Zapisz program w pliku pod nazwą *Zliczanie_znaków*.

3.3. Zapisywanie algorytmu wyszukiwania wzorca w tekście metodą naiwną w językach C++ i Python

Program realizujący algorytm wyszukiwania wzorca w tekście napiszemy zgodnie z opisem podanym w punkcie 1.3. tego tematu.

W wyniku działania programu chcemy otrzymać informację, czy znaleziono wzorzec, a jeśli tak, to na jakiej pozycji (indeks znaku tekstu, od którego znaleziono wzorzec). Ponadto program ma wyszukać wszystkie wzorce w podanym tekście, czyli podać pozycje wszystkich znalezionych wzorców.

Do wyszukiwania wzorca w tekście zdefiniujemy funkcję *znajdz_tekst()* z trzema parametrami: *tekst*, *wzorzec*, *pozycja_startowa*. Trzeci parametr określa, od którego znaku *tekstu* ma być wyszukiwany wzorzec.

W funkcji *znajdz_tekst()* zastosujemy pętle zagnieżdżone **`for`**:

- w pętli zewnętrznej przeglądamy kolejne znaki tekstu, począwszy od *pozycji_startowej*,
- w pętli wewnętrznej przeglądamy kolejne znaki wzorca, porównując je ze znakami tekstu. W przypadku rozbieżności, przerywamy pętlę (instrukcja **`break`**).

Zanim napiszemy cały program, przeanalizujemy działanie przykładowej funkcji znajdującej wzorzec w tekście (rys. 14a i 14b).

C++

```
int znajdz_tekst(string tekst, string wzorzec, int pozycja_startowa)
{
    int dl1 = tekst.length();
    int dl2 = wzorzec.length();

    for(int i = pozycja_startowa; i < dl1-dl2+1; i++)
    {
        int j;
        for(j = 0; j < dl2; j++)
            if(tekst[i+j] != wzorzec[j])
                break;
        if(j == dl2)
            return i;
    }
    return -1;
}
```

***Rys. 14a**. Przykład funkcji znajdującej wzorzec w tekście (C++)*

Python

```python
def znajdz_tekst(tekst, wzorzec, pozycja_startowa):
    dl1 = len(tekst)
    dl2 = len(wzorzec)

    for i in range (pozycja_startowa, dl1-dl2+1):
        for j in range(dl2):
            if tekst[i+j] != wzorzec[j]:
                break
        if j == dl2:
            return i
    return -1
```

***Rys. 14b**. Przykład funkcji znajdującej wzorzec w tekście (Python)*

Analiza realizacji algorytmu wyszukiwania wzorca w tekście z wykorzystaniem funkcji `znajdz_tekst`

```
przeszukiwany_tekst = "matematyka"        dl1 = 10
wzorzec = "mat"                           dl2 = 3
pozycja startowa = 0
dl1 - dl2 + 1 = 10 - 3 + 1 = 8
```

Wartości				
`i`	`j`	`tekst[i+j]`	`wzorzec[j]`	
0		m	-	`i` < 8, wejście do pętli wewnętrznej
0	0	m	m	`j` < 3, wejście do porównania
0	0	m	m	`tekst[i+j] == 'm'` `wzorzec[j] == 'm'` nie ma rozbieżności, kolejna iteracja pętli wewnętrznej, zwiększenie `j` o 1
0	1	a	a	`tekst[i+j] == 'a'` `wzorzec[j] == 'a'` nie ma rozbieżności, kolejna iteracja pętli wewnętrznej, zwiększenie `j` o 1
0	2	t	t	`tekst[i+j] == 't'` `wzorzec[j] == 't'` nie ma rozbieżności, zwiększenie `j` o 1
0	3	m	-	`j` nie jest mniejsze od 3, zakończenie pętli wewnętrznej
0	3	m	-	spełniony jest warunek `j == dl2`, a zatem funkcja kończy się, zwracając aktualną wartość `i` (0)

***Tabela 6.** Analiza rozwiązania – znalezienie pierwszego wystąpienia wzorca w tekście*

Ćwiczenie 19. Analizujemy realizację algorytmu wyszukiwania wzorca w tekście z wykorzystaniem funkcji

Utwórz tabelę podobną do tabeli 6. i dokończ analizę działania funkcji wyszukiwania wzorca „mat" w słowie „matematyka". Załóż, że zmienna *pozycja_startowa* przyjmuje wartości 1, 4.

Funkcję *znajdz_tekst()* wywołamy w programie głównym. Po wykonaniu ćwiczenia 19. powinniśmy zauważyć, że jednokrotne wywołanie funkcji znajduje tylko pierwsze wystąpienie wzorca. Dlatego w programie głównym należy jeszcze zastosować odpowiednio instrukcję pętli, aby program powtarzał wyszukiwanie, aż wzorzec przestanie być znajdowany, i podał wszystkie wystąpienia wzorca (rys. 15.)

Ćwiczenie 20. Piszemy program realizujący algorytm wyszukiwania wzorca w tekście

1. Na podstawie opisu algorytmu wyszukiwania wzorca w tekście (punkt 1.3. tego tematu) napisz specyfikację zadania. Na tej podstawie utwórz program realizujący ten algorytm. Zdefiniuj funkcję *znajdz_tekst()* z trzema parametrami (rys. 14a lub 14b).
2. Wywołaj funkcję *znajdz_tekst()* w programie głównym z parametrami aktualnymi (argumentami) *tekst*, *tekst_do_wyszukania*, *pozycja_startowa*.
3. Wyprowadzaj również odpowiednie komunikaty, np. tak jak pokazano na rysunku 15.
4. Zapisz program w pliku pod nazwą *Wyszukiwanie_wzorca*.
5. Przetestuj działanie programu dla kilku różnych danych.

Wskazówka: Funkcję *znajdz_tekst* można wywołać w następujący sposób:

C++

```
pozycja = znajdz_tekst(przeszukiwany_tekst,
tekst_do_wyszukania, pozycja);
```

Python

```
pozycja = znajdz_tekst(przeszukiwany_tekst,
tekst_do_wyszukania, pozycja)
```

Zastanów się, od której pozycji należy rozpocząć kolejne wyszukiwanie po znalezieniu wzorca.

```
Wprowadz tekst do przeszukania: matematyka
Wprowadz tekst do wyszukania: mat
Znaleziono: | matematyka | mat | 0 |
Znaleziono: | matematyka | mat | 4 |
```

***Rys. 15.** Przykładowy wynik działania programu – ćwiczenie 20.*

Warto zapamiętać

- Algorytm porównywania dwóch tekstów może polegać na porównywaniu kolejnych znaków tych tekstów, zaczynając od pierwszego znaku każdego z nich.

- Aby szukać danego znaku w tekście i zliczać jego wystąpienia, przeglądamy tekst znak po znaku, porównując jego znaki z szukanym znakiem i dodatkowo je zliczamy.
- W przypadku gdy wykonywanie operacji jest uzależnione od różnych wartości zmiennej, w języku C++ można zastosować instrukcję wyboru.
- Naiwne wyszukiwanie wzorca polega na porównywaniu kolejnych znaków tekstu i wzorca. Zaczynamy od sprawdzenia pierwszego znaku tekstu i pierwszego znaku wzorca: jeśli są zgodne, sprawdzamy zgodność drugiego znaku tekstu i drugiego znaku wzorca, gdy nie są zgodne, sprawdzamy zgodność drugiego znaku tekstu i pierwszego znaku wzorca itd. – aż do znalezienia wzorca.
- Typy danych dobiera się do rozwiązywanego zadania, uwzględniając rodzaj informacji, które mają być przetwarzane, oraz sposób ich przetwarzania. Jeśli algorytm wymaga zastosowania danych tekstowych, stosujemy zmienne typu tekstowego.
- Do pojedynczego znaku łańcucha odwołujemy się tak samo, jak do elementu tablicy (C++) lub listy (Python), czyli używając indeksu, np. `napis[i]`.

Pytania i polecenia

1. Na czym polega algorytm porównywania tekstów? Wyjaśnij na przykładzie.
2. W jaki sposób można wyszukać i zliczać wybrany znak w tekście?
3. Na czym polega naiwny algorytm wyszukiwania wzorca w tekście? Wyjaśnij na przykładzie.
4. W jaki sposób stosujemy zmienne tekstowe w języku C++ lub Python?
5. Jak odwołujemy się do pojedynczego elementu łańcucha? Pokaż na przykładzie.
6. Wyjaśnij, w jaki sposób można zapisać algorytm porównywania tekstów w wybranym języku programowania.
7. Wyjaśnij, w jaki sposób można zapisać algorytm znajdowania wzorca w tekście w wybranym języku programowania.

Zadania

1. Wyszukaj wzorzec „rok” w słowie „„mmmrokmmrokmmmmmmmmmrokm”. Przeprowadź analizę rozwiązania, podobnie jak w ćwiczeniu 8.
2. Napisz program, który umożliwi zamianę wszystkich znaków „a” na „b” oraz wszystkich znaków „b” na „a” we wprowadzonym tekście. Skorzystaj z programu pokazanego na rysunku 11a. lub 11b.
3. Napisz program, który wyświetla na ekranie indeks tego znaku łańcucha wprowadzanego z klawiatury, który jest spacją. Jeśli nie ma takiego elementu, program wyświetla komunikat „brak spacji”.
4. Napisz program zliczający, ile znaków spacji występuje w tekście wprowadzanym z klawiatury (*tekst*). Liczba znaków spacji (*liczba_sp*) ma być wyświetlona na ekranie monitora.

Wskazówka:

C++

```
if(tekst[i] == ' ')
  liczba_sp++;
```

Python

```
if tekst[i] == ' ':
  liczba_sp = liczba_sp + 1
```

5. Napisz program zliczający wszystkie samogłoski w podanym tekście.
6. Napisz program sprawdzający, czy podane przez użytkownika polskie imię jest imieniem męskim czy żeńskim.
   Wskazówka: Typowe polskie imiona żeńskie kończą się na literę „a". Niektóre imiona męskie również kończą się na literę „a", ale twój program nie musi tego uwzględniać.
7. Zmodyfikuj program zapisany w ćwiczeniu 17., aby zliczany znak był wprowadzany z klawiatury po uruchomieniu programu.
   Wskazówka: Do definicji funkcji *zlicz_znak* dodaj drugi parametr oznaczający znak.
8. Napisz program, który dla wprowadzonego znaku „i" wyprowadza na ekran twoje imię, dla „n" – twoje nazwisko, a dla znaku innego niż „i" oraz „n" – komunikat „błędny znak".
9. Napisz program zliczający znaki: „*", „%" i „x", wśród *n* znaków wprowadzanych z klawiatury.

Dla zainteresowanych

10. Napisz program, który obliczy, z ilu wyrazów składa się wprowadzony tekst (zakładamy, że jest to zdanie zbudowane z wyrazów oddzielonych spacjami oraz zakończone kropką).
11. Napisz program, który przeczyta tekst wprowadzony z klawiatury (zdanie składające się z wyrazów oddzielonych spacjami, zakończone kropką) i wyprowadzi każdy wyraz w oddzielnym wierszu na ekranie.
12. Napisz program sprawdzający, czy dany ciąg znaków tworzy palindrom.
    Wskazówki:
    - Palindrom to ciąg znaków, który czyta się tak samo od lewej, jak i od prawej strony. Palindronami mogą być słowa, liczby czy dowolne ciągi znaków, np. *kajak*, *potop*, *zaraz, ala*, *12521*, *565*, *gk@kg*, *kos-sok*.
    - Aby sprawdzić, czy ciąg znaków jest palindromem, możemy porównać pierwszy znak z ostatnim, drugi z przedostatnim itd. Aby nie porównywać tych samych znaków podwójnie, czyli ostatniego z pierwszym, przedostatniego z drugim itd., należy ograniczyć sprawdzanie do połowy długości ciągu.

Temat C2. Algorytmy szyfrowania

1. W jakim celu stosujemy szyfry?
2. Podstawowe pojęcia dotyczące szyfrowania
3. Szyfrowanie tekstu metodą podstawieniową na przykładzie szyfru Cezara
4. Szyfrowanie tekstu metodą przestawieniową
5. Programowanie algorytmów szyfrowania w językach C++ i Python
   - 5.1. Miniszyfrowanie – tworzenie anagramów
   - 5.2. Programowanie algorytmu szyfrowania podstawieniowego
   - 5.3. Programowanie algorytmu szyfrowania przestawieniowego
6. Więcej o szyfrowaniu
   - 6.1. Szyfrowanie symetryczne i asymetryczne
   - 6.2. Maszyna Enigma
   - 6.3. Algorytm RSA

Warto powtórzyć

1. W jaki sposób można zadeklarować zmienne tekstowe w języku C++?
2. Jak używamy zmiennych tekstowych w języku Python?
3. W jaki sposób wprowadzamy dane tekstowe w języku C++ lub Python?
4. Jak odwołujemy się do pojedynczego znaku łańcucha w języku C++ lub Python?
5. Która funkcja umożliwia wyznaczenie długości łańcucha w języku C++ lub Python?

1. W jakim celu stosujemy szyfry?

Wyobraźmy sobie, że nie mamy dostępu do nowych technologii. Jak przekazać ważną wiadomość, aby jej treść nie trafiła w niepowołane ręce?

Możemy zapisać wiadomość w taki sposób, aby tylko nadawca i odbiorca mogli ją zrozumieć – wykorzystamy w tym celu szyfr.

Na przykład wiadomość „JUTROIDZIEMYDOKINA" możemy zaszyfrować jako „QFGILRWARVNBWLPRMZ", wykorzystując schemat szyfrowania pokazany w tabeli 1.

S **Szyfr**

Algorytm szyfrowania i deszyfrowania.

A	Y	C	W	E	U	G	S	I	Q	K	O	M
Z	B	X	D	V	F	T	H	R	J	P	L	N

Tabela.1. *Przykład prostego schematu szyfrowania*

Ćwiczenie 1. Szyfrujemy wiadomości

1. Wyjaśnij, na czym polega schemat szyfrowania pokazany w tabeli 1.
2. Zaszyfruj wiadomość do koleżanki lub kolegi. Wykorzystaj schemat szyfrowania z tabeli 1.

Ćwiczenie 2. Deszyfrujemy wiadomość

1. Rozszyfruj wiadomość WLALYZXAVMRZLGIAVXRVQ, wykorzystując tabelę 1.
2. Wyjaśnij sposób rozszyfrowywania wiadomości.

Szyfrowanie S

Proces przetwarzania wiadomości na **szyfrogram**, pozwalający na ukrycie jej tekstu.

Deszyfrowanie D

Proces odtwarzania tekstu oryginalnego na podstawie treści szyfrogramu.

Szyfry stosuje się w celu utajnienia wiadomości, aby nie mogła zostać odczytana przez osobę postronną. Pierwsze szyfry wykorzystywano do przekazywania rozkazów i innych informacji wojennych, a czasami także... listów miłosnych. Z czasem sposoby szyfrowania stawały się coraz bardziej skomplikowane, aby utrudnić złamanie szyfru.

W dzisiejszych czasach szyfry wykorzystuje się zdecydowanie częściej niż kiedyś. Spotykamy się z nimi codziennie – służą na przykład do ochrony wrażliwych danych (dane osobowe, hasła dostępu, dane finansowe). Zupełnie nieświadomie korzystamy z szyfrowania, rozmawiając przez telefon komórkowy, na stronach internetowych czy wysyłając e-mail.

2. Podstawowe pojęcia dotyczące szyfrowania

Szyfrowanie to technika umożliwiająca bezpieczne przesyłanie wiadomości. **Nadawca wiadomości** chciałby mieć pewność, że nikt niepowołany nie pozna jej treści, ani też jej nie zmodyfikuje.

Niezaszyfrowaną wiadomość nazywamy **tekstem jawnym**, natomiast zaszyfrowaną – **szyfrogramem**.



Rys. 1. *Schemat procesu szyfrowania i deszyfrowania*

Nauka zajmująca się zabezpieczaniem wiadomości to **kryptografia**, natomiast **kryptoanaliza** to nauka o łamaniu tych zabezpieczeń. Razem tworzą one **kryptologię** – osobną dziedzinę matematyki. Szyfrowanie i deszyfrowanie wykonuje się za pomocą odpowiednich funkcji matematycznych, często przy użyciu komputera.

3. Szyfrowanie tekstu metodą podstawieniową na przykładzie szyfru Cezara

Szyfr podstawieniowy to taki, w którym każdy znak tekstu jawnego jest zastępowany innym znakiem. Najbardziej znanym przykładem szyfru podstawieniowego jest **szyfr Cezara**.

W ćwiczeniu 1. przedstawiliśmy algorytm szyfrowania oparty na metodzie podstawieniowej.

Autorstwo szyfru Cezara przypisuje się Juliuszowi Cezarowi. Szyfr ten polega na zastąpieniu każdej litery tekstu jawnego znakiem stojącym w alfabecie o trzy pozycje w prawo względem znaku źródłowego. I tak A zostaje zastąpione przez D, B przez E, ... W przez Z, X przez A, Y przez B i Z przez C. Na rysunku 2. pokazano szyfr Cezara dla alfabetu łacińskiego, który zawiera 26 liter.

```
  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
  ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
→ → → ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

Rys. 2. *Szyfr Cezara*

Szyfr Cezara jest szyfrem **przesuwającym**. W tej metodzie szyfrowania można zastosować liczbę inną niż 3, która wskaże, o ile pozycji należy się przesunąć względem znaku wyjściowego. W ten sposób dla tego samego tekstu jawnego możemy uzyskać różne szyfrogramy, w zależności od tego, jaką liczbę zastosujemy (tabela 2.). Aby odszyfrować tekst, odbiorca powinien znać liczbę pełniącą funkcję **klucza** szyfru (rys. 3.).

Tekst jawny	Klucz	Szyfrogram
KOTEK	3	NRWHN
KOTEK	5	PTYJP
KOTEK	13	XBGRX

Tabela 2. *Przykład wiadomości zaszyfrowanych metodą podstawieniową (szyfr przesuwający) z różnymi kluczami*

K **Klucz**

Informacja służąca do szyfrowania lub deszyfrowania tekstu.



Rys. 3. *Schemat procesu szyfrowania i deszyfrowania z kluczem*

Ćwiczenie 3. Szyfrujemy wyraz szyfrem Cezara

1. Zaszyfruj napis MATEMATYKA szyfrem Cezara.
2. Wyjaśnij zastosowany sposób szyfrowywania.

Ćwiczenie 4. Odszyfrowujemy słowa zapisane metodą Cezara

1. Na podstawie wskazówek przedstawionych na rysunku 2. rozszyfruj słowa:
   a. VCBIU
   b. LQIRUPDWBND
   c. NUBSWRJUDILD
2. Wyjaśnij zastosowany sposób rozszyfrowywania.

Ćwiczenie 5. Szyfrujemy zdanie szyfrem Cezara

Korzystając z szyfru Cezara, zaszyfruj zdanie: „Szyfr Cezara to najbardziej znany szyfr podstawieniowy".

4. Szyfrowanie tekstu metodą przestawieniową

Zanim poznamy inny sposób szyfrowania, wykonajmy ćwiczenie 6.

Ćwiczenie 6. Odszyfrowujemy cytaty

1. Przeanalizuj sposób szyfrowania pokazany w tabeli 3. Wyjaśnij, na czym polega ta metoda.
2. Korzystając z przykładów z tabeli 3., odszyfruj cytaty, posługując się podanym kluczem:
   a. NAWOEIWATJEERERMJAJZOŚZSEŻĆYAKNDDMI (klucz 5/7),
   b. NOAZSUIWMETSEOJRTŁCKEAWYHOSZYSSCZZCZŁHCUHĘ (klucz 6/7).

Wskazówka: Autorzy cytatów: a. Heraklit z Efezu, b. Adam Mickiewicz (*Pan Tadeusz*).

Tekst jawny	Klucz	Zapis wierszami	Szyfrogram
ALA MA KOTA	3/3	A L A M A K O T A	AMOLATAKA
SZKOŁA PONADPODSTAWOWA	3/7	S Z K O Ł A P O N A D P O D S T A W O W A	SOSZNTKAAODWŁPOAOWPDA

Tabela 3. *Przykłady wiadomości zaszyfrowanych metodą przestawieniową – ćwiczenie 6.*

W ćwiczeniu 6. posłużyliśmy się szyfrem przestawieniowym, który najprościej zrealizować, zapisując tekst jawny wierszami o ustalonej długości, a czytając go kolumnami (przy ustalonej długości wierszy powstają kolumny o określonej liczbie znaków). Kluczem może być liczba wierszy lub kolumn (w przykładach powyżej podane zostały obie te liczby) – wszystko zależy od ustaleń.

Szyfr przestawieniowy to taki, w którym w tekście zaszyfrowanym pojawiają się wszystkie znaki tekstu jawnego, ale w innej kolejności.

Ćwiczenie 7. Stosowanie szyfru przestawieniowego

Zaszyfruj cytat „To be or not to be – that is the question", stosując szyfrowanie przestawieniowe. Ustaw znaki tekstu jawnego w dziesięciu kolumnach.

Wskazówka: W szyfrowaniu przestawieniowym pomijamy spacje.
Tekst jawny: TOBEORNOTTOBETHATISTHEQUESTION

Uwaga

„To be or not to be – that is the question"
– W. Szekspir, *Hamlet*, akt III, scena 1.

5. Programowanie algorytmów szyfrowania w językach C++ i Python

5.1. Miniszyfrowanie – tworzenie anagramów

Anagram tworzymy poprzez przestawienie liter lub sylab wyrazu bądź zdania, wykorzystując wszystkie litery (głoski lub sylaby) wyrazu bądź zdania źródłowego.

Najprostszy anagram można utworzyć, układając litery w odwrotnej kolejności, np. *baobab – baboab*. Można też przestawić tylko sylaby, co np. w wyrazie *ma·ra*, daje anagram: *ra·ma*.

Możemy w ten sposób utworzyć szyfr, którego działanie będzie polegało na przykład na wyprowadzeniu wyrazu (ciągu znaków) wspak, np. „poezja" – jako „ajzeop".

Ten rodzaj szyfru to szyfrowanie przestawieniowe – polega na zmianie kolejności znaków tekstu.

Ćwiczenie 8. Tworzymy anagramy i szukamy sposobu utworzenia anagramu

1. W kilku grupach twórzcie różne anagramy, np. jedna osoba wymyśla anagram słowa, a pozostałe osoby z grupy próbują odgadnąć, jakie to słowo oraz jaki zastosowano klucz (sposób, w jaki utworzono anagram).
2. Przedstawcie wybrane rozwiązania na forum klasy.

{ Chcemy napisać w wybranym języku program, który tworzy najprostszy anagram, czyli ustawia litery w odwrotnej kolejności. Jak to zrobić? }

Aby napisać program tworzący taki anagram, wystarczy wiedzieć, jak odwołać się do pojedynczego znaku napisu (co omówiliśmy w temacie C1), a także jak użyć instrukcji iteracyjnej do wyświetlania napisu od końca do początku.

Ćwiczenie 9. Piszemy program tworzący anagramy w wybranym języku programowania

1. Przypomnij, w jaki sposób można odwołać się do pojedynczego znaku tekstu w języku C++ lub w języku Python.
2. W wybranym języku programowania napisz program tworzący anagram słowa podanego z klawiatury. Anagram ma być przestawieniem liter w odwrotnej kolejności. Program ma umożliwiać tworzenie anagramów *n* napisów wprowadzanych z klawiatury.
3. Zapisz program w pliku pod nazwą *Miniszyfrowanie*.
4. Przetestuj program dla różnych danych.

Wskazówka: Instrukcja pętli może mieć postać:

C++

```
for(int i = napis.length() - 1; i >= 0; i--)
    cout << napis[i];
```

Python

W języku Python można użyć funkcji wbudowanej `reversed` do iteracji w odwrotnej kolejności po napisie lub liście.

```
for znak in reversed(napis):
    print(znak)
```

5.2. Programowanie algorytmu szyfrowania podstawieniowego

{ Chcemy napisać program realizujący algorytm szyfrowania metodą Cezara.
W jaki sposób napisać w wybranym języku programowania funkcję szyfrującą? }

Aby zaprogramować algorytm, w którym mamy zastąpić daną literę inną literą umiejscowioną w alfabecie o trzy pozycje w prawo względem znaku źródłowego, musimy wiedzieć, jak wyznaczyć tę pozycję litery w alfabecie. Ponadto działania musimy powtórzyć tyle razy, ile wynosi długość tekstu do zaszyfrowania – mamy więc do czynienia z iteracją.

Jak wiemy, każda litera ma swój odpowiednik liczbowy w kodzie ASCII, np. litera A ma kod 65, a Z – 90. W tabeli kodów ASCII litery alfabetu łacińskiego ustawiono po kolei, co możemy wykorzystać do szyfru. Chcemy jednak literom alfabetu przyporządkować kolejne liczby naturalne (od 0 do 25), czyli zamienić literę na liczbę określającą jej pozycję w alfabecie. Jak to zrobić? Można od kodu ASCII danej litery odjąć kod ASCII litery A równy 65. Na przykład:

dla B $66 - 65 = 1$,
dla L $76 - 65 = 11$,
dla Z $90 - 65 = 25$.

Otrzymaliśmy w ten sposób dla każdej litery jej pozycję w alfabecie (przyjęto numerowanie od zera, aby ułatwić wykonywanie operacji dodawania modulo 26).

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

T	U	V	W	X	Y	Z
19	20	21	22	23	24	25

Tabela 4. *Przyporządkowanie literom kolejnych liczb naturalnych*

Aby zamieniać literę na jej pozycję w alfabecie (czyli znak na kod) i odwrotnie (kod na znak), zdefiniujemy dwie funkcje: *znak_na_kod()* i *kod_na_znak()* – rysunki 4a (C++) i 4b (Python).

W języku C++ znak jest tożsamy z jego kodem ASCII, dlatego nie musimy w programie zamieniać liczby na jej kod ASCII – zrobi to automatycznie kompilator. Użyjemy tylko wbudowanej funkcji `toupper()`, która zwraca znak przekazany jako parametr funkcji zamieniony z małej litery na wielką (wielkie litery nie są zmieniane).

C++

```
unsigned int znak_na_kod(char znak)
{
    return toupper(znak) - 'A';
}

unsigned char kod_na_znak(int kod_znaku)
{
    return kod_znaku + 'A';
}
```

Rys. 4a. *Przykładowe funkcje zmieniające literę na jej pozycję w alfabecie i odwrotnie (C++)*

W języku Python, inaczej niż w języku C++, znak nie jest tożsamy z liczbą, dlatego do zamiany litery na jej kod Unicode (i odwrotnie) musimy użyć wbudowanych funkcji:

- `ord()` – zamienia znak na kod Unicode,
- `chr()` – zamienia kod Unicode na odpowiadający mu znak.

Użyjemy również metody `upper()`, która zwraca znak zamieniony z małej litery na wielką.

Python

```
def znak_na_kod(znak):
    return ord(znak.upper()) - ord('A')

def kod_na_znak(kod_znaku):
    return chr(kod_znaku + ord('A'))
```

Rys. 4b. *Przykładowe funkcje zmieniające literę na jej pozycję w alfabecie i odwrotnie (Python)*

! Uwaga

W przypadku liter alfabetu łacińskiego, cyfr i znaków interpunkcyjnych dostępnych na klawiaturze, kod Unicode tych znaków jest taki sam, jak ich kod ASCII. Zbiór kodów ASCII jest podzbiorem zbioru kodów Unicode.

Ćwiczenie 10. Testujemy działanie funkcji

1. Przetestuj działanie funkcji pokazanych na rysunkach 4a lub 4b. Napisz program, definiując te funkcje i wywołując je w programie głównym z parametrem aktualnym *litera*. Znak ma być wprowadzany z klawiatury.
2. Zapisz program w pliku pod nazwą *Zamiany*.
3. Sprawdź działanie programu dla liter E oraz e. Co zauważasz? Uzasadnij odpowiedź.

Wskazówka: (Python) Pamiętaj, że w języku Python możesz w trybie interaktywnym (w oknie powłoki Pythona) sprawdzać działanie różnych instrukcji.

Zastanówmy się teraz, jak otrzymać pozycję zakodowanego znaku. Czy wystarczy do liczby określającej pozycję litery dodać wartość klucza, czyli 3?

Dla litery L: 11 + 3 = 14 po zakodowaniu otrzymamy literę O,
ale dla litery Z: 25 + 3 = 28 po zakodowaniu powinniśmy otrzymać literę C, która w alfabecie ma pozycję 2.

Obliczenie samej sumy nie wystarczy. Aby otrzymywać zawsze poprawną pozycję zaszyfrowanego znaku, musimy obliczać resztę z dzielenia sumy pozycji znaku tekstu jawnego i klucza przez 26.

Zatem dla litery Z poprawne działanie to: (25 + 3) % 26 = 28 % 26 = 2 (gdzie operator % oznacza resztę z dzielenia).

Dlatego w funkcji *zaszyfruj()* realizującej algorytm szyfrowania metodą Cezara (rys. 5a i 5b) należy umieścić odpowiednio przypisanie:

C++

```
unsigned int kod_znaku_zaszyfrowany = (kod_znaku + 3) % 26;
```

Python

```
kod_znaku_zaszyfrowany = (kod_znaku + 3) % 26
```

Funkcję *zaszyfruj()* zdefiniujemy z jednym parametrem *napis*, określającym tekst jawny. Zastosujemy pętlę **for**, w której przeglądamy kolejne znaki tekstu, począwszy od 0. Zanim napiszemy cały program, spróbujmy przeanalizować działanie tej funkcji (rys. 5a i 5b).

C++

```
17  string zaszyfruj(string napis)
18  {
19      string wynik = "";
20      for(int i = 0; i < napis.length(); i++)
21      {
22          unsigned int kod_znaku = znak_na_kod(napis[i]);
23          unsigned int kod_znaku_zaszyfrowany =  (kod_znaku + 3) % 26;
24          wynik += kod_na_znak(kod_znaku_zaszyfrowany);
25      }
26      return wynik;
27  }
```

Rys. 5a. *Przykładowa funkcja realizująca algorytm szyfrowania szyfrem Cezara (C++)*

Python

```
def zaszyfruj(napis):
    wynik = ""
    for znak in napis:
        kod_znaku = znak_na_kod(znak)
        kod_znaku_zaszyfrowany = (kod_znaku + 3) % 26
        wynik += kod_na_znak(kod_znaku_zaszyfrowany)
    return wynik
```

Rys. 5b. *Przykładowa funkcja realizująca algorytm szyfrowania szyfrem Cezara (Python)*

Ćwiczenie 11. Piszemy program realizujący algorytm szyfrowania szyfrem Cezara

1. Napisz w wybranym języku programowania program umożliwiający zaszyfrowanie szyfrem Cezara tekstu wprowadzonego przez użytkownika. Wykorzystaj w programie funkcje zapisane w pliku *Zamiany* w ćwiczeniu 10. oraz funkcje szyfrujące pokazane na rysunkach 5a (C++) lub 5b (Python). Wyjaśnij znaczenie poszczególnych wierszy wybranej funkcji szyfrującej.
2. Wywołaj funkcję w programie głównym z parametrem aktualnym.
3. Zapisz program w pliku pod nazwą *Szyfr_Cezara*.
4. Przetestuj program dla różnych danych.

Ćwiczenie 12. Formułujemy algorytm deszyfrowania

1. Napis „SLHV” został zaszyfrowany metodą Cezara. Na przykładzie odszyfrowywania tego słowa uzasadnij, dlaczego kod znaku odszyfrowanego tekstu obliczamy tak, jak pokazano w poniższych instrukcjach przypisania.

   C++

   ```
   unsigned int kod_znaku_odszyfrowany = (kod_znaku - 3 + 26) % 26;
   ```

   Python

   ```
   kod_znaku_odszyfrowany = (kod_znaku - 3 + 26) % 26
   ```

2. Sformułuj algorytm odszyfrowywania tekstu zaszyfrowanego metodą Cezara.

Ćwiczenie 13. Piszemy funkcję deszyfrowania podstawieniowego

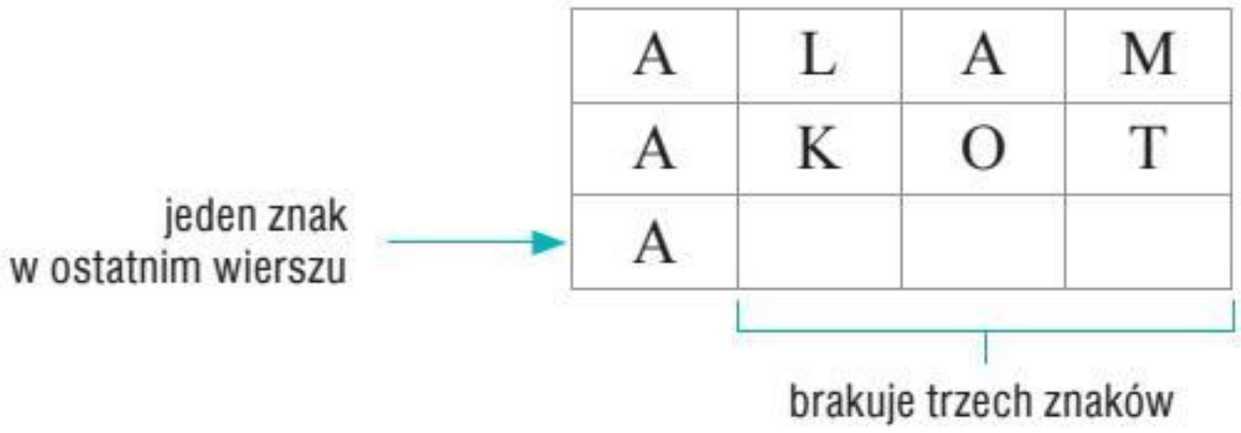
1. Uzupełnij program zapisany w ćwiczeniu 11. o funkcję deszyfrującą tekst wprowadzony przez użytkownika.
2. Wywołaj funkcję w programie głównym.
3. Zapisz plik pod tą samą nazwą i przetestuj działanie programu dla różnych danych.

5.3. Programowanie algorytmu szyfrowania przestawieniowego

{ Chcemy napisać program realizujący algorytm szyfrowania przestawieniowego.
W jaki sposób możemy napisać funkcję szyfrującą w wybranym języku programowania? }

Aby zaszyfrować dany tekst metodą przestawieniową z użyciem kolumn, musimy zadbać o to, by w każdej kolumnie było tyle samo znaków. W przypadku tekstu, którego długość nie jest podzielna przez liczbę kolumn, musimy taki tekst dopełnić wybranym znakiem.

Na przykład, aby zaszyfrować tekst jawny ALAMAKOTA z użyciem czterech kolumn, trzeba będzie dopełnić tekst trzema znakami do dwunastu znaków (rys. 6.).



Rys. 6. *Przykład tekstu jawnego ALAMAKOTA do zaszyfrowania z użyciem czterech kolumn*

Zdefiniujemy funkcję pomocniczą *dopelnij()* z trzema parametrami: *napis*, *znak* i *liczba_znakow*, która zwraca tekst *napis* dopełniony znakami do długości równej *liczba_znakow*.

C++

```
 7  string dopelnij(string napis, char znak, int liczba_znakow)
 8  {
 9      string wynik = napis;
10      for(int i = napis.length(); i < liczba_znakow; i++)
11          wynik += znak;
12      return wynik;
13  }
```

Rys. 7a. *Przykładowa pierwsza funkcja pomocnicza do szyfrowania przestawieniowego (C++)*

Python

```
def dopelnij(napis, znak, liczba_znakow):
    wynik = napis
    for i in range (len(napis), liczba_znakow):
        wynik += znak
    return wynik
```

Rys. 7b. *Przykładowa pierwsza funkcja pomocnicza do szyfrowania przestawieniowego (Python)*

Aby skorzystać z funkcji *dopelnij()*, musimy wiedzieć, jaka powinna być długość tekstu z dopełnieniem. W tym celu obliczamy resztę z dzielenia długości tekstu przez liczbę kolumn, która to wartość odpowiadać będzie liczbie znaków w ostatnim wierszu. Jeżeli będzie ona niezerowa – długość zwiększamy o brakującą liczbę znaków.

W naszym przykładzie tekst jawny ALAMAKOTA ma długość 9, czyli:

9 % 4 = 1, czyli w ostatnim wierszu brakuje: 4 – 1 = 3 znaków.

Do tekstu jawnego dodajemy brakujące znaki, czyli długość tekstu z dopełnieniem wynosi: 9 + 3 = 12 znaków.

C++

```
15  int liczba_znakow_z_dopelnieniem(string napis, int liczba_kolumn)
16  {
17      int dlugosc = napis.length();
18      int liczba_znakow_ostatni_wiersz = dlugosc % liczba_kolumn;
19      if(liczba_znakow_ostatni_wiersz != 0)
20          dlugosc += (liczba_kolumn - liczba_znakow_ostatni_wiersz);
21      return dlugosc;
22  }
```

Rys. 8a. *Przykładowa druga funkcja pomocnicza do szyfrowania przestawieniowego (C++)*

Python

```python
def liczba_znakow_z_dopelnieniem(napis, liczba_kolumn):
    dlugosc = len(napis)
    liczba_znakow_ostatni_wiersz = dlugosc % liczba_kolumn
    if liczba_znakow_ostatni_wiersz != 0:
        dlugosc += (liczba_kolumn - liczba_znakow_ostatni_wiersz)
    return dlugosc
```

Rys. 8b. *Przykładowa druga funkcja pomocnicza do szyfrowania przestawieniowego (Python)*

Zapisując algorytm szyfrowania metodą przestawieniową, możemy zauważyć, że czytanie tekstu kolumnami polega w istocie na odczytywaniu co *n*-tego znaku, gdzie *n* odpowiada kluczowi szyfru określonemu jako liczba kolumn. Dla powyższego przykładu, przyjmując, że znaki numerowane są od zera, powinniśmy zatem odczytać znaki o numerach: 0, 4, 8 (pierwsza kolumna), 1, 5, 9 (druga kolumna) itd. W naszym przykładzie użyjemy znaku podkreślenia „_" do dopełnienia tekstu (rys. 9.). Tego samego znaku użyjemy również w funkcji szyfrującej (rys. 10a i 10b).

Znak	A	L	A	M	A	K	O	T	A	_	_	_
Numer znaku	0	1	2	3	4	5	6	7	8	9	10	11

Rys. 9. *Znaki tekstu z dopełnieniem do dwunastu znaków*

Ćwiczenie 14. Szyfrujemy tekst

Zanim napiszesz program, zaszyfruj tekst ALAMAKOTA metodą przestawieniową z użyciem czterech kolumn.

Do szyfrowania tekstu utworzymy funkcję *zaszyfruj()*. Aby wybrać z tekstu co *n*-ty znak, wykorzystujemy zmienną *akt_pozycja*, w której przechowywana jest aktualna pozycja znaku tekstu niezaszyfrowanego. Pozycję tę zwiększamy o parametr *liczba_kolumn*. Jeżeli wartość zmiennej *akt_pozycja* będzie większa lub równa długości tekstu, zmniejszamy ją o długość tekstu i zwiększamy o jeden. Proces ten powtarzamy tak długo, aż wykorzystamy wszystkie znaki tekstu niezaszyfrowanego.

C++

```
24  string zaszyfruj(string napis, int liczba_kolumn)
25  {
26      string wynik = "";
27      napis = dopelnij(napis, '_', liczba_znakow_z_dopelnieniem(napis, liczba_kolumn));
28      int dlugosc = napis.length();
29      int akt_pozycja = 0;
30      for(int i = 0; i < dlugosc; i++)
31      {
32          wynik += napis[akt_pozycja];
33
34
35
36      }
37      return wynik;
38  }
```

Rys. 10a*. Przykładowa funkcja realizująca algorytm szyfrowania przestawieniowego (C++)*

Python

```
def zaszyfruj(napis, liczba_kolumn):
    wynik = ""
    napis = dopelnij(napis, '_', liczba_znakow_z_dopelnieniem(napis, liczba_kolumn))
    dlugosc = len(napis)
    akt_pozycja = 0
    for i in range(dlugosc):
        wynik += napis[akt_pozycja]
        [redacted]
    return wynik
```

***Rys. 10b**. Przykładowa funkcja realizująca algorytm szyfrowania przestawieniowego (Python)*

Ćwiczenie 15. Piszemy program realizujący algorytm szyfrowania przestawieniowego

1. W wybranym języku programowania napisz program szyfrujący szyfrem przestawieniowym kolumnowym tekst wprowadzony przez użytkownika. Liczba kolumn powinna być podawana przez użytkownika.
2. Przykładowe funkcje pomocnicze i szyfrujące pokazano na rysunkach 7a, 8a, 10a (C++) i 7b, 8b, 10b (Python). Możesz się wzorować na tych rozwiązaniach, przy czym w funkcjach szyfrujących pokazanych na rysunkach 10a i 10b brakuje po trzy wiersze programu. Uzupełnij je, korzystając z opisu umieszczonego przed rysunkami.
3. Wywołaj funkcję szyfrującą w programie głównym.
4. Zapisz program w pliku pod nazwą *Szyfrowanie_przestawieniowe*.
5. Przetestuj program dla różnych danych, w tym dla tekstu „ALAMAKOTA".

Deszyfrowanie tekstu wykorzystuje ten sam algorytm, co szyfrowanie. Z tekstu zaszyfrowanego odczytujemy jednak nie co n-ty znak, lecz co m-ty znak, gdzie m odpowiada liczbie wierszy ($m \cdot n = liczba\ znaków\ z\ dopełnieniem$).

Ćwiczenie 16. Piszemy funkcję deszyfrowania przestawieniowego

1. Uzupełnij program zapisany w ćwiczeniu 15. o funkcję deszyfrującą tekst wprowadzony przez użytkownika. Liczba kolumn powinna być podawana przez użytkownika. Wywołaj funkcję w programie głównym.
2. Zapisz plik pod tą samą nazwą.
3. Przetestuj program dla różnych danych.

6. Więcej o szyfrowaniu

6.1. Szyfrowanie symetryczne i asymetryczne

Podczas szyfrowania danych z kluczem możemy spotkać się z dwoma rodzajami algorytmów szyfrujących: **symetrycznym** i **asymetrycznym**.

Szyfrowanie symetryczne wykorzystuje jeden klucz – ten sam zarówno do szyfrowania, jak i deszyfrowania wiadomości, np. dla szyfru przestawieniowego kluczem jest liczba kolumn.

W **algorytmach asymetrycznych** wiadomość dla danego adresata zaszyfrować może każdy, używając **klucza publicznego** tego adresata. Natomiast odczytanie wiadomości możliwe jest tylko przy użyciu innego klucza – **klucza prywatnego**, który zna jedynie odbiorca. Z algorytmów asymetrycznych korzysta się np. przy uwierzytelnianiu transakcji bankowych czy w certyfikatach bezpieczeństwa stron HTTPS. Popularne protokoły szyfrowania, które zapobiegają przechwytywaniu i zmienianiu przesyłanych danych, wykorzystywane m. in. na stronach internetowych, to SSL lub TLS.



Pamiętaj, aby sprawdzać czy strona, na której podajesz dane wrażliwe, tj. hasło lub dane karty kredytowej, np. strona płatności w sklepie internetowym, wykorzystuje aktualny certyfikat szyfrowania HTTPS (znak kłódki lub *https://* w adresie strony). Jeśli strona go nie ma, twoje dane mogą zostać przechwycone podczas ich przesyłania.

6.2. Maszyna Enigma

Metodę podstawieniową wykorzystano w słynnej maszynie szyfrującej Enigma, która w sposób mechaniczny dokonuje kolejnych podstawień alfabetycznych. Podstawowy model maszyny Enigma wynalazł Arthur Scherbius w Berlinie w 1918 roku. Szyfrowała ona wiadomość, wykonując kolejno pewną liczbę podmian znaków i wykorzystując przy tym połączenia elektryczne. Jej najważniejszymi elementami były obracające się na jednej osi bębny szyfrujące.



Rys. 11. *Maszyna Enigma*

Jeszcze przed rozpoczęciem II wojny światowej Enigmą zainteresowało się niemieckie wojsko, które po zmodyfikowaniu jej konstrukcji i zasad działania wprowadziło maszynę do własnego użytku.

W kolejnych latach w różnych krajach trwały prace nad złamaniem niemieckich szyfrów. Panowało przekonanie, że najlepsi w tej dziedzinie będą językoznawcy i lingwiści. Polacy jako pierwsi zatrudnili w tym celu profesjonalnych matematyków, byli to Jerzy Różycki, Henryk Zygalski oraz Marian Rejewski, którym w 1932 roku udało się rozwiązać szyfr Enigmy. Polacy mogli odczytywać korespondencję niemiecką, jednak nie było to proste, bowiem Niemcy stale udoskonalali maszynę i procedury szyfrowania. Podczas II wojny światowej łamaniem szyfrów Enigmy zajmował się w Wielkiej Brytanii zespół matematyków kierowany przez Alana Turinga, jednego z pionierów informatyki.

6.3. Algorytm RSA

Jednym z aktualnie najpopularniejszych asymetrycznych algorytmów szyfrowania jest algorytm RSA. Jego nazwa pochodzi od pierwszych liter nazwisk jego twórców: Rona Rivesta, Adiego Shamira oraz Leonarda Adlemana, którzy zaprojektowali go w 1977 roku.

Jest to pierwszy algorytm, który może być stosowany zarówno do szyfrowania, jak i do podpisów elektronicznych.

W celu wygenerowania pary kluczy prywatnego i publicznego w algorytmie stosuje się działania i funkcje matematyczne na dużych liczbach pierwszych.

Obecnie RSA stosuje się na przykład w systemie międzynarodowych przekazów bankowych SWIFT oraz w przeglądarkach internetowych. Również wspomniane wcześniej certyfikaty bezpieczeństwa stron HTTPS bazują na tego typu algorytmie.

Warto zapamiętać

- Aby zabezpieczać informacje, można posługiwać się różnymi algorytmami szyfrującymi, m.in. algorytmem podstawieniowym i przestawieniowym.
- Szyfr, w którym każdy znak tekstu jawnego jest zastępowany innym znakiem, nazywamy szyfrem podstawieniowym. Najbardziej znany przykład szyfru podstawieniowego to szyfr Cezara.
- Szyfr, w którym w tekście zaszyfrowanym pojawiają się wszystkie znaki tekstu jawnego, ale w innej kolejności, nazywamy szyfrem przestawieniowym.
- Gdy w wyrazie lub zdaniu przestawimy litery lub sylaby, wykorzystując wszystkie litery (głoski lub sylaby) tego wyrazu lub zdania – powstaje anagram.
- Szyfrowanie symetryczne wykorzystuje jeden klucz – ten sam zarówno do szyfrowania, jak i deszyfrowania wiadomości. W algorytmach asymetrycznych wiadomość zaszyfrować może każdy, używając klucza publicznego. Wiadomość można odczytać przy użyciu klucza prywatnego, który zna jedynie odbiorca.

Pytania i polecenia

1. Czym jest szyfr?
2. Czym różni się tekst jawny od szyfrogramu?
3. Czym zajmuje się kryptografia, a czym kryptoanaliza?
4. W jaki sposób działają techniki szyfrowania wiadomości? Podaj kilka przykładów.
5. Wyjaśnij na wymyślonym przez siebie przykładzie, na czym polega szyfrowanie podstawieniowe.
6. Wyjaśnij na wymyślonym przez siebie przykładzie, na czym polega szyfrowanie przestawieniowe.
7. Czym jest anagram?
8. Czym się różni szyfrowanie symetryczne od asymetrycznego?
9. Do czego służyła maszyna Enigma? Która metoda szyfrowania była w niej wykorzystywana?

Zadania

1. Odszyfruj tekst: „VCBIUFHCDUDMHVWSURVWB".
2. Korzystając z metod szyfrowania omówionych w tym temacie, zaszyfruj na dwa sposoby swoje dane osobowe (imię, nazwisko, miejsce zamieszkania).
3. Zadanie do wykonania w grupie dwuosobowej. Każda z osób wymyśla własny algorytm szyfrowania, następnie przekazuje zaszyfrowany tekst drugiej osobie do odszyfrowania.
4. Do programu zapisanego w ćwiczeniu 13. dodaj możliwość wprowadzania z klawiatury dowolnej wartości klucza. Zapisz plik pod tą samą nazwą.
5. Do programu zapisanego w zadaniu 4. dodaj możliwość wyboru szyfrowania lub deszyfrowania bądź zakończenia programu.
6. Do programu zapisanego w ćwiczeniu 16. dodaj możliwość wyboru szyfrowania lub deszyfrowania bądź zakończenia programu.

Dla zainteresowanych

7. Napisz program wyświetlający wszystkie anagramy utworzone na podstawie wyrazu „pora".
8. Steganografia to nauka o ukrywaniu informacji wewnątrz innej informacji. Zapoznaj się z tym zagadnieniem, korzystając z dodatkowej literatury i źródeł internetowych.
9. Korzystając z Internetu i dodatkowej literatury, znajdź informacje na temat Enigmy. **Wskazówka**: Zadanie można wykonać w grupie kilku osób, które następnie mogą w dyskusji podzielić się wiedzą.
10. Korzystając z Internetu i dodatkowej literatury, dowiedz się więcej na temat algorytmu z kluczem jawnym RSA.

Temat C3. Metoda połowienia, podejście zachłanne i rekurencja

1. Metoda połowienia
   - 1.1. Stosowanie metody połowienia do wyszukiwania elementu w zbiorze uporządkowanym
   - 1.2. Zapisywanie algorytmu wyszukiwania elementu w zbiorze uporządkowanym metodą połowienia w językach C++ i Python
2. Metoda zachłanna
   - 2.1 Jak znaleźć najlepszy sposób osiągnięcia celu?
   - 2.2. Stosowanie metody zachłannej do wydawania reszty
   - 2.3. Zapisywanie algorytmu wydawania reszty metodą zachłanną w językach C++ i Python
3. Na czym polega rekurencja?
4. Algorytm obliczania silni
   - 4.1. Iteracyjne obliczanie silni
   - 4.2. Rekurencyjne obliczenie silni liczby naturalnej
5. Rekurencyjne obliczanie wartości elementów ciągu Fibonacciego

Warto powtórzyć

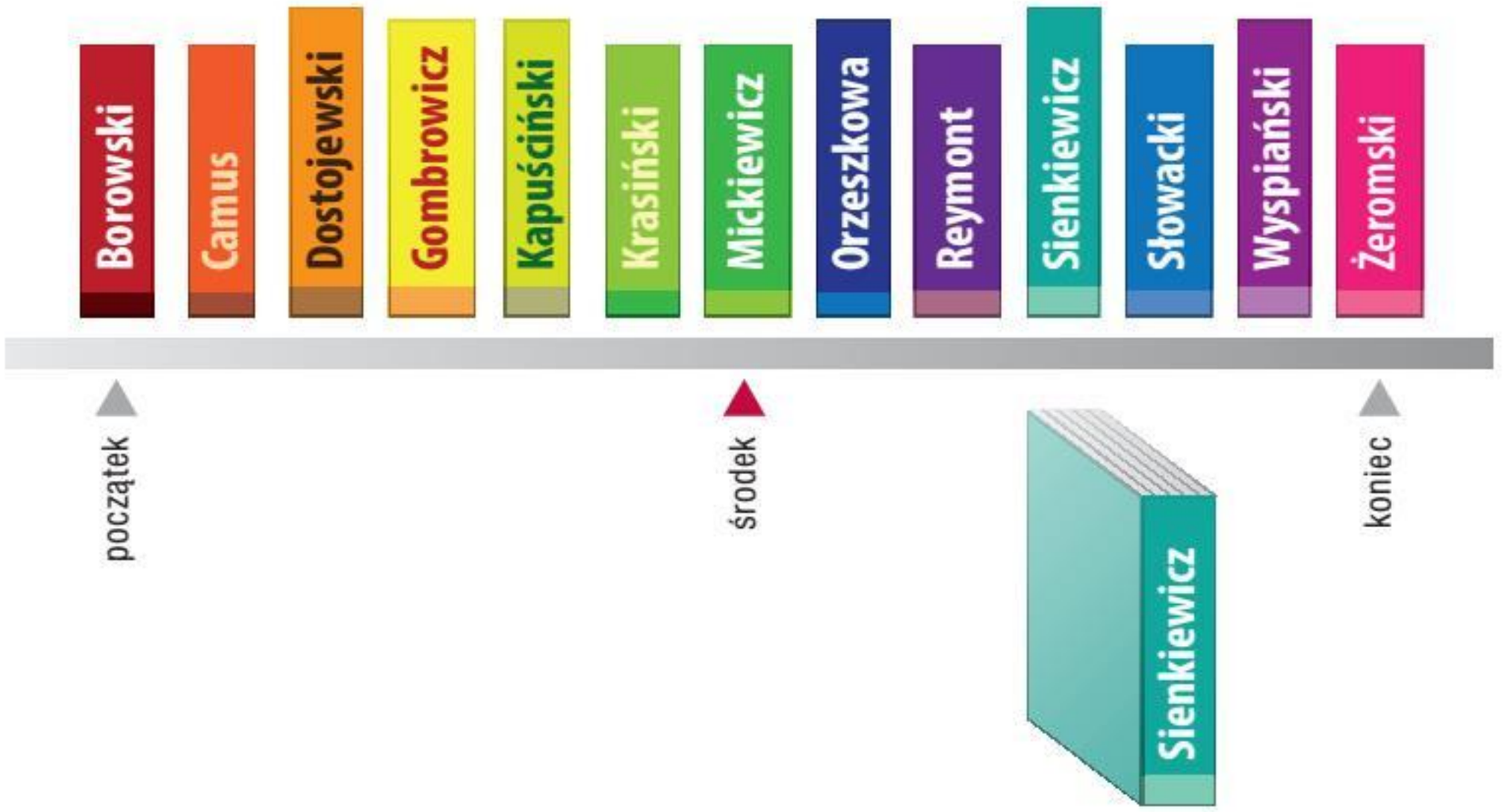
1. Czym jest iteracja?
2. W jaki sposób zapisujemy iterację w języku C++ lub w języku Python?
3. Na czym polega algorytm wyznaczania liczb Fibonacciego?

1. Metoda połowienia

1.1. Stosowanie metody połowienia do wyszukiwania elementu w zbiorze uporządkowanym

Z wyszukiwaniem przez połowienie spotykamy się w życiu codziennym, np. gdy szukamy jakiejś rzeczy w zbiorze, w którym panuje ustalony porządek. Wyobraźmy sobie półkę z książkami, które ułożono alfabetycznie według nazwisk autorów, zaczynając od lewej strony (rys. 1.). Chcemy znaleźć książkę autora, którego nazwisko zaczyna się na literę S. W jaki sposób jej szukamy?

Czy przeglądamy półkę od samego początku? Raczej nie. Zwykle odruchowo sprawdzamy, czy szukanej książki nie ma pośrodku. Jeśli trafimy na inną książkę, np. autora, którego nazwisko zaczyna się na literę K, to nie przeszukujemy lewej połowy półki, tylko będziemy szukać po prawej stronie. Szukamy dotąd, aż znajdziemy potrzebną książkę. Bezwiednie postępujemy zgodnie z algorytmem wyszukiwania przez połowienie.



Rys. 1. *Przykładowa ilustracja do ćwiczenia 1.*

Ćwiczenie 1. Szukamy książki na półce

1. Wspólnie z koleżankami i kolegami ułóżcie na półce trzynaście książek różnych autorów w porządku alfabetycznym według nazwiska twórcy. Następnie wybrane osoby powinny wyszukać książkę konkretnego autora.
2. Wyjaśnij sposób szukania książki autora, którego nazwisko zaczyna się na literę S, np. Henryka Sienkiewicza. Ponumeruj miejsca położenia książek na półce i podaj pozycję znalezionej książki.

Uwaga: Przyjmij, że na półce znajduje się tylko jedna książka danego autora.



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

Analiza algorytmu wyszukiwania metodą połowienia

Załóżmy, że mamy zbiór dziesięciu liczb uporządkowany w kolejności od najmniejszej do największej (rys. 2.). Poszukiwana liczba to 44. 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 na rys. 2. strzałką). W przypadku, gdy liczba elementów zbioru jest parzysta, to spośród dwóch środkowych elementów (w naszym przypadku liczb 51 i 59) jako „środkowy" wybieramy ten, który jest pierwszy w kolejności (czyli 51). 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. 2. *Przykład wyszukiwania liczby w zbiorze uporządkowanym metodą połowienia*

Szukamy liczby 44.				
Lp.	**„Środkowy element"**	**Pytanie**	**Odpowiedź**	**Działanie**
1.	„Środkowym" elementem zbioru jest liczba 51.	Czy liczba 51 jest większa od 44?	TAK	Dalej przeszukujemy tylko lewą część zbioru, a odrzucamy prawą oraz element środkowy 51.
2.				
3.				
4.	–	–	–	Pozostał nam jeden element i jest to szukana liczba 44, która jest czwartą wartością w zbiorze.

Tabela 1. *Przykładowa analiza rozwiązania – wzór tabeli do ćwiczenia 2.*

Ćwiczenie 2. Sprawdzamy działanie algorytmu wyszukiwania liczby w zbiorze uporządkowanym metodą przez połowienie

1. Przygotuj odpowiednie pomoce dydaktyczne (rys. 2.) i przedstaw wspólnie z koleżankami i kolegami z klasy algorytm wyszukiwania liczby w zbiorze uporządkowanym metodą przez połowienie.
2. Przeprowadź analizę rozwiązania, tworząc i uzupełniając tabelę podobną do tabeli 1. Skorzystaj z rysunku 2. i podanego opisu algorytmu. Odpowiedz na pytania:
   a. Jakie powtarzające się operacje zauważasz w rozwiązaniu?
   b. Kiedy kończymy powtarzanie operacji połowienia zbioru?

1.2. Zapisywanie algorytmu wyszukiwania elementu w zbiorze uporządkowanym metodą połowienia w językach C++ i Python

Chcemy napisać program realizujący algorytm wyszukiwania elementu w zbiorze uporządkowanym metodą połowienia. W jaki sposób napisać taki program w wybranym języku programowania (C++ lub Python)?

Program realizujący algorytm wyszukiwania elementu w zbiorze uporządkowanym metodą połowienia napiszemy zgodnie z opisem podanym w punkcie 1.1. tego tematu. Program ma wyszukiwać w zbiorze uporządkowanym daną podaną przez użytkownika i wyświetlać na ekranie jej pozycję na liście (indeks).

Zaczynamy od ustalenia, w jaki sposób będziemy pamiętać elementy zbioru. Elementy zbioru wprowadzimy do tablicy (C++) lub listy (Python).

Zakładamy, że wprowadzamy dane od najmniejszej do największej (inaczej program nie będzie działał poprawnie). Bez takiego założenia należałoby dodać również funkcję sortowania danych (ustawiania w określonym porządku).

Zdefiniujemy funkcję *znajdz_dana()* z parametrem *wartosc* (rys. 3a i 3b).

- W funkcji wyznaczamy pozycję elementu środkowego:

  C++
  ```
  srodek = (poczatek + koniec) / 2;
  ```
  Python
  ```
  srodek = (poczatek + koniec) // 2
  ```
- Jeśli element środkowy (*a*[*srodek*]) jest szukaną daną (zmienna *wartosc*), to funkcja zwraca wartość zmiennej *srodek*, czyli pozycję danego elementu w przeciwnym wypadku:
  - jeśli szukana dana jest mniejsza od środkowej, odrzucamy prawą połowę zbioru (liczby większe od elementu środkowego lub mu równe) – ustalamy nowy koniec: `koniec = srodek - 1;`
  - w przeciwnym wypadku odrzucamy lewą połowę zbioru (liczby mniejsze od elementu środkowego lub mu równe) i ustalamy nowy początek: `poczatek = srodek + 1`.
- Połowienie powtarzamy, dopóki początek zbioru jest mniejszy od końca zbioru lub mu równy (`poczatek <= koniec`) – o ile wcześniej nie znajdziemy szukanego elementu (wtedy komputer wykona instrukcję **`return`** `srodek`).
- Jeśli w przeszukiwanym zbiorze nie ma szukanej liczby, funkcja *znajdz_dana()* zwraca wartość –1.

C++

```cpp
int znajdz_dana(int wartosc)
{
    int poczatek, koniec, srodek;
    poczatek = 0;
    koniec = N - 1;
    do
    {
        srodek = (poczatek + koniec) / 2;
        if(a[srodek] == wartosc)
            return srodek;
        else
            [redacted]
            else
                [redacted]
    }
    while(poczatek <= koniec);

    return -1;
}
```

Rys. 3a. *Definicja funkcji wyszukiwania liczby w zbiorze uporządkowanym (C++)*

Python

```python
def znajdz_dana(wartosc):
    poczatek = 0
    koniec = N - 1
    while poczatek <= koniec:
        srodek = (poczatek + koniec) // 2
        if a[srodek] == wartosc:
            return srodek
        else:
            [redacted]
            else:
                [redacted]
    return -1
```

Rys. 3b. *Definicja funkcji wyszukiwania liczby w zbiorze uporządkowanym (Python)*

Ćwiczenie 3. Piszemy program realizujący algorytm wyszukiwania liczby w zbiorze uporządkowanym

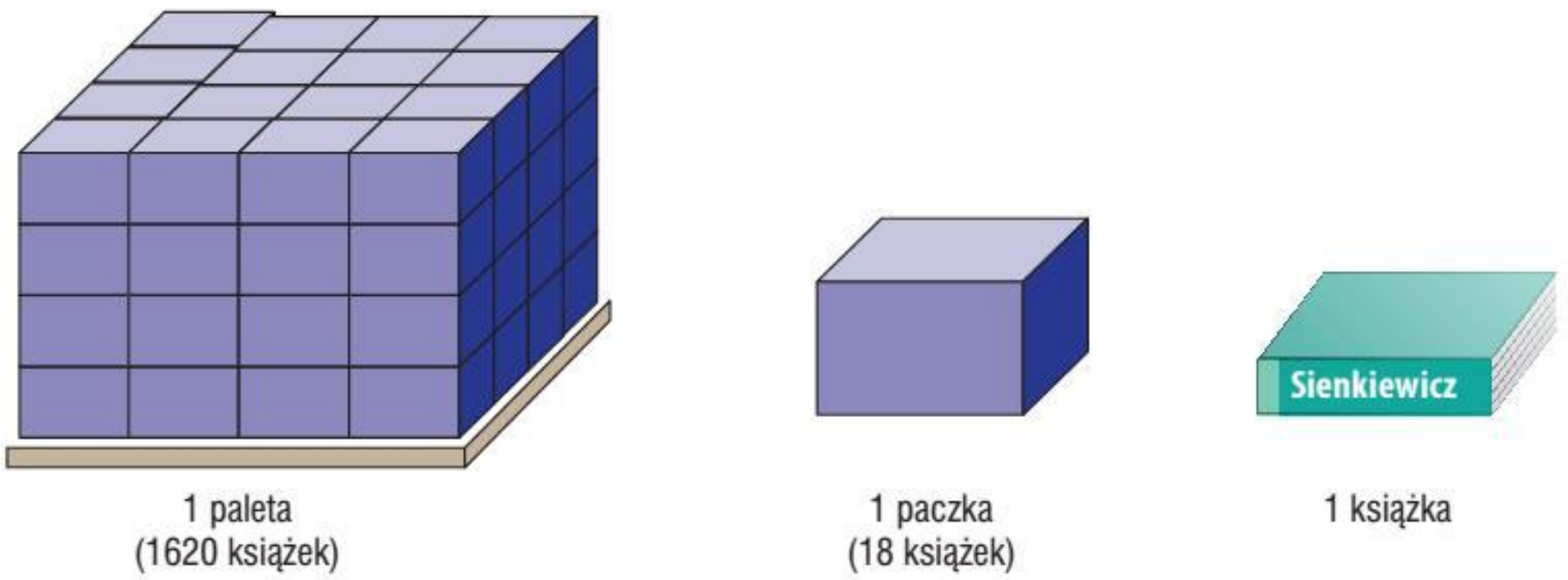
1. Napisz program realizujący algorytm wyszukiwania elementu w zbiorze uporządkowanym metodą połowienia. Zapisz specyfikację tego zadania. Dla uproszczenia przyjmij, że w zbiorze jest pięć elementów. Dlaczego elementy zbioru należy pamiętać w tablicy (C++) lub w liście (Python)? Uzasadnij odpowiedź.
2. Na rysunkach 3a i 3b pokazano przykładowe funkcje, ale są one niedokończone. Zastanów się, czego w nich brakuje? W wybranym języku (C++ lub Python) zdefiniuj funkcję *znajdz_dana()* i uzupełnij ją. Dodatkowo wyjaśnij:
   a. Dlaczego w funkcji *znajdz_dana()* zastosowano instrukcję iteracyjną **`do...while`** (C++) lub **`while`** (Python)?
   b. Dlaczego w instrukcji iteracyjnej umieszczono instrukcję warunkową, dodatkowo zagnieżdżoną?
3. Wywołaj funkcję w programie głównym.
4. Zapisz program w pliku pod nazwą *Wyszukiwanie_polowienie* i przetestuj działanie programu dla różnych danych.

2. Metoda zachłanna

2.1. Jak znaleźć najlepszy sposób osiągnięcia celu?

Jeśli można usprawnić i przyspieszyć jakieś działanie, to zwykle staramy się to zrobić. Próbujemy wymyślić sposób jak najszybszego osiągnięcia wyznaczonego celu. Zacznijmy od następującego problemu.

> Jesteś kierowcą samochodu dostawczego i musisz przewieźć 3500 książek z drukarni do magazynu. Książki zapakowano w paczki, które umieszczono na paletach. W jednej paczce mieści się 18 książek, natomiast na palecie – 1620 książek. W samochodzie możesz umieszczać tylko pełne palety lub paczki bądź pojedyncze egzemplarze. Jak sprawnie i szybko zapakować książki do samochodu?



Rys. 4. *Przykładowa ilustracja – ćwiczenie 4.*

Czy pakowanie lepiej zacząć od pojedynczych książek? Czy może od paczek z książkami? Przede wszystkim chcemy jak najmniej razy umieszczać książki w samochodzie, a więc w pierwszej kolejności powinniśmy korzystać z opakowań mieszczących jak najwięcej książek, czyli palet (oczywiście paletę umieszcza się w samochodzie, wykorzystując specjalny podnośnik). Jeśli nie uda się umieścić wszystkich książek w paletach, trzeba będzie dodać te w mniejszych opakowaniach, czyli paczki, a następnie pojedyncze książki, aż umieścimy 3500 książek.

Ćwiczenie 4. Szukamy rozwiązania problemu umieszczenia książek w samochodzie

1. Przyjmując założenia podane w sformułowanym problemie, wyjaśnij sposób umieszczenia 3500 książek w samochodzie. Ile pełnych palet, ile paczek, a ile pojedynczych książek znajdzie się w samochodzie? Pokaż sposób wykonywania obliczeń.
2. Jakie zauważasz powtarzające się operacje? Wyróżnij problem cząstkowy.

Ćwiczenie 5. Korzystamy z rozwiązania jednego problemu do rozwiązania innego problemu

Musisz przewieźć 50 palet ładunku (którego nie możesz rozdzielać). Dysponujesz pojazdami o ładowności 20 palet, 8 palet i 1 paleta. Ile i jakich samochodów użyjesz, aby jak najlepiej wykorzystać przestrzeń? Czy można zastosować sposób z ćwiczenia 4.? Uzasadnij odpowiedź.

Dla każdego problemu staraliśmy się znaleźć najkorzystniejsze rozwiązanie, postępując w sposób, który ma cechy zachłanności.

Algorytm zachłanny – algorytm, który w celu wyznaczenia rozwiązania w każdym kroku dokonuje zachłannego, tj. najkorzystniejszego w danym kroku, rozwiązania częściowego.

2.2. Stosowanie metody zachłannej do wydawania reszty

Z problemem wydawania reszty spotkamy się wielokrotnie. Mimo że dziś często używamy kart płatniczych, to umiejętność obliczenia reszty przydaje się, nawet jeśli kasa podpowiada sprzedawcy, ile reszty ma wydać. Czy możemy algorytm zachłanny wykorzystać podczas wydawania reszty?



Resztę wydajemy, używając jak najmniejszej liczby banknotów lub bilonu, czyli możemy w tym przypadku zastosować ten sam algorytm, co podczas ładowania książek do samochodu. Zauważmy, że ten sam algorytm możemy wykorzystać w różnych sytuacjach.

Metoda zachłanna zakłada użycie do wydania reszty największych dostępnych nominałów (ale mniejszych od lub równych wydawanej wartości). Zanim zaczniemy wydawać resztę, musimy określić dostępne nominały (zbiór nominałów).

Analiza algorytmu wydawania reszty metodą zachłanną

Algorytm zaczynamy od sprawdzenia, czy w zbiorze nominałów jest wartość N mniejsza od reszty R lub jej równa. Jeśli odnaleziony nominał N jest równy R, to algorytm się kończy. Jeśli nie, to wybieramy największą możliwą liczbę L sztuk znalezionego nominału N mniejszego od R (zakładamy, że zbiór nominałów jest posortowany malejąco). Liczbę tę otrzymujemy, dzieląc R przez wartość odnalezionego nominału (liczba nominałów to oczywiście część całkowita z dzielenia). Następnie od reszty R odejmujemy wartość znalezionego nominału pomnożoną przez liczbę nominałów wynikającą z dzielenia. Szukanie powtarzamy wśród kolejnych nominałów dla pomniejszonej wartości R do czasu, aż wypłacimy całą sumę. Mamy tu więc odniesienie do iteracji.

Korzystając z podanego opisu, znajdziemy nominały, którymi zostanie wydana reszta w wysokości 69 zł.

Reszta $R = 69$. Zbiór dostępnych nominałów to: {50, 20, 10, 5, 2, 1}.					
1.	Największy nominał, nie większy niż R to 50 zł.	Część całkowita z dzielenia 69 przez 50 to: 1.	Wypłacamy jeden banknot o nominale 50 zł.	$R = 69 - 50 \cdot 1 = 19$	Zostało 19 zł reszty do wydania.
2.	Największy nominał, nie większy niż aktualne R to 10 zł.	Część całkowita z dzielenia 19 przez 10 to: 1.	Wypłacamy jeden banknot o nominale 10 zł.	$R = 19 - 10 \cdot 1 = 9$	Zostało 9 zł reszty do wydania.
3.					
4					

Tabela 2. *Przykładowa analiza rozwiązania – wzór tabeli do ćwiczenia 6.*

Ćwiczenie 6. Sprawdzamy działanie algorytmu wydawania reszty metodą zachłanną

1. Przygotuj odpowiednie pomoce dydaktyczne i przedstaw wspólnie z koleżankami i kolegami z klasy algorytm wydawania reszty metodą zachłanną
2. Przeprowadź analizę rozwiązania, tworząc i uzupełniając tabelę podobną do tabeli 2. Skorzystaj z podanego opisu algorytmu. Odpowiedz na pytania:
   a. Jakie powtarzające się operacje zauważasz w rozwiązaniu?
   b. Jak ostatecznie wydano resztę?

2.3. Zapisywanie algorytmu wydawania reszty metodą zachłanną w językach C++ i Python

W zachłannym algorytmie wydawania reszty wykorzystujemy iterację. Szukanie nominału powtarzamy wśród kolejnych nominałów dla pomniejszonej wartości reszty, aż wypłacimy całą sumę – wykorzystamy instrukcję `while`.

Listę dostępnych nominałów umieścimy w tablicy (C++) lub w liście (Python). Elementy tablicy lub listy powinny być posortowane malejąco.

C++

```
const int nominaly[] = {100, 50, 20, 10, 5, 2, 1};
const int l_nominalow = 7;
```

Python

```
nominaly = [100, 50, 20, 10, 5, 2, 1]
```

Na rysunkach 5a i 5b pokazano fragmenty funkcji w językach C++ i Python. Funkcja ma za zadanie wypisać na ekranie liczbę banknotów poszczególnych nominałów, niezbędnych do wydania reszty.

Przyjęto następujące oznaczenia dla zmiennych:

- *aktualny_nominal* – aktualnie sprawdzany nominał banknotu,
- *reszt*a – reszta do wydania,
- *l_banknotow* – liczba wydawanych banknotów o wartości *aktualny_nominal*.

Do obliczenia liczby banknotów należy zastosować:

- w języku C++ operator dzielenia /, którego działanie zależy od typu danych (w przypadku liczb całkowitych wynik również będzie całkowity),
- w języku Python – operator dzielenia całkowitego //.

C++

```
while(reszta > 0 && aktualny_nominal < l_nominalow)
{
    while(aktualny_nominal < l_nominalow && reszta < nominaly[aktualny_nominal])
        aktualny_nominal++;

    if(aktualny_nominal < l_nominalow && reszta >= nominaly[aktualny_nominal])
    {
        int l_banknotow = reszta / nominaly[aktualny_nominal];
        cout << l_banknotow << " x " << nominaly[aktualny_nominal] << " zł" << endl;
        reszta -= l_banknotow * nominaly[aktualny_nominal];
    }
}
```

Rys. 5a. *Fragment funkcji wydawania reszty metodą zachłanną (C++)*

Python

```
while reszta > 0 and aktualny_nominal < len(nominaly):
    while aktualny_nominal < len(nominaly) and reszta < nominaly[aktualny_nominal]:
        aktualny_nominal = aktualny_nominal + 1
    if aktualny_nominal < len(nominaly) and reszta >= nominaly[aktualny_nominal]:
        l_banknotow = reszta // nominaly[aktualny_nominal]
        print(nominaly[aktualny_nominal], "x", l_banknotow)
        reszta = reszta - l_banknotow * nominaly[aktualny_nominal]
```

Rys. 5b. *Fragment funkcji wydawania reszty metodą zachłanną (Python)*

Ćwiczenie 7. Piszemy program realizujący algorytm zachłanny wydawania reszty

1. Na podstawie opisu algorytmu wydawania reszty metodą zachłanną napisz program realizujący ten algorytm. Zapisz specyfikację zadania. Zdefiniuj funkcję niezwracającą wartości *znajdz_wypisz_reszte()* z parametrem *reszta*. Na rysunkach 5a i 5b pokazano fragmenty tych funkcji w językach C++ i Python. Objaśnij znaczenie poszczególnych instrukcji.
2. Wywołaj funkcję w programie głównym.
3. Zapisz program w pliku pod nazwą *Wydawanie_reszty*, a następnie przetestuj program dla kilku różnych danych.

3. Na czym polega rekurencja?

Podczas rozwiązywania problemu możemy skorzystać z rozwiązania, które wypracowaliśmy wcześniej przy podobnym problemie. Możemy też podzielić problem na mniejsze podproblemy, które będzie nam łatwiej rozwiązać, a później przejść do rozwiązania całego problemu.

Możemy również w problemie i jego algorytmie wyodrębnić części podobne do całości i zmienić algorytm tak, aby korzystał z siebie samego.

Na przykład, w definicji liczby naturalnej następuje odwołanie do tej definicji.

a) 1 *jest liczbą naturalną*,

b) *następnik liczby naturalnej jest liczbą naturalną*.

Z **rekurencją** mamy do czynienia, gdy, określając jakieś pojęcie, odwołujemy się w definicji do niego samego.

Rekurencję stosujemy często w definicjach matematycznych, np. w definicji liczby naturalnej.

Ale zjawiska rekurencyjne występują również w naszym otoczeniu. Na rysunku 6. rekurencja jest przedstawiona w postaci „zagłębiających się w siebie", podobnych, coraz to mniejszych figur geometrycznych.

Podobny efekt możemy uzyskać dzięki odbiciom w lustrze. Wystarczy, patrząc w jedno lustro, nakierować na nie drugie, najlepiej mniejsze, owalne. Uzyskamy wówczas ciąg odbić mieszczących się jedno w drugim.



Rys. 6. *Wizualny efekt zjawiska rekurencji*

Ćwiczenie 8. Podajemy przykłady zjawisk rekurencyjnych

Podaj przykłady zjawisk rekurencyjnych, które można spotkać w życiu codziennym lub które znasz z przedmiotów szkolnych.

4. Algorytm obliczania silni

Aby zrozumieć, na czym polega rekurencja, porównamy realizację iteracyjną i rekurencyjną tego samego algorytmu. Porównanie przeprowadzimy na przykładzie algorytmu obliczania silni liczby naturalnej.

4.1. Iteracyjne obliczanie silni liczby naturalnej

Silnia liczby naturalnej n to iloczyn wszystkich liczb naturalnych dodatnich nie większych niż n.

Iteracyjna definicja silni liczby naturalnej n:

$$n! = \begin{cases} 1 & \text{dla } n = 0 \\ 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n & \text{dla } n > 0 \end{cases}$$

W definicji silni występuje powtarzanie poleceń, mamy więc do czynienia z iteracją, w której określono z góry liczbę kroków. Dlatego w iteracyjnej realizacji algorytmu obliczania silni zastosujemy instrukcję **`for`**.

C++

```
 7  int silnia_iter(int n)
 8  {
 9      int wynik = 1;
10
11      if(n > 1)
12          for(int i = 2; i <= n; i++)
13              wynik *= i;
14      return wynik;
15  }
```

Rys. 7a. *Iteracyjna definicja funkcji obliczającej silnię liczby naturalnej (C++)*

Python

```
def silnia_iter(n):
        wynik = 1;

        if n > 1:
                for i in range(2, n + 1):
                        wynik *= i
        return wynik
```

Rys. 7b. *Iteracyjna definicja funkcji obliczającej silnię liczby naturalnej (Python)*

Ćwiczenie 9. Zapisujemy w postaci programu iteracyjną realizację algorytmu obliczającego silnię liczby naturalnej n

1. Napisz specyfikację zadania algorytmu obliczania silni.
2. Napisz program iteracyjnej realizacji algorytmu obliczania silni. Wykorzystaj funkcję *silnia_iter()* z rysunku 7a lub 7b. Omów jej działanie – wyjaśnij znaczenie poszczególnych instrukcji.
3. Zapisz program w pliku pod nazwą *Silnia_iteracyjnie*.
4. Przetestuj program dla kilku różnych danych.

4.2. Rekurencyjne obliczenie silni liczby naturalnej

Zgodnie z zasadą rekurencji, w definicji silni liczby n pojawia się definicja tego samego pojęcia, czyli silni, tyle że liczby $n-1$.

Rekurencyjna definicja silni liczby naturalnej n:

$$n! = \begin{cases} 1 & \text{dla } n = 0 \\ (n-1)! \cdot n & \text{dla } n > 0 \end{cases}$$

Obliczenia wynikające z definicji rekurencyjnej silni dla $n = 3$:

$3! = 2! \cdot 3$

$2! = 1! \cdot 2$

$1! = 0! \cdot 1.$

Dana funkcja jest **rekurencyjna**, jeśli zawiera odwołanie do samej siebie. Aby zapisać rekurencyjną realizację wybranego algorytmu, musimy zapisać algorytm w postaci funkcji, która może wywołać samą siebie.

Definiujemy funkcję z jednym parametrem n. W definicji funkcji *silnia_rek* następuje jej wywołanie wewnątrz niej samej z parametrem $(n-1)$.

C++

```
int silnia_rek(int n)
{
    if( n == 0 )
        return 1;
    else
        return n*silnia_rek(n - 1);
}
```

Rys. 8a. *Rekurencyjna definicja funkcji obliczającej silnię liczby naturalnej (C++)*

Python

```
def silnia_rek(n):
    if( n == 0 ):
        return 1
    else:
        return n*silnia(n - 1)
```

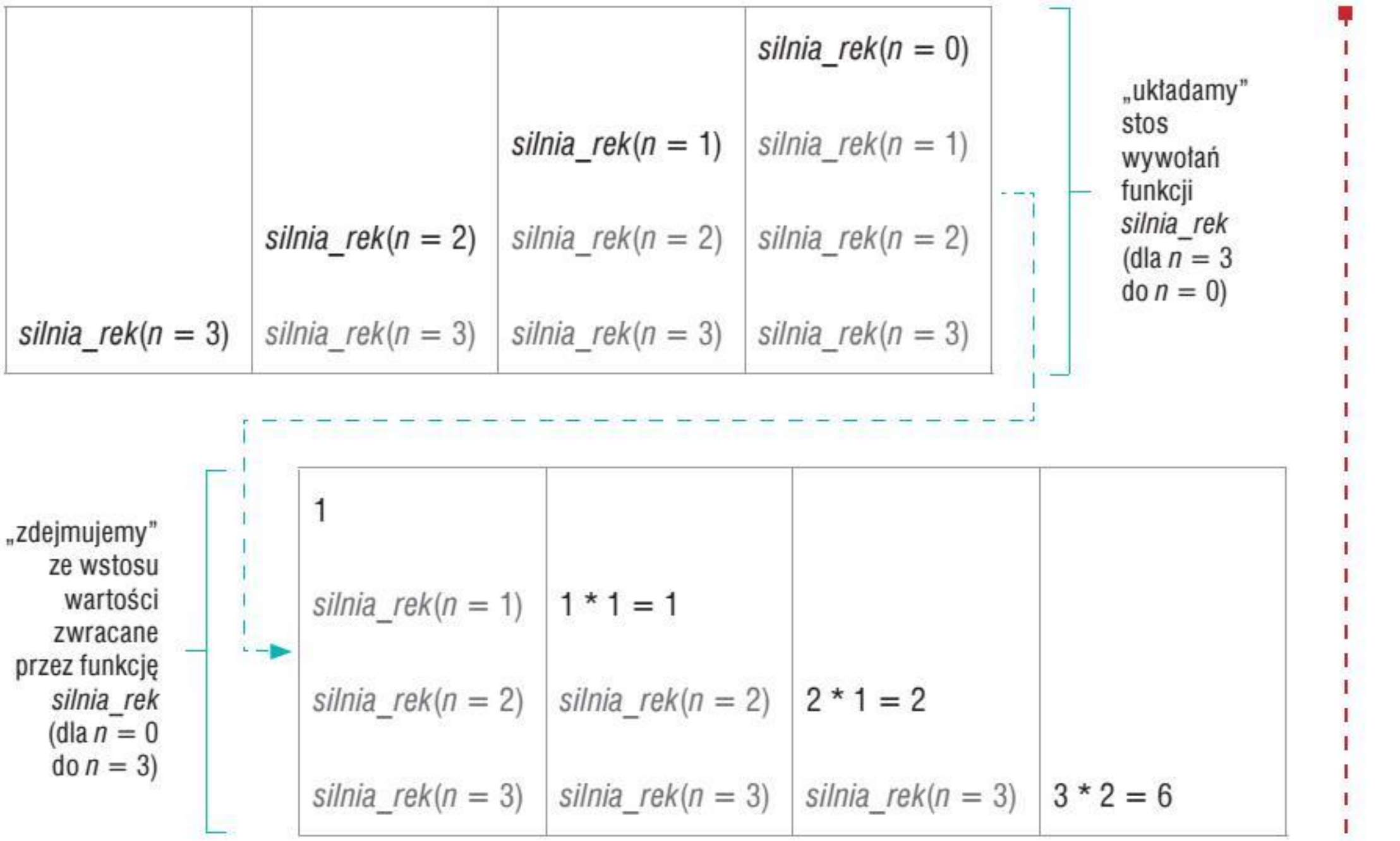
Rys. 8b. *Rekurencyjna definicja funkcji obliczającej silnię liczby naturalnej (Python)*

Na rysunku 9. zobrazowano schematycznie, w jaki sposób zmieniają się wartości funkcji *silnia_rek()* podczas tworzenia tzw. **stosu** wywołań rekurencyjnych funkcji *silnia_rek()*, a potem podczas ich „zdejmowania" (począwszy od ostatnio wywołanej funkcji).



S **Stos w informatyce**

Struktura przechowywania danych, w której dostępny jest tylko element znajdujący się na wierzchołku stosu (tak jak w stosie książek na biurku – dostępna jest książka położona na wierzchu jako ostatnia).



Rys. 9. *Schemat wywołań rekurencyjnych funkcji* silnia_rek() *dla* n=*3*

Ćwiczenie 10. Zapisujemy w postaci programu rekurencyjną realizację algorytmu obliczającego silnię liczby naturalnej *n*

1. W wybranym języku programowania napisz program rekurencyjnej realizacji algorytmu obliczania silni. Wykorzystaj funkcję *silnia_rek()* z rysunku 8a lub 8b. Omów jej działanie – wyjaśnij znaczenie poszczególnych instrukcji.
2. Zapisz program w pliku pod nazwą *Silnia_rekurencyjnie*.
3. Przetestuj program dla kilku różnych danych.

Ćwiczenie 11. Modyfikujemy program

1. Zmodyfikuj program zapisany w ćwiczeniu 10., aby obliczał i wyświetlał w jednej kolumnie wartości silni dla liczb od 1 do 12.
2. Zapisz plik pod tą samą nazwą.

Wskazówka: Aby obliczyć i wyświetlić na ekranie dwanaście kolejnych wartości silni, można wywołanie funkcji *silnia_rek()* umieścić w pętli z parametrem aktualnym *i*, gdzie *i* jest zmienną globalną.

C++

```
for(i=0; i<=12; i++)
   cout << i << "!=" << silnia_rek(i);
```

Python

```
for i in range(13):
    print(i, '!=', silnia_rek(i))
```



Rekurencja i iteracja różnią się zasadniczo. Powtórzenia w rekurencji są innego rodzaju niż powtórzenia właściwe dla iteracji.

Powtórzenia w rekurencji „zagłębiają się w siebie" na zasadzie powrotu do tej samej definicji (funkcji), czyli wywoływania jej wewnątrz niej samej, ale z innymi parametrami.

W informatyce rekurencja jest szczególnym rodzajem powtarzania operacji bez konieczności stosowania pętli.

5. Rekurencyjne obliczanie wartości elementów ciągu Fibonacciego

Ciągiem Fibonacciego nazywamy ciąg liczb naturalnych, w którym każda liczba (z wyjątkiem pierwszej i drugiej) jest sumą dwóch poprzednich.

$$F_n = \begin{cases} 1 & \text{dla } n \in \{1, 2\} \\ F_{n-1} + F_{n-2} & \text{dla } n \geq 3 \end{cases}$$



Podana definicja ciągu Fibonacciego jest rekurencyjna, gdyż odwołuje się do siebie samej.

Aby obliczyć poszczególne liczby Fibonacciego, można posłużyć się zarówno algorytmem iteracyjnym, jak i rekurencyjnym. Jednak w tym przypadku algorytm w postaci rekurencyjnej wymaga wykonania dużo większej liczby kroków obliczeniowych niż ten sam algorytm w postaci iteracyjnej.

Funkcja realizująca rekurencyjny algorytm obliczania liczb Fibonacciego powinna być wywoływana dwukrotnie wewnątrz siebie samej: raz z parametrem $(n - 1)$, drugi raz z parametrem $(n - 2)$. Na rysunkach 10a i 10b pokazano przykładowe definicje funkcji *fib_rek()* w wybranych językach, do samodzielnego uzupełnienia.

C++

```
 6  long fib_rek(int n)
 7  {
 8      if( n < 3 )
 9          return [redacted]
10      else
11          return [redacted]
12  }
```

***Rys. 10a**. Rekurencyjna definicja funkcji obliczającej liczby Fibonacciego (C++)*

Python

```
def fib_rek(n):
    if( n < 3 ):
        return [redacted]
    else:
        return [redacted]
```

***Rys. 10b**. Rekurencyjna definicja funkcji obliczającej liczby Fibonacciego (Python)*

Ćwiczenie 12. Przedstawiamy schemat wywołań rekurencyjnych funkcji *fib_rek*

1. Zastanów się, w jaki sposób należy uzupełnić funkcje pokazane na rysunkach 10a i 10b. Zapisz wybraną funkcję i uzupełnij ją.
2. Wykonaj, podobnie jak pokazano na rysunku 9., schemat rekurencyjnego wyznaczania sześciu pierwszych liczb Fibonacciego.

Ćwiczenie 13. Zapisujemy w postaci programu rekurencyjną realizację algorytmu obliczającego liczby Fibonacciego

1. Korzystając z wybranej definicji funkcji zapisanej w ćwiczeniu 12., napisz program w wersji rekurencyjnej, obliczający czterdzieści pierwszych liczb Fibonacciego.
2. Zapisz program w pliku pod nazwą *Fibonacci_rekurencyjnie*.

Warto zapamiętać

- 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.
- Algorytm zachłanny to algorytm, który w celu wyznaczenia rozwiązania w każdym kroku dokonuje zachłannego, tj. najkorzystniejszego w danym kroku rozwiązania częściowego.
- Z rekurencją spotykamy się w definicjach, w których definiowane pojęcie odwołuje się do samego siebie.
- Aby funkcja została zdefiniowana rekurencyjnie, musi zawierać w swojej definicji odwołanie do samej siebie.
- W algorytmach rekurencyjnych (podobnie jak w iteracyjnych) musi być wprowadzony warunek zakończenia wywołań rekurencyjnych.
- Algorytmy obliczania silni i wyznaczania liczb Fibonacciego można realizować w języku programowania, definiując funkcje rekurencyjne.
- W technice iteracji i rekurencji mamy do czynienia z powtórzeniami, jednak są to powtórzenia innego rodzaju. W przypadku iteracji stosujemy pętle, natomiast w rekurencji – nie (powtórzenia „zagłębiają się w siebie" na zasadzie powrotu do tej samej definicji).

Pytania i polecenia

1. Na czym polega metoda połowienia? Podaj przykład wyszukiwania metodą połowienia z codziennego życia.
2. Omów metodę zachłanną wydawania reszty wynoszącej 178 zł. Dostępne nominały to 100 zł, 50 zł, 20 zł, 10 zł, 5 zł, 2 zł, 1 zł.
3. Na czym polega rekurencja?
4. Dlaczego w pewien sposób porównuje się iterację do rekurencji?
5. Korzystając ze schematu pokazanego na rysunku 9., omów rekurencyjną realizację algorytmu obliczania silni dla $n = 4$. Na wybranym przykładzie omów sposób definiowania funkcji rekurencyjnej.
6. Pokaż różnicę między iteracją a rekurencją na przykładzie algorytmu wybranego spośród omówionych w tym podręczniku.

Zadania

1. Wyjaśnij, dlaczego przy liczbach Fibonacciego umieszczono zdjęcie królików.
2. W wybranym języku programowania napisz program realizujący zachłanny algorytm przewozu książek z ćwiczenia 4. Liczbę książek do przewiezienia oraz liczbę książek mieszczących się na palecie i w paczce wprowadzaj z klawiatury.
3. W wybranym języku programowania napisz program realizujący zachłanny algorytm przewozu ładunku z ćwiczenia 5. Liczbę palet do przewiezienia, liczbę samochodów i ich ładowność wprowadzaj z klawiatury.
4. Stosując rekurencyjną definicję obliczania silni, napisz program obliczający wartość symbolu Newtona: $\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$, gdzie n i k – liczby naturalne.
5. W wybranym języku programowania napisz program realizujący iteracyjny algorytm obliczania liczb Fibonacciego.

Dla zainteresowanych

6. Sformułuj samodzielnie problem, do którego rozwiązania można zastosować podejście zachłanne. Napisz program realizujący ten algorytm.
7. Korzystając z rozwiązania zadania ., napisz program obliczający, na ile sposobów można wybrać *k*-osobową reprezentację spośród *s* uczniów jednej klasy (*k* i *s* wprowadzane z klawiatury). Umieść w programie odpowiednie opisy, wyświetlające się przy wprowadzaniu danych i wyprowadzaniu wyniku.
8. Wykorzystując program w języku C++ zapisany w ćwiczeniu 9., oblicz wartość 16! i 17!. Porównaj otrzymane wartości z wartościami wyznaczonymi inną metodą (np. za pomocą kalkulatora naukowego lub arkusza kalkulacyjnego). Co zauważasz?
9. Zapisz w wybranym języku programowania rekurencyjną realizację algorytmu Euklidesa w wersji z odejmowaniem.

Temat C4. Algorytmy na liczbach w językach C++ i Python

1. Zamiana liczby dwójkowej na dziesiętną
2. Zamiana liczby dziesiętnej na dwójkową
3. Zamiana liczb między systemami pozycyjnymi
   - 3.1. Obliczanie wartości dziesiętnej liczby
   - 3.2. Zapisywanie liczby dziesiętnej w systemie liczbowym o określonej podstawie
4. Działania na ułamkach z wykorzystaniem NWD i NWW
   - 4.1. NWD i NWW
   - 4.2. Reprezentacja ułamka w programie komputerowym
   - 4.3. Skracanie ułamków zwykłych
   - 4.4. Dodawanie i odejmowanie ułamków zwykłych
   - 4.5. Mnożenie i dzielenie ułamków zwykłych

Warto powtórzyć

1. W jaki sposób wyznaczamy rozwinięcie dwójkowe liczby dziesiętnej? Pokaż na przykładzie.
2. Jak oblicza się wartość dziesiętną liczby binarnej?
3. Do czego służy algorytm Euklidesa?
4. Na czym polega algorytm Euklidesa w wersji z odejmowaniem, a na czym w wersji z dzieleniem?
5. W jaki sposób zapisujemy algorytm Euklidesa w wybranym języku programowania (C++ lub Python)? Pokaż na przykładzie jednej z wersji algorytmu.

1. Zamiana liczby dwójkowej na dziesiętną

W temacie A1 omówiliśmy system dwójkowy oraz algorytmy zamiany liczb pomiędzy systemami dziesiętnym i dwójkowym. W tym temacie zaprogramujemy te algorytmy w językach C++ i Python.

Aby obliczyć wartość dziesiętną liczby zapisanej w systemie dwójkowym, dokonujemy rozwinięcia liczby według wzoru podanego w temacie A1 (str. 12), przyjmując jako podstawę systemu liczbę 2. W algorytmie powtarzamy operację dla każdej cyfry liczby binarnej, mamy więc do czynienia z iteracją, dlatego zastosujemy instrukcję pętli. Pisząc program, należy zastanowić się, jakiego typu powinna być zmienna, w której będziemy pamiętać liczbę dwójkową (binarną).

Na rysunkach 1a i 1b pokazano fragmenty funkcji zamieniającej liczbę dwójkową na dziesiętną. Korzystamy ze wzoru na wartość dziesiętną liczby dwójkowej (podanego w temacie A1). Aby uniknąć operacji potęgowania (dla obliczenia 2^n), kolejne potęgi dwójki otrzymujemy, podwajając wcześniej obliczoną potęgę (`potega *= 2`).

C++

```
potega = 1;
for(int i = binarna.length() - 1; i >= 0; --i)
{
    dziesietna += (binarna[i] - '0')*potega;
    potega *= 2;
}
```

Rys. 1a. *Fragment programu obliczającego wartość dziesiętną liczby dwójkowej (C++)*

Python

```
dziesietna = 0
potega = 1
for i in range(len(binarna)-1, -1, -1):
   dziesietna += (ord(binarna[i]) - ord('0'))*potega
   potega *= 2
```

Rys. 1b. *Fragment programu obliczającego wartość dziesiętną liczby dwójkowej (Python)*

Ćwiczenie 1. Piszemy program obliczający wartość dziesiętną liczby dwójkowej

1. W wybranym języku programowania utwórz program obliczający wartość dziesiętną liczby zapisanej w systemie dwójkowym (binarnej). Napisz specyfikację tego zadania. Zastanów się, w jaki sposób musi być zapamiętana liczba dwójkowa (jakiego typu powinna być zmienna, w której będziemy pamiętać liczbę dwójkową). Zdefiniuj funkcję *dwojkowa_na_dziesietna()*. Przeanalizuj fragment programu pokazany na rysunku 1a lub 1b (wyjaśnij znaczenie poszczególnych instrukcji) i użyj go w funkcji. Funkcję wywołaj w programie głównym.
2. Zapisz program w pliku pod nazwą *Dwa_na_10*.
3. Przetestuj program dla kilku różnych danych.

2. Zamiana liczby dziesiętnej na dwójkową

Jak wiemy z tematu A1, aby wyznaczyć postać dwójkową liczby dziesiętnej, powtarzamy operację dzielenia z resztą tej liczby przez dwa i zapisujemy wszystkie reszty z dzielenia, aż do otrzymania ilorazu równego zero. Rozwinięcie dwójkowe liczby to zapisane kolejno (począwszy od ostatniej) reszty z dzieleń. W tym algorytmie występuje powtarzanie poleceń, dlatego użyjemy instrukcji pętli.

Na rysunkach 2a i 2b pokazano fragmenty funkcji wyznaczającej rozwinięcie dwójkowe liczby dziesiętnej.

W języku C++ zapis: `(char)(dziesietna % 2 + '0')` oznacza zamianę wyrażenia występującego po `(char)` na typ zapisany w nawiasie, czyli w tym przypadku **`char`**.

C++

```
string wynik = "";
do
{
    wynik = (char)(dziesietna % 2 + '0') + wynik;
    dziesietna /= 2;
} while(dziesietna > 0);
```

Rys. 2a. *Fragment programu wyznaczającego rozwinięcie dwójkowe liczby dziesiętnej (C++)*

Python

```
if dziesietna == 0:
    wynik = '0'
else:
    while dziesietna > 0:
        wynik = chr(dziesietna % 2 + ord('0')) + wynik
        dziesietna //= 2
```

Rys. 2b. *Fragment programu wyznaczającego rozwinięcie dwójkowe liczby dziesiętnej (Python)*



Ćwiczenie 2. Piszemy program wyznaczający rozwinięcie dwójkowe liczby dziesiętnej

1. W wybranym języku programowania napisz program wyznaczający rozwinięcie dwójkowe liczby dziesiętnej. Zdefiniuj funkcję *dziesietna_na_dwojkowa()*. Napisz specyfikację tego zadania. Przeanalizuj fragment programu pokazany na rysunku 2a lub 2b (wyjaśnij znaczenie poszczególnych instrukcji) i użyj go w funkcji. Funkcję wywołaj w programie głównym.
2. Zapisz program w pliku pod nazwą *Dziesięć_na_2*.
3. Przetestuj program dla kilku różnych danych.
4. Wyjaśnij, dlaczego użyto różnych instrukcji iteracyjnych w programach pokazanych na rysunkach 1a i 2a lub na rysunkach 1b i 2b.

3. Zamiana liczb między systemami pozycyjnymi

Zastanówmy się teraz, jak można przedstawić algorytm zamiany liczb między dowolnymi systemami pozycyjnymi.

Skoro wiemy, jak zamienić liczbę z systemu dwójkowego na dziesiętny i odwrotnie, to czy możemy wykorzystać taki sposób podejścia do zamiany liczb pomiędzy innymi systemami?

Może wystarczy obliczyć najpierw wartość dziesiętną liczby zapisanej w dowolnym systemie pozycyjnym, a następnie zapisać ją w docelowym systemie?

Zacznijmy od zapisania listy kroków zamiany liczb pomiędzy systemami liczbowymi.

Przykład 1. Algorytm zamiany liczb między dowolnymi pozycyjnymi systemami liczbowymi

Zadanie: Zamień liczbę *k* zapisaną w systemie pozycyjnym o podstawie p_1 na liczbę w systemie pozycyjnym o podstawie p_2.

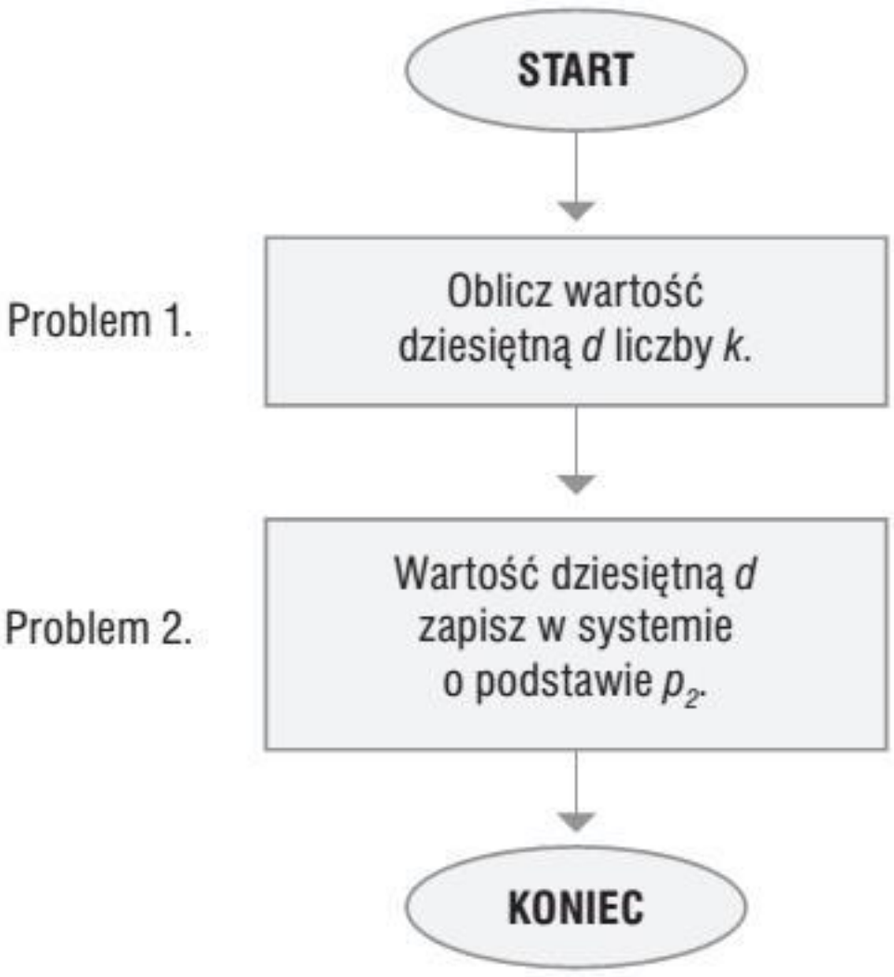
Dane: Liczba *k* zapisana w systemie pozycyjnym o podstawie p_1.

Wynik: Liczba *k* zapisana w systemie pozycyjnym o podstawie p_2.

Lista kroków:

1. Zacznij algorytm.
2. Oblicz wartość dziesiętną *d* liczby *k*.
3. Obliczoną wartość dziesiętną *d* zapisz w systemie o podstawie p_2.
4. Zakończ algorytm.

Rozwiązanie problemu sprowadza się do rozwiązania dwóch podproblemów (kroki 2. i 3. listy kroków algorytmu podanego w przykładzie 1.).



Rys. 3. *Podział problemu na podproblemy w algorytmie zamiany liczb pomiędzy systemami liczbowymi*

W następnych punktach tematu podamy specyfikację zadań i listy kroków rozwiązania problemów cząstkowych, aby ułatwić ich zapisanie w postaci funkcji w wybranym języku programowania.

3.1. Obliczanie wartości dziesiętnej liczby

Jak pamiętamy, rozwinięcie liczby zapisanej w systemie pozycyjnym o określonej podstawie ma postać: $(a_{n-1}\, a_{n-2} \ldots a_1 a_0)_p = a_{n-1} \cdot p^{n-1} + a_{n-2} \cdot p^{n-2} + \ldots + a_1 \cdot p^1 + a_0 \cdot p^0$, gdzie *p* to podstawa systemu liczbowego.

Rozwinięcie to jest wielomianem. Najszybszym sposobem obliczenia wartości wielomianu jest skorzystanie ze **schematu Hornera**.

Wzór na obliczanie wartości wielomianu, zwany **schematem Hornera**:

$W(x) = (...((a_0x + a_1)x + a_2)x...+ a_{n-1})x + a_n$.

Przykład 2. Algorytm obliczania wartości dziesiętnej liczby

Zadanie: Oblicz wartość dziesiętną liczby k.
Dane: Liczba $k = (a_{n-1}\, a_{n-2} \ldots a_1 a_0)_{p_1}$, zapisana w systemie pozycyjnym o podstawie p_1.
Wynik: Wartość dziesiętna d liczby k.
Lista kroków:

1. Zacznij algorytm.
2. Zmiennej d przypisz wartość 0: $d = 0$.
3. Zmiennej i przypisz wartość $n - 1$: $i = n - 1$.
4. Zmiennej d przypisz wartość $d \cdot p_1 + a_i$: $d = d \cdot p_1 + a_i$.
5. Zmniejsz i o 1: $i = i - 1$.
6. Jeśli $i >= 0$, przejdź do kroku 4.
7. Wyprowadź d.
8. Zakończ algorytm.

Ćwiczenie 3. Analizujemy listę kroków algorytmu obliczania wartości dziesiętnej liczby

Przetestuj dla kilku różnych danych działanie algorytmu obliczania wartości dziesiętnej liczby zapisanego w postaci listy kroków w przykładzie 2.

Wskazówka: Analizę możesz przeprowadzić w tabeli podobnej do tabel tworzonych do analizy innych algorytmów w poprzednich tematach.

Aby algorytm był uniwersalny i obsługiwał systemy pozycyjne o podstawie większej od 10, przyjmiemy, że do zapisania cyfr w tych systemach używa się kolejnych liter alfabetu (temat A1).

Jedyna trudność w realizacji tego algorytmu w języku programowania to konieczność reprezentowania liczby k jako ciągu znaków.

W języku Python wpisanie instrukcji:

`liczba = input("wprowadź cyfry liczby")` spowoduje, że po wprowadzaniu z klawiatury 345 w zmiennej `liczba` będzie pamiętany ciąg znaków „345”, a nie liczba.

Ćwiczenie 4. Piszemy program obliczający wartość dziesiętną liczby

1. Na podstawie listy kroków podanej w przykładzie 2. napisz program obliczający wartość dziesiętną liczby k, zapisanej w systemie o podstawie p. Zdefiniuj odpowiednią funkcję o dwóch parametrach: *liczba* typu tekstowego, w którym będzie przekazywana liczba zapisana w postaci kolejnych cyfr, oraz *podstawa*.
2. Zapisz program w pliku pod nazwą *Dowolny _na_dziesiętny*. Przetestuj program dla kilku różnych danych.
3. Wywołaj funkcję w programie głównym i przetestuj działanie programu dla różnych danych.

Wskazówka:

C++

W języku C++ wartości typu **`char`** mają wartości liczbowe odpowiadające kodowi ASCII znaku. Przyjmując, że *c* jest znakiem cyfry (np. szesnastkowej), wartość tej cyfry reprezentują wyrażenia:

`c - '0'` dla cyfr 0...9,
`c - 'A' + 10` dla „cyfr" A...Z,
`c - 'a' + 10` dla „cyfr" a...z.

Python

W języku Python znaki nie mają bezpośredniej wartości liczbowej i aby jej użyć wykorzystujemy funkcję `ord()`. Przyjmując, że *c* jest znakiem cyfry (np. szesnastkowej), wartość tej cyfry reprezentują wyrażenia:

`ord(c) - ord('0')` dla cyfr 0...9,
`ord(c) - ord('A') + 10` dla „cyfr" A...Z,
`ord(c) - ord('a') + 10` dla „cyfr" a...z.

Zamieniając znak na odpowiadającą mu wartość, należy najpierw sprawdzić, do którego z podanych wyżej przedziałów należy cyfra, i użyć odpowiedniego wyrażenia. W tym celu warto utworzyć osobną funkcję.

3.2. Zapisywanie liczby dziesiętnej w systemie liczbowym o określonej podstawie

W temacie A1 opisaliśmy algorytm zamiany liczby dziesiętnej na dwójkową. Liczbę dziesiętną dzieliliśmy przez 2, a uzyskane reszty z dzielenia, czytane od końca, tworzyły rozwinięcie dwójkowe. Algorytm ten można uogólnić dla dowolnej podstawy *p*, dzieląc liczbę nie przez 2, lecz przez podstawę systemu *p*.

Przykład 3. Algorytm zapisywania liczby dziesiętnej w systemie liczbowym o określonej podstawie

Zadanie: Zapisz liczbę dziesiętną w systemie o dowolnej podstawie.
Dane: Nieujemna liczba dziesiętna *d*.
Wynik: Reprezentacja *b* liczby *d*, zapisana w systemie pozycyjnym o podstawie p_2.
Lista kroków:

1. Zacznij algorytm.
2. Przypisz *b* wartość ciągu pustego.
3. Dopisz cyfrę odpowiadającą reszcie z dzielenia *d* przez p_2 do ciągu *b* **z przodu**.
4. Przypisz *d* część całkowitą z dzielenia *d* przez p_2.
5. Jeżeli *d* jest większe od 0, przejdź do punktu 3.
6. Wyprowadź *b*.
7. Zakończ algorytm.

Ćwiczenie 5. Analizujemy listę kroków algorytmu zapisywania liczby dziesiętnej w systemie liczbowym o określonej podstawie

Przetestuj działanie algorytmu zapisywania liczby dziesiętnej w systemie liczbowym o określonej podstawie, zapisanego w postaci listy kroków w przykładzie 3., dla kilku różnych danych.

Wskazówka: Analizę możesz przeprowadzić w tabeli podobnej do tabel tworzonych w poprzednich tematach.

Ćwiczenie 6. Piszemy program zapisujący liczbę dziesiętną w systemie liczbowym o określonej podstawie

1. Na podstawie listy kroków podanej w przykładzie 3. utwórz program zapisujący liczbę dziesiętną *d* w systemie liczbowym o podstawie *p*. Zdefiniuj odpowiednią funkcję.
2. Zapisz program w pliku pod nazwą *Dziesiętny _na_dowolny*. Przetestuj program dla kilku różnych danych.

Wskazówki:

W realizacji tego algorytmu problemem jest konieczność zamiany wartości liczbowej na kod znaku.

C++

`k + '0'` dla $k = 0..9$
`k + 'A' - 10` dla $k >= 10$

Python

Aby uzyskać znak z wartości liczbowej, konieczne jest użycie funkcji `chr()`.

`chr(k + ord('0'))` dla $k = 0..9$
`chr(k + ord('A') - 10)` dla $k >= 10$

Warto utworzyć odpowiednią funkcję, zwracającą znak reprezentujący cyfrę podaną jako parametr funkcji.

4. Działania na ułamkach z wykorzystaniem NWD i NWW

W obliczeniach, w których posługujemy się ułamkami zwykłymi, ma zastosowanie znajdowanie największego wspólnego dzielnika. Do zapisania działań na ułamkach zwykłych w języku programowania będziemy zatem potrzebowali funkcji obliczających największy wspólny dzielnik **NWD** i najmniejszą wspólną wielokrotność **NWW**.

4.1. NWD i NWW

NWD

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

NWW

Najmniejsza wspólna wielokrotność dwóch lub więcej liczb całkowitych jest to najmniejsza liczba różna od zera, która jest jednocześnie wielokrotnością obu liczb.

Ćwiczenie 7. Piszemy funkcję wyznaczającą NWD

1. Napisz program, w którym zdefiniujesz funkcję *nwd()* wyznaczającą i zwracającą NWD dwóch nieujemnych liczb całkowitych *a* i *b*, będących parametrami tej funkcji. Skorzystaj z algorytmu rekurencyjnego lub iteracyjnego.
2. Wywołaj funkcję w programie głównym i przetestuj działanie programu dla różnych danych.
3. Zapisz program w pliku pod nazwą *Ułamki*.

Wskazówka: Aby zapisać definicję funkcji *nwd()* w wersji rekurencyjnej, możesz skorzystać z rozwiązania zadania 9. z tematu C3.

Aby wyznaczyć NWW dwóch liczb całkowitych, wystarczy pomnożyć je przez siebie, a następnie podzielić przez ich NWD.

Zależność między NWD a NWW przedstawia wzór:

$$NWW(m, n) = \frac{m \cdot n}{NWD(m, n)}$$

Ćwiczenie 8. Piszemy funkcję wyznaczającą NWW

1. Otwórz plik *Ułamki* zapisany w ćwiczeniu 7. Zdefiniuj funkcję *nww()* wyznaczającą i zwracającą NWW dwóch nieujemnych liczb całkowitych *a* i *b*, będących parametrami tej funkcji.
2. Wywołaj funkcję w programie głównym i przetestuj działanie programu dla różnych danych.
3. Zapisz plik pod tą samą nazwą.

Wskazówka: W definicji funkcji *nww()* użyj funkcji *nwd()*.

4.2. Reprezentacja ułamka w programie komputerowym

Ułamek zwykły opisują dwie liczby całkowite: licznik i mianownik przedzielone kreską ułamkową. W programie komputerowym licznik i mianownik ułamka można zapamiętać np. w postaci tablicy dwuelementowej lub rekordu (w języku C++), a w języku Python – np. w postaci klasy.

Reprezentacja ułamka w języku C++

Rekord w języku programowania to struktura danych służąca do przechowywania wielu danych tego samego typu lub różnych typów, przy czym poszczególne dane są identyfikowane przez nazwę pola rekordu (jest to rozwiązanie analogiczne do stosowanego w bazach danych).

W języku C++ rekordy deklarujemy za pomocą słowa kluczowego **`struct`**, w nawiasach klamrowych, podając typy i nazwy pól rekordu. Zadeklarować możemy zmienną (zmienne) typu rekordowego (nazwę zmiennej lub nazwy zmiennych podajemy po klamrze zamykającej) lub typ zmiennych rekordowych (nazwę typu podajemy po słowie kluczowym **`struct`**).

W przypadku utworzenia typu rekordowego, nazwy tego typu można używać w deklaracjach zmiennych podobnie jak w przypadku standardowych typów języka C++ (np. **`int`**, **`float`**, **`char`**, **`bool`**, **`string`**).

Deklaracja zmiennej *moj_ulamek* mającej pola *licznik* i *mianownik* typu całkowitego:

C++

```
struct
{
    int licznik;
    int mianownik;
} moj_ulamek;
```

! Uwaga

Z pojęciem rekordu spotkaliśmy się w tematach dotyczących baz danych, gdzie rekord również składał się z pól różnego bądź tego samego typu.

Deklaracja typu rekordowego *ulamek*, mającego pola *licznik* i *mianownik* typu całkowitego, oraz dwóch zmiennych *u1* i *u2* typu *ulamek*:

C++

```
struct ulamek
{
    int licznik;
    int mianownik;
};
ulamek u1, u2;
```

Do pól odwołujemy się, podając nazwę zmiennej typu rekordowego i nazwę pola, rozdzielone kropką, np.:

```
u1.licznik = 1;
u1.mianownik = 2;
cout << u1.licznik << "/" << u1.mianownik << endl;
```

Reprezentacja ułamka w języku Python

Klasa w języku programowania Python, podobnie jak rekord w C++, umożliwia przechowywanie danych w polach. Klasa może również mieć metody definiujące jej zachowanie. Można powiedzieć, że klasa definiuje szablon, na podstawie którego program może utworzyć **obiekty** tej klasy. Obiekt danej klasy często nazywa się **instancją** tej klasy. Można również powiedzieć, że typem takiego obiektu jest dana klasa.

W języku Python klasę definiuje się za pomocą słowa kluczowego `class`. W ramach klasy można zdefiniować metody, w tym specjalną metodę inicjalizującą `__init__`, która zostanie wywołana przy tworzeniu obiektu danej klasy.

W poniższej ramce pokazano przykład klasy *Ulamek* i utworzono dwie instancje tej klasy. W metodzie `__init__` utworzono dwa pola (*licznik* i *mianownik*), którym przy tworzeniu instancji zostały przypisane wartości argumentów (odpowiednio: *licz*, *mian*). Dodatkowy argument *self* służy do odwołania się do tworzonej instancji obiektu.

Aby utworzyć zmienną danej klasy (instancję), używamy instrukcji przypisania, w której po prawej stronie znajdują się: nazwa klasy i parametry początkowe obiektu, np.:

```
u1 = Ulamek(1, 2)
```

Uwaga !

W języku C++ również istnieją klasy i obiekty.

Python

```
class Ulamek:
    def __init__(self, licz, mian):
        self.licznik = licz
        self.mianownik = mian
u1 = Ulamek(1, 2)
u2 = Ulamek(3, 5)
```

Do pól klasy odwołujemy się, podając nazwę zmiennej danej klasy (instancji) i nazwę pola, rozdzielone kropką, np.:

```
u1.licznik = 1
u1.mianownik = 2
```

Aby łatwo wyświetlić na ekranie dane z obiektu takiej klasy, można użyć metody `format`:

```
print("{}/{}".format(u1.licznik, u1.mianownik))
```

Zamiast pierwszych nawiasów klamrowych `{}` zostanie wstawiony pierwszy parametr wywołania `format`, czyli `u1.licznik`. Zamiast drugich nawiasów klamrowych zostanie wstawiony drugi parametr, czyli `u1.mianownik`. Końcowym efektem będzie wyświetlenie na ekranie: 1/2.

4.3. Skracanie ułamków zwykłych



Aby skrócić ułamek, należy licznik i mianownik podzielić przez ich NWD. Jeżeli NWD jest równy 1, to ułamek jest nieskracalny i oczywiście nie trzeba wykonywać dzielenia.



Ćwiczenie 9. Piszemy funkcję skracającą ułamki zwykłe

1. Otwórz plik *Ułamki* zapisany w ćwiczeniu 8. Napisz funkcję *skroc()* skracającą ułamek przekazany jako parametr i zwracającą skrócony ułamek. Wzory takich funkcji pokazano na rysunkach 4a i 4b. Objaśnij znaczenie poszczególnych wierszy wybranej funkcji.
2. Wywołaj funkcję w programie głównym i przetestuj działanie programu dla różnych danych.
3. Zapisz plik pod tą samą nazwą.

Wskazówka: Wykorzystaj strukturę *ulamek* (C++) lub klasę *Ulamek* (Python), których definicję podaliśmy w punkcie 4.2.

C++

```
64  ulamek skroc(ulamek a)
65  {
66      ulamek wynik = a;
67      int nwd_ulamka = nwd(a.licznik, a.mianownik);
68      wynik.licznik /= nwd_ulamka;
69      wynik.mianownik /= nwd_ulamka;
70      return wynik;
71  }
```

Rys. 4a. *Funkcja skracająca ułamki zwykłe (C++)*

Python

```
def skroc(a):
    nwd_ulamka = nwd(a.licznik, a.mianownik)
    return Ulamek(a.licznik // nwd_ulamka, a.mianownik // nwd_ulamka)
```

Rys. 4b. *Funkcja skracająca ułamki zwykłe (Python)*

4.4. Dodawanie i odejmowanie ułamków zwykłych

Wzór na obliczenie sumy ułamków to:

$$\frac{l_1}{m_1} + \frac{l_2}{m_2} = \frac{l_1 \cdot m_2 + l_2 \cdot m_1}{m_1 \cdot m_2}$$

Można również wyznaczyć najpierw NWW mianowników m_1 i m_2: $nww(m_1, m_2)$. Wtedy:

$$\frac{l_1}{m_1} + \frac{l_2}{m_2} = \frac{l_1 \cdot \frac{nww}{m_1} + l_2 \cdot \frac{nww}{m_2}}{nww}$$

W praktyce, o ile nie wykonujemy działań na dużych liczbach, przekraczających zakres dopuszczalnych wartości danego typu zmiennej, wygodniej jest korzystać z pierwszej metody, ponieważ wyznaczenie NWW jest stosunkowo czasochłonne (wymaga uprzedniego iteracyjnego lub rekurencyjnego wyznaczenia NWD). Otrzymany ułamek można skrócić.

C++

```
73  ulamek dodaj(ulamek a, ulamek b)
74  {
75      ulamek wynik;
76      wynik.licznik = a.licznik * b.mianownik + b.licznik * a.mianownik;
77      wynik.mianownik = a.mianownik * b.mianownik;
78      return skroc(wynik);
79  }
```

Rys. 5a. *Funkcja dodająca ułamki zwykłe (C++)*

Python

```
def dodaj(a, b):
    wynik = Ulamek(a.licznik * b.mianownik + b.licznik
            * a.mianownik, a.mianownik * b.mianownik)
    return skroc(wynik)
```

Rys. 5b. *Funkcja dodająca ułamki zwykłe (Python)*

Ćwiczenie 10. Piszemy funkcję dodającą ułamki

1. Otwórz plik *Ułamki* zapisany w ćwiczeniu 9. Napisz funkcję dodającą ułamki. Wzory takich funkcji pokazano na rysunkach 5a i 5b. Objaśnij znaczenie poszczególnych wierszy wybranej funkcji.
2. Wywołaj funkcję w programie głównym i przetestuj działanie programu dla różnych danych.
3. Zapisz plik pod tą samą nazwą.

Podobnie obliczymy różnicę ułamków, w powyższych wzorach zamieniając dodawanie na odejmowanie.

Ćwiczenie 11. Piszemy funkcję odejmującą ułamki

1. Otwórz plik *Ułamki* zapisany w ćwiczeniu 10. Napisz funkcję *odejmij*(*u1*, *u2*), odejmującą zmienną *u2* typu *ulamek* (klasy *Ułamek*) od zmiennej *u1* tego samego typu i zwracającą różnicę ułamkową tych zmiennych. Implementując funkcje *dodaj()* i *odejmij()*, pamiętaj o skróceniu otrzymanego wyniku. Wykorzystaj funkcję *skroc()* z ćwiczenia 9.
2. Wywołaj funkcję w programie głównym i przetestuj działanie programu dla różnych danych.
3. Zapisz plik pod tą samą nazwą.

Uwaga

W ułamku mianownik musi być zawsze różny od zera. Dla potrzeb programu przyjęliśmy, że licznik również jest różny od zera.

4.5. Mnożenie i dzielenie ułamków zwykłych

Wzór na obliczenie iloczynu ułamków to:

$$\frac{l_1}{m_1} \cdot \frac{l_2}{m_2} = \frac{l_1 \cdot l_2}{m_1 \cdot m_2}$$

Otrzymany ułamek należy skrócić.

Ćwiczenie 12. Piszemy funkcję mnożącą ułamki

1. Otwórz plik *Ułamki* zapisany w ćwiczeniu 11. Napisz funkcję *pomnoz()*, mnożącą *u1* i *u2* typu *ulamek* (klasy *Ułamek*) i zwracającą iloczyn ułamkowy tych zmiennych.
2. Wywołaj funkcję w programie głównym i przetestuj działanie programu dla różnych danych.
3. Zapisz plik pod tą samą nazwą.

Aby obliczyć iloraz ułamków, należy obliczyć iloczyn pierwszego ułamka i odwrotności drugiego ułamka:

$$\frac{l_1}{m_1} : \frac{l_2}{m_2} = \frac{l_1}{m_1} \cdot \frac{m_2}{l_2} = \frac{l_1 \cdot m_2}{m_1 \cdot l_2}$$

Ćwiczenie 13. Piszemy funkcję dzielącą ułamki

1. Otwórz plik *Ułamki* zapisany w ćwiczeniu 12. Napisz funkcję *podziel(u1, u2)*, dzielącą *u1* przez *u2* typu *ulamek* (klasy *Ułamek*) i zwracającą iloraz ułamkowy tych zmiennych.
2. Wywołaj funkcję w programie głównym i przetestuj działanie programu dla różnych danych.
3. Zapisz plik pod tą samą nazwą.

Wskazówka: Implementując funkcje *pomnoz()* i *podziel()*, pamiętaj o skróceniu otrzymanego wyniku. Wykorzystaj funkcję *skroc()* z ćwiczenia 9.

Warto zapamiętać

- Algorytm zamiany liczb między dowolnymi systemami pozycyjnymi możemy zapisać w wybranym języku programowania.
- Do zapisania działań na ułamkach w wybranym języku programowania wykorzystujemy funkcje obliczające największy wspólny dzielnik NWD i najmniejszą wspólną wielokrotność NWW.
- Ułamek zwykły opisujemy przez dwie liczby całkowite: licznik i mianownik przedzielone kreską ułamkową. W programie komputerowym można je zapamiętać np. w postaci tablicy dwuelementowej lub rekordu (w języku C++), w postaci klasy (w języku Python).

Pytania i polecenia

1. Jak możemy zamieniać liczby między dwoma systemami pozycyjnymi o dowolnych podstawach?
2. W jaki sposób wykonujemy operacje na ułamkach zwykłych w wybranym języku programowania?

Zadania

1. Wyjaśnij sposób realizacji algorytmu zamiany liczby dwójkowej na dziesiętną i odwrotnie w wybranym języku programowania.
2. Jakie podproblemy możemy wyróżnić w algorytmie zamiany liczby między dwoma systemami pozycyjnymi o dowolnych podstawach?
3. Wyjaśnij sposób realizacji algorytmu zamiany liczby między dwoma systemami pozycyjnymi o dowolnych podstawach.
4. Które algorytmy możemy wykorzystać do zapisania w wybranym języku programowania działań na ułamkach zwykłych?
5. W jaki sposób możemy prezentować ułamki zwykłe w wybranym języku programowania?
6. Wyjaśnij na przykładzie sposób dodawania ułamków zwykłych. Omów, jak w wybranym języku programowania napisać program realizujący to działanie.

Dla zainteresowanych

7. Napisz funkcję *normalizuj_ujemne()*, przekształcającą ułamek przekazany jako parametr funkcji, tak by mianownik nigdy nie był ujemny.
8. Dodaj do funkcji operujących na ułamkach sprawdzanie, czy ich parametry są poprawne (mianownik nie może być zerem).
9. Napisz kalkulator ułamkowy. Program powinien umożliwić wprowadzenie wartości dwóch ułamków oraz zapytać, jaką operację użytkownik chce wykonać na ułamkach.
10. Napisz funkcję *poteguj()* z dwoma parametrami, które są ułamkami zwykłymi. Skorzystaj ze wzoru na potęgę o wykładniku wymiernym.
11. Napisz program realizujący rekurencyjny algorytm zamiany liczby dziesiętnej na postać binarną.
12. Napisz w wybranym języku programowania funkcje *szesn_na_dzies()* i *dzies_na_szesn()*, umożliwiające zamianę liczb z systemu szesnastkowego na dziesiętny i odwrotnie. Zadbaj o właściwą postać parametrów i wyników funkcji.

Wokół Internetu i projektów

D

Moduł

Temat D1. Elementy robotyki – projekt

Temat D2. Więcej na temat tworzenia stron internetowych – projekt



Temat D1. Elementy robotyki – projekty

1. Wprowadzenie
   1.1. Przygotowanie do wykonania projektu
   1.2. Czym jest Arduino?
2. Projekt sterowania nawadnianiem ogrodu
3. Zadania projektowe
   Zadanie 1. Wykonanie systemu „Podlewanie ogrodu”
   Zadanie 2. Rozszerzenie systemu „Podlewanie ogrodu”
   Zadanie 3. System „Zamknij okno” w pokoju
   Zadanie 4.* System „Wycisz radio” po otwarciu drzwi do pokoju
   Zadanie 5. Budowanie robota – zabawki

Warto powtórzyć

1. Podaj przykłady sytuacji warunkowych z życia codziennego.
2. W jaki sposób zapisujemy sytuacje warunkowe w językach programowania?
3. Czym jest iteracja?
4. Jak w językach programowania realizujemy iteracją?
5. W jakich sytuacjach stosujemy funkcję niezwracającą wartości?

1. Wprowadzenie

Robotyka łączy zagadnienia z takich dziedzin, jak: inżynieria mechaniczna, elektrotechnika, **automatyka** i informatyka. Celem robotyki jest tworzenie robotów, w tym sterowanie ich ruchem. Robotyka pozwala automatyzować różne czynności i procesy w przemyśle czy – coraz częściej – w życiu codziennym.

1.1. Przygotowanie do wykonania projektu

W temacie B4 zaprojektowaliśmy ogród, wykorzystując narzędzie informatyczne – odpowiedni program komputerowy. Zakładamy, że projekt został już zrealizowany w rzeczywistości – zasiano trawę i posadzono rośliny.

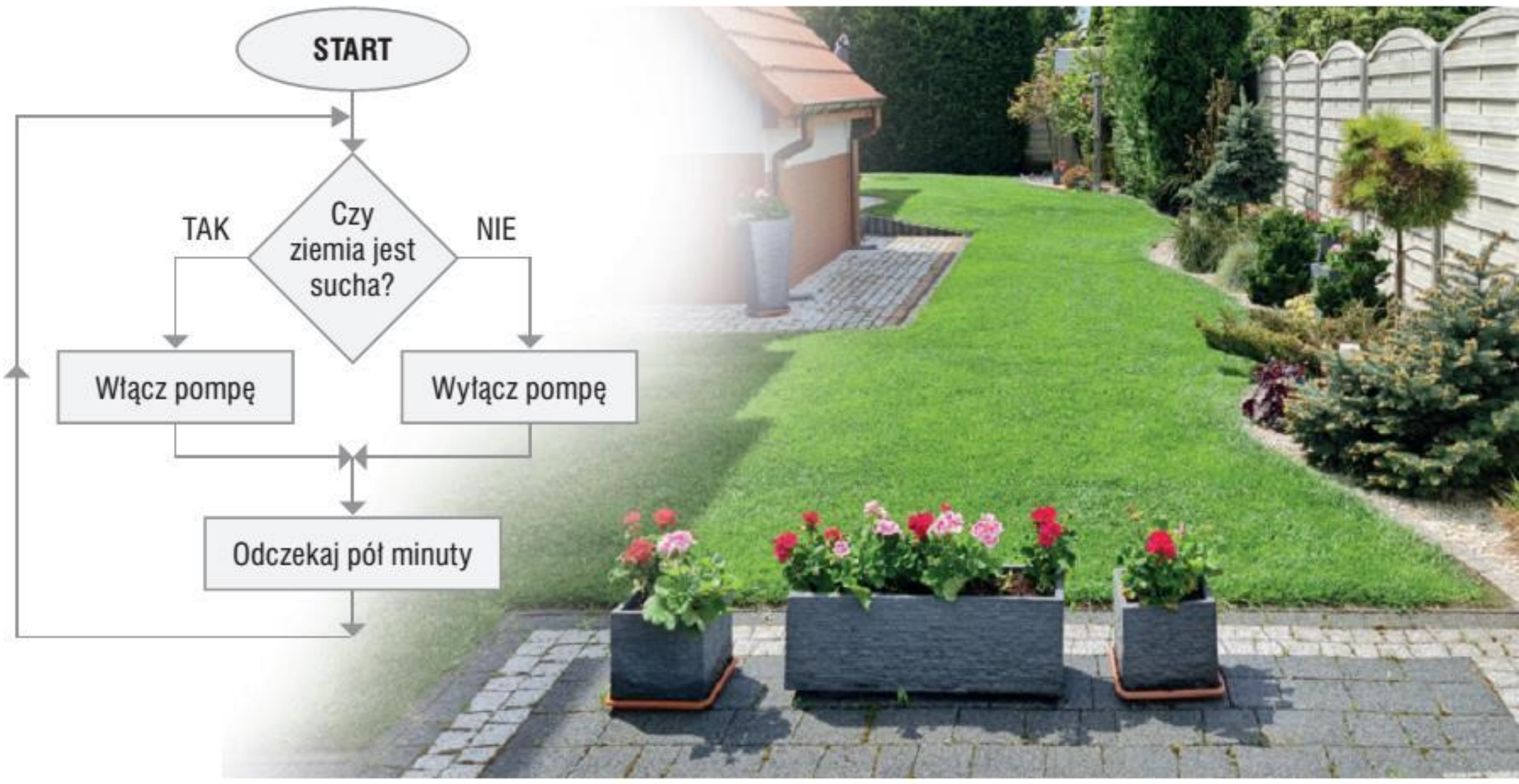
Teraz przyszedł czas na zadbanie o nasz ogród, czyli m.in. jego bieżące podlewanie. Przygotujemy projekt systemu sterowania nawadnianiem ogrodu pt. „Podlewanie ogrodu”.

Zakładamy, że podlewamy, gdy przez dłuższy czas nie pada deszcz (czyli ziemia jest sucha) oraz gdy nasz ogród przed dłuższy czas jest nasłoneczniany. Czynności te oczywiście **powtarzamy**, gdy ziemia jest sucha, czyli spada jej wilgotność. Mamy tu więc do czynienia z **sytuacją warunkową** i **iteracją** (powtarzaniem czynności podlewania).

Pokażemy, jak zrealizować ten schemat postępowania, wykonując zadania projektowe z punktu 3. tego tematu z wykorzystaniem układu **Arduino**. Omówimy najpierw, w jaki sposób przygotować prosty system sterowania nawadnianiem ogrodu (zadanie 1.), który będziemy następnie rozbudowywać (zadanie 2.).

Doświadczenia nabyte przy pracy nad tym systemem posłużą do wykonania innych systemów (zadania 3. i 4.), a w zadaniu 5. zastosujemy układ Arduino do zaprojektowania prostego, poruszającego się robota.

Podczas pracy nad projektami wykorzystamy poznane dotychczas zagadnienia związane z technikami algorytmicznymi (sytuacja warunkowa, iteracja) oraz ich realizacjami w językach programowania.



***Rys 1.** Schemat blokowy algorytmu podlewania ogrodu*

1.2. Czym jest Arduino?

Arduino to rodzina układów zawierających **mikrokontrolery**, które umożliwiają budowę robotów i prostych systemów automatyki.

Mikrokontroler to układ mający wejścia i wyjścia, do których można podłączyć różne urządzenia (np. diodę LED, przycisk, wyświetlacz, czujnik temperatury). Układ ten programuje się, aby ustalić, jak mają się zachowywać wyjścia urządzenia na podstawie parametrów wejściowych, w tym związanych z czasem – najprostszy przykład to włączenie światła, gdy jest ciemno albo gdy jest ciemno i wykryto ruch w pomieszczeniu.

Uwaga !

Programy dla Arduino możemy tworzyć w środowisku Arduino IDE (dostępnym bezpłatnie na stronie *arduino.cc*).

Pin w elektronice P

Styk, przez który wprowadzamy lub wyprowadzamy sygnały z urządzenia.

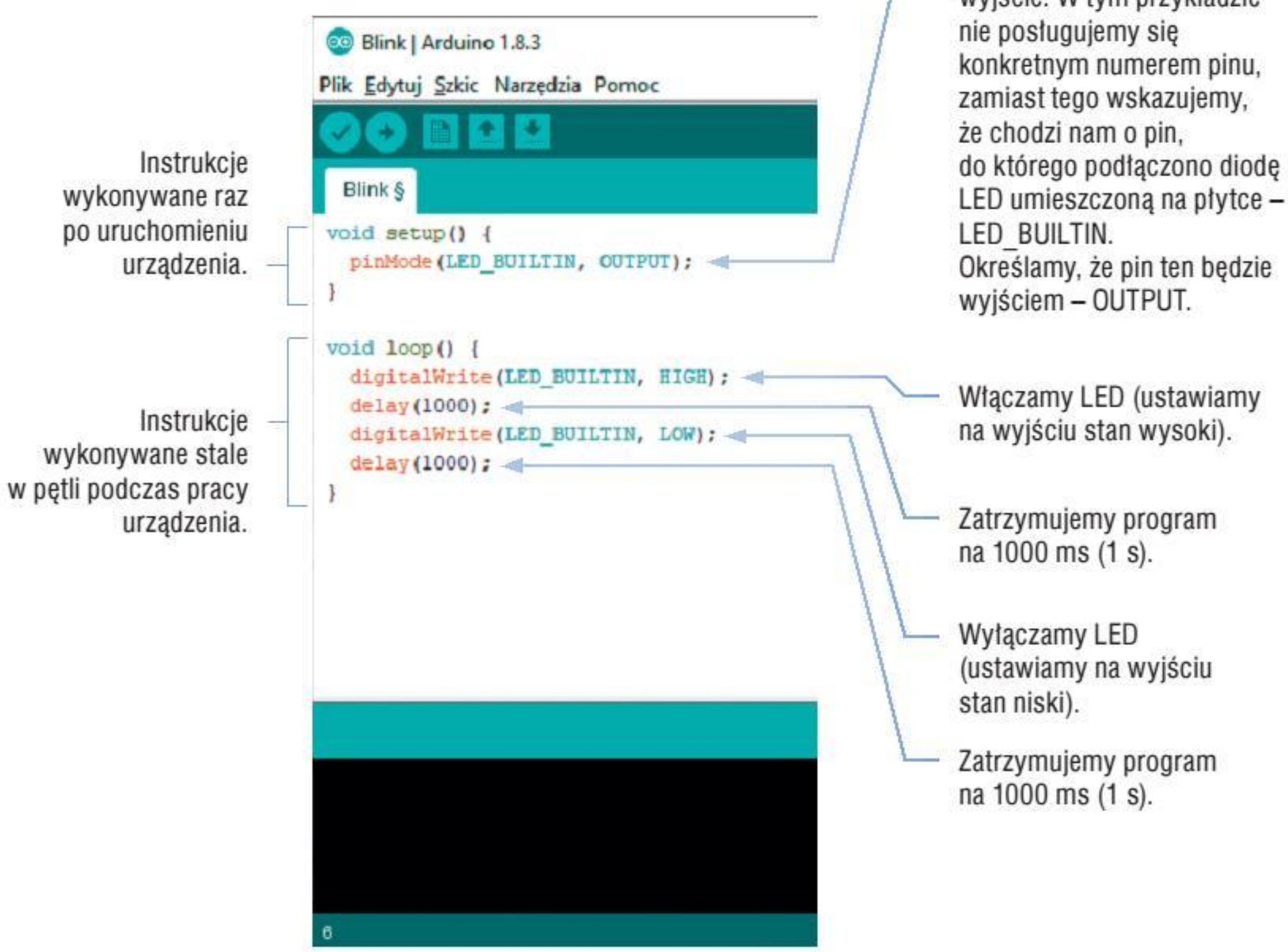
W układach opartych o Arduino często wykorzystuje się gotowe moduły elektroniczne, które pozwalają na budowę złożonych urządzeń z wykorzystaniem jedynie elementarnej wiedzy z zakresu elektroniki (używamy tylko przewodów do łączenia modułów). Początkującym użytkownikom pomogą liczne przykłady rozwiązań dostępne w Internecie.

Typowa płytka Arduino zawiera mikrokontroler, cyfrowe i analogowe linie wejścia/wyjścia oraz interfejs USB do połączenia z komputerem. Komputer służy do programowania kontrolera. Można go też wykorzystać do interakcji z Arduino w czasie działania, choć Arduino po zaprogramowaniu działa jako samodzielny układ.

Programowanie Arduino opiera się o język programowania zbliżony do języka C.

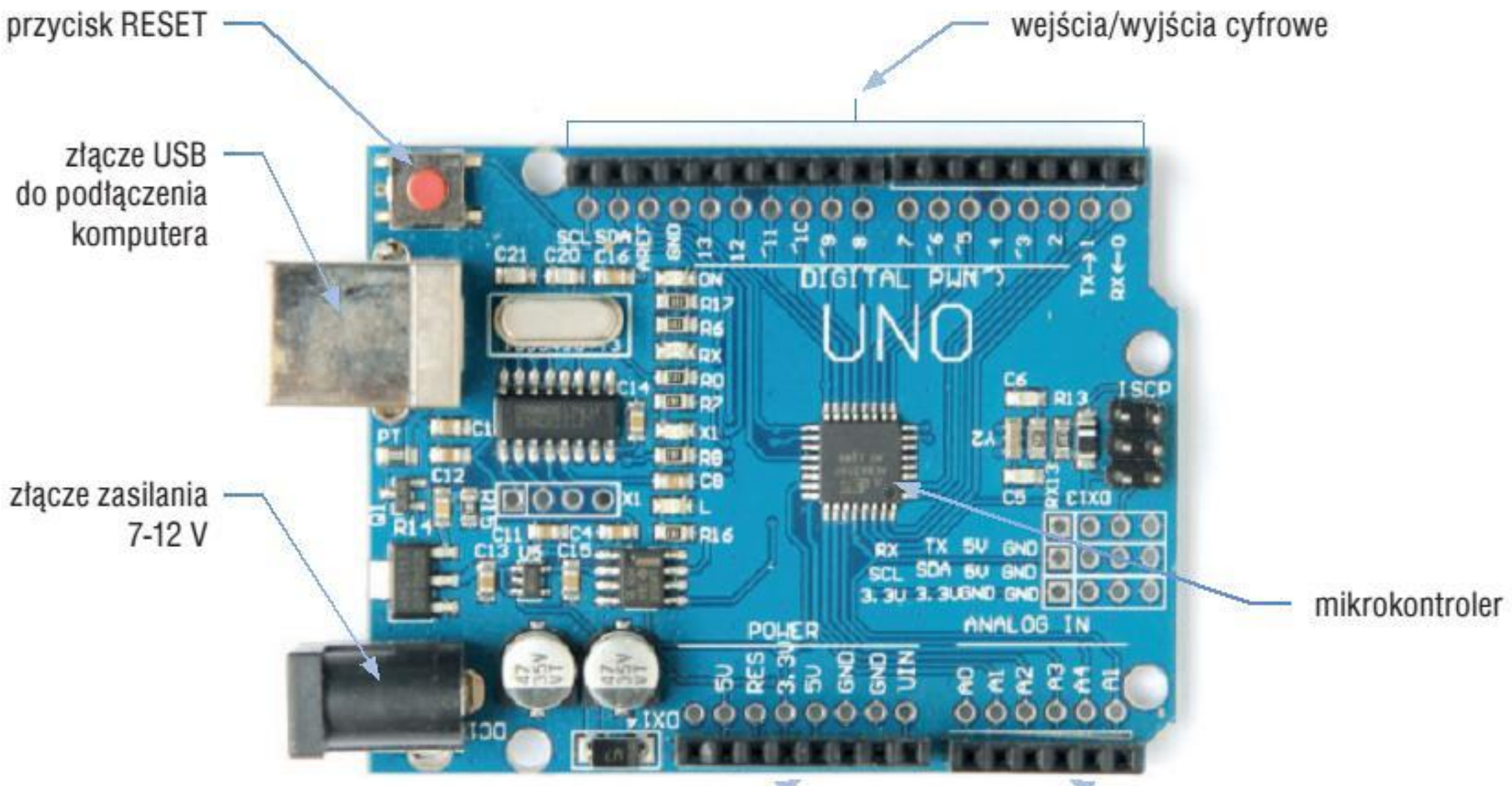
W każdym programie dla Arduino powinny znajdować się dwie funkcje (rys. 2.):

- `setup()` – zawiera polecenia wykonywane na początku programu (po uruchomieniu urządzenia),
- `loop()` – funkcja wykonywana w pętli po wykonaniu funkcji `setup()`.



Rys. 2. *Przykładowy program dla Arduino – mruganie diody umieszczonej na płytce*

W programie pokazanym na rysunku 2. w funkcji `setup()` wywołujemy funkcję `pinMode()` z dwoma parametrami: pierwszy określa numer **pinu**, drugi – sposób działania pinu (wejście lub wyjście układu).



Rys. 3. *Płytka zgodna z Arduino Uno*

2. Projekt sterowania nawadnianiem ogrodu

Celem projektu „Podlewanie ogrodu" jest zaprojektowanie prostego systemu sterowania nawadnianiem ogrodu. Na początku musimy odpowiedzieć na pytanie, jakie akcje chcemy wykonywać i na podstawie jakich parametrów podejmować decyzje o wykonaniu akcji.

Akcja

W przypadku nawadniania ogrodu jedyną akcją, którą musimy podjąć, jest włączenie lub wyłączenie nawadniania (przez włączenie pompy albo otwarcie elektrozaworu). Gdybyśmy nawadniali szklarnię, moglibyśmy jeszcze sterować wentylatorem lub otwieraniem okien w przypadku zbyt wysokiej temperatury.

Warunki wykonania akcji

Kiedy podlewamy rośliny? Oczywiście, gdy jest sucho. Nie jest to jednak jedyny warunek. Jeśli jest silne słońce, powstrzymujemy się od podlewania. Podobnie, gdy właśnie zaczął padać deszcz – wtedy nie podlewamy (nawet gdy ziemia jest sucha), tylko czekamy, aż deszcz wykona pracę za nas.

Aby móc wykonywać rzeczywiste pomiary (m.in. temperatury czy wilgotności gleby), musimy wybrać odpowiednie moduły i elementy, których użyjemy do naszego projektu (pokazano je w tabeli 1.).

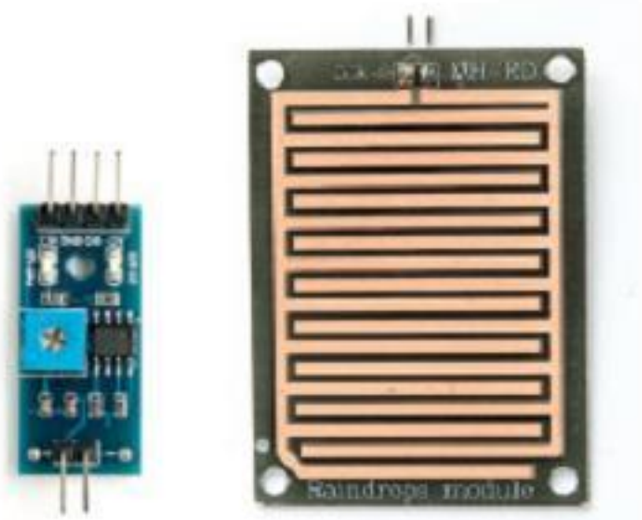
Czujnik deszczu oraz jego sonda	
Czujnik wilgotności gleby oraz jego sonda	
Czujnik światła	
Przekaźnik do sterowania urządzeniami zasilanymi prądem o dużym napięciu i natężeniu – np. lampą, grzałką czy pompą.	
Pompa	

Tabela 1. *Moduły i elementy użyte w projekcie sterowania nawadnianiem ogrodu*

Jeśli chcielibyśmy włączać nawadnianie o określonych godzinach, potrzebny będzie nam także moduł zegara czasu rzeczywistego (RTC – z ang. *Real Time Clock*). W naszym projekcie możemy jednak bazować na czujniku oświetlenia – będziemy podlewać, kiedy zrobi się ciemno.

Do łączenia prototypowych układów służą płytki stykowe (rys. 4.) i przewody zakończone specjalnymi twardymi końcówkami. W płytkach styki są połączone kolumnami, a dwie skrajne linie z każdej strony wierszami – służą one do zasilania. Przewody podłączamy poprzez wbicie ich w odpowiednie pola.

! Uwaga

Dla początkujących użytkowników, chcących spróbować różnych możliwości, powstały gotowe układy zawierające mikrokontroler Arduino i zestaw czujników oraz urządzeń wyjściowych. Można w ten sposób skupić się na programowaniu, nie martwiąc się o połączenia elektryczne.



Rys. 4. *Płytka stykowa - brązowymi liniami zaznaczono wewnętrzne połączenia poszczególnych pól*

Na rysunku 6. pokazano zmontowany układ nawadniania ogrodu połączony za pomocą płytki stykowej. Zastosowano czujnik wilgotności gleby, czujnik opadów oraz czujnik natężenia oświetlenia (został umieszczony na płytce stykowej). Podłączono też przekaźnik (urządzenie wyjściowe), który będzie włączał pompę. Zanim jednak zastosujemy ten układ, warto przygotować jego schemat (rys. 5.).



pompa

przekaźnik

mikrokontroler

zasilanie pompy

zasilanie mikrokontrolera (przez złącze USB lub przez złącze zasilania)

sonda czujnika deszczu

czujnik deszczu

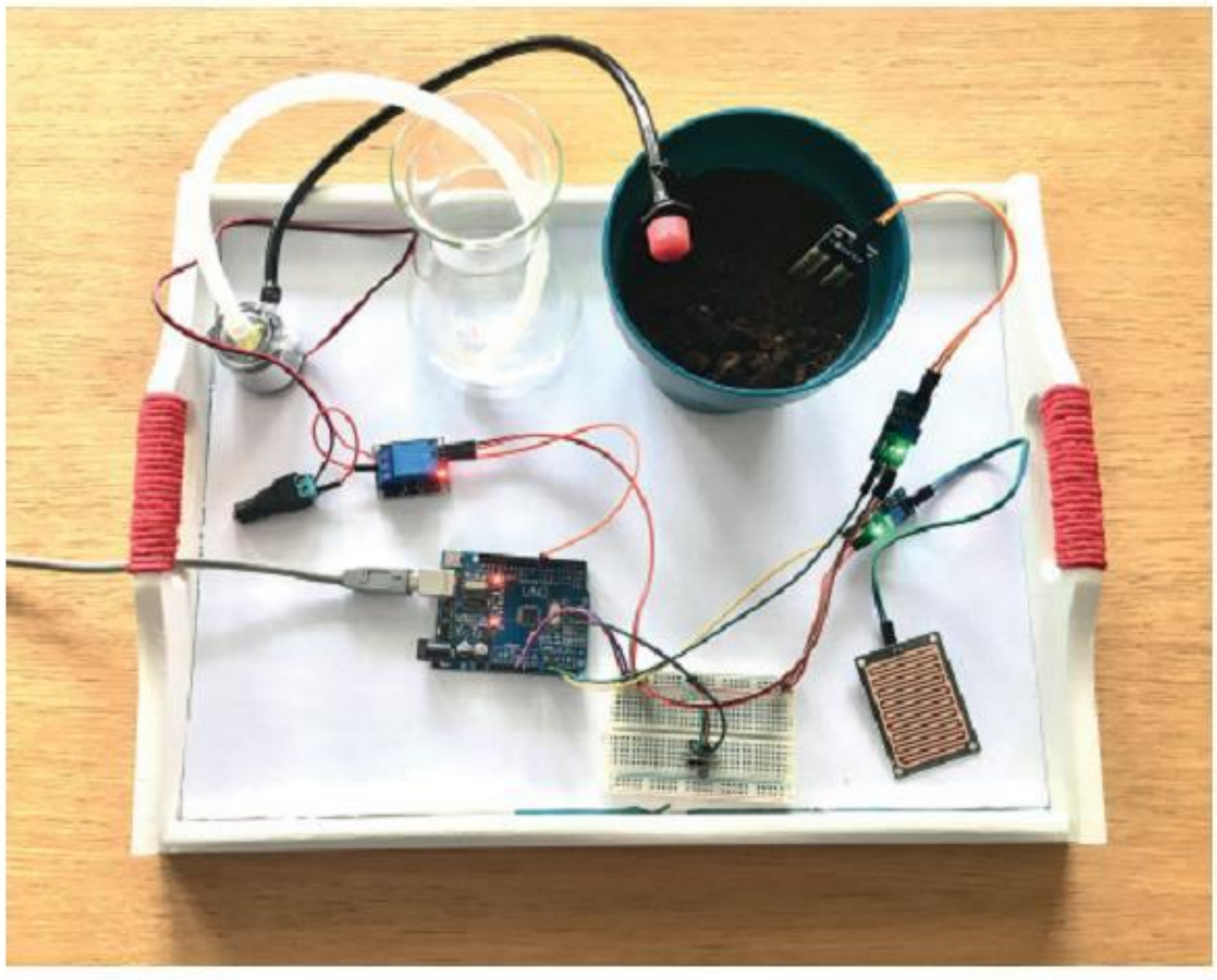
czujnik wilgotności gleby

czujnik światła (fotorezystor)

sonda czujnika wilgotności gleby

Kolorowymi liniami zaznaczono połączenia modułów (przewody). Czarne i czerwone linie to zasilanie. Aby rozdzielić zasilanie z Arduino na poszczególne moduły, możemy posłużyć się płytką stykową. Przewody sygnałowe z poszczególnych modułów możemy poprowadzić bezpośrednio do Arduino.

Rys. 5. *Schemat połączeń modułów składających się na system sterowania nawadnianiem ogrodu*

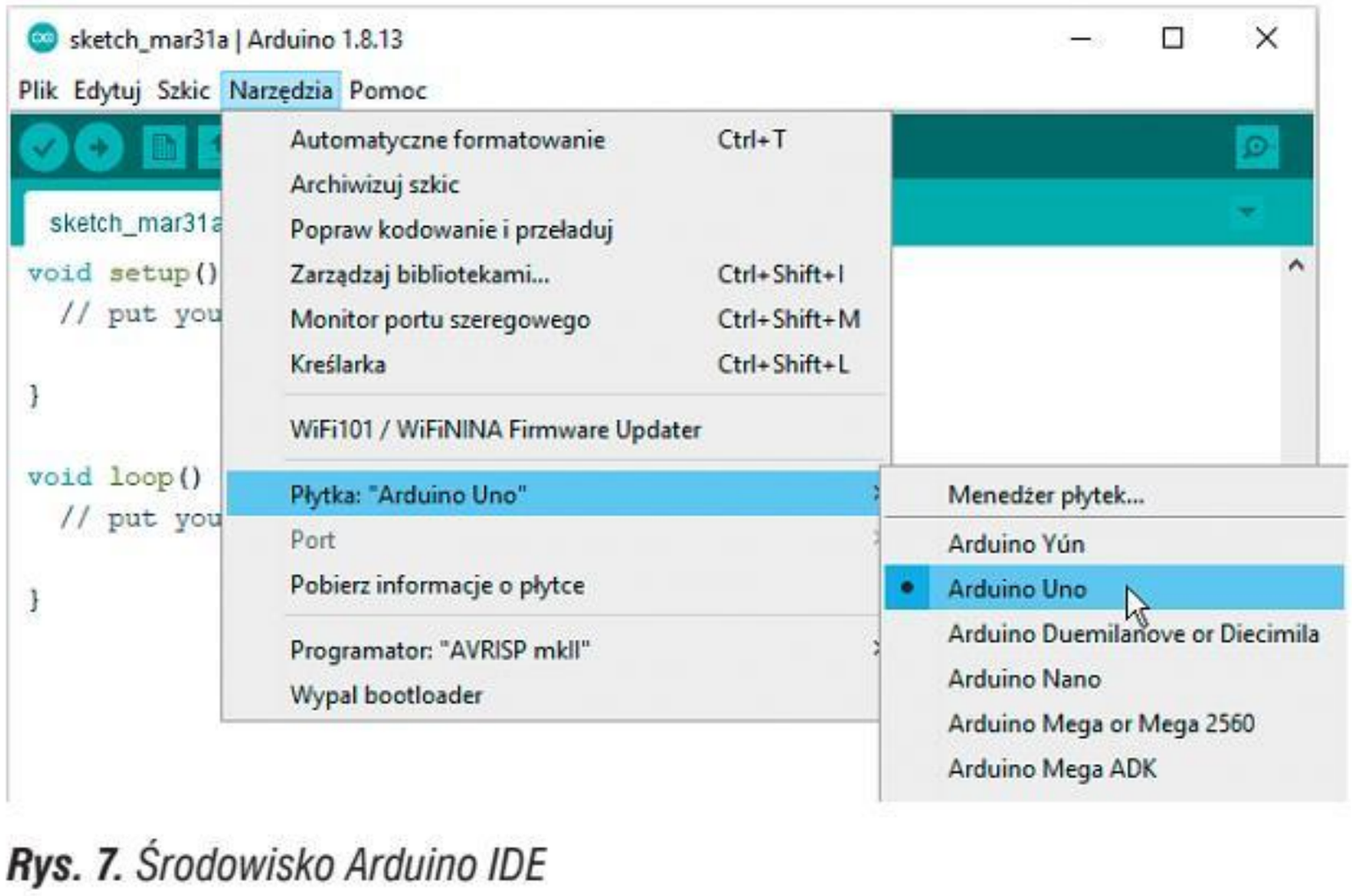


Rys. 6. *Prototyp układu nawadniania ogrodu*

Uwaga

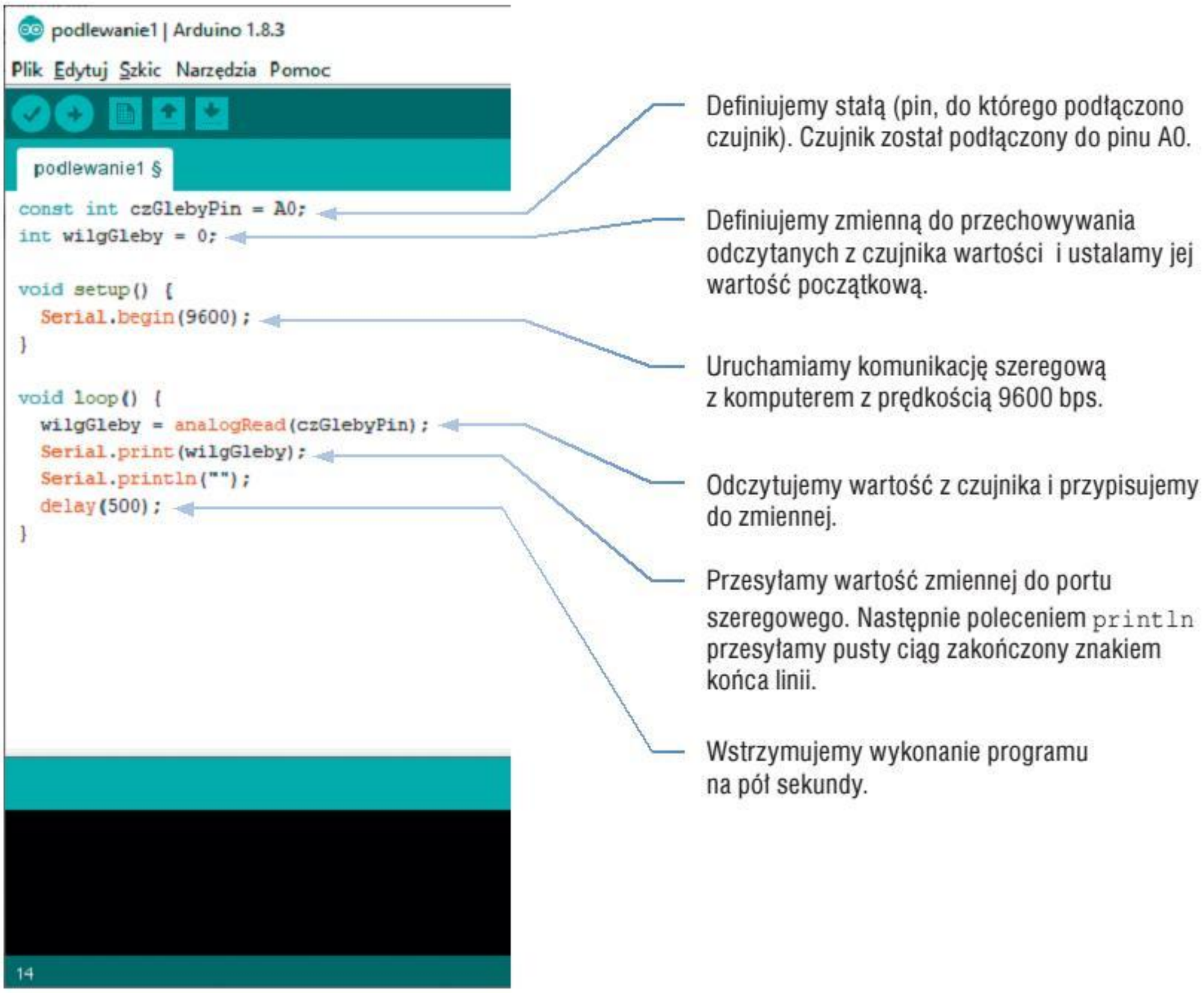
Do celów testowych można zamiast przekaźnika i pompy zastosować diodę LED, sygnalizującą, że należy włączyć pompę. W tym czasie można wodą z konewki nawadniać doniczkę, w której umieszczono czujnik wilgotności gleby (rys. 6.) lub wkładać czujnik do szklanki z wodą.

W środowisku Arduino IDE musimy wybrać płytkę (Arduino Uno) oraz (w tym samym menu) port, do którego jest podłączona (prawdopodobnie będzie tylko jeden do wyboru) – rys. 7.



Rys. 7. *Środowisko Arduino IDE*

Napiszemy program, który odczyta wartości z czujnika wilgotności gleby. Przykładowy program łącznie z komentarzami pokazano na rysunku 8.

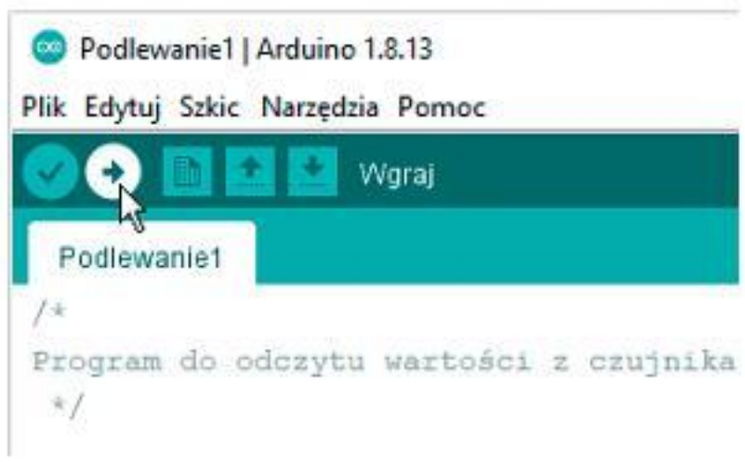


Rys. 8. *Program odczytujący wartości z czujnika wilgotności gleby*

Kolejnym krokiem jest przesłanie programu do płytki z mikrokontrolerem.



Aby przesłać program do płytki, należy kliknąć strzałkę w lewym górnym rogu lub wybrać z menu **Szkic** opcję **Wgraj** (rys. 9.). W dolnej części okna programu powinien po chwili pojawić się komunikat „Ładowanie zakończone".



Rys. 9. *Menu programu z ikoną przesyłania*

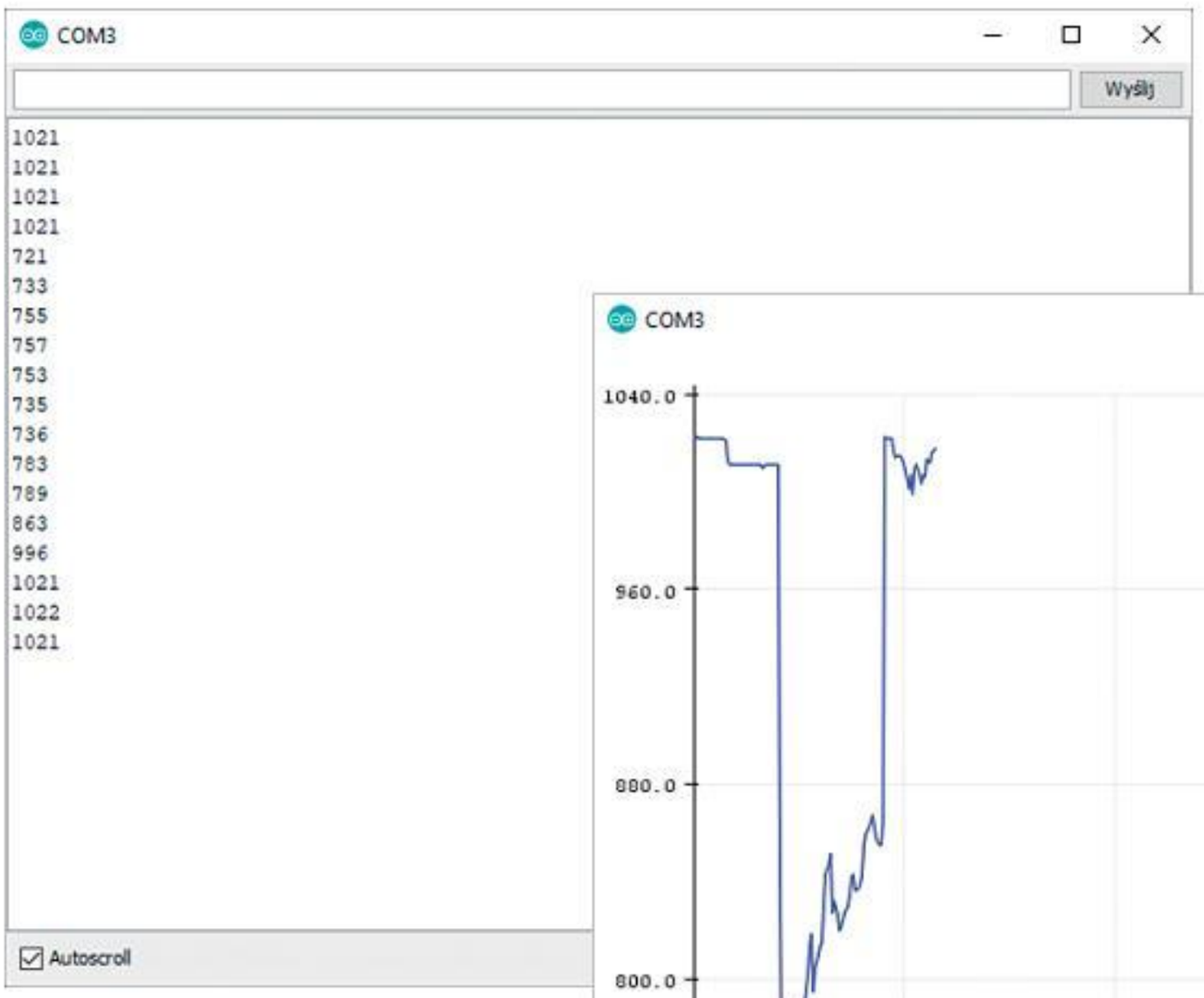
W programie umieściliśmy polecenia odpowiedzialne za komunikację z komputerem i wyświetlanie wyników pomiarów. Aby je zobaczyć, należy wybrać z menu **Narzędzia** opcję **Monitor portu szeregowego**.

Należy pamiętać, aby w prawym dolnym rogu nowo otwartego okna (rys. 10b) wybrać szybkość, którą określiliśmy w naszym programie (9600 bps).

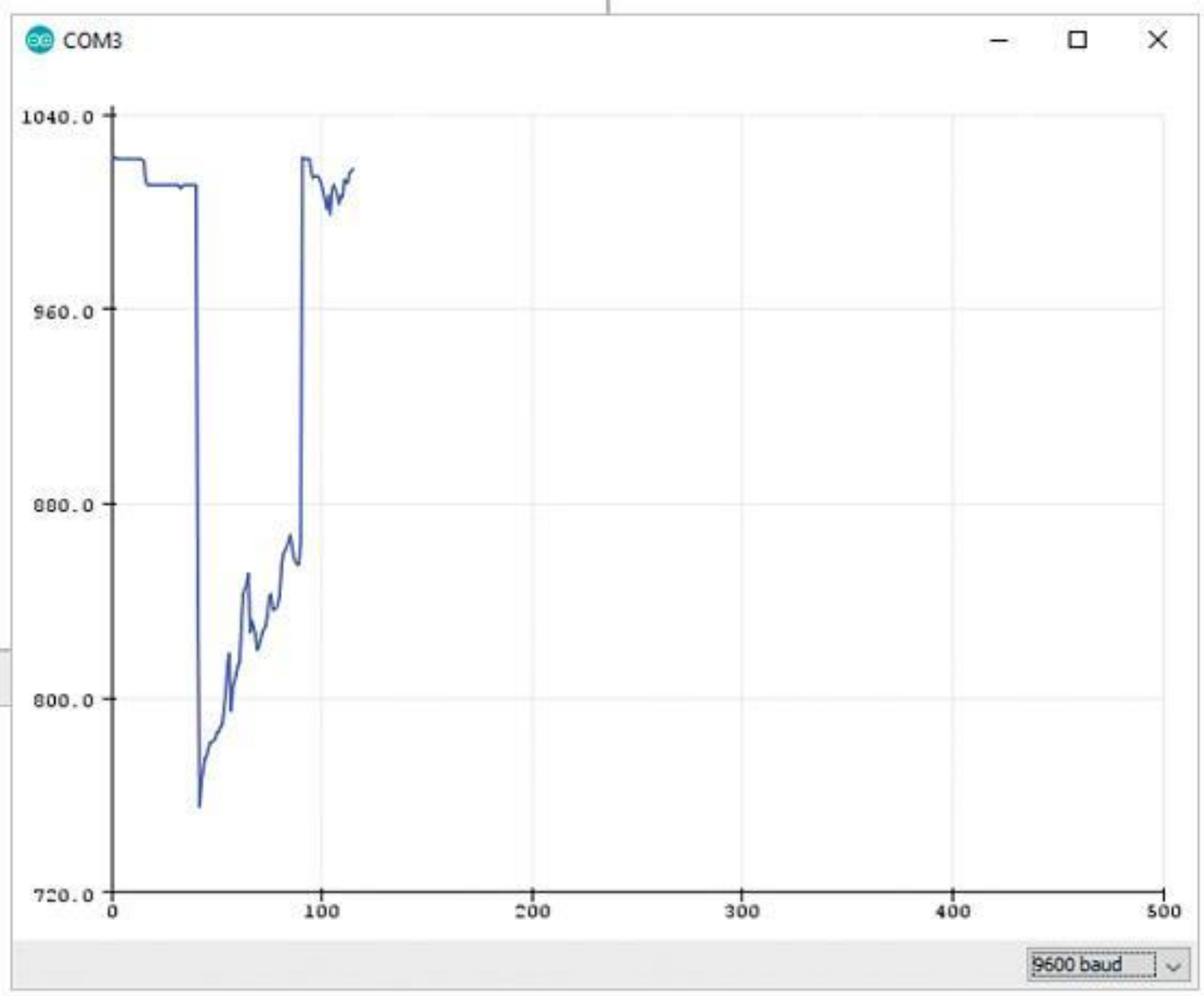
Zgodnie z oczekiwaniami, odczytane wartości pojawiają się w kolejnych wierszach (rys. 10a). Możemy te dane zobrazować na wykresie, aby łatwiej obserwować zmiany (rys. 10b). Należy w tym celu zamknąć monitor portu szeregowego i z menu **Narzędzia** wybrać opcję **Kreślarka**.

! Uwaga

Aby obserwować zmiany wartości z czujnika wilgotności gleby, możemy zanurzać stopniowo czujnik w wodzie lub dotykać jednocześnie obu styków zwilżonymi palcami.



Rys. 10a. *Monitor portu szeregowego*



Rys. 10b. *Kreślarka – obrazowanie danych z monitora portu szeregowego na wykresie*

Zauważmy, że przy wzroście wilgotności odczytane wartości maleją – odczytujemy, jak bardzo ziemia jest sucha, a nie jak bardzo jest mokra – należy o tym pamiętać, tworząc warunek uruchamiający nawadnianie.

Możemy teraz dodać do naszego programu instrukcje odpowiedzialne za sterowanie wyjściem, czyli uruchamianie pompy.

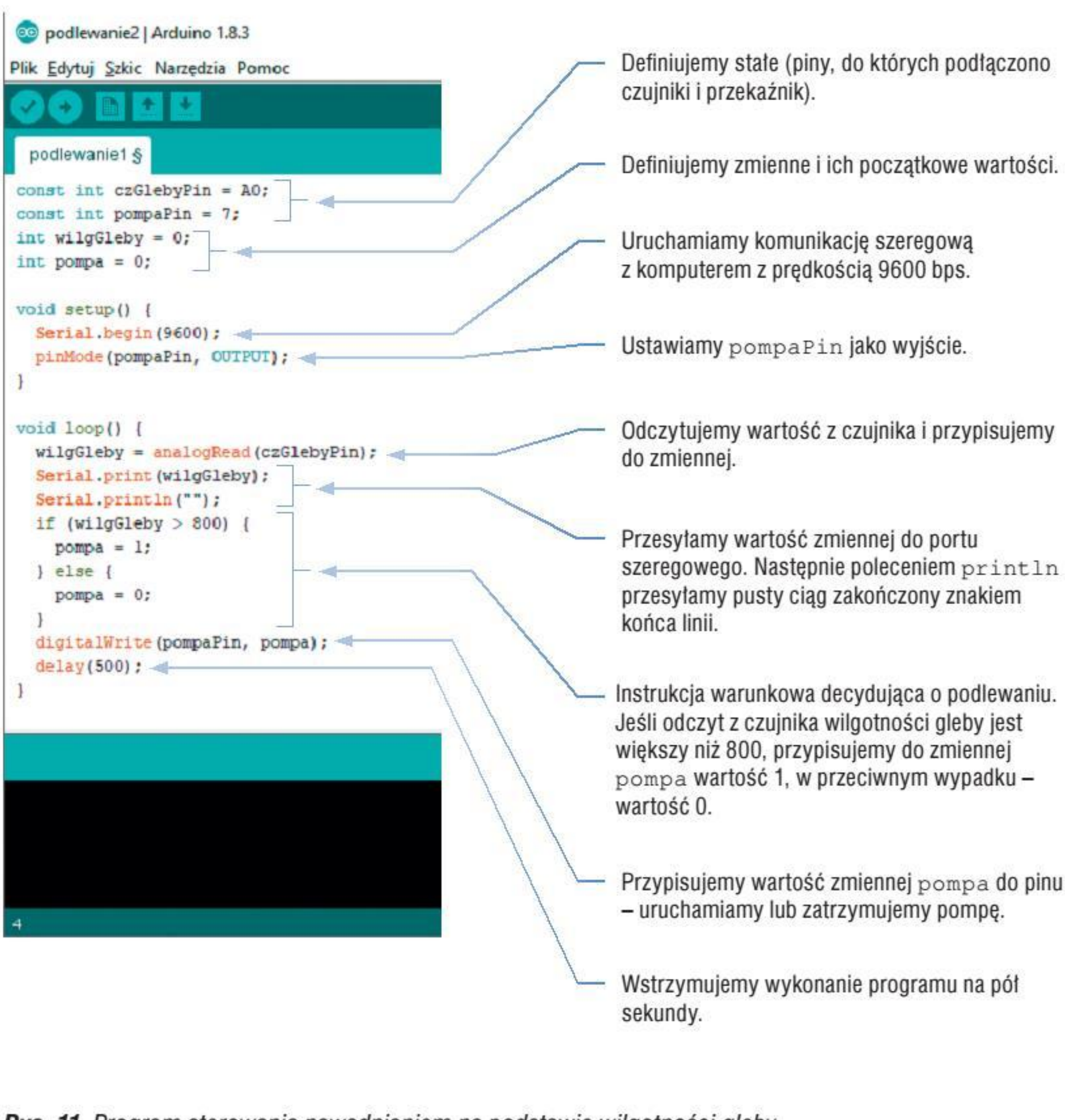
Musimy dodać następujące linie programu:

```
const int pompaPin = 7;
int pompa = 0;
```

i w funkcji `setup()`:

```
pinMode(pompaPin, OUTPUT);
```

Możemy teraz dodać do programu instrukcję warunkową sterującą pompą (zgodnie ze schematem blokowym pokazanym na rysunku 1.) – podlewanie powinno zacząć działać.



***Rys. 11**. Program sterowania nawadnianiem na podstawie wilgotności gleby*

3. Zadania projektowe

Zadanie 1. Wykonanie systemu „Podlewanie ogrodu"

Celem zadania projektowego jest wykonanie prostego systemu sterującego nawadnianiem ogrodu „Podlewanie ogrodu", omówionego w punkcie 2. tego tematu.

Zadania szczegółowe

1. Zapoznaj się z opisem systemu i przygotuj odpowiednie pomoce.
2. Wykonaj przykładowy program – mruganie wbudowanej diody świecącej (rys 2.).
3. Wykonaj układ według wskazówek podanych w punkcie 2. tematu.
4. Przetestuj system (testowanie możesz przeprowadzić w podobny sposób, jak pokazano na rysunku 6. i opisano w uwadze w ramce).

Wskazówka dotycząca realizacji projektu

Zaprogramowany układ można odłączyć od komputera i zasilać z powerbanku przez wejście USB lub z baterii albo zasilacza przez złącze zasilania.



Pamiętaj, aby korzystać tylko z bezpiecznych napięć!
Jeśli system będzie działał prawidłowo i zechcesz podłączyć go do pompy ogrodowej, pamiętaj, że może to zrobić wyłącznie elektryk z odpowiednimi uprawnieniami.

Zadanie 2. Rozszerzenie systemu „Podlewanie ogrodu"

Celem drugiego zadania projektowego jest rozszerzenie systemu „Podlewanie ogrodu" wykonanego w zadaniu 1.

Zadania szczegółowe

1. W systemie „Podlewanie ogrodu" utworzonym w zadaniu 1. czujnik wilgotności gleby jest zasilany cały czas. Zmień połączenia, aby zasilić go z jednego z wyjść mikrokontrolera. Zaprogramuj to wyjście w taki sposób, aby czujnik był zasilany tylko na czas pomiaru – pomoże to uniknąć szybkiej korozji czujnika.
2. Dodaj do projektu czujnik nasłonecznienia (fotorezystor) oraz czujnik deszczu. Sprawdź odczyty z tych modułów i dostosuj algorytm – podlewamy tylko w nocy i jeśli nie pada.
3. Dodaj dodatkowy czujnik wilgotności gleby. Podlewanie powinno być uruchamiane, gdy którykolwiek z czujników wykryje zbyt niską wilgotność.
4. Dodaj do projektu drugi obwód pompy. Rozdziel algorytm tak, aby każdy z czujników wilgotności gleby sterował osobną pompą (czujnik światła i deszczu jest wspólny).
5. Przetestuj system.

Zadanie 3. System „Zamknij okno" w pokoju

W temacie B4 utworzyliśmy projekt własnego pokoju. Doświadczenia nabyte podczas przygotowania projektu systemu sterowania nawadnianiem ogrodu z zastosowaniem układu Arduino możemy wykorzystać do zautomatyzowania niektórych czynności w pokoju.

Celem projektu jest utworzenie systemu „Zamknij okno", sygnalizującego za pomocą diody potrzebę zamknięcia okna zależnie od temperatury.

W projekcie „Podlewanie ogrodu" poznaliśmy już odczyt wartości sygnałów analogowych. Teraz wykorzystamy te umiejętności do pomiaru temperatury w pokoju. Zastosujemy czujniki analogowe LM35, które na wyjściu podają napięcie zależne od temperatury. Do sygnalizacji potrzeby zamknięcia okna można wykorzystać diodę LED.

Zadania szczegółowe

1. Utwórz system sygnalizacji „Zamknij okno". Zastosuj w projekcie dwa czujniki temperatury – jeden wewnątrz pomieszczenia, drugi za oknem. W zależności od temperatury sygnalizuj, czy okno powinno zostać zamknięte. Potrzebę zamknięcia okna sygnalizuj:
   a. przy temperaturze wewnętrznej poniżej 20 stopni i temperaturze zewnętrznej niższej niż wewnętrzna (ochrona przed wychłodzeniem pomieszczenia),
   b. przy temperaturze wewnętrznej powyżej 24 stopni i temperaturze zewnętrznej wyższej niż wewnętrzna (ochrona przed przegrzaniem pomieszczenia).
2. Przetestuj system.

Wskazówki dotyczące realizacji projektu

- Aby odczytać temperaturę, musimy najpierw wyliczyć napięcie otrzymane na wejściu Arduino: `float napiecie = odczyt * (5000 / 1024.0);`
  gdzie 5000 to napięcie odniesienia w miliwoltach (maksymalne jakie można podać na wejście), a 1024 to rozdzielczość (tyle poziomów rozpoznaje Arduino – odczytana wartość 0 to 0 V, a 1023 to 5 V).
- Następnie napięcie musimy przeliczyć na temperaturę:
  `float temperatura = napiecie / 10;`

```
int odczyt = analogRead(sensorPin);
float napiecie = odczyt * (5000 / 1024.0);
float temperatura = napiecie / 10;
```

Zadanie 4.* System „Wycisz radio" po otwarciu drzwi do pokoju

Dla zainteresowanych

Celem czwartego zadania jest zbudowanie urządzenia wyciszającego radio po otwarciu drzwi do pokoju.

Radioodbiornik musi być wyposażony w pilota zdalnego sterowania. Na pilocie powinien znajdować się przycisk „mute" do szybkiego wyciszenia dźwięku.

Zadania szczegółowe

Utwórz system wyciszający radio po otwarciu drzwi do pokoju.

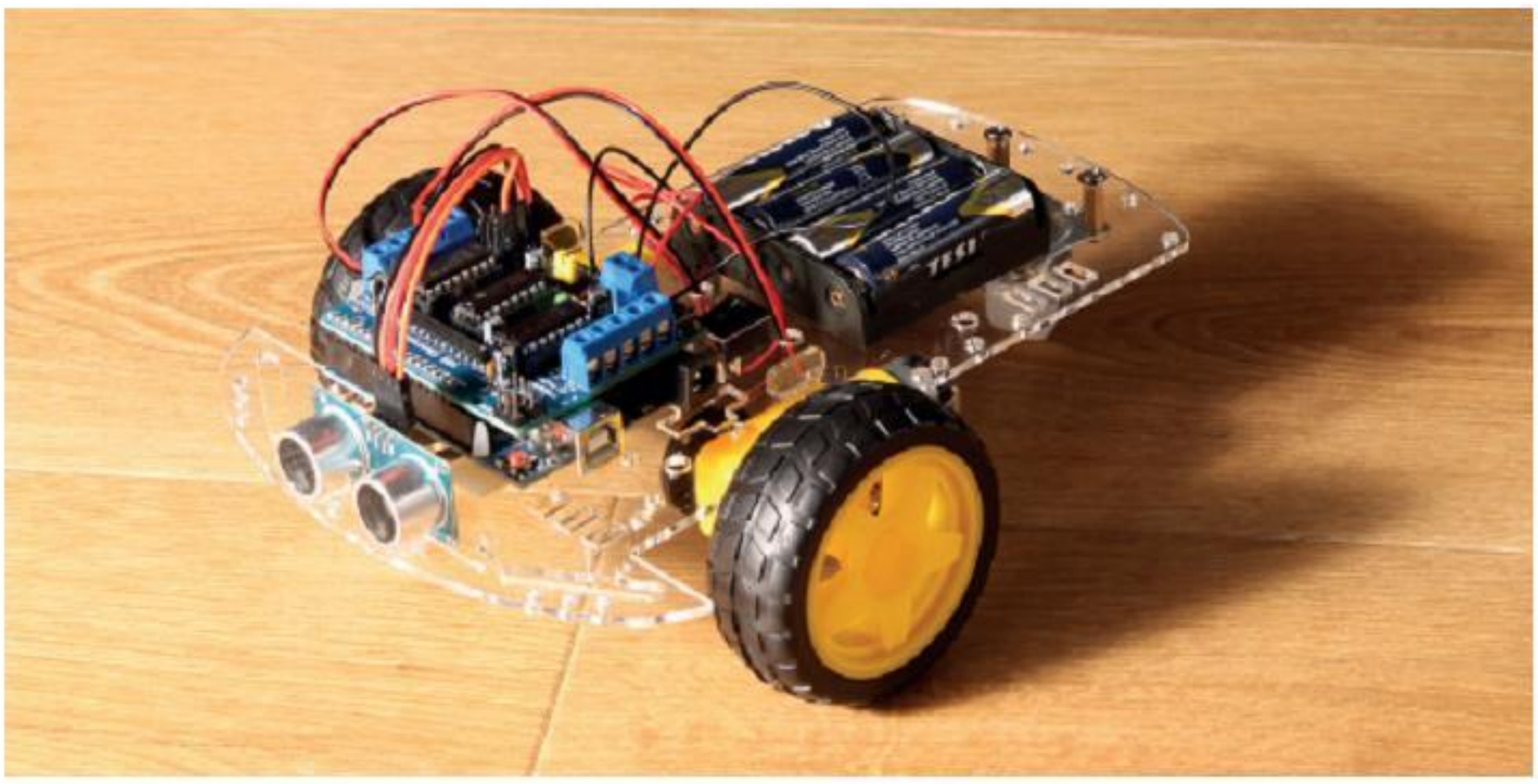
1. Zbuduj urządzenie wykorzystując płytkę Arduino, moduł odbiornika podczerwieni (np. VS1838B), diodę LED emitującą światło podczerwone (IR) oraz czujnik otwarcia drzwi (np. kontaktron). Przy tworzeniu programu możesz wspierać się przykładami dołączonymi do biblioteki IRremote.
2. Napisz program, który wyświetla w **Monitorze portu szeregowego** kody odebrane przez odbiornik podczerwieni. Znajdź kod, który jest przypisany do przycisku „mute" na pilocie od swojego radioodbiornika.
3. Napisz program, który po otwarciu drzwi (przy zastosowaniu kontaktronu będzie to rozwarcie styków czujnika) wyemituje przy pomocy diody LED IR kod przypisany do przycisku „mute".
4. Przetestuj system..

Wskazówki dotyczące realizacji projektu

- Do celów testowych czujnik otwierania drzwi można stymulować, zwierając i rozwierając styki na płytce stykowej.
- W Arduino IDE będziemy musieli skorzystać z biblioteki IRremote (**Szkic/Dołącz bibliotekę**). Możesz skorzystać z przykładowych programów (**Plik/Przykłady/IRremote**), aby przetestować odbieranie i wysyłanie sygnałów.
- Sygnał IR wysyłamy do radia tylko w momencie otwarcia drzwi, a nie zawsze, gdy są otwarte. Musimy więc w zmiennej przechowywać poprzedni stan drzwi i sygnał wysyłać tylko w momencie, gdy stan drzwi zmienia się z zamkniętych na otwarte.

Zadanie 5. Budowanie robota – zabawki

Celem zadania projektowego jest zbudowanie robota-zabawki poruszającego się na kołach i omijającego przeszkody.



***Rys. 12**. Przykładowy zmontowany robot-zabawka wykorzystujący układ Arduino*

Zadania szczegółowe

1. Zaplanuj i przygotuj odpowiednie pomoce.
2. Zmontuj układ, podłączając silniki do wybranych wyjść (np. M3 i M4).
3. Korzystając z wyprowadzeń wejść analogowych na nakładce ze sterownikiem silników, podłącz czujnik odległości.
4. Napisz program mierzący odległość przedmiotu od czujnika. Utwórz funkcję dokonującą pomiaru i zapisującą zmierzoną w centymetrach odległość do zmiennej (przedmiotu).
5. Zainstaluj bibliotekę AFMotor (z ang. *Adafruit Motor Shield Library*).
6. Korzystając ze zdobytych doświadczeń, napisz program, w którym, przy braku przeszkód przed robotem, oba silniki kół będą się kręcić do przodu. W przypadku przeszkody w bliskiej odległości, jedno koło powinno się zatrzymać, a drugie kręcić do tyłu. Możesz dodać instrukcję `delay(500);`, aby ruch robota do tyłu połączony z obracaniem trwał przynajmniej pół sekundy.
7. Przetestuj system.

Zadania dla zainteresowanych

8. Dodaj do robota odbiciowy czujnik podczerwieni skierowany na podłogę i zmień program, aby przy braku odbicia (podniesienie robota lub niebezpieczeństwo upadku ze schodów) robot zatrzymywał się.
9. Dodaj czujniki odległości skierowane lekko na boki, aby bardziej precyzyjnie identyfikować położenie przeszkody i lepiej podejmować decyzje o skręcaniu. Możesz też pozostać przy jednym czujniku, ale obracać go za pomocą serwomechanizmu.

Wskazówki dotyczące realizacji projektu

- Do wykonania zadania będziesz potrzebować: płytkę zgodną z Arduino UNO, podwozie z dwoma silnikami, sterownik silników Motor Shield L293D, ultradźwiękowy czujnik odległości.
- Pin czujnika odległości oznaczony Trig podłącz do wejścia A0, natomiast pin oznaczony Echo – do wejścia A1. Pomiar odległości polega na wysłaniu impulsu ultradźwiękowego i zmierzeniu czasu, po jakim czujnik usłyszy jego odbicie (echo) – dlatego musimy użyć dwóch wejść mikrokontrolera (jednego użyjemy jako wyjście, drugiego jako wejście).
- Możesz poszukać przykładowych programów rozwiązujących podobne problemy w Internecie.
- Aby zainstalować bibliotekę AFMotor, wybierz opcję **Szkic/Dołącz bibliotekę/Zarządzaj bibliotekami**. Razem z biblioteką zainstalują się przykładowe programy. Otwórz przykład MotorTest i w instrukcji `AF_DCMotor motor(4);` wstaw numer swojego silnika, aby przetestować jego działanie.

Aby sterować dwoma silnikami, musisz zadeklarować je osobno, np.:

```
SilnikLewy motor(3);
SilnikPrawy motor(4);
```

Jeśli nie masz zamiaru zmieniać prędkości silników, możesz zadeklarować ich prędkość tylko raz na początku w funkcji `setup()`.

Temat D2. Więcej na temat tworzenia stron internetowych – projekt

1. Strony internetowe – usystematyzowanie wiadomości
   1.1. Blogi
   1.2. Systemy zarządzania treścią
   1.3. Szablony do tworzenia stron
2. Przygotowanie projektu
3. Etapy tworzenia strony internetowej
   3.1. Projekt graficzny strony
   3.2. Wdrożenie – realizacja projektu
   3.3. Testowanie
4. Zadania projektowe
   Zadanie 1. Nowy wygląd strony internetowej o Wyspach Kanaryjskich
   Zadanie 2. Tworzenie strony internetowej wzbogaconej o dodatkowe elementy

Warto powtórzyć

1. Jaka jest ogólna struktura pliku w języku HTML?
2. Do czego służą arkusze stylów CSS?
3. Jakie są sposoby dołączania arkuszy stylów CSS do dokumentów HTML?

1. Strony internetowe – usystematyzowanie wiadomości

Do stworzenia prostej witryny wystarczą: edytor tekstu, edytor grafiki oraz serwer WWW i program umożliwiający przesyłanie na niego plików (klient FTP).

Rozbudowane witryny tworzy się zwykle za pomocą specjalistycznych narzędzi. Witryny takie korzystają również z baz danych i języków skryptowych. Na podstawie projektu witryny przygotowuje się szablony poszczególnych podstron, które następnie wypełnia się tekstami. W końcowej fazie sprawdza się poprawność tekstów i funkcjonowanie strony, w szczególności na różnych urządzeniach i w różnych przeglądarkach internetowych.

1.1. Blogi

Sieciowy dziennik (ang. *weblog*, w skrócie blog) może założyć każdy: osoby młodsze i starsze, znane i nieznane, m.in. politycy, sportowcy, artyści, uczniowie, studenci. Autorzy blogów, zwani blogerami, opisują swoje życie codzienne, komentują zdarzenia z otaczającego świata, wypowiadają się np. na temat kultury, oświaty, polityki, gospodarki, sportu. Poza tekstem blogi mogą zawierać rysunki, fotografie, filmy, linki do innych stron.

Blogi zawierają poglądy autorów na różne tematy, często o subiektywnym, emocjonalnym zabarwieniu. Nierzadko blogerzy piszą również o osobistych przeżyciach. Właśnie tym prywatnym charakterem blog odróżnia się od innych stron internetowych.

W Internecie spotykamy również blogi tematyczne, poświęcone określonej dziedzinie, często zakładane przez specjalistów, np. z zakresu medycyny, prawa, oświaty. Takie blogi mają zazwyczaj wysoki poziom merytoryczny. Blogi mogą mieć jednego autora lub wielu. Ludzie piszący blogi tworzą społeczność, zwaną **blogosferą**.

Ćwiczenie 1. Szukamy w Internecie blogów o wybranej tematyce

Odszukaj w Internecie kilka blogów tematycznych związanych z twoim hobby.

Blog jest rodzajem strony internetowej. Podstawową zaletą takiej metody publikowania treści w Internecie jest łatwość korzystania z niej. Do prowadzenia blogu nie jest wymagana znajomość protokołów sieciowych, języków programowania, nie trzeba też mieć odpowiednich programów narzędziowych – wystarczy przeglądarka internetowa, w której, na odpowiedniej stronie WWW, wpisuje się i formatuje teksty notatek.

Jest wiele serwisów oferujących możliwość darmowego prowadzenia blogów. Jedną z najpopularniejszych platform blogowych jest serwis *wordpress.com*. Założenie blogu nie jest skomplikowane – wystarczy wykonać kilka czynności, które opisano na stronie umożliwiającej zakładanie blogów. Należy zwrócić szczególną uwagę na regulamin danego portalu i dokładnie zapoznać się z jego zapisami.

Aby założyć blog, należy:

- wejść na stronę portalu, który umożliwia założenie blogu,
- podać niezbędne dane, m.in.: nazwę, tytuł blogu, login, hasło,
- wybrać kategorię, opisać krótko blog i wybrać szablon,
- zapoznać się z regulaminem portalu i zaakceptować go,
- zatwierdzić wprowadzone dane, naciskając odpowiedni przycisk.

Po potwierdzeniu rejestracji w serwisie możemy wybrać wygląd blogu, korzystając z jednego z wielu dostępnych **szablonów**. Potem pozostaje już tylko wypełnić zawartość blogu treścią.

Z uwagi na fakt, że rośnie liczba użytkowników korzystających na co dzień ze smartfonów, coraz większą popularnością cieszą się również tzw. **mikroblogi**, w których długość notatek jest ograniczona do kilkudziesięciu (maksymalnie kilkuset) znaków. Najpopularniejszym serwisem tego rodzaju jest Twitter.

Forma blogu nie zawsze jest odpowiednia dla treści, które chcemy przedstawić. W przypadku, gdy chcemy stworzyć rozbudowaną witrynę internetową, wyposażoną w mechanizmy interakcji z użytkownikiem, niezbędne jest użycie innych narzędzi.

1.2. Systemy zarządzania treścią

Bardziej rozbudowane niż blogi są systemy zarządzania treścią (CMS – z ang. *Content Management System*). CMS to oprogramowanie umożliwiające tworzenie stron internetowych przez osoby, które nie mają szczegółowej wiedzy z zakresu technologii internetowych. Systemy tego typu pozwalają zarządzać witryną internetową z wykorzystaniem przeglądarki internetowej, bez konieczności poznawania języka HTML.

Wygląd serwisu modyfikuje się za pomocą szablonów (inaczej zwanych motywami), co ułatwia dokonywanie szybkich zmian – wystarczy wybrać inny szablon. Uprawnieni redaktorzy wprowadzają do systemu i edytują informacje (artykuły), a następnie zapisują je w bazie danych. Na podstawie szablonu oraz treści pochodzących z bazy danych system dynamicznie generuje stronę internetową w języku HTML.

Systemy zarządzania treścią to zarówno programy komercyjne, jak i dostępne bezpłatnie, np. Joomla!, Drupal, WordPress – rozpowszechniane na licencji GNU GPL. Niektóre programy mogą być przeznaczone do konkretnych zastosowań, np. MediaWiki – do tworzenia encyklopedii, a WordPress – do tworzenia i obsługi blogów.

Ćwiczenie 2. Szukamy wybranych serwisów internetowych

Korzystając z wyszukiwarki internetowej, znajdź polskie serwisy internetowe wykonane na platformie Joomla!

1.3. Szablony do tworzenia stron

W Internecie są dostępne szablony stron internetowych – zarówno wersje darmowe, jak i odpłatne.

Szablony strony internetowej, podobnie jak szablony dokumentów w edytorze tekstu czy prezentacji w programie PowerPoint, to przygotowane projekty graficzne dostosowane do różnych rodzajów stron, np. prywatnych, firmowych, edukacyjnych.

Szablon jest pewnego rodzaju szkieletem składającym się z zaprojektowanych stron i podstron. Szablony są często tak przygotowane, żeby można je było łatwo dopasować do potrzeb klienta, np. do różnych rodzajów działalności.

Szablony są utworzone w języku HTML. Należy je wypełnić odpowiednimi treściami, aby strona internetowa była kompletna. Stworzenie strony internetowej na podstawie szablonu HTML wymaga przynajmniej podstawowej znajomości języka HTML, dlatego warto znać znaczniki tego języka.

Na stronach internetowych oferujących szablony znajduje się zazwyczaj przewodnik (poradnik) wyjaśniający, w jaki sposób korzystać z szablonu.

2. Przygotowanie projektu

Podczas pracy nad projektami wykorzystamy omówione dotychczas zagadnienia dotyczące tworzenia stron internetowych, m.in.: stosowanie podstawowych znaczników HTML, wstawianie tabel, obrazów, stosowanie list, posługiwanie się arkuszem stylów CSS.

Wskażemy na początek kilka możliwości uatrakcyjnienia strony internetowej, które wykorzystamy do wzbogacenia strony o Wyspach Kanaryjskich (zadanie 1.). Doświadczenia nabyte przy pracy nad tą stroną posłużą nam do wykonania własnej strony internetowej (zadanie 2.).

3. Etapy tworzenia strony internetowej

Tworząc stronę internetową, trzeba wiedzieć, jaka ma być jej zawartość i w jakim celu chcemy ją wykonać. Może to być na przykład strona o tematyce edukacyjnej w celu dzielenia się wiedzą albo sklep internetowy do sprzedaży produktów. To wszystko określa projekt wyglądu strony. Niezależnie od tematyki strony, ważne, żeby strona była funkcjonalna i przyjazna użytkownikowi – w ten sposób więcej osób będzie chciało ją oglądać i chętniej będzie do niej wracać.

Nad rozbudowanymi stronami, które oglądamy na co dzień w Internecie, pracują wieloosobowe zespoły, w których każdy ma inne zadanie do wykonania, np.:

- określenie wymagań (czyli: jakiego typu ma być utworzona strona, w jakim celu powstaje, kto będzie użytkownikiem) oraz przygotowanie na tej podstawie treści;
- projekt graficzny strony internetowej wraz z uwzględnieniem opisów dotyczących kolorów, grubości obramowań, marginesów itp.;
- wdrożenie (realizacja) projektu – zmiana wyglądu strony internetowej (z ang. *front-end*), z wykorzystaniem języka HTML, stylów CSS, języka JavaSript;
- programowanie po stronie serwera (z ang. *back-end*); zaprogramowanie struktury danych strony, algorytmów, przetwarzania danych z formularzy itp.;
- testowanie – sprawdzenie zgodności strony z projektem i poprawności działania strony.

Wymienione zadania może realizować jedna osoba albo wieloosobowy, wyspecjalizowany zespół. Wszystko zależy od tego, jak rozbudowana ma być tworzona strona internetowa.

3.1. Projekt graficzny strony

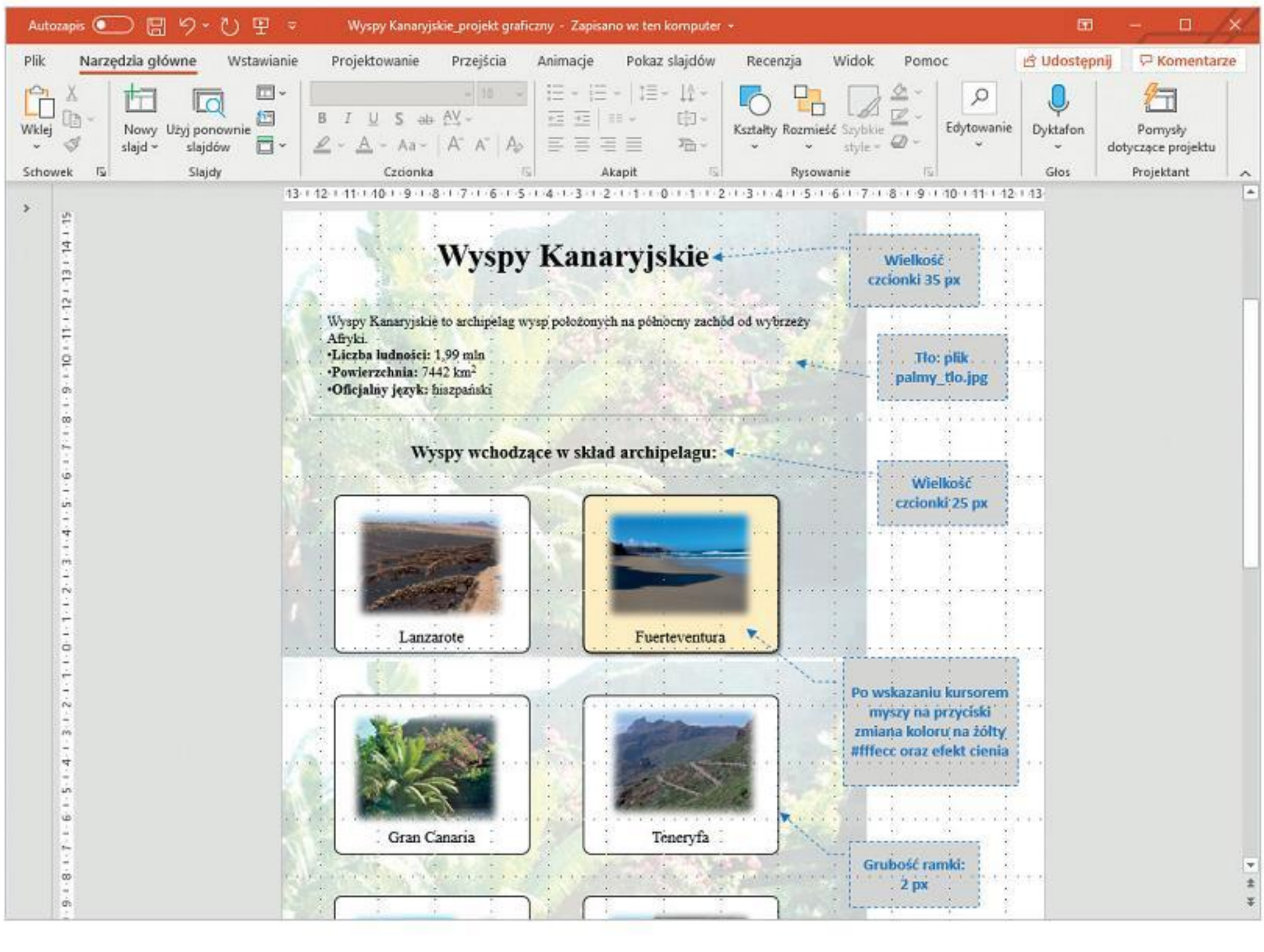
Praca nad projektem graficznym ma kilka etapów. W pierwszym kroku należy przygotować treści, które mają się znaleźć na stronie. Następnie powinien zostać określony plan pracy. Jeżeli projekt jest prosty, ten etap ogranicza się często do rozpisania poszczególnych podstron i określenia terminów wykonania. Na tej podstawie można rozpocząć pracę nad projektem graficznym.

Zadaniem grafika jest zaprojektowanie wyglądu strony na podstawie dostarczonych treści. Grafik określa kolory poszczególnych elementów, wielkości i rodzaje czcionek oraz przygotowuje zdjęcia i inne grafiki (rys. 1.).

Na etapie projektowania ustala się również, które elementy powinny zmieniać wygląd po ich wskazaniu kursorem myszy i kliknięciu.

Projekty graficzne można tworzyć w dowolnym programie graficznym (np. Adobe Photoshop, GIMP czy Paint lub PowerPoint). Profesjonalne projekty powstają często z wykorzystaniem gotowych narzędzi (np. Adobe XD, InVision, Figma) umożliwiających równoczesne projektowanie kilku osobom oraz pozwalających na utworzenie projektu, który od razu podpowiada atrybuty stylów CSS.

Projektując wygląd strony, należy zwrócić uwagę nie tylko na to, czy jest atrakcyjna wizualnie, ale również czy jest funkcjonalna oraz czy angażuje uwagę odbiorcy. W komercyjnych projektach w celu zaprojektowania strony, która jest dobrze odbierana przez użytkowników, zatrudnia się specjalistów: projektantów UX (z ang. – *User Experience Designer*).



Rys. 1. *Fragment przykładowego projektu graficznego strony internetowej. W ramkach umieszczono wskazówki dla programisty.*

3.2. Wdrożenie – realizacja projektu

Przy realizacji prostej strony programista może zacząć pracę, gdy ma strukturę strony oraz treści do zamieszczenia. Na tej podstawie może wprowadzić teksty oraz utworzyć podstrony. Po otrzymaniu projektu graficznego przechodzi do zmiany wyglądu strony, np. za pomocą stylów CSS.

Ustalenie tła strony

Tłem strony może być obraz lub wybrany kolor. Powinno być ono dobrane w taki sposób, aby umożliwiało wyróżnienie głównych elementów strony, np. produktów sprzedawanych w sklepie internetowym.

Jeśli chcemy, żeby tłem był obraz, dodajemy do arkusza stylów:

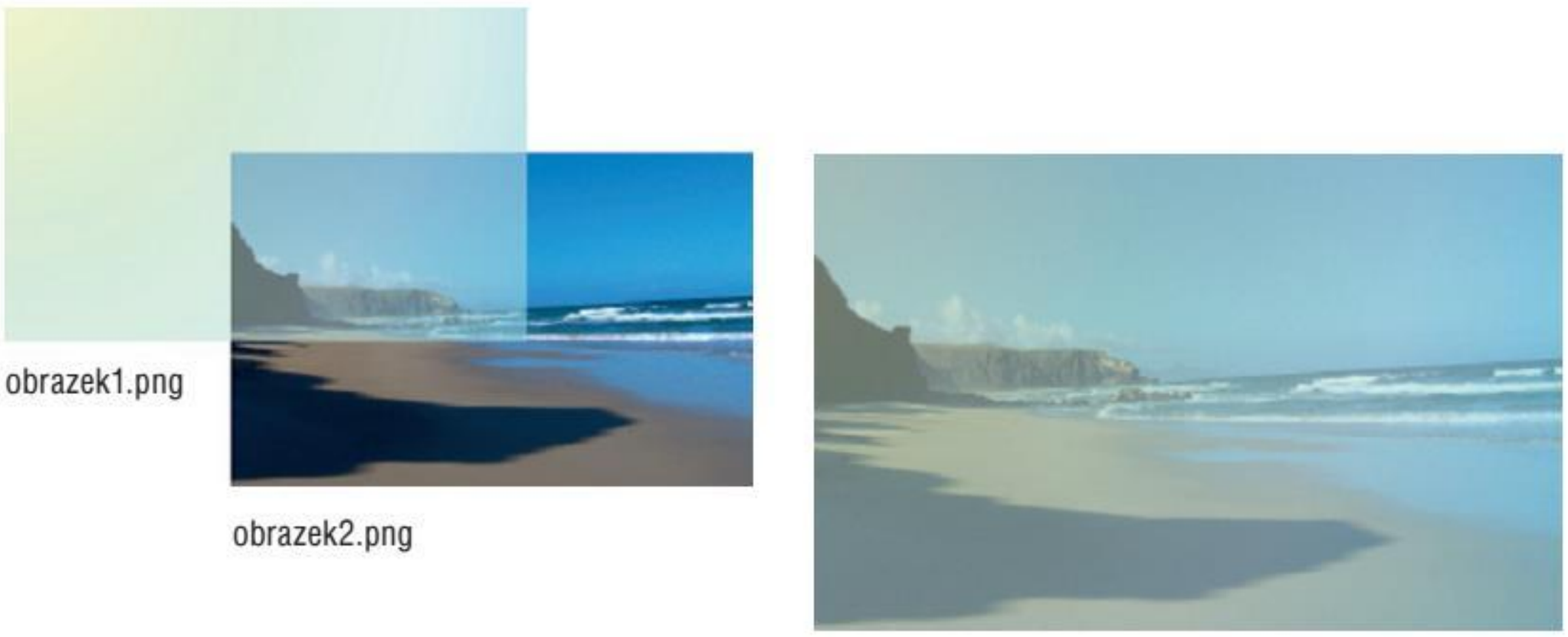
```
body { background-image: url("obrazek.png"); }
```

Plik *obrazek.png* (lub o innej nazwie) programista powinien dostać od grafika.

Można użyć kilku obrazów tła, np.:

```
body { background-image: url("obrazek1.png"), url("obrazek2.png"); }
```

Wtedy jeden obraz nałoży się na drugi. Wykorzystując plik z przezroczystym tłem, można osiągnąć ciekawe efekty (rys. 2.).



obrazek1.png

obrazek2.png

Rys. 2. *Efekt zastosowania dwóch obrazów tła*

Ustawienie listy w wierszach i kolumnach na przykładzie listy uporządkowanej

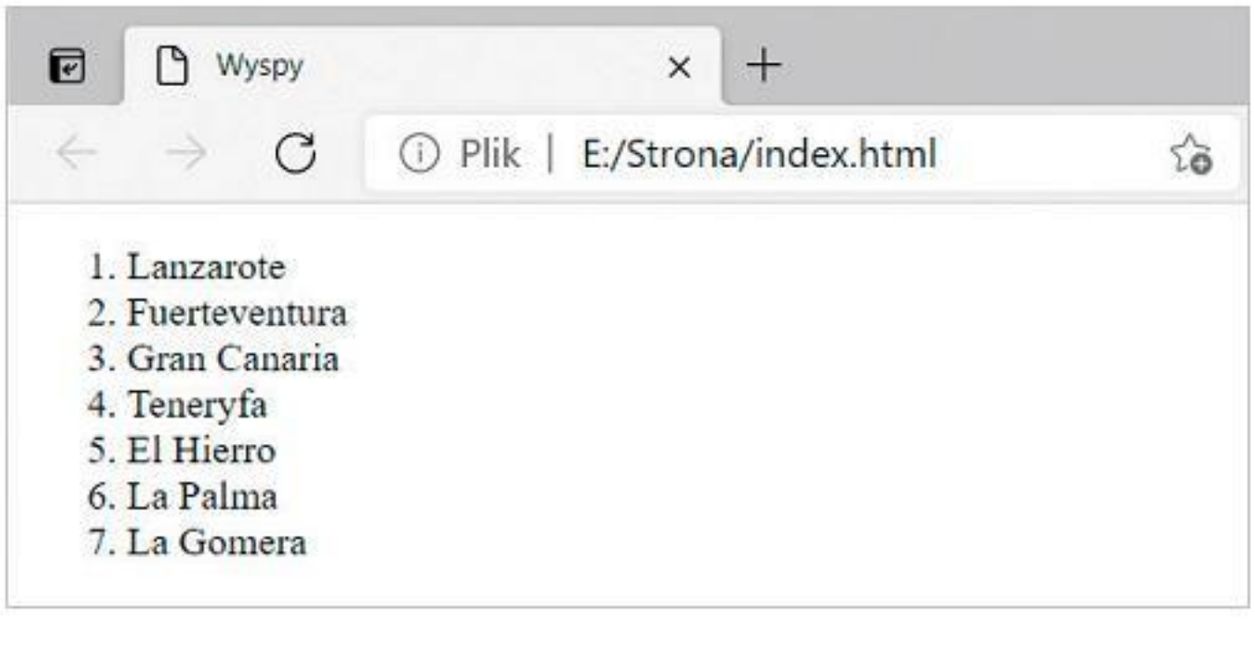
Jednym z ważniejszych elementów strony jest menu, dzięki któremu wiemy, jaka jest jej zawartość i jak się po niej poruszać. Menu może być przedstawione na wiele sposobów. Najczęściej jest to poziomy pasek na górze strony; przy prostych stronach mogą to być przyciski z grafiką (rys. 1.).

Przyciski menu w języku HTML można stworzyć, zmieniając atrybuty stylu listy (uporządkowanej lub nieuporządkowanej), a następnie dodając do jej elementów obramowanie oraz inne efekty. Na przykładzie elementów listy uporządkowanej przedstawimy, jak ustawić je w kilku wierszach i w dwóch kolumnach, aby później zrealizować projekt przedstawiony na rysunku 1.

Krok 1.

Elementy menu można zapisać w postaci listy uporządkowanej (rys. 3.).

```
<ol>
    <li>Lanzarote</li>
    <li>Fuerteventura</li>
    <li>Gran Canaria</li>
    <li>Teneryfa</li>
    <li>El Hierro</li>
    <li>La Palma</li>
    <li>La Gomera</li>
</ol>
```

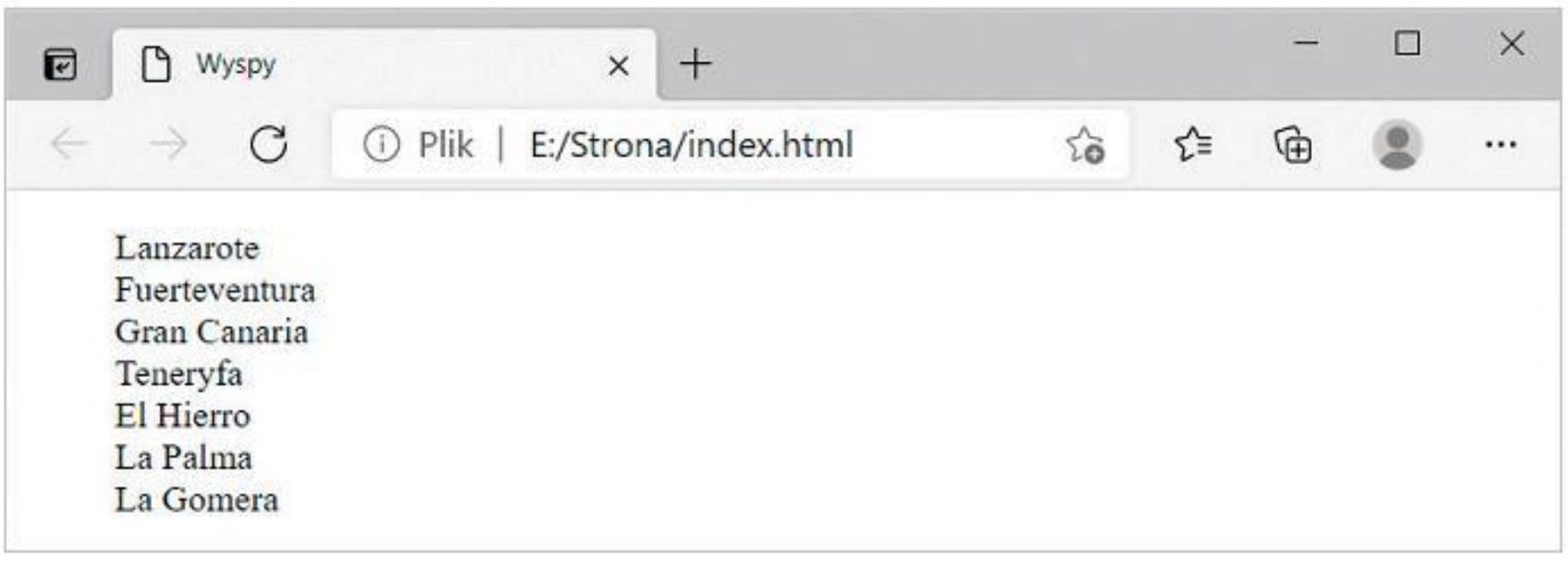


Rys. 3. *Przykład elementów menu w postaci listy uporządkowanej*

Krok 2.

Aby lista uporządkowana nie miała automatycznej numeracji, należy zmienić rodzaj stylu listy (rys. 4.).

```
ol { list-style: none; }
```

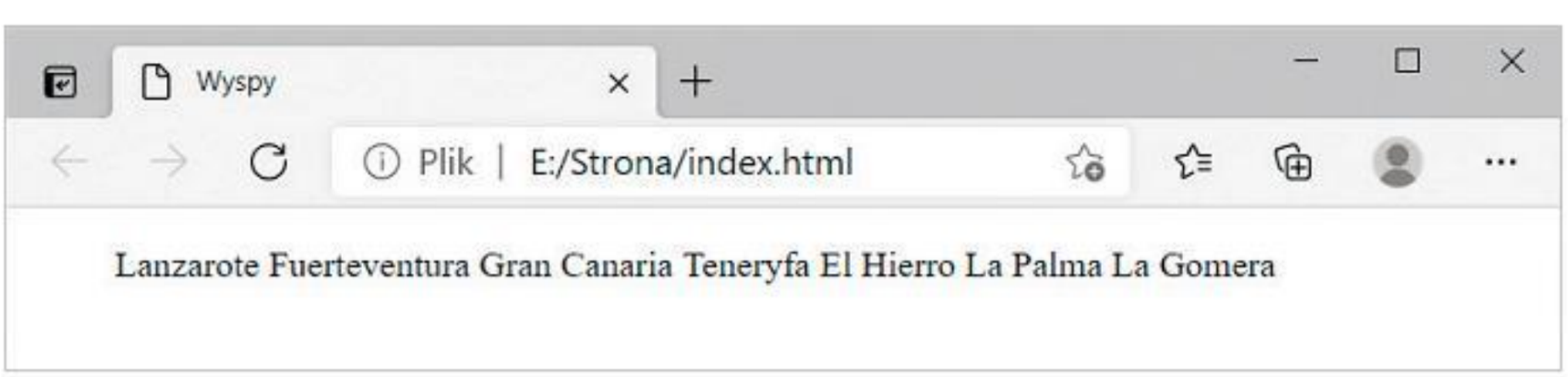


Rys. 4. *Widok listy uporządkowanej po usunięciu numeracji*

Krok 3.

Można ustawić elementy w wierszu, a nie kolejno pod sobą (rys. 5.).

```
ol li { display: inline-block; }
```

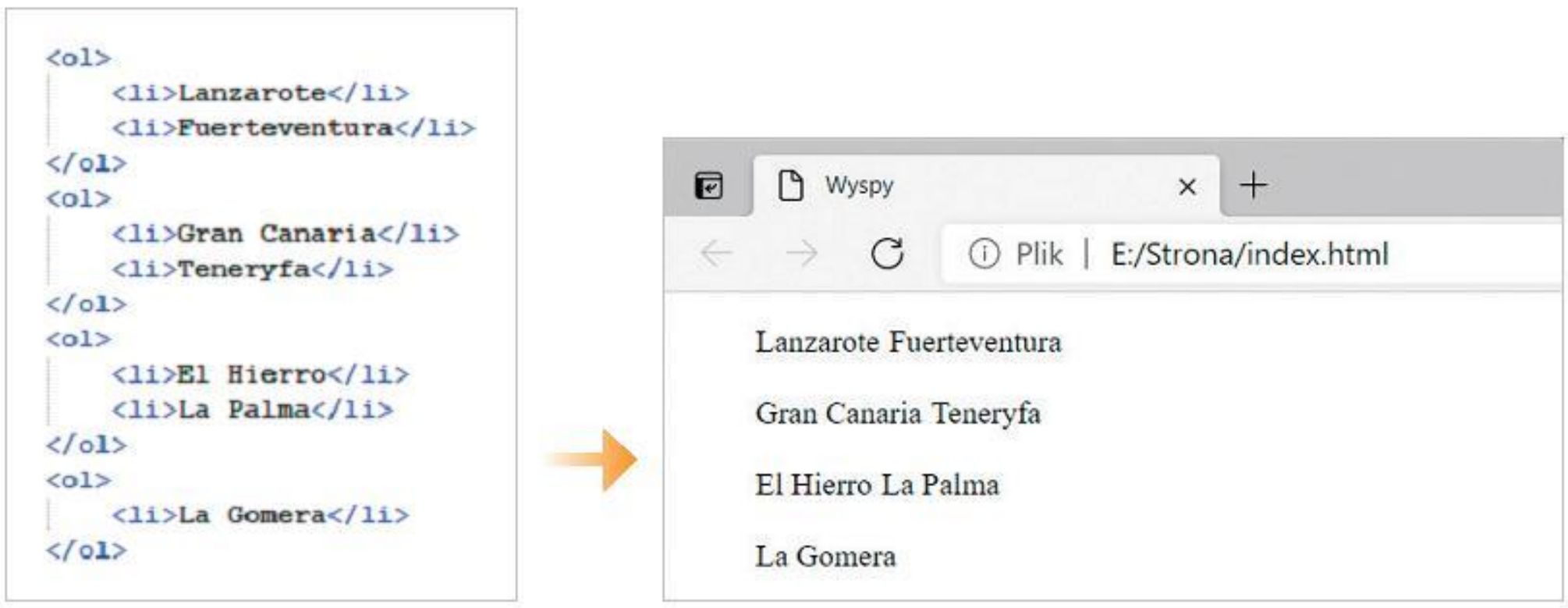


Rys. 5. *Widok listy uporządkowanej po ustawieniu elementów w wierszu*

Krok 4.

Jeśli chcemy, aby elementy były w kilku wierszach i kolumnach, to po ustawieniu elementów listy w jednym wierszu można rozdzielić je na kilka list. Liczba elementów w danej liście określa ich liczbę w danym wierszu. W ten sposób przygotowane listy można później zamienić na kolumny.

```
<ol>
   <li>Element listy</li>
   <li>Element listy</li>
…
</ol>
<ol>
   <li>Element listy</li>
   <li>Element listy</li>
   …
</ol>
…
```



Rys. 6. *Efekt ustawienia elementów listy w kilku wierszach*

Obramowanie elementów listy

Przyciski na stronie można stworzyć z elementów listy poprzez dodanie obramowania oraz innych atrybutów. Podstawowym atrybutem obramowania w stylach CSS jest `border`, w którym określamy grubość, styl i kolor obramowania. Inne przydatne atrybuty przedstawiono w tabeli 1.

Nazwa atrybutu	Funkcja
`border:` *`grubość styl kolor`*	Określa grubość, styl i kolor obramowania. Grubość może być wyrażona w pikselach lub wartościami `thin`, `medium`, `thick`. Przykładowe wartości stylu: `solid`, `double`, `dotted`, `dashed`. Kolory najlepiej określić nazwą jednego z kolorów należących do palety kolorów CSS lub zapisem heksadecymalnym. **Przykłady:** `{ border: 2px solid #575555; }` `{ border: thin dashed red; }`
`border-radius:` *`wielkość_zaokrąglenia`*	Określa zaokrąglenie brzegów elementu. **Przykład:** `{ border-radius: 4px; }`
`padding:` *`góra prawo dół lewo`*	Określa rozmiar marginesów wewnętrznych. **Przykład:** `{ padding: 6px 12px 6px 12px; }` lub w skróconej wersji: `{ padding: 6px 12px; }` – rozmiar górnego oraz dolnego wewnętrznego marginesu wynosi 6 pikseli, natomiast rozmiar prawego oraz lewego wewnętrznego marginesu elementu to 12 pikseli.
`margin:` *`góra prawo dół lewo`*	Określa rozmiar marginesów zewnętrznych. **Przykład:** `{ margin: 2px 5px; }` – rozmiar górnego oraz dolnego zewnętrznego marginesu wynosi 2 piksele, natomiast rozmiar prawego oraz lewego zewnętrznego marginesu to 5 pikseli.
`background-color:` *`kolor_tła`*	Określa kolor tła elementu. **Przykład:** `{ background-color: #ffffff; }`
`box-shadow:` *`przesunięcie_w_poziomie przesunięcie_w_pionie rozmycie rozpiętość kolor inset`*	Określa efekt cienia elementu. **Przykład:** `{ box-shadow: 3px 4px 5px 6px #ccc; }` – efekt cienia jest przesunięty w prawo o 3 piksele i w dół o 4 piksele, ma stopień rozmycia równy 5 pikseli, a rozmiar rozpiętości powiększony o 6 pikseli; jest w kolorze szarym (`#ccc`). Brak składnika inset oznacza, że cień zostanie utworzony na zewnątrz.
`transition:` *`element czas_trwania tempo czas_opóźnienia`*	Określa efekt przejścia elementu. **Przykład:** `{transition: background .5 s ease-in-out; }` – animacja tła trwa 0,5 sekundy w tempie `ease-in-out`.

Tabela 1. *Najczęściej wykorzystywane atrybuty CSS i wybrane przykłady stosowania*

Efekty po wskazaniu kursorem myszy

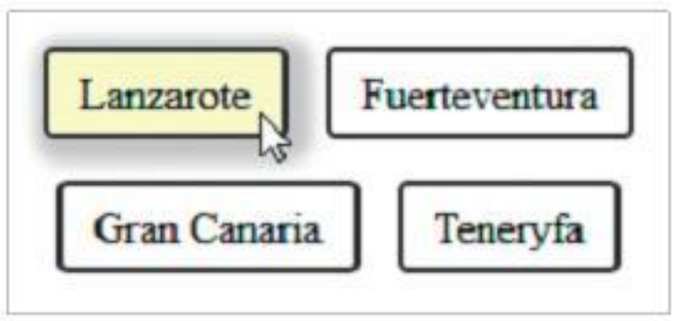
Efekty pojawiające się po wskazaniu elementu kursorem myszy mogą być opisywane przez te same atrybuty co obramowanie (tabela 1.), tylko z innymi wartościami. Na przykład może zmieniać się kolor tła, sposób cieniowania lub styl obramowania. Aby dodać efekty, należy wpisać polecenie:

```
element:hover { atrybut }
```

Na przykład:

```
ol li:hover {
   background-color: #fffecc;
   transition: background .25s ease-in-out;
   box-shadow: 3px 3px 5px 6px #ccc;
}
```

Podane atrybuty oznaczają, że po wskazaniu kursorem myszy element zmieni tło na jasnożółte (`#fffecc`) w 0,25 sekundy w tempie `ease-in-out` oraz pojawi się cień (rys. 7.).



Rys. 7. *Przykład efektu pojawiającego się po wskazaniu elementu kursorem myszy*

Jak można zauważyć na rysunku 8b, ostateczny efekt nie jest idealnym odzwierciedleniem projektu (np. inne są odległości między ramkami, inaczej wygląda tło). Taki rezultat może zostać zaakceptowany, ale musi go zatwierdzić osoba odpowiadająca za ostateczny wygląd strony. Dlatego bardzo ważna jest konsultacja programisty z projektantem oraz testowanie.

3.3. Testowanie

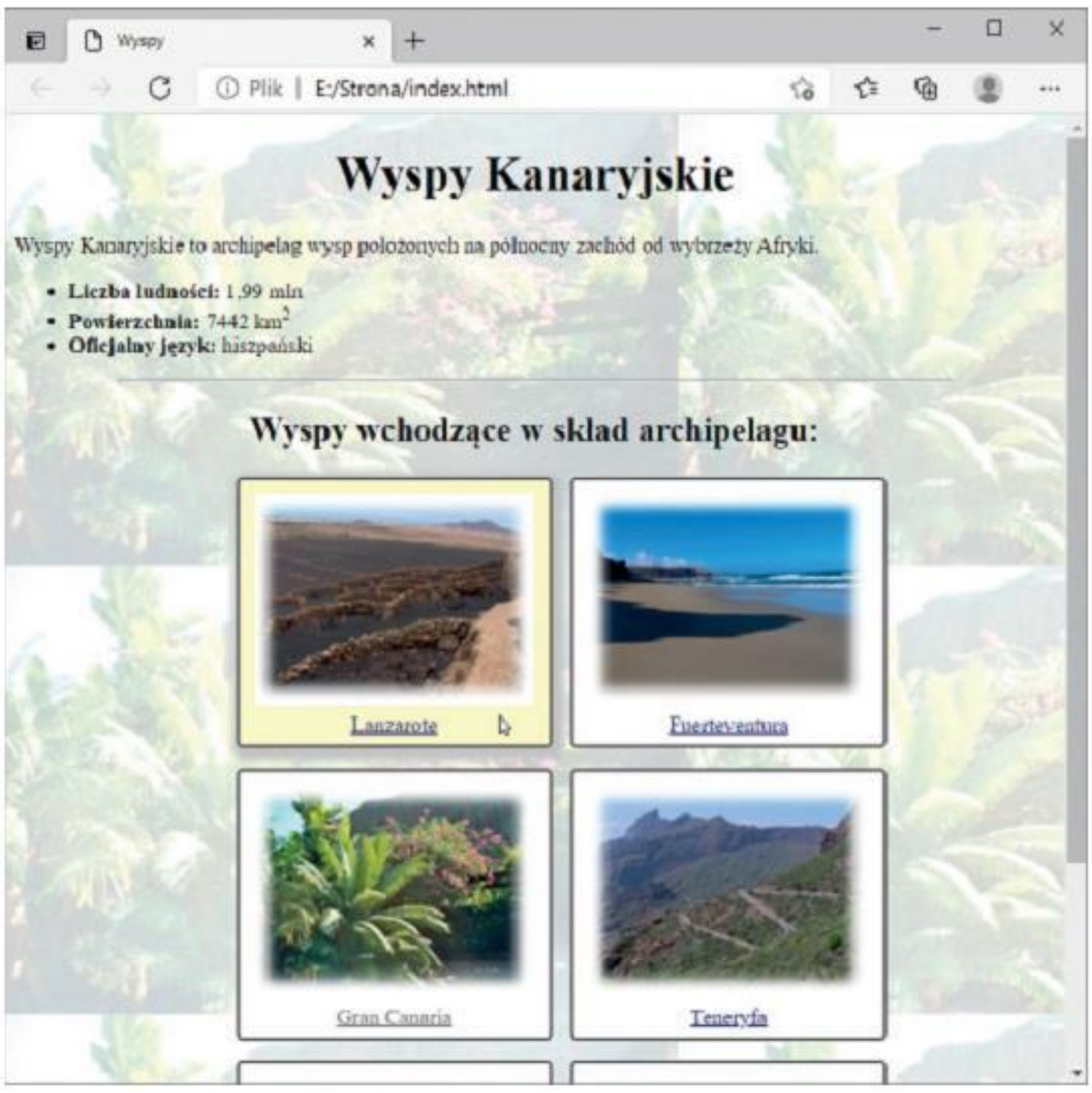
Aby mieć pewność, że strona spełnia wymagania, należy ją przetestować. Oznacza to, że trzeba skontrolować, czy projekt został wdrożony zgodnie ze specyfikacją. Należy sprawdzić m.in. wszystkie funkcjonalności strony oraz jej wygląd na ekranach o różnych rozdzielczościach. Może być za to odpowiedzialny tester oprogramowania lub inny programista sprawdzający kod. W komercyjnych projektach ostateczny wygląd oraz funkcjonalność strony sprawdza klient (najczęściej firma, dla której przygotowuje się stronę).

Plik Edycja Szukaj Widok Format Składnia Ustawienia Narzędzia Makra Uruchom Wty

index.html

```
<!DOCTYPE HTML>
<html lang="pl">
    <head>
    <meta charset="utf-8">
    <title>Wyspy</title>
        <style>
            a { color: #000080; }
            a:hover { color: #ff0000; }
            a:visited  { color: #808080; }
            body { background-image: url("palmy_tlo.jpg"); }
            h1 { font-size: 35px; }
            h2 { font-size: 25px; }
            ol {
                list-style: none;
                text-align: center;
            }
            ol li {
                display: inline-block;
                border: 2px solid #444;
                border-radius: 4px;
                padding: 6px 12px;
                margin: 0 5px;
                background-color: #ffffff;
            }
            ol li:hover {
                background-color: #fffecc;
                box-shadow: 3px 3px 5px 6px #ccc;
                transition: background .25s ease-in-out;
            }
            ol li img {
                display: block;
                margin: 5px auto;
            }
        </style>
    </head>
```

***Rys. 8a**. Fragment kodu źródłowego wraz ze stylami CSS widoczny w oknie programu Notepad ++ – przykład wdrożenia projektu strony o Wyspach Kanaryjskich*



***Rys. 8b**. Strona widoczna w przeglądarce*

4. Zadania projektowe

Zadanie 1. Nowy wygląd strony internetowej o Wyspach Kanaryjskich

Celem projektu jest wzbogacenie prostej strony internetowej o Wyspach Kanaryjskich o dodatkowe elementy (plik otrzymasz od nauczyciela).

Przed przystąpieniem do rozdzielenia zadań projektowych warto wybrać osoby, które będą odpowiedzialne za grafikę, wdrożenie i testowanie. Każde z tych zadań może wykonywać kilka osób.

Zadania szczegółowe

1. Zaprojektuj stronę główną według własnego pomysłu. Możesz zainspirować się projektem pokazanym na rysunku 1.
2. Zaprojektuj podstrony o poszczególnych wyspach. Najlepiej wykonaj jeden wzór podstrony, który powielisz dla kolejnych wysp.
3. Zrealizuj projekt. Jeśli masz pytania, skonsultuj się z osobami, które przygotowały grafikę. Zawsze możesz dostosować koncepcję do swoich możliwości.
4. Przetestuj stronę oraz popraw błędy.

Uwagi dotyczące realizacji projektu

- Projekt można wykonać w dowolnym programie graficznym.
- Pamiętaj, że na każdym etapie ważna jest współpraca między wszystkimi osobami zaangażowanymi w realizację projektu.
- Do realizacji projektu wystarczy znajomość języka HTML i poznanych stylów CSS.

Zadanie 2. Tworzenie strony internetowej wzbogaconej o dodatkowe elementy

Celem projektu jest zaprojektowanie oraz utworzenie prostej strony internetowej na wybrany temat.

Zadania szczegółowe

1. Określ tematykę strony internetowej oraz cel jej istnienia (np. prezentacja zdjęć, blog, dzielenie się wiedzą).
2. Przygotuj projekt strony wraz ze wszystkimi elementami graficznymi. Utwórz menu w formie poziomego paska na górze strony.
3. Zrealizuj projekt.
4. Przetestuj stronę oraz popraw błędy.

Uwagi dotyczące realizacji projektu

- Pamiętaj, że możesz dodać efekty, które pojawią się po wskazaniu elementu kursorem myszy.
- Proste menu w formie paska (rys. 9.) wykonasz, wykorzystując między innymi poznane atrybuty CSS, np.:

```
ol { list-style: none; font-family: Arial; }
ol li { display: inline-block; padding: 6px 12px; }
ol li:hover { background-color: #b3d2e6; }
hr { margin: -18px 40px; }
```

- Dobrą praktyką jest, aby funkcjonalność strony testowała inna osoba, a nie sam twórca.

Przykłady innych zadań projektowych

1. Utwórz stronę internetową, w której podzielisz się wiedzą na jeden z tematów:
   a. Matura z matematyki.
   b. Dowolny sport olimpijski wraz z historią występów reprezentantów Polski.



Rys. 9. *Przykład prostego menu strony internetowej*

   c. Twoje hobby.
   d. Ochrona środowiska.
2. Utwórz stronę internetową, na której umieścisz informacje dotyczące twojego ulubionego sportu.
3. Wybierz dziewięć zdjęć swojego autorstwa, które najlepiej podsumowują aktualny rok szkolny. Zaprojektuj i utwórz stronę, na której umieścisz zdjęcia wraz z podpisami.
4. Znajdź w Internecie przykładowy szablon do tworzenia stron internetowych. Zapoznaj się ze sposobem projektowania stron za jego pomocą. Zrealizuj projekt strony z zadania 1. Wymień w kilku punktach, czym różni się tworzenie stron z wykorzystaniem szablonu od tworzenia stron w edytorze plików tekstowych, np. Notepad++.



Słownik pojęć

B

Baza danych
Zbiór danych, czyli odpowiednio zorganizowanych informacji, które można przetwarzać za pomocą programów (str. 40).

D

Deszyfrowanie
Proces odtwarzania tekstu oryginalnego na podstawie treści szyfrogramu (str. 124).

Druk 3D
Drukowanie przestrzenne, które polega na wytwarzaniu fizycznego obiektu poprzez nanoszenie kolejnych warstw materiału (tzw. metoda przyrostowa lub technologia addytywna) (str. 20).

E

eCall
Ogólnoeuropejski system szybkiego powiadamiania o wypadkach drogowych (str. 31).

K

Klucz
Informacja służąca do szyfrowania lub deszyfrowania tekstu (str.125).

N

NWD
Największy wspólny dzielnik dwóch lub więcej liczb całkowitych (różnych od zera) to największa liczba naturalna, przez którą dzieli się bez reszty każda z tych liczb (str. 160).

NWW
Najmniejsza wspólna wielokrotność dwóch liczb lub więcej całkowitych jest to najmniejsza liczba różna od zera, która jest jednocześnie wielokrotnością obu liczb (str. 160).

P

Pin w elektronice
Styk, przez który wprowadzamy lub wyprowadzamy sygnały z urządzenia (str. 170).

S

SQL
(z ang. *Structured Query Language*)
Strukturalny język zapytań wykorzystywany do obsługi baz danych (str. 70).

Stos w informatyce
Struktura przechowywania danych, w której dostępny jest tylko element znajdujący się na wierzchołku stosu (tak jak w stosie książek na biurku – dostępna jest książka położona na wierzchu jako ostatnia) (str. 149).

System liczbowy
Sposób zapisywania i odczytywania liczb (str. 12).

Szyfr
Algorytm szyfrowania i deszyfrowania (str.123).

Szyfrowanie
Proces przetwarzania wiadomości na **szyfrogram**, pozwalający na ukrycie jej tekstu (str.124).

W

Wzorzec
Spójny ciąg znaków (str. 105).

Książka jest przeznaczona do nauczania informatyki w szkołach ponadpodstawowych na poziomie podstawowym.

Składa się z czterech modułów, a każdy z nich z tematów o wyrazistej strukturze merytorycznej i metodycznej. Modułowa budowa pozwala realizować tematy w podanym układzie, ale również dostosować kolejność ich realizacji do warunków szkoły i grupy uczniów.

Prezentowane metody, pojęcia i przykłady uwzględniają różnorodność oprogramowania, jaka występuje w szkołach.

Uczeń poznaje zagadnienia w sposób systematyczny i uporządkowany, a rozwiązując niebanalne zadania, rozwija wyobraźnię twórczą i uczy się logicznego myślenia.

Książka zawiera:

- ponad 160 ćwiczeń i zadań sprawdzających, z tego 70 z programowania
- ponad 50 pytań sprawdzających
- 7 zadań projektowych

ISBN 978-83-953551-7-2