Struktury – zadania maturalne

- Wprowadzenie
- Przeczytaj
- Prezentacja multimedialna
- Sprawdź się
- Dla nauczyciela

Źródło: Alain Pham, domena publiczna.

Ze względu na swoje własności struktury wykorzystywane są przede wszystkim do przechowywania powiązanych ze sobą danych różnego typu. Czym są oraz jak działają, dowiedzieliśmy się w e-materiale Wprowadzenie do struktur. Z kolei w e-materiale Struktury omówiliśmy ich zastosowania w zadaniach algorytmicznych. Teraz spróbujemy rozwiązać zadania typu maturalnego wymagające znajomości struktury.

Implementacje struktur w różnych językach programowania zostały omówione w e-materiałach:

- Struktury w języku C++,
- Struktury w języku Java,
- Struktury w języku Python.

Twoje cele

- Sprawdzisz zdobyte umiejętności oraz wiedzę z zakresu struktur, rozwiązując zadania typu maturalnego.
- Rozwiążesz złożone problemy z wykorzystaniem struktur.
- Dobierzesz struktury danych, które pozwolą na rozwiązanie problemu w danym języku programowania.

Przeczytaj

Zadanie 1. Spis pingwinów

Ogród zoologiczny w Bajtowie przeprowadza spis 270 pingwinów znajdujących się na wszystkich wybiegach.

Dane w spisie dotyczące pingwinów zawierają:

- `Id` – identyfikator ID (unikatowy dla każdego osobnika) lub „0”, gdy brak danych,
- `Płeć` – „samiec” lub „samica”,
- `Gatunek` – nazwa gatunku w postaci ciągu znaków,
- `Wiek` – zapisany w miesiącach (liczba naturalna zapisana z zerami wiodącymi, o długości dokładnie trzech znaków).

Aby uporządkować dane w spisie, pracownicy zoo zapisali zebrane informacje w pliku `pingwiny.txt` – każdy osobnik w nowej linii, każda z czterech informacji oddzielona pojedynczym znakiem odstępu.

Plik `pingwiny.txt`.
Plik o rozmiarze 7.26 KB w języku polskim

W spisie brakuje jednak informacji dotyczących niektórych numerów ID – luki zostały oznaczone jako „0”. Wiadomo natomiast, że ID każdego pingwina tworzone jest w następujący sposób:

1. pierwsza cyfra to cyfra oznaczająca płeć – „1” to samiec, „2” to samica,
2. druga, trzecia i czwarta cyfra to kod ASCII pierwszego znaku nazwy gatunku pingwina (np. dla wartości 97 powinno to być 097),
3. piąta, szósta i siódma cyfra to wiek pingwina w miesiącach (np. dla osobnika w wieku sześciu miesięcy cyfry wieku podane są w następujący sposób: 006).

W wybranym przez siebie języku programowania napisz program, który:

1. uzupełni brakujące dane dotyczące pingwinów,
2. posortuje informacje niemalejąco według numerów ID, a następnie zapisze do pliku `gatunki.txt` gatunki pingwinów, których dane w spisie znajdują się na pozycjach 1, 20 oraz 200, każdy w osobnej linii.

Do oceny oddajesz:

- plik `gatunki.txt` zawierający odpowiedź (gatunki pingwinów z pliku `pingwiny.txt`, które po posortowaniu według ID znalazły się na 1., 20. oraz 200. miejscu),

- plik(i) z komputerową realizacją zadania (kodem programu).

> Przedstaw rozwiązanie w postaci programu w języku C++, Java lub Python. Odpowiedź do zadania znajdziesz po omówieniu rozwiązania.

Rozwiązanie

Rozwiązanie zadania przedstawimy w postaci pseudokodu.

Stworzenie struktury oraz import danych

```
1  struktura Pingwin
2      Id: liczba całkowita
3      Płeć: tekst
4      Gatunek: tekst
5      Wiek: liczba całkowita
6
7  utwórz tablicę Pingwiny[]
8
9  dla i = 0, 1, ..., 269 wykonuj:
10     Informacje[0..3] ← wczytaj i-tą linię z pliku "pingwiny.txt"
11     Pingwiny[i] ← nowy element typu Pingwin
12     Pingwiny[i].Id ← Informacje[0]
13     Pingwiny[i].Płeć ← Informacje[1]
14     Pingwiny[i].Gatunek ← Informacje[2]
15     Pingwiny[i].Wiek ← Informacje[3]
```

1. Tworzymy strukturę opisującą dane dotyczące pingwinów oraz określamy typy danych zawartych w niej elementów. **[linie od 1 do 5]**
2. Importujemy dane z każdej linii pliku oraz zapisujemy je do struktur reprezentujących pingwiny. **[linie od 9 do 15]**

Ze względu na istotne różnice między językami programowania dostępnymi na egzaminie maturalnym przedstawiamy przykładowe implementacje struktury `Pingwin`:

- C++

```
1  struct Pingwin {
2      int Id;
3      string Płeć;
4      string Gatunek;
5      int Wiek;
```

```
};
```

- Java

```java
class Pingwin {
    public int Id;
    public String Płeć;
    public String Gatunek;
    public int Wiek;
}
```

- Python (z użyciem struktury słownika wypełnionej przykładowymi danymi)

```python
Pingwin = {
    "Id" : 1,
    "Płeć" : "samiec",
    "Gatunek" : "grubodzioby",
    "Wiek" : 102
}
```

Rozwiązanie zadań

> **Ważne!**
>
> Ze względu na sortowanie struktur względem jednego z ich pól trudniej skorzystać z wbudowanych funkcji sortowania, dlatego też musimy sami zaimplementować wybrany algorytm sortowania. W podanym rozwiązaniu do posortowania danych użyty został algorytm sortowania bąbelkowego.

```
funkcja zamień(x, y)
    pom ← x
    x ← y
    y ← pom
```

Rozpoczynamy od definicji funkcji `zamień(x, y)`, która zamieni ze sobą wartości dwóch argumentów do niej przekazanych. **[linia od 1 do 4]**

Zapiszemy algorytm sortowania bąbelkowego tablicy.

```
```

```
1  funkcja sortuj(Pingwiny[])
2      n ← długość(Pingwiny[])
3      dla i = 0, 1, 2, ... n - 1 wykonuj:
4          zamiana ← fałsz
5          dla j = 1, 2, ..., n - i - 1 wykonuj:
6              jeżeli Pingwiny[j - 1].Id > Pingwiny[j].Id :
7                  zamień(Pingwiny[j - 1], Pingwiny[j])
8                  zamiana ← prawda
9          jeżeli zamiana = fałsz:
10             przerwij pętlę
```

1. Definiujmy funkcję sortowania bąbelkowego. **[linia 1]**
2. Zapisujemy zewnętrzną pętlę przechodzącą po wszystkich pozycjach tablicy. Wewnątrz definiujemy zmienną logiczną `zamiana`, której wartość będzie decydować, czy zakończyć sortowanie (unikniemy pustych przebiegów pętli, jeśli tablica będzie posortowana wcześniej). **[linie 2, 3, 4]**
3. W pętli wewnętrznej przechodzącej po nieuporządkowanych elementach (`i` pingwinów od końca znajduje się już na właściwych pozycjach) porównujemy ID pingwinów. Jeżeli osobnik na wcześniejszej pozycji ma większe ID, zamieniamy go z sąsiadem po prawej i aktualizujemy wartość zmiennej `zamiana`. **[linie 5–8]**
4. W przypadku niewystąpienia jakiejkolwiek zamiany wartość zmiennej `zamiana` pozostanie równa `fałsz`, co oznacza, że tablica jest już posortowana i można przerwać zewnętrzną pętlę. **[linie 9, 10]**

Sortowanie możemy wykonać wyłącznie wtedy, kiedy wszystkie pingwiny będą miały nadane właściwe identyfikatory. Powinniśmy zatem sprawdzić, które ich nie mają, a następnie im je nadać. Zapiszemy zatem pętlę, która sprawdzi, które pingwiny nie mają nadanych poprawnie identyfikatorów, a następnie przypisze im je na podstawie płci, gatunku oraz wieku. Ostatnim krokiem będzie posortowanie całej struktury oraz zapisanie odpowiednich danych do pliku wynikowego.

```
1  dla i = 0, 1, ..., długość(Pingwiny[]) - 1 wykonuj:
2      jeżeli Pingwiny[i].Id = 0:
3          jeżeli Pingwiny[i].Płeć = "samiec":
4              Pingwiny[i].Id ← 1
5          w przeciwnym razie:
6              Pingwiny[i].Id ← 2
7
8          Pingwiny[i].Id ← Pingwiny[i].Id * 1000
9          Pingwiny[i].Id ← Pingwiny[i].Id + ASCII(Pingwiny[i].Gatun
10
```

```
11          Pingwiny[i].Id ← Pingwiny[i].Id * 1000
12          Pingwiny[i].Id ← Pingwiny[i].Id + Pingwiny[i].Wiek
13
14  sortuj(Pingwiny)
15
16  zapisz Pingwiny[0].Gatunek, Pingwiny[19].Gatunek, Pingwiny[199].G
```

1. W celu rozwiązania zadania przechodzimy po wszystkich indeksach tablicy `Pingwiny`, a następnie sprawdzamy, które numery ID są błędnie zapisane (wpisane do struktury jako 0) **[linie 1, 2]**. Co istotne, ID można było tworzyć i uzupełniać na bieżąco, podczas wczytywania danych. W kodzie ilustrującym tworzenie struktury w **linii 12** można dodać tworzenie ID, umieszczając tam **linie 2–12** z powyższego kodu.
2. Gdy natrafimy na błędny numer ID, należy go stworzyć zgodnie ze wzorem opisanym w zadaniu:

- Jeżeli osobnik jest samcem, pierwszą cyfrą będzie 1. Jeśli zaś jest samicą, pierwszą cyfrą będzie 2. **[linie od 3 do 6]**
- Następnie mnożymy wartość zmiennej strukturalnej ID przez 1000 (dopisując na końcu trzy zera – w ten sposób nie musimy rozważać, czy kod ASCII jest trzy-, czy dwucyfrowy). Dodajemy teraz wartość kodu ASCII pierwszej litery gatunku osobnika do jego ID. **[linie 8, 9]**
- Tym samym sposobem ponownie dopisujemy trzy zera do wartości zmiennej ID, a następnie dodajemy wiek osobnika. **[linie 11, 12]**

3. Sortujemy teraz całą strukturę – w przedstawionym przykładzie użyty został algorytm sortowania bąbelkowego. **[linia 14]**
4. Następnie zapisujemy odpowiednio dane do pliku wynikowego (pamiętamy, że w językach programowania dostępnych na egzaminie maturalnym indeksowanie elementów tablicy zaczyna się od zera). **[linia 16]**

Odpowiedź do zadania

Odpowiedź dla danych z pliku `pingwiny.txt`:

```
1  adeli
2  bialobrewy
3  krolewski
```

Słownik

> **indeks**

uporządkowany (zwykle alfabetycznie) spis pojęć, nazwisk, nazw, tematów wraz z numerami stron, na których występują

pętla zagnieżdżona

(ang. *nested loop*) sposób zapisu pętli, gdzie jedna pętla znajduje się w drugiej; przechodzenie pomiędzy pętlami odbywa się w następujący sposób: pętla zewnętrzna rozpoczyna pierwsze wykonanie, następuje przejście po wszystkich iteracjach pętli wewnętrznej, wówczas pętla wewnętrzna kończy swoje działanie i pętla zewnętrzna rozpoczyna drugie wykonanie

sortowanie bąbelkowe

(ang. *bubble sort*) prosty algorytm sortowania polegający na porównywaniu dwóch kolejnych elementów i zamianie ich kolejności, jeśli zaburza ona porządek, w jakim sortuje się tablicę; algorytm kończy się, gdy nie wykonano żadnych zamian

Prezentacja multimedialna

Zadanie 2. Fabryka leków

W miejscowości Matrycowo znajduje się fabryka zajmująca się produkcją leków na różne dolegliwości. Zarząd fabryki postanowił uporządkować produkowane rodzaje tabletek za pomocą struktury przechowującej odpowiednio uporządkowane dane.

Każda tabletka opisana jest za pomocą następujących danych:

- `IdLeku` – liczba naturalna z przedziału $\langle 100, 199 \rangle$ lub `"brak"`, jeżeli brak danych; identyfikator leku;
- `Waga` – liczba naturalna z przedziału $\langle 1000, 5500 \rangle$ lub `"brak"`, jeżeli brak danych; waga leku w miligramach;
- `Liczba` – liczba naturalna z przedziału $\langle 3, 9 \rangle$ lub `"brak"`, jeżeli brak danych; wartość określająca, po ile tabletek pakowanych jest w pojedynczym blistrze (listku);
- `Kształt` – ciąg znaków; kształt tabletki; możliwe wartości: `"owalny"`, `"okragly"`, `"prostokatny"`, `"fasolkowaty"`.

> **Ważne!**
>
> Jeden lek może występować w różnych tabletkach, różniących się wagą, liczbą tabletek lub kształtem – wtedy `IdLeku` dwóch tabletek jest identyczny, np.:
>
> ```
> 102 3500 4 okragly
> 102 1000 9 prostokatny
> ```

Niestety z powodu awarii część danych została utracona. W każdym wierszu zostało uszkodzone maksymalnie jedno pole. Dane o kształcie zostały zachowane w całości.

Po wstępnej analizie okazało się, że informacje o liczbie tabletek w blistrze uległy uszkodzeniu tylko dla tabletek owalnych lub prostokątnych. Firma przy pakowaniu

posługuje się następującą zasadą:

Tabletki o owalnym lub prostokątnym kształcie dla nieparzystego `IdLeku` są pakowane w blistrze po 7 sztuk, natomiast dla parzystego `IdLeku` – po 9 sztuk.

Po głębszej analizie specjalistom udało się ustalić, że:

Waga tabletek uległa uszkodzeniu tylko w przypadku tabletek okrągłych. Każda taka tabletka, zgodnie z jej specyfikacją, waży 3500 mg.

Finalnie udało się ustalić, że `IdLeku` tabletek został zniszczony tylko dla tabletek fasolkowatych. W dostarczonej dokumentacji widnieje zasada:

`IdLeku` fasolkowatych tabletek to zawsze „17”, gdzie symbol „*” to liczba tabletek w pojedynczym blistrze.*

Plik `leki.txt` składa się z 300 wierszy. W każdym wierszu znajdują się kolejno oddzielone od siebie znakiem spacji trzy liczby naturalne (lub ciągi znaków `"brak"` w przypadku braku danych) oraz łańcuch znaków. Informacje o kształcie nie zostały utracone w żadnym z wierszy.

Plik `leki.txt`.
Plik o rozmiarze 6.22 KB w języku polskim

> **Polecenie 1**
> W wybranym przez siebie języku programowania napisz program, który:
>
> 1. poda kształty – mogą się powtarzać – wszystkich tabletek o `IdLeku` równym `IdLeku` pierwszej lub drugiej tabletki w spisie uporządkowanym niemalejąco według `IdLeku`, a wyniki zapisze w pliku `wyniki.txt`, rozdzielając je pojedynczym znakiem odstępu;
> 2. poda liczbę takich tabletek, których kształt zaczyna się na „o”, a których liczba w blistrze wynosi 7, i zapisze odpowiedź w drugiej linii pliku `wyniki.txt`;
> 3. poda liczbę takich tabletek, których `IdLeku` jest równy 174 i zapisze odpowiedź w trzeciej linii pliku `wyniki.txt`.

Do oceny oddajesz:

- plik `wyniki.txt` zawierający odpowiedzi do poleceń,
- plik(i) z komputerową realizacją zadania (kodem programu).

> **Praca domowa**
>
> Przedstaw rozwiązanie zadania w postaci programu w języku C++, Java lub Python. Zadbaj o prawidłowe wczytanie danych z pliku tekstowego do programu. Odpowiedź do zadania znajdziesz pod prezentacją rozwiązania.

Rozwiązanie

> Rozwiązanie zadania przedstawimy za pomocą pseudokodu.

Odpowiedź do zadania

Odpowiedź do zadania dla danych z pliku `leki.txt`:

```
prostokatny prostokatny
30
```

```
3 33
```

> **Polecenie 2**
> Dodaj do swojego programu komentarze tak, żeby był zrozumiały dla osoby, która nie potrafi programować.

Sprawdź się

Zadanie 3. Sklep papierniczy

Sklep Nowa Linia ma szeroki asortyment produktów papierniczych.

Każdy artykuł w sklepie opisany jest następującymi danymi:

- `id produktu` – liczba całkowita dodatnia, unikatowy numer każdego produktu,
- `rodzaj` – „zeszyt”, „kalendarz”, „notatnik”,
- `rozmiar` – „A3”, „A4”, „A5”,
- `liczba kartek` – liczba całkowita z zakresu $\langle 6, 200 \rangle$,
- `okladka` – „twarda”, „miekka”.

Przedstawione informacje dla 500 artykułów zostały zapisane w pliku `papierniczy.txt`, każdy artykuł w osobnej linii. Informacje dotyczące artykułów zostały oddzielone od siebie pojedynczymi znakami odstępu.

Plik `papierniczy.txt`.
Plik o rozmiarze 12.43 KB w języku polskim

Korzystając z danych w pliku, wykonaj zadania:

1. Niektóre dane produktów zostały błędnie wpisane. Zapisz, ile artykułów z błędnymi danymi znajduje się w pliku. Błędne dane rozpoznajemy po tym, że nie spełniają podanych niżej zasad:

- Zeszyt w miękkiej okładce nie ma więcej kartek niż 110.
- Liczba kartek w notatniku jest liczbą z zakresu $\langle 30, 85 \rangle$.
- Kalendarze w rozmiarze A3 występują tylko w miękkiej okładce.
- Kalendarze mają zawsze 6 lub 12 kartek.

> **Uwaga!**
>
> Produkt łamiący więcej niż jedno kryterium liczymy tylko raz. Produkty z błędnymi danymi nie powinny być uwzględniane podczas wykonywania kolejnych zadań!

2. Produkty należą do jednej kategorii, jeżeli mają jednakowe następujące dane:

- `rodzaj`,
- `rozmiar`,
- `liczba kartek`,
- `okładka`.

Dla każdego rodzaju (zeszyt, notatnik oraz kalendarz) zapisz największą liczbę produktów danej kategorii. Kolejne liczby rozdziel znakami odstępu.

3. Zapisz nazwę rodzaju produktu, który zawiera łącznie najwięcej różnych kategorii produktów oraz łączną liczbę kartek produktów różnych kategorii tego rodzaju.
4. Sprawdź, czy istnieje produkt o liczbie kartek, która jest liczbą pierwszą. Jeżeli tak, zapisz „tak”, w przeciwnym wypadku zapisz „nie”.

Kolejne odpowiedzi zapisz do pliku `wyniki.txt`, każdy podpunkt w nowej linii (wyniki w ramach podpunktów oddziel pojedynczymi znakami odstępu).

Do oceny oddajesz:

- plik `wyniki.txt` zawierający odpowiedzi do poleceń,
- plik(i) z komputerową realizacją zadania (kodem programu).

Pokaż ćwiczenia:

Ćwiczenie 1

JĘZYK C++

Twoje zadania

1. Napisz program, który podaje wyniki do podpunktów od 1. do 4. Kolejne odpowiedzi zapisz w nowych liniach, wyniki w ramach jednego podpunktu oddziel pojedynczym znakiem odstępu.

Ćwiczenie 2

JĘZYK JAVA

Twoje zadania

1. Napisz program, który podaje wyniki do podpunktów od 1. do 4. Kolejne odpowiedzi zapisz w nowych liniach, wyniki w ramach jednego podpunktu oddziel pojedynczym znakiem odstępu.

Ćwiczenie 3

JĘZYK PYTHON

Twoje zadania

1. Napisz program, który podaje wyniki do podpunktów od 1. do 4. Kolejne odpowiedzi zapisz w nowych liniach, wyniki w ramach jednego podpunktu oddziel pojedynczym znakiem odstępu.

Wyniki dla pliku `papierniczy.txt`:

```
73
3 4 59
notatniki 8951
tak
```

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Struktury – zadania maturalne

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych;

3) objaśnia dobrany algorytm, uzasadnia poprawność rozwiązania na wybranych przykładach danych i ocenia jego efektywność;

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Sprawdzisz zdobyte umiejętności oraz wiedzę z zakresu struktur, rozwiązując zadania typu maturalnego.
- Rozwiążesz złożone problemy z wykorzystaniem struktur.
- Dobierzesz struktury danych, które pozwolą na rozwiązanie problemu w danym języku programowania.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie dla języka C++, w tym kompilator GCC/G++ 4.5 (lub nowszej wersji) i Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio;
- oprogramowanie dla języka Java SE 8 (lub nowszej wersji), w tym Eclipse 4.4 (lub nowszej wersji);
- oprogramowanie dla języka Python 3 (lub nowszej wersji), w tym PyCharm lub IDLE.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Struktury – zadania maturalne". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wyświetla uczniom temat zajęć oraz cele. Prosi, by na ich podstawie uczniowie sformułowali kryteria sukcesu.
2. Prowadzący prosi uczniów, aby zgłaszali swoje propozycje pytań do tematu. Jedna osoba może zapisywać je na tablicy. Gdy uczniowie wyczerpią swoje pomysły, a pozostały jakieś ważne kwestie do poruszenia, nauczyciel je dopowiada.

Faza realizacyjna:

1. **Praca z tekstem.** Metodą burzy mózgów uczniowie referują najważniejsze informacje, jakie znaleźli w sekcji „Przeczytaj", przygotowując się do lekcji. W razie potrzeby nauczyciel wyjaśnia niezrozumiałe kwestie.
   Na forum klasy uczniowie analizują przedstawione w niej rozwiązanie zadania 1. Spis pingwinów. Następnie w parach implementują algorytm w wybranym języku programowania.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Prezentacja multimedialna". Uczniowie analizują rozwiązanie zadania 2. Fabryka leków.
3. **Ćwiczenia umiejętności.** Uczniowie wykonują zadanie 3. Sklep papierniczy z sekcji „Sprawdź się". Ich zadaniem jest utworzenie struktury, a następnie wykonanie podpunktów od 1 do 4. Kolejne odpowiedzi powinny zostać rozdzielone pojedynczym znakiem odstępu.
   Ćwiczenie realizują w jednym z dostępnych języków programowania.

Faza podsumowująca:

1. Nauczyciel ponownie wyświetla na tablicy temat i cele lekcji zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji następuje omówienie ewentualnych problemów z rozwiązaniem ćwiczeń z sekcji „Sprawdź się".

Praca domowa:

1. Uczniowie dodają do swoich rozwiązań z sekcji „Sprawdź się" komentarze tak, by nawet osoba, która nie zna programowania, mogła zrozumieć zachodzące procesy.
2. Uczniowie implementują w wybranym języku programowania rozwiązanie zadania przedstawione w sekcji „Prezentacja multimedialna".
3. Uczniowie wykonują polecenie 2 z sekcji „Prezentacja multimedialna".

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka C++.
- Oficjalna dokumentacja techniczna dla oprogramowania Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.
- Oficjalna dokumentacja techniczna dla oprogramowania Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.
- Oficjalna dokumentacja techniczna dla języka Java SE 8 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla języka Python 3 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania PyCharm lub IDLE.

Wskazówki metodyczne:

- Treści w sekcji „Przeczytaj” można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.