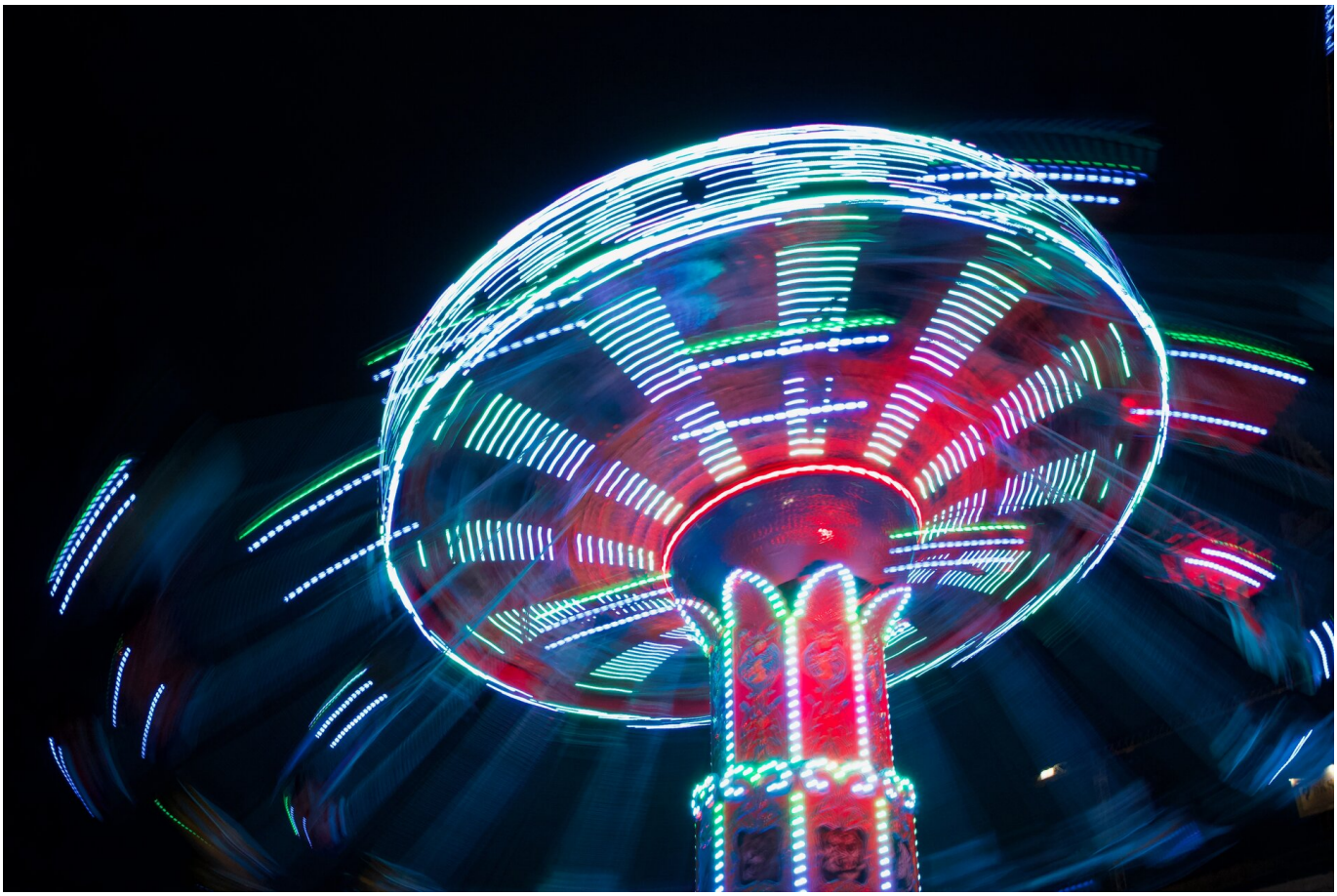
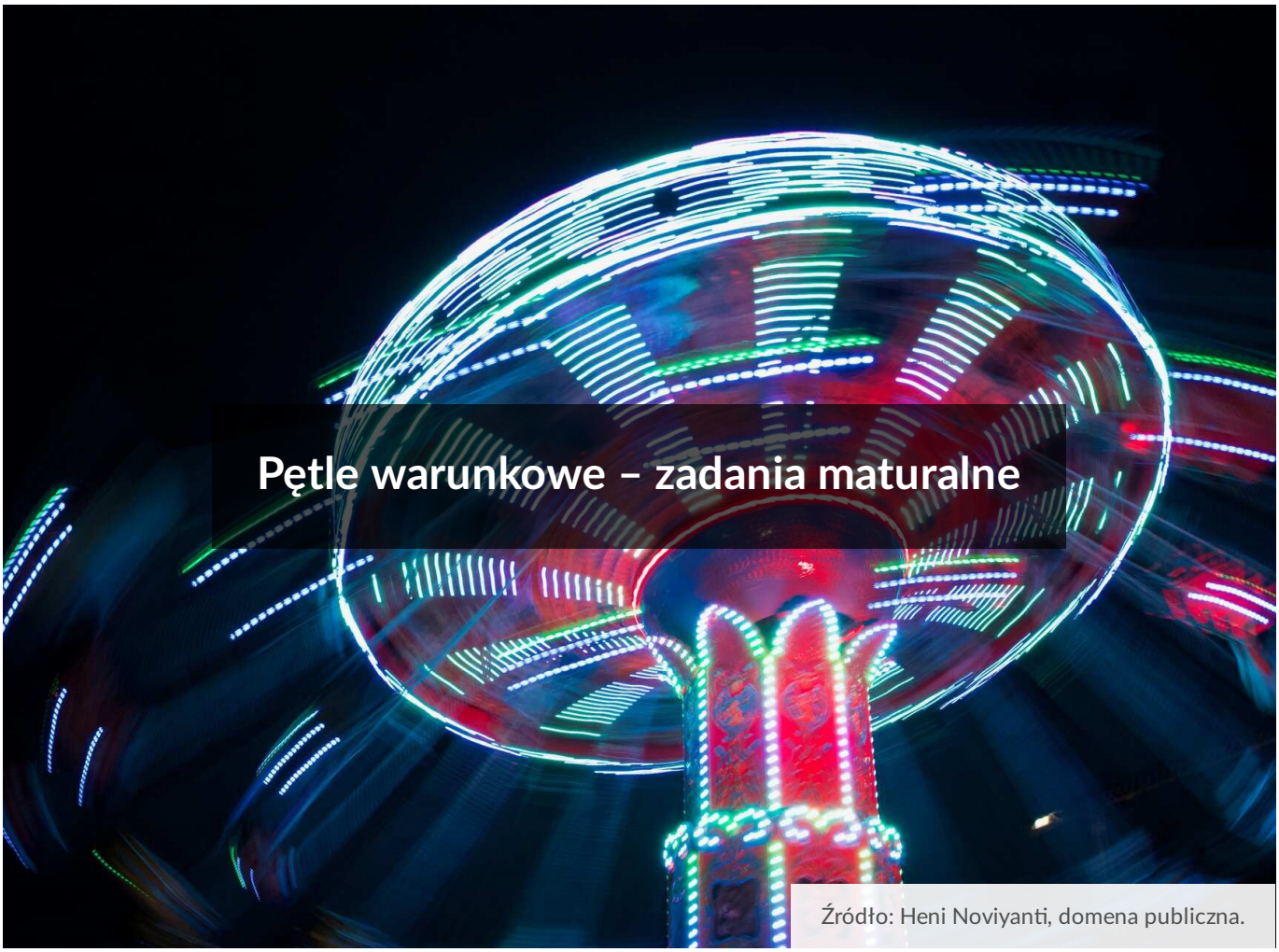


Pętle warunkowe – zadania maturalne



Źródło: Heni Noviyanti, domena publiczna.

Pętle należą do grupy instrukcji sterujących działaniem programu. Pozwalają one na cykliczne wykonanie instrukcji. Poznaliśmy już działanie pętli warunkowej `while`.

W tym e-materiale zapoznamy się z przykładowymi zadaniami maturalnymi dotyczącymi tego zagadnienia.

Implementacje pętli `while` w poszczególnych językach programowania przedstawiamy w e-materiałach:

- Pętle warunkowe w języku C++,
- Pętle warunkowe w języku Java,
- Pętle warunkowe w języku Python.

> **Twoje cele**
>
> - Przeanalizujesz sposób rozwiązywania typowych zadań z wykorzystaniem pętli `while`.
> - Rozwiążesz samodzielnie kilka zadań, wykorzystując pętlę `while`.
> - Prześledzisz schemat oceniania zadań maturalnych z informatyki.

Przeczytaj

Powtórzenie wiadomości o pętli `while`

Pętla `while` jest specjalną instrukcją programistyczną służącą do powtarzania określonych czynności. To alternatywne rozwiązanie dla pętli `for`, która działa bardzo podobnie. Stosuje się ją zazwyczaj w sytuacjach, gdy kluczową kwestią jest liczba wykonanych powtórzeń. Z kolei pętli `while` używa się wówczas, gdy liczba wykonywanych powtórzeń nie jest znana, natomiast ważne jest spełnienie określonego warunku.

Przykładowe zastosowania pętli `while`:

1. operacje matematyczne na danej liczbie wykonywane do momentu osiągnięcia określonej wartości,
2. wypisanie `n` liczb parzystych większych od 0,
3. odliczanie pozostałego czasu trwania sesji.

Implementacja pierwszego przykładu w pseudokodzie:

```
liczba ← 0

dopóki liczba < 5 wykonuj:
    liczba ← liczba + 1
```

Implementacja drugiego przykładu w pseudokodzie:

```
n ← 5
licznik ← 0
i ← 1

dopóki licznik < n wykonuj:
    jeżeli i mod 2 = 0:
        wypisz i
        licznik ← licznik + 1
    i ← i + 1
```

Implementacja trzeciego przykładu w pseudokodzie:

```
ile_sekund ← 60

```

```
dopóki ile_sekund > 0 wykonuj:
    wypisz ("Pozostało " + ile_sekund + " sekund trwania sesji")
    czekaj jedną sekundę
    ile_sekund ← ile_sekund - 1
```

Zadanie 1. Silniowy system pozycyjny

Pojęcie silni dla liczb naturalnych większych od zera definiuje się następująco:

$$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-1) \cdot n$$

Silniowy system pozycyjny to pozycyjny sposób zapisu liczb naturalnych, w którym mnożniki dla kolejnych pozycji definiowane są przez silnie kolejnych liczb naturalnych, czyli:

$$:)_! = (x_n\, x_{n-1}\, x_{n-2} \ldots x_2\, x_1)_! = x_n \cdot n! + x_{n-1} \cdot \ (n-1)! + \ldots + x_2 \cdot 2! + x_1 \cdot$$

W systemie silniowym współczynnik x_i, który odpowiada mnożnikowi $i!$, spełnia zależność $0 \leq x_i \leq i$. Zapis każdej liczby w silniowym systemie pozycyjnym jest jednoznaczny, tzn. każdą liczbę naturalną można zapisać tylko w jeden i dokładnie jeden sposób. W omawianym zadaniu będziemy mieć do czynienia wyłącznie z takimi liczbami, dla których współczynniki x_i spełniają zależność $0 \leq x_i \leq 9$.

Poniżej przedstawiono algorytm z lukami, który zamienia zapis liczb z systemu dziesiętnego na system silniowy. Uzupełnij luki w tym algorytmie.

```
silnia ← 1
k ← 1
dopóki (silnia < x), wykonuj:
    k ← k + 1
    silnia ← silnia * k
jeżeli ........................................
    silnia ← silnia div k
    k ← k – 1
s ←" "
dopóki (k > 0), wykonuj:
    cyfra ← ..............................................
    s ← s ◦ tekst(cyfra)
    x ← ..................................................
    silnia ← ..........................................
    k ← k – 1
```

Uwaga:

- `tekst(x)` – funkcja zamieniającą liczbę `x` na jej zapis tekstowy
- `" "` – pusty napis
- `u ◦ v` – sklejenie dwóch napisów: `u` oraz `v`

Specyfikacja problemu:

Dane:

- `x` – liczba całkowita dodatnia zapisana w systemie dziesiętnym

Wynik:

- `s` – napis reprezentujący liczbę `x` zapisaną w systemie silniowym

> Zadanie zostało opracowane przez Centralną Komisję Egzaminacyjną i zostało opublikowane w zbiorze zadań maturalnych z informatyki. Cały zbiór można znaleźć na stronie internetowej CKE.

Rozwiązanie

Przeanalizujmy algorytm krok po kroku. Pierwsza z luk do uzupełnienia pojawia się tuż po obliczeniu silni, której docelowa wartość ma być nie mniejsza niż `x`:

```
silnia ← 1
k ← 1
dopóki (silnia < x), wykonuj:
    k ← k + 1
    silnia ← silnia * k
jeżeli ........................................
    silnia ← silnia div k
    k ← k – 1
```

Instrukcja umieszczona w bloku `jeżeli` powoduje zmniejszenie wartości zmiennych `silnia` oraz `k`. W celu otrzymania pierwszego współczynnika z silniowego systemu pozycyjnego odszukujemy jak największą wartość silni, jednak nie większą niż konwertowana liczba.
W związku z tym jedynym możliwym warunkiem dla bloku jest `silnia > x`:

```
silnia ← 1
k ← 1
dopóki (silnia < x), wykonuj:
    k ← k + 1
    silnia ← silnia * k
jeżeli silnia > x:
```

```
    silnia ← silnia div k
    k ← k – 1
```

Kolejne luki w algorytmie pojawiają się po zadeklarowaniu pustego łańcucha znaków i bezpośrednio dotyczą konwersji do silniowego systemu pozycyjnego:

```
silnia ← 1
k ← 1
dopóki (silnia < x), wykonuj:
    k ← k + 1
    silnia ← silnia * k
jeżeli silnia > x:
    silnia ← silnia div k
    k ← k – 1
s ←" "
dopóki (k > 0), wykonuj:
    cyfra ← ..........................................
    s ← s ◦ tekst(cyfra)
    x ← ..............................................
    silnia ← ........................................
    k ← k – 1
```

Luka w linii 11. dotyczy obliczania kolejnej cyfry w silniowym systemie pozycyjnym. Zgodnie z definicją będzie to wynik dzielenia całkowitego przez odpowiednią silnię będącą współczynnikiem przy danej cyfrze, która jest zapisana w zmiennej `silnia`:

```
silnia ← 1
k ← 1
dopóki (silnia < x), wykonuj:
    k ← k + 1
    silnia ← silnia * k
jeżeli silnia > x:
    silnia ← silnia div k
    k ← k – 1
s ←" "
dopóki (k > 0), wykonuj:
    cyfra ← x div silnia
    s ← s ◦ tekst(cyfra)
    x ← ..............................................
    silnia ← ........................................
    k ← k – 1
```

Następnym krokiem będzie pomniejszenie wartości zmiennej `x` w celu wyznaczenia cyfr na kolejnych pozycjach. Zmniejszamy ją zatem o zapisaną już część w postaci silniowej, czyli iloczyn zmiennych `cyfra` oraz `silnia` (której wartość odpowiada współczynnikowi przy cyfrze):

```
1  silnia ← 1
2  k ← 1
3  dopóki (silnia < x), wykonuj:
4      k ← k + 1
5      silnia ← silnia * k
6  jeżeli silnia > x:
7      silnia ← silnia div k
8      k ← k – 1
9  s ←" "
10 dopóki (k > 0), wykonuj:
11     cyfra ← x div silnia
12     s ← s ◦ tekst(cyfra)
13     x ← x - cyfra * silnia
14     silnia ← ........................................
15     k ← k – 1
```

Ostatnia luka stanowi pomniejszanie silni w celu otrzymania kolejnego współczynnika pozycyjnego – podobny krok wykonujemy w linii 7. W związku z tym na zmiennej `silnia` wykonujemy dzielenie całkowite przez zmienną `k`:

```
1  silnia ← 1
2  k ← 1
3  dopóki (silnia < x), wykonuj:
4      k ← k + 1
5      silnia ← silnia * k
6  jeżeli silnia > x:
7      silnia ← silnia div k
8      k ← k – 1
9  s ←" "
10 dopóki (k > 0), wykonuj:
11     cyfra ← x div silnia
12     s ← s ◦ tekst(cyfra)
13     x ← x - cyfra * silnia
14     silnia ← silnia div k
15     k ← k – 1
```

Schemat oceniania

- **3 pkt** – za poprawne uzupełnienie wszystkich luk,
- **2 pkt** – za poprawne uzupełnienie 3 luk,
- **1 pkt** – za poprawne uzupełnienie 2 luk,
- **0 pkt** – za poprawne uzupełnienie 1 lub 0 luk, lub za brak odpowiedzi.

Słownik

pętla `for`

instrukcja sterująca, która pozwala wielokrotnie wykonać ustalony zestaw poleceń; o liczbie powtórzeń decyduje specjalna zmienna, zwyczajowo nosząca nazwę `i`

pętla `while`

pętla, która wykonuje instrukcje w niej zawarte tak długo, jak zadane jej wyrażenie logiczne jest prawdziwe

sesja

w kontekście stron internetowych: przechowuje przez pewien czas na serwerze szczegóły połączenia między klientem a serwerem, np. dane konta po zalogowaniu

silnia

$n!$ – iloczyn kolejnych liczb naturalnych, mniejszych lub równych n

Prezentacja multimedialna

Zadanie 2. Rozkład na czynniki pierwsze

Michał pracuje w urzędzie statystycznym Republiki Procesorowej. W ostatnich dniach zostały przeprowadzone badania na temat zużycia prądu we wszystkich gospodarstwach domowych. Wyniki zapisano w jednostce kilowatogodzin w pliku `zuzycie.txt`, każde gospodarstwo w osobnej linii. Na polecenie dyrektora zakładu Michał ma rozłożyć zgromadzone dane na czynniki pierwsze.

Plik o rozmiarze 199.00 B w języku polskim

> **Polecenie 1**
>
> Napisz program, który rozłoży dane z pliku `zuzycie.txt` na czynniki pierwsze i zapisze je w odpowiadających liniach pliku `rozklad.txt`, oddzielając poszczególne czynniki pojedynczymi znakami odstępu.

Do oceny oddajesz:

- plik `rozklad.txt` zawierający odpowiedź (dane z pliku `zuzycie.txt` rozłożone na czynniki pierwsze),
- plik(i) z komputerową realizacją zadania (kodem programu).

> **Polecenie 2**
>
> Przedstaw rozwiązanie w formie programu napisanego w wybranym języku: C++, Java lub Python. Odpowiedź do zadania znajdziesz pod prezentacją.

Rozwiązanie

> Rozwiązanie zadania przedstawimy za pomocą pseudokodu.

1

Program z pętlą for może pomóc w uporządkowaniu danych zebranych podczas badań statystycznych.
Źródło: pixabay.com, domena publiczna.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PYbiQMVVj

Rozwiązanie rozpocznijmy od wczytania danych z pliku:

```
zuzycie[0..49] ← wczytaj
dane z pliku "zuzycie.txt"
```

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PYbiQMVVj

Następnie każdą z liczb rozkładamy na czynniki pierwsze, które będziemy na bieżąco zapisywać do pliku wynikowego. Posłuży do tego pętla `dla`:

```
zuzycie[0..49] ← wczytaj
dane z pliku "zuzycie.txt"

dla i = 0, 1, ..., 49
wykonuj:
```

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PYbiQMVVj

Rozkład na czynniki pierwsze wykonamy w pętli `dopóki`, do momentu osiągnięcia przez liczbę wartości 1. Pierwszym możliwym czynnikiem pierwszym jest liczba 2:

```
zuzycie[0..49] ← wczytaj dane z pliku "zuzycie.txt"

dla i = 0, 1, ..., 49 wykonuj:
    czynnik ← 2
    dopóki zuzycie[i] > 1, wykonuj:
```

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PYbiQMVVj

Począwszy od pierwszego możliwego czynnika pierwszego, dzielimy liczbę przez obecnie sprawdzany czynnik, dopóki reszta z dzielenia będzie wynosić 0. Za każdym razem zapisujemy dzielnik do pliku wynikowego razem z pojedynczym znakiem odstępu. Operator `mod` oznacza resztę z dzielenia:

```
zuzycie[0..49] ← wczytaj dane z pliku "zuzycie.txt"

dla i = 0, 1, ..., 49 wykonuj:
    czynnik ← 2
    dopóki zuzycie[i] > 1, wykonuj:
```

```
        dopóki zuzycie[i]
mod czynnik = 0, wykonuj:
            zuzycie[i] ←
zuzycie[i] / czynnik
            zapisz czynnik
do pliku "rozklad.txt"
            zapisz znak
odstępu do pliku
"rozklad.txt"

```

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PYbiQMVVj

Jeżeli obecny czynnik nie dzieli już sprawdzanej liczby, zwiększamy go o 1, przechodząc do kolejnego możliwego czynnika. Całą procedurę powtarzamy do momentu osiągnięcia przez sprawdzaną liczbę wartości 1:

```
zuzycie[0..49] ← wczytaj
dane z pliku "zuzycie.txt"

dla i = 0, 1, ..., 49
wykonuj:
    czynnik ← 2
    dopóki zuzycie[i] > 1,
wykonuj:
        dopóki zuzycie[i]
mod czynnik = 0, wykonuj:
            zuzycie[i] ←
zuzycie[i] / czynnik
            zapisz czynnik
do pliku "rozklad.txt"
            zapisz znak
odstępu do pliku
"rozklad.txt"
        czynnik ← czynnik
+ 1
```

6

Pętla for jest pętlą typu wyliczeniowego, w której stosuje się licznik w postaci liczby całkowitej int, zmieniany o pewien stały krok, dopóki nie zostanie spełniony z góry określony warunek zakończenia pętli.
Źródło: en.wikipedia.org, domena publiczna.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PYbiQMVVj

Przedstawiony algorytm ma jedną poważną wadę: po przekroczeniu pierwiastka sprawdzanej liczby najbliższym ewentualnym czynnikiem pierwszym jest sama liczba (jeśli jest liczbą pierwszą). Poprawmy zatem algorytm, dokładając do zewnętrznej pętli `dopóki` warunek na zmienną `czynnik` oraz instrukcję po pętli przechwytującą przypadek, jeśli sprawdzana liczba jest liczbą pierwszą. W tym celu dołóżmy zmienną `kopia` przechowującą oryginalną wartość sprawdzanej liczby. Po zapisaniu ostatniego czynnika dokładamy znak nowej linii na zakończenie rozkładu danej liczby:

```
1 zuzycie[0..49] ← wczytaj
  dane z pliku "zuzycie.txt"
2
3 dla i = 0, 1, ..., 49
  wykonuj:
4     czynnik ← 2
5     kopia ← zuzycie[i]
```

```
    dopóki zuzycie[i] > 1 oraz czynnik*czynnik <= kopia, wykonuj:
        dopóki zuzycie[i] mod czynnik = 0, wykonuj:
            zuzycie[i] ← zuzycie[i] / czynnik
            zapisz czynnik do pliku "rozklad.txt"
        czynnik ← czynnik + 1
    jeżeli zuzycie[i] > 1:
        zapisz zuzycie[i] do pliku "rozklad.txt"
    zapisz znak nowej linii do pliku "rozklad.txt"
```

Tak zapisany algorytm gwarantuje pełną pulę punktów na egzaminie.

Odpowiedź do zadania

`rozklad.txt`

Plik o rozmiarze 378.00 B w języku polskim

Sprawdź się

Zadanie 3. Konwersja do postaci binarnej

Kasia i Tomek w ramach projektu szkolnego wymieniali korespondencję pocztową z Markiem, mieszkańcem Bajtocji. Niestety, mieszkańcy Bajtocji porozumiewają się w systemie binarnym, podczas gdy systemem ojczystym Kasi i Tomka jest system dziesiętny. W związku z tym wszystkie swoje wiadomości muszą oni konwertować do postaci binarnej, aby Marek mógł je zrozumieć. Jedna z wiadomości zapisana jest w pliku `konwersja.txt`, każda z liczb w osobnej linii.

Plik o rozmiarze 792.00 B w języku polskim

Napisz program, który przetłumaczy dane z pliku `konwersja.txt` do postaci binarnej, a wyniki zapisze do pliku `informacja.txt`.

Do oceny oddajesz:

- plik `informacja.txt` zawierający odpowiedź (dane z pliku `konwersja.txt` zapisane w postaci binarnej),
- 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 swojego programu. Odpowiedź do zadania dla danych z pliku znajdziesz pod sekcją ćwiczeń.

Pokaż ćwiczenia:

> C++

Ćwiczenie 1

Java

Ćwiczenie 2

Python

Ćwiczenie 3

Schemat oceniania

- **2 pkt** – za poprawną odpowiedź,
- **0 pkt** – za błędną odpowiedź lub brak odpowiedzi.

Odpowiedź do zadania

`informacja.txt`

Plik o rozmiarze 2.28 KB w języku polskim

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Pętle warunkowe – zadania maturalne

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres podstawowy i rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

> 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

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.
>
> Zakres podstawowy. Uczeń:
>
> 1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje poprawność programów dla różnych danych; w szczególności programuje algorytmy z punktu I.2);
>
> Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:
>
> 2) stosuje zasady programowania strukturalnego i obiektowego w rozwiązywaniu problemów;
>
> 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):

- Przeanalizujesz sposób rozwiązywania typowych zadań z wykorzystaniem pętli `while`.
- Rozwiążesz samodzielnie kilka zadań, wykorzystując pętlę `while`.
- Prześledzisz schemat oceniania zadań maturalnych z informatyki.

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.** Uczniowie powtarzają wiadomości o pętlach warunkowych.

Faza wstępna:

1. Prowadzący wyświetla na tablicy interaktywnej zawartość sekcji „Wprowadzenie” i omawia cele do osiągnięcia w trakcie lekcji.
2. **Rozpoznanie wiedzy uczniów.** Uczniowie tworzą pytania dotyczące tematu zajęć, na które odpowiedzą w trakcie lekcji.

Faza realizacyjna:

1. **Praca z tekstem.** Uczniowie przystępują do cichego czytania tekstu e-materiału. Indywidualnie zapoznają się z treścią w sekcji „Przeczytaj”. Analizują rozwiązanie zadania 1 i omawiają je na forum klasy.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Prezentacja multimedialna”, uczniowie wspólnie analizują treść zadania, następnie zapoznają się z jego rozwiązaniem omówionym w prezentacji. Na forum klasy nauczyciel wyjaśnia niezrozumiałe kwestie.
3. **Ćwiczenie umiejętności.** Uczniowie indywidualnie wykonują zadanie 3 z sekcji „Sprawdź się”, pisząc program w wybranym języku programowania. Następnie porównują swoje rozwiązanie z inną osobą, programującą w tym samym języku.

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 implementują w wybranym języku programowania rozwiązanie zadania 2 z sekcji „Prezentacja multimedialna”.

Materiały pomocnicze:

- 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 języka C++.
- Oficjalna dokumentacja techniczna dla kompilatora GCC/G++ 4.5 (lub nowszej wersji).
- 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 Eclipse 4.4 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania PyCharm lub IDLE.

Wskazówki metodyczne:

- Uczniowie mogą wykorzystać treści w sekcjach: „Przeczytaj”, „Prezentacja multimedialna”, „Sprawdź się” jako materiał do lekcji powtórkowej.