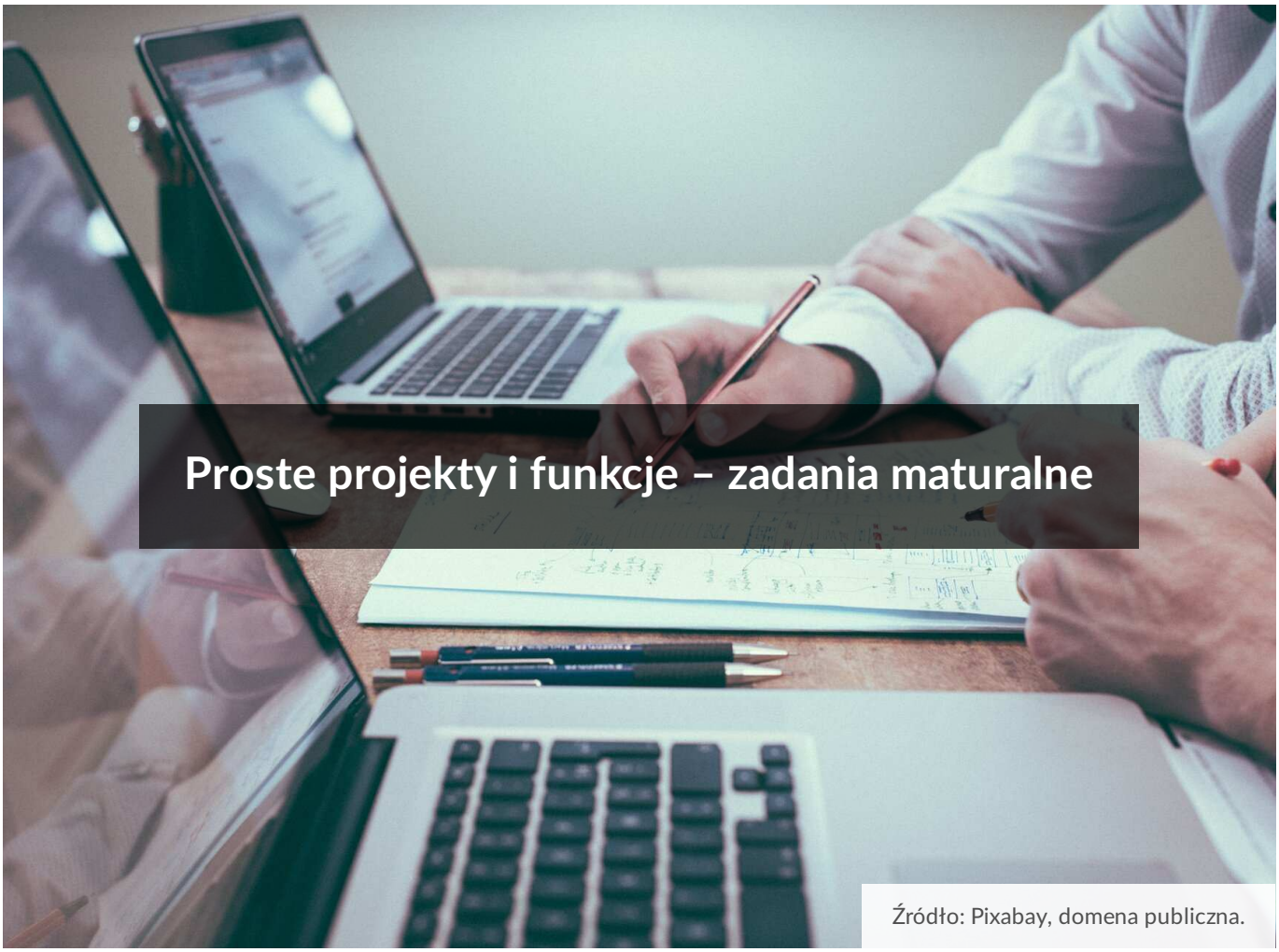
Proste projekty i funkcje – zadania maturalne



Źródło: Pixabay, domena publiczna.

Funkcje pozwalają rozbić program na bloki logiczne odpowiedzialne za wykonywanie określonych zadań. Zastosowanie funkcji sprawia, że kod programu staje się bardziej czytelny, łatwiej jest go modyfikować oraz odnajdować i usuwać pojawiające się w nim błędy. W tym e-materiale zapoznamy się z wykorzystaniem funkcji w zadaniu maturalnym.

Implementację projektów w wybranych językach programowania znajdziesz w e-materiałach:

- Proste projekty i funkcje w języku C++,
- Proste projekty i funkcje w języku Java,
- Proste projekty i funkcje w języku Python.

Więcej zadań? Przejdź do e-materiału Proste projekty i funkcje – ćwiczenia.

Twoje cele

- Przeanalizujesz sposób rozwiązywania zadań maturalnych z wykorzystaniem własnych funkcji.
- Rozwiążesz samodzielnie kilka zadań maturalnych.
- Dowiesz się, na co należy zwrócić uwagę przy rozwiązywaniu zadań.

Przeczytaj

Zadanie 1. Numery PESEL

> Zadanie zostało stworzone przez Centralną Komisję Egzaminacyjną i pojawiło się na egzaminie maturalnym z informatyki (poziom podstawowy, część II) w 2019 roku. Cały arkusz można znaleźć na stronie internetowej CKE.

W pliku `dane.txt` zapisano 1000 numerów PESEL osób urodzonych w latach $1945 - 2003$, po jednym numerze w wierszu.

Fragment pliku `dane.txt` (10 pierwszych wierszy pliku):

```
87062374513
70042949368
57070843873
68100217281
80071113568
58121156393
55042672243
73091424567
89100324122
60062195366
```

Plik `przyklad.txt` zawiera 15 przykładowych numerów PESEL.

Napisz program(y), za pomocą którego(ych) uzyskasz odpowiedzi do poniższych zadań. Odpowiedzi zapisz w pliku `wyniki6.txt`, a odpowiedź do każdego zadania poprzedź numerem oznaczającym to zadanie.

Plik `przyklad.txt`

Plik o rozmiarze 195.00 B w języku polskim

Zadanie 1.1

Przedostatnia, dziesiąta cyfra numeru PESEL określa płeć jego właściciela. W numerach PESEL kobiet cyfra ta jest parzysta, u mężczyzn – nieparzysta. Podaj liczbę numerów PESEL kobiet i liczbę numerów PESEL mężczyzn z pliku `dane.txt`.

Przykład:

Dla numerów PESEL zawartych w pliku `przyklad.txt` odpowiedzią jest 8 numerów dla kobiet i 7 dla mężczyzn.

> **Polecenie 1**
>
> Napisz program, który obliczy, ile numerów PESEL w pliku `dane.txt` należy do kobiet, a ile do mężczyzn, i zapisze odpowiedź w pliku `wyniki6_1.txt` w osobnych liniach.

Do oceny oddajesz:

- plik `wyniki6_1.txt` z odpowiedzią (dwie liczby naturalne; liczba numerów PESEL należąca do kobiet i liczba numerów PESEL należących do mężczyzn)
- plik(i) zawierający(e) kody źródłowe.

Kompletne dane do zadania znajdują się w załączonym poniżej pliku `dane.txt`

Plik o rozmiarze 12.70 KB w języku polskim

> Rozwiąż zadanie dla danych wejściowych znajdujących się w załączonym pliku. Wykorzystaj w tym celu jeden z języków programowania: C++, Java lub Python. Odpowiedź do zadania znajdziesz w osobnym pliku tekstowym umieszczonym po omówieniu przykładowego rozwiązania.

Rozwiązanie

> Na egzaminie maturalnym mamy możliwość wybrania dowolnego z dostępnych języków programowania, jednak tutaj nie będziemy dokonywać implementacji rozwiązania tego zadania w konkretnym języku. Zamiast tego przedstawimy rozwiązanie zapisane za pomocą pseudokodu. Twoim zadaniem będzie przełożenie go na język, w którym programujesz.

Pierwszym krokiem, jaki musimy wykonać, tworząc nasz program, jest wczytanie danych z pliku. Dane te zapiszemy w tablicy o rozmiarze równym liczbie numerów PESEL w pliku. Ponieważ mamy obliczyć liczbę numerów PESEL należących do kobiet i liczbę numerów PESEL należących do mężczyzn, stworzymy również zmienne do przechowywania tych wartości. Zmienne te inicjujemy wartością 0.

```
1 numery_pesel[0..999] ← wczytaj dane z pliku "dane.txt"
2 liczba_kobiet ← 0
3 liczba_mezczyzn ← 0
```

Na płeć osoby, do której należy numer PESEL, wskazuje parzystość jego przedostatniej cyfry. Dane w plikach przechowywane są w postaci tekstowej. Aby korzystać z nich jak z liczb, trzeba je przekonwertować do typu liczbowego. Na potrzeby tego zadania wystarczy znać

zatem wartość wyłącznie przedostatniej cyfry numeru PESEL, zatem stworzymy funkcję pomocniczą, która pozwoli nam na odczytanie wartości danej cyfry zapisanej jako znak. Będzie ona zwracała kod ASCII znaku pomniejszony o 48 (wartość dziesiętna kodu znaku '0').

```
funkcja liczba_ze_znaku(znak)
    ascii(znak) - 48
```

> **Ważne!**
>
> W językach programowania C++, Java oraz Python istnieją funkcje wbudowane (będące odpowiednikiem zaprezentowanej funkcji `liczba_ze_znaku()`, które pozwalają na konwersję znaku na liczbę.
>
> C++
>
> ```
> int(znak)
> ```
>
> Java
>
> ```
> Integer.parseInt(znak)
> ```
>
> Python
>
> ```
> int(znak)
> ```

Po zapewnieniu możliwości traktowania pojedynczych cyfr numerów PESEL jak liczb możemy przystąpić do rozwiązania właściwego zadania. W tym celu będziemy potrzebować pętli, która pozwoli na przejście po całej tablicy z numerami PESEL.

```
numery_pesel[0..999] ← wczytaj dane z pliku "dane.txt"
liczba_kobiet ← 0
liczba_mezczyzn ← 0

dla i = 0, 1, 2, ..., 999 wykonuj
```

Następnie dla każdego numeru PESEL w tablicy sprawdzamy parzystość jego przedostatniej cyfry. Jeżeli jest ona parzysta, oznacza to, że PESEL należy do kobiety i musimy zwiększyć

o 1 wartość zmiennej `liczba_kobiet`. Jeśli zaś liczba ta nie jest parzysta, rozważany numer PESEL należy do mężczyzny i musimy zwiększyć o 1 wartość zmiennej `liczba_mezczyzn`.

```
numery_pesel[0..999] ← wczytaj dane z pliku "dane.txt"
liczba_kobiet ← 0
liczba_mezczyzn ← 0

dla i = 0, 1, 2, ..., 999 wykonuj
    jeżeli liczba_ze_znaku(numery_pesel[i][9]) mod 2 = 0
        liczba_kobiet ← liczba_kobiet + 1
    w przeciwnym razie
        liczba_mezczyzn ← liczba_mezczyzn + 1
```

Gdy już policzymy, ile numerów PESEL należy do kobiet, a ile do mężczyzn, pozostaje nam zapisać te wyniki do pliku.

Na końcu zamykamy pliki.

Oto kompletny kod rozwiązujący to zadanie:

```
funkcja liczba_ze_znaku(znak)
    zwróć kod ASCII podanego argumentu - 48

numery_pesel[0..999] ← wczytaj dane z pliku "dane.txt"
liczba_kobiet ← 0
liczba_mezczyzn ← 0

dla i = 0, 1, 2, ..., 999 wykonuj
    jeżeli liczba_ze_znaku(numery_pesel[i][9]) mod 2 = 0
        liczba_kobiet ← liczba_kobiet + 1
    w przeciwnym razie
        liczba_mezczyzn ← liczba_mezczyzn + 1

zapisz liczba_kobiet do pliku "wyniki6_1.txt"
zapisz liczba_mezczyzn do pliku "wyniki6_1.txt" w nowej linii
zamknij plik "wyniki6_1.txt"
zamknij plik "dane.txt"
```

Schemat oceniania

- **3 pkt** – za podanie poprawnej odpowiedzi.
- **2 pkt** – za podanie tylko poprawnej liczby mężczyzn lub kobiet.
- **0 pkt** – za błędną odpowiedź albo za brak odpowiedzi.

Uwaga: Nie przyznaje się 1 pkt.

Odpowiedź

Odpowiedź do zadania została zawarta w pliku tekstowym dostępnym w załączniku.

Plik o rozmiarze 8.00 B w języku polskim

Zadanie 1.2

Cyfry trzecia i czwarta numeru PESEL oznaczają miesiąc urodzenia, przy czym osoby urodzone przed rokiem 2000 mają numer miesiąca podany w sposób naturalny, czyli od 01 do 12, a osoby urodzone w roku 2000 i później mają do numeru miesiąca dodaną liczbę 20, czyli od 21 do 32. Dla danych z pliku `dane.txt` podaj, ile osób łącznie urodziło się w listopadzie.

Przykład:

Dla numerów PESEL zawartych w pliku `przyklad.txt` odpowiedzią jest liczba 2.

> **Polecenie 2**
>
> Napisz program, który obliczy, ile numerów PESEL w pliku `dane.txt` należy osób, które urodziły się w listopadzie, i zapisze odpowiedź w pliku `wyniki6_2.txt`.

Do oceny oddajesz:

- plik `wyniki6_2.txt` z odpowiedzią (liczba naturalna; liczba numerów PESEL w pliku `dane.txt` należących osób, które urodziły się w listopadzie)
- plik(i) zawierający(e) kody źródłowe

Kompletne dane do zadania znajdują się w załączonym pliku `dane.txt`

Plik o rozmiarze 12.70 KB w języku polskim

> Rozwiąż zadanie dla danych wejściowych znajdujących się w załączonym pliku. Wykorzystaj w tym celu jeden z języków programowania: C++, Java lub Python. Odpowiedź do zadania znajdziesz w osobnym pliku tekstowym umieszczonym po omówieniu przykładowego rozwiązania.

Rozwiązanie

Podobnie jak w zadaniu 1.1, tutaj również zaczynamy od wczytania danych z pliku i stworzenia zmiennej do przechowywania wyniku:

```
numery_pesel[0..999] ← wczytaj dane z pliku "dane.txt"
wynik ← 0
```

Do rozwiązania tego zadania potrzebne nam będą dwie funkcje pomocnicze. Jedna z nich została przedstawiona w poprzednim zadaniu. Jest to funkcja odczytująca wartość cyfry zapisanej jako znak:

```
funkcja liczba_ze_znaku(znak)
    zwróć kod ASCII zmiennej znak - 48
```

Drugą funkcją pomocniczą będzie funkcja, która z podanego w argumencie numeru PESEL będzie wyodrębniała liczbę oznaczającą miesiąc urodzenia jego właściciela. Funkcja będzie zwracała miesiąc wyznaczony z trzeciej i czwartej cyfry numeru PESEL, do czego wykorzystamy przedstawioną wyżej funkcję `liczba_ze_znaku`. Miesiąc urodzenia właściciela numeru PESEL będzie obliczany jako suma czwartej cyfry numeru PESEL i jego trzeciej cyfry pomnożonej przez 10.

```
funkcja odczytaj_miesiac(pesel)
    a1 ← 10
    a0 ← 1
    x1 ← liczba_ze_znaku(pesel[2])
    x0 ← liczba_ze_znaku(pesel[3])
    zwróć a1 * x1 + a0 * x0
```

Następnie, skoro mamy już przygotowane funkcje pomocnicze, możemy przystąpić do rozwiązania właściwego zadania. W tym celu potrzebna nam będzie pętla, która przejdzie po wszystkich odczytanych z pliku numerach PESEL.

```
numery_pesel[0..999] ← wczytaj dane z pliku "dane.txt"
wynik ← 0

dla i = 0, 1, 2, ..., 999 wykonuj
```

Kiedy mamy już gotową pętlę, następnym krokiem będzie zliczenie wystąpień miesiąca listopad w przetwarzanych numerach PESEL. W tym celu dla każdego numeru PESEL

musimy odczytać liczbę symbolizującą ten miesiąc i przyrównać ją do oczekiwanych wartości, czyli 11 i 31. W przypadku ich dopasowania zwiększamy przygotowany na początku rozwiązywania licznik przechowujący wynik.

```
numery_pesel[0..999] ← wczytaj dane z pliku "dane.txt"
wynik ← 0

dla i = 0, 1, 2, ..., 999 wykonuj
    miesiac ← odczytaj_miesiac(numery_pesel[i])
    jeżeli miesiac = 11 lub miesiac = 31
        wynik ← wynik + 1
```

Skoro mamy już ustaloną liczbę osób urodzonych w listopadzie, pozostaje zapisać wynik do pliku `wyniki6_2.txt`.

Na końcu zamykamy pliki.

Oto pełen kod służący do rozwiązania zadania:

```
funkcja liczba_ze_znaku(znak)
    zwróć kod ASCII podanego argumentu - 48

funkcja odczytaj_miesiac(pesel)
    a1 ← 10
    a0 ← 1
    x1 ← liczba_ze_znaku(pesel[2])
    x0 ← liczba_ze_znaku(pesel[3])
    zwróć a1 * x1 + a0 * x0

numery_pesel[0..999] ← wczytaj dane z pliku "dane.txt"
wynik ← 0

dla i = 0, 1, 2, ..., 999 wykonuj
    miesiac ← odczytaj_miesiac(numery_pesel[i])
    jeżeli miesiac = 11 lub miesiac = 31
        wynik ← wynik + 1

zapisz wynik do pliku "wyniki6_2.txt"
zamknij plik "wyniki6_2.txt"
zamknij plik "dane.txt"
```

Schemat oceniania

- **3 pkt** – za podanie poprawnej odpowiedzi.
- **2 pkt** – za podanie liczby urodzonych w listopadzie tylko w XX lub tylko w XXI wieku.
- **2 pkt** – jeśli zdający pomyli numery cyfr i zamiast 3 i 4 sprawdzi 4 i 5 (otrzyma wynik 80) lub pomyli cyfry 3 z 4 (wynik 79).
- **1 pkt –** za sprawdzenie przez zdającego tylko jednej cyfry.
- **0 pkt** – za błędną odpowiedź albo za brak odpowiedzi.

Odpowiedź

Odpowiedź do zadania została zawarta w pliku tekstowym dostępnym w załączniku.

Plik o rozmiarze 2.00 B w języku polskim

> **Ważne!**
>
> Rozwiązanie zadania możesz zapisać, wykorzystując jeden program.
>
> Propozycja rozwiązania:

```
funkcja liczba_ze_znaku(znak)
    zwróć kod ASCII podanego argumentu - 48

funkcja podpunkt_1(numery_pesel)
    liczba_kobiet ← 0
    liczba_mezczyzn ← 0

    dla i = 0, 1, 2, ..., długość(numery_pesel) - 1 wykonuj
        jeżeli liczba_ze_znaku(numery_pesel[i][9]) mod 2 = 0
            liczba_kobiet ← liczba_kobiet + 1
        w przeciwnym razie
            liczba_mezczyzn ← liczba_mezczyzn + 1

    zwróć liczba_kobiet, liczba_mezczyzn

funkcja odczytaj_miesiac(pesel)
    a1 ← 10
    a0 ← 1
    x1 ← liczba_ze_znaku(pesel[2])
    x0 ← liczba_ze_znaku(pesel[3])
    zwróć a1 * x1 + a0 * x0

funkcja podpunkt_2(numery_pesel)
```

```
    wynik ← 0
    dla i = 0, 1, 2, ..., długość(numery_pesel) - 1 wykonuj
        miesiac ← odczytaj_miesiac(numery_pesel[i])
        jeżeli miesiac = 11 lub miesiac = 31
            wynik ← wynik + 1
    zwróć wynik

numery_pesel[0..999] ← wczytaj dane z pliku "dane.txt"

liczba_kobiet, liczba_mezczyzn ← podpunkt_1(numery_pesel)
wynik ← podpunkt_2(numery_pesel)

zapisz liczba_kobiet do pliku "wyniki6_1.txt"
zapisz liczba_mezczyzn do pliku "wyniki6_1.txt" w nowej linii
zapisz wynik do pliku "wyniki6_2.txt"

zamknij plik "wyniki6_1.txt"
zamknij plik "wyniki6_2.txt"
zamknij plik "dane.txt"
```

Słownik

konwersja typów

specjalna konstrukcja programistyczna umożliwiająca zmianę typu pewnej danej na inny typ danych; jej przykładem jest zamiana liczby całkowitej na napis reprezentujący tę liczbę lub odwrotnie; określana również jako rzutowanie typu, przekształcenie typu

Prezentacja multimedialna

Zadanie 1.3

> Zadanie zostało stworzone przez Centralną Komisję Egzaminacyjną i pojawiło się na egzaminie maturalnym z informatyki (poziom podstawowy, część II) w 2019 roku. Cały arkusz można znaleźć na stronie internetowej CKE.

Poprawność numeru PESEL można obliczyć według podanego poniżej algorytmu:

$$1 \cdot a_1 + 3 \cdot a_2 + 7 \cdot a_3 + 9 \cdot a_4 + 1 \cdot a_5 + 3 \cdot a_6 + 7 \cdot a_7 + 9 \cdot a_8 + 1 \cdot a_9 + 3 \cdot a_{10} + a_{11}$$

gdzie a_i to kolejne cyfry numeru PESEL.

Jeżeli powyższy wynik jest liczbą podzielną przez 10, to numer PESEL jest poprawny, w przeciwnym razie jest błędny.

Podaj wszystkie błędne numery PESEL w pliku `dane.txt`.

Przykład:

Wśród numerów PESEL zawartych w pliku `przyklad.txt` błędny jest numer `97092746487`.

Plik `przyklad.txt`

Plik o rozmiarze 195.00 B w języku polskim

Napisz program, który wyznaczy wszystkie niepoprawne numery PESEL w pliku `dane.txt` i zapisze je w pliku `wynik6_3.txt` w kolejnych liniach.

Do oceny oddajesz:

- plik `wyniki6_3.txt` z odpowiedzią (błędnymi numerami PESEL)
- plik(i) zawierający(e) kody źródłowe

Kompletne dane do zadania znajdują się w załączonym pliku `dane.txt`

Plik o rozmiarze 12.70 KB w języku polskim

Polecenie 1

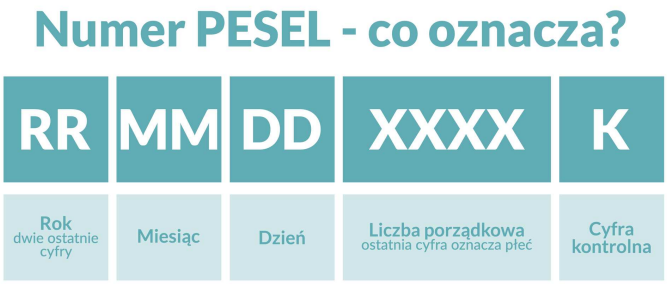
Rozwiąż zadanie dla danych wejściowych znajdujących się w załączonym pliku. Wykorzystaj w tym celu jeden z języków programowania: C++, Java lub Python. Odpowiedź do zadania znajdziesz w osobnym pliku tekstowym umieszczonym po omówieniu przykładowego rozwiązania.

Przeanalizuj kolejne kroki prezentacji i porównaj je ze swoim rozwiązaniem.

Rozwiązanie

Rozwiązanie zadania przedstawimy w formie pseudokodu.

1



Każda cyfra numeru PESEL ma konkretne znaczenie.
Źródło: Contentplus Sp. z o. o., CC BY-SA 3.0.

Materiał audio dostępny pod adresem:

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

Zapiszemy za pomocą pseudokodu algorytm, który wypisze wszystkie błędne numery PESEL z podanego pliku.

2

Materiał audio dostępny pod adresem:

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

Zaczniemy od wczytania danych z podanego pliku `dane.txt`.

3

Materiał audio dostępny pod adresem:

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

Kolejnym krokiem jest stworzenie funkcji `liczba_ze_znaku`, która odczyta wartości cyfry zapisanej w formie znaku. W funkcji wykorzystamy funkcję wbudowaną `ascii(znak)`, która będzie zwracała kod ASCII znaku. Następnie zwrócona wartość zostanie pomniejszona o kod znaku „0”, co w rezultacie da poszukiwaną cyfrę.

Odpowiedniki funkcji `ascii(znak)` zostały opisane w sekcji „Przeczytaj”.

4

Materiał audio dostępny pod adresem:

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

Aby sprawdzić wszystkie zapisane w pliku numery PESEL, należy sprawdzić zawartość całej tablicy `numery_pesel`. Zastosujemy do tego pętlę, dzięki której będziemy mieli dostęp do każdego z analizowanych numerów PESEL.

5

Materiał audio dostępny pod adresem:

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

Napisaliśmy już pętlę, za pomocą której będziemy przetwarzać pobrane z pliku numery PESEL. Napiszmy zatem funkcję pozwalającą na sprawdzenie poprawności numeru PESEL. Numer PESEL jest poprawny, jeżeli wynik działania obliczony za pomocą wzoru przedstawionego na slajdzie jest podzielny przez 10.

$$1 \cdot a_1 + 3 \cdot a_2 + 7 \cdot a_3 + 9 \cdot a_4 + 1 \cdot a_5 + 3 \cdot a_6 + 7 \cdot a_7 + 9 \cdot a_8$$

gdzie a_i to kolejne cyfry numeru PESEL.

6

Materiał audio dostępny pod adresem:

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

Napiszmy funkcję `czy_poprawny` przyjmującą jako argument zmienną `pesel`. Funkcja zwróci wartość logiczną `prawda`, jeśli numer PESEL będzie poprawny (w przeciwnym razie zwróci wartość `fałsz`). Tworzymy zmienną `suma_kontrolna` do przechowywania aktualnej wartości obliczanego wyrażenia. Następnie tworzymy pętlę, w której przetwarzamy po kolei cyfry numeru PESEL, a następnie w zależności od ich położenia (indeksu) mnożymy je przez odpowiednie współczynniki liczbowe i dodajemy do zmiennej `suma_kontrolna`. Po wyznaczeniu sumy kontrolnej sprawdzamy, czy jest ona podzielna przez 10. Jeśli tak, zwracamy wartość `prawda`, w przeciwnym wypadku zwracamy wartość `fałsz`. W funkcji `czy_poprawny` wykorzystamy napisaną wcześniej funkcję pomocniczą `liczba_ze_znaku` do zamiany poszczególnych analizowanych znaków numeru PESEL na cyfry.

7

Materiał audio dostępny pod adresem:

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

Przystępujemy do identyfikacji błędnych numerów PESEL. Dla każdego numeru PESEL wywołujemy funkcję `czy_poprawny` sprawdzającą jego poprawność. Jeżeli funkcja zwróci wartość `fałsz`, wówczas sprawdzany PESEL zapisujemy do pliku wynikowego `wynik6_3.txt`. Pamiętajmy o tym, by każdy niepoprawny PESEL zapisać w osobnej linii. Zamykamy pliki.

8

MIESIĄC	OZNACZENIE MIESIĄCA ZALEŻNE OD ROKU URODZENIA				
	1800-1899	1900-1999	2000-2099	2100-2199	2200-2299
Styczeń	81	01	21	41	61
Luty	82	02	22	42	62
Marzec	83	03	23	43	63
Kwiecień	84	04	24	44	64
Maj	85	05	25	45	65
Czerwiec	86	06	26	46	66
Lipiec	87	07	27	47	67
Sierpień	88	08	28	48	68
Wrzesień	89	09	29	49	69
Październik	90	10	30	50	70
Listopad	91	11	31	51	71
Grudzień	92	12	32	52	72

Oznaczenia miesiąca różnią się zależnie od tego, w którym roku urodził się posiadacz numeru PESEL.

Materiał audio dostępny pod adresem:

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

Całe rozwiązanie zadania znajduje się na slajdzie:

Schemat oceniania

- **4 pkt** – za podanie poprawnego zestawienia 9 numerów PESEL
- **3 pkt** – za podanie zestawienia szukanych numerów PESEL z 1 błędem.
- **2 pkt** – za podanie 5-7 niepoprawnych numerów PESEL.
- **1 pkt** – znalezienie przynajmniej jednego niepoprawnego numeru PESEL.
- **0 pkt** – za błędną odpowiedź albo za brak odpowiedzi.

> Schemat oceniania pochodzi z arkusza odpowiedzi do egzaminu maturalnego z informatyki (poziom podstawowy) z 2019 roku. Cały arkusz można znaleźć na stronie internetowej Centralnej Komisji Egzaminacyjnej.

Odpowiedź

Odpowiedź do zadania została zawarta w pliku tekstowym dostępnym w załączniku.

`wyniki6_3.txt`
Plik o rozmiarze 115.00 B w języku polskim

> **Polecenie 2**
>
> Przygotuj notatkę na temat tego, kiedy i dlaczego wprowadzono numery PESEL.

> **Polecenie 3**
>
> Dodaj do swojego programu komentarze tak, żeby był zrozumiały dla osoby, która nie potrafi programować.

Sprawdź się

Zadanie 2

Na planecie OxkhO przyjęto, że imiona zaczynają się i kończą taką samą literą. Do tradycji należy również to, że pierwsza i ostatnia litera imienia zawsze są zapisywane jako wielkie. Po tym, jak mieszkańcy planety nawiązali kontakt z Ziemią, zapożyczyli część imion. W pliku `kosmiczne_imiona.txt` zapisano 100 najpopularniejszych imion, jakie nadawano w danym roku dzieciom urodzonym na planecie OxkhO.

Plik składa się ze 100 wierszy. W każdym z nich zapisano za pomocą małych i wielkich liter alfabetu łacińskiego łańcuchy znaków.

Plik o rozmiarze 765.00 B w języku polskim

Napisz program, który sprawdzi, które imiona są tradycyjnymi dla tej planety, czyli zaczynają i kończą się tą samą wielką literą. Przyjmij założenie, że imię musi składać się z przynajmniej dwóch liter.

Odpowiedzią powinien być ciąg napisów `TAK` lub `NIE`, w zależności od spełnienia warunku przez dane imię. Ciąg napisów `TAK` i `NIE` oddzielonych pojedynczymi znakami odstępu powinien zostać zapisany w pliku `wynik6_4.txt`.

Do oceny oddajesz:

- plik `wyniki6_4.txt` zawierający odpowiedź (wyrazy `TAK` i `NIE` oddzielone pojedynczymi znakami odstępu)
- plik(i) zawierający(e) kody źródłowe

> Rozwiąż zadanie dla danych wejściowych znajdujących się w załączonym pliku.
> Wykorzystaj w tym celu jeden z języków programowania: C++, Java lub Python.
> Odpowiedź do zadania znajdziesz w osobnym pliku tekstowym umieszczonym pod sekcją ćwiczeń.

Działanie programu możesz przetestować na przykładowych danych znajdujących się w testerkach poniżej.

Pokaż ćwiczenia:

Ćwiczenie 1

JĘZYK C++

Twoje zadania

1. Napisz program, który sprawdza, czy imiona przechowywane w zmiennej `imiona` zaczynają i kończą się tą samą wielką literą. Przyjmij założenie, że imię składa się co najmniej z dwóch liter.

```
#include <iostream>

using namespace std;

// tutaj dopisz kod

int main() {
    string imiona [] = {
        "AaaaaA",
        "Ababss",
        "Sdckmlk",
        "MMMMM",
        "kKK2KKK",
        "Lklksld",
```

```
```

Ćwiczenie 2

JĘZYK JAVA

Twoje zadania

1. Napisz program, który sprawdza, czy imiona przechowywane w zmiennej `imiona` zaczynają i kończą się tą samą wielką literą. Przyjmij założenie, że imię składa się co najmniej z dwóch liter.

```java
public class Main {

    // tutaj dopisz swój kod

    public static void main(String [] args) {
        String [] imiona = {
            "AaaaaA",
            "Ababss",
            "Sdckmlk",
            "MMMMM",
            "kKK2KKK",
            "Lklksld",
            "Ls",
            "JjjJ",
```

```
1
```

Ćwiczenie 3

JĘZYK PYTHON

Twoje zadania

1. Napisz program, który sprawdza, czy imiona przechowywane w zmiennej `imiona` zaczynają i kończą się tą samą wielką literą. Przyjmij założenie, że imię składa się co najmniej z dwóch liter.

```
imiona = ["AaaaaA",
    "Ababss",
    "Sdckmlk",
    "MMMMM",
    "kKK2KKK",
    "Lklksld",
    "Ls",
    "JjjJ",
    "P",
    "UuuuuuU"]

# Tutaj dodaj kod. Do wypisania użyj poniższej linijki.
# print("TAK", end = " ")
# lub
```

```

```

Odpowiedź do zadania

Odpowiedź do zadania znajduje się w załączonym pliku.

Plik o rozmiarze 399.00 B w języku polskim

Zadanie 3

W pliku `napisy.txt` znajduje się 100 różnych wyrazów.

Napisz program sprawdzający dla wszystkich napisów znajdujących się w pliku `napisy.txt`, czy cały napis składa się z tych samych znaków, a jeżeli tak, to jaki jest to znak. Odpowiedzią powinien być ciąg napisów `NIE` lub `TAK` z występującym znakiem. Napis `TAK` powinien być połączony z literą pojedynczym myślnikiem (np. `TAK-k`). Napisy powinny być oddzielone pojedynczym znakiem odstępu i zapisane w pliku `wynik6_5.txt`. Sprawdzanie napisu ma być umieszczone w osobnej funkcji. Uwzględnij wielkość liter.

Plik o rozmiarze 984.00 B w języku polskim

Do oceny oddajesz:

- plik `wyniki6_5.txt` zawierający odpowiedź (wyrazy `TAK` połączone z literą i `NIE` oddzielone pojedynczymi znakami odstępu)
- plik(i) zawierający(e) kody źródłowe

> Rozwiąż zadanie dla danych wejściowych znajdujących się w załączonym pliku.
> Wykorzystaj w tym celu jeden z języków programowania: C++, Java lub Python.
> Odpowiedź do zadania znajdziesz w osobnym pliku tekstowym umieszczonym pod sekcją ćwiczeń.

Działanie programu możesz przetestować na przykładowych danych znajdujących się w testerkach poniżej.

Ćwiczenie 4

JĘZYK C++

Twoje zadania

1. Program w języku C++ wypisuje po spacji dla każdego napisu wartość `TAK-znak` lub `NIE` w zależności od tego, czy napis składa się z takich samych znaków.

```cpp
#include <iostream>

using namespace std;

// tutaj dopisz funkcję

int main() {
    string napisy[] = {
        "aaaaa",
        "ababss",
        "sdckmlk",
        "MMMMM",
        "KKK2KKK",
        "lklksld",
```

```cpp

```

Ćwiczenie 5

JĘZYK JAVA

Twoje zadania

1. Program w języku Java wypisuje po spacji dla każdego napisu wartość `TAK-znak` lub `NIE` w zależności od tego, czy napis składa się z takich samych znaków.

```java
public class Main {

    // tutaj dopisz kod

    public static void main(String [] args) {
        String [] napisy = {
            "aaaaa",
            "ababss",
            "sdckmlk",
            "MMMMM",
            "KKK2KKK",
            "lklksld",
            "ls",
            "jjj",
```

```
1
```

Ćwiczenie 6

JĘZYK PYTHON

Twoje zadania

1. Zaimplementuj program w języku Python, który wypisuje po spacji dla każdego napisu wartość `TAK-znak` lub `NIE` w zależności od tego, czy napis składa się z takich samych znaków.

```
napisy = ["aaaaa", "ababss", "sdckmlk", "MMMMM", "KKK2KKK",
"lklksld", "ls", "jjj", "p", "uuuuuuu"]

# Tutaj dodaj kod. Do wypisania użyj poniższej linijki.
# print()
```

```

```

Odpowiedź do zadania

Odpowiedź do zadania znajduje się w załączonym pliku.

Plik o rozmiarze 421.00 B w języku polskim

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Proste projekty i funkcje – zadania maturalne

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres 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;

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 zadań maturalnych z wykorzystaniem własnych funkcji.
- Rozwiążesz samodzielnie kilka zadań maturalnych.
- Dowiesz się, na co należy zwrócić uwagę przy rozwiązywaniu zadań.

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;
- telefony z dostępem do internetu;
- 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: „Proste projekty i funkcje – zadania maturalne". Uczniowie zapoznają się z treściami w sekcji „Przeczytaj" w kontekście programowania.

Faza wstępna:

1. Wyświetlenie przez nauczyciela tematu i celów lekcji. Określenie wiążących dla uczniów kryteriów sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel prosi wybranego ucznia lub uczniów o przedstawienie sytuacji problemowej związanej z tematem lekcji oraz w odniesieniu do poznanych języków programowania.

Faza realizacyjna:

1. Uczniowie analizują przykład z sekcji „Przeczytaj" i powtarzają zaprezentowane rozwiązanie na swoim komputerze.
2. **Praca z multimedium.** Nauczyciel czyta polecenie nr 1 z sekcji „Prezentacja multimedialna". Prosi uczniów, aby wykonali je w parach. Następnie wybrana osoba prezentuje propozycję odpowiedzi, a pozostali uczniowie ustosunkowują się do niej. Nauczyciel w razie potrzeby uzupełnia ją, udziela też uczniom informacji zwrotnej.
3. **Ćwiczenie umiejętności.** Prowadzący zapowiada uczniom, że będą rozwiązywać ćwiczenie nr 2: „Napisz program sprawdzający w każdym napisie w tablicy, czy wartość pierwszego i ostatniego znaku w danym napisie jest taka sama. Odpowiedzią powinien być ciąg napisów `TAK` lub `NIE`, w zależności od spełnienia warunku przez dany napis. Użyj w programie przynajmniej jednej funkcji własnej. Przykład odpowiedzi: `TAK TAK NIE TAK NIE NIE` dla danych wejściowych: `{"aa", "anna", "mama", "sos", "smok", "nazwisko"}` Zrealizuj powyższe zadanie w jednym z wybranych języków programowania." z sekcji „Sprawdź się". Każdy z uczniów robi to samodzielnie. Po ustalonym czasie następuje porównanie napisanych kodów podczas wspólnego omówienia rozwiązań.
4. Liga zadaniowa – uczniowie pracując w parach, wykonują ćwiczenie nr 3: „Napisz program sprawdzający w każdym napisie z tablicy, czy cały napis składa się z tych samych znaków, a jeżeli tak, to jaki jest to znak. Odpowiedzią powinien być ciąg napisów `NIE` lub `TAK` z występującym znakiem. Sprawdzanie napisu ma być umieszczone w osobnej funkcji. Uwzględnij wielkość liter. Przykład odpowiedzi: `TAK-a NIE TAK-m TAK-K NIE NIE` dla danych wejściowych: `{"aa", "anna", "mmmm", "KKK", "smok", "nazwisko"}` Zrealizuj powyższe zadanie w jednym z wybranych języków programowania." z sekcji „Sprawdź się", a następnie dzielą się swoimi wynikami przez porównywanie napisanego kodu z inną grupą, która również zakończyła zadanie.

Faza podsumowująca:

1. Nauczyciel wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji podsumowuje przebieg zajęć, a także wskazuje mocne i słabe strony pracy uczniów.

2. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności, omawia ewentualne problemy podczas rozwiązania ćwiczeń z programowania w języku C++.

Praca domowa:

1. Uczniowie proponują alternatywne rozwiązanie ćwiczenia 2 z sekcji „Sprawdź się”.

Materiały pomocnicze:

- 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 języka Java SE 8 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Eclipse 4.4 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla języka Python 3 (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.