Sortowanie szybkie – zadania maturalne

Bibliografia:

- Źródło: oprac. własne.

Źródło: Kelvyn Ornettte Sol Marte, domena publiczna.

W algorytmie sortowania szybkiego (ang. *quick sort*) stosowana jest metoda „dziel i zwyciężaj". Polega ona na dzieleniu zadania na mniejsze i łatwiejsze do rozwiązania podproblemy.

Sortowanie szybkie jest przydatne zwłaszcza podczas porządkowania dużej liczby danych należących do nieokreślonego lub obszernego zakresu. Średnia złożoność obliczeniowa algorytmu sortowania szybkiego wynosi $O(n \log n)$.

Więcej informacji o sortowaniu szybkim znajdziesz w e-materiale Sortowanie szybkie.

Implementacja tego algorytmu w wybranych językach programowania została przedstawiona w e-materiałach:

- Sortowanie szybkie w języku C++,
- Sortowanie szybkie w języku Java,
- Sortowanie szybkie w języku Python.

Twoje cele

- Przeanalizujesz przykładowe zadania maturalne wykorzystujące algorytm sortowania szybkiego zapisany za pomocą pseudokodu.

- Zaimplementujesz algorytm sortowania szybkiego przy rozwiązywaniu różnych problemów.
- Rozwiążesz przykładowe zadania maturalne, stosując algorytm *quick sort*.

Przeczytaj

Algorytm sortowania szybkiego

Sortowanie szybkie polega na wybraniu elementu rozdzielającego (tzw. pivot) oraz podzieleniu tablicy zawierającej dane na dwie części. W jednej zostają umieszczone elementy o wartościach większych niż pivot, w drugiej zaś — elementy o wartościach od niego mniejszych. Następnie elementy mniejsze i większe sortowane są osobno według tej samej zasady. Sortowanie odbywa się rekurencyjnie do momentu, gdy każda z podtablic zostanie uporządkowana. Dokładne omówienie algorytmu sortowania szybkiego zostało przedstawione w e-materiale Sortowanie szybkie.

Zadanie 1. Wystawa obrazów

W pewnym muzeum organizowana jest wystawa obrazów poświęconych figurom geometrycznym. Okoliczne muzea wypożyczyły placówce części własnych ekspozycji pasujących do tematyki wystawy.

Zadanie 1.1

Muzeum chce ustalić, w którym roku powstało najwięcej dzieł z obecnej wystawy. Obrazy te zostaną oprawione w złote ramy.

Plik `lata.txt` zawiera 76 wierszy zawierających rok powstania poszczególnych obrazów pokazywanych na wystawie (każdy rok zapisano w osobnym wierszu).

Plik o rozmiarze 456.00 B w języku polskim

Napisz program, który wyznaczy rok, z którego pochodzi najwięcej dzieł, oraz wskaże, ile złotych ram na obrazy z tego roku powinno przygotować muzeum. Przyjmij, że jest tylko jeden rok z maksymalną liczbą powstałych dzieł. Zapisz obie wartości do pliku `ramy.txt`.

Do oceny oddajesz:

- plik `ramy.txt` zawierający odpowiedź (rok, w którym powstało najwięcej prezentowanych obrazów, oraz liczbę złotych ram potrzebnych do ich oprawienia)
- plik(i) z komputerową realizacją zadania (kodem programu)

> **Praca domowa**
>
> Przedstaw rozwiązanie zadania w postaci programu w wybranym języku (C++, Java lub Python). Zadbaj o prawidłowe wczytanie i zapisanie danych z/do pliku tekstowego.

> Odpowiedź do zadania znajdziesz w osobnym pliku umieszczonym pod omówieniem pseudokodu.

Rozwiązanie

> Rozwiązanie zadania przedstawimy w postaci pseudokodu, ponieważ na egzaminie maturalnym można korzystać z wybranego języka programowania: C++, Java lub Python.

W celu łatwiejszego zliczenia obrazów pochodzących z danego roku, posortujemy dane z pliku. Użyjemy do tego algorytmu sortowania szybkiego.

> **Ważne!**
>
> W przedstawionym algorytmie jako punkt rozdzielający przyjmuje się ostatnią wartość w tablicy.

```
1 funkcja sortowanie_szybkie(Lata[], lewy, prawy)
2     jeżeli lewy < prawy:
3         punkt_podziału ← podział(Lata, lewy, prawy)
4         sortowanie_szybkie(Lata, lewy, punkt_podziału - 1)
5         sortowanie_szybkie(Lata, punkt_podziału + 1, prawy)
```

1. Definiujemy funkcję, która będzie dzielić tablicę na dwie części, a następnie sortować je osobno. **[linia 1]**
2. Sprawdzamy, czy tablica, którą mamy podzielić, nie jest jednoelementowa (jeżeli jest, indeksy `lewy` i `prawy` są sobie równe). **[linia 2]**
3. Wyznaczamy punkt podziału (pivot); jednocześnie zapisujemy dane w tablicy z uwzględnieniem punktu podziału. **[linia 3]**
4. Rekurencyjnie uruchamiamy funkcję podziału dla utworzonych podtablic. **[linie 4, 5]**

```
1 funkcja podział(Lata[], lewy, prawy)
2     pivot ← Lata[prawy]
3     mniejsza ← lewy - 1
4     dla większa = lewy, lewy + 1, ... prawy - 1 wykonuj:
5         jeżeli Lata[większa] < pivot:
6             mniejsza ← mniejsza + 1
7             zamień Lata[mniejsza] z Lata[większa]
8     zamień Lata[mniejsza + 1] z Lata[prawy]
9     zwróć (mniejsza + 1)
```

1. Definiujemy funkcję podziału, która będzie ustawiać wartości mniejsze niż pivot po lewej stronie tablicy, zaś wartości większe – po prawej; następnie funkcja zwróci

pozycję wartości rozdzielającej tablicę. **[linia 1]**
2. Ustawiamy pozycję elementu rozdzielającego oraz wartość zmiennej `mniejsza`. Pamiętajmy o tym, że przed zamianą pozycji komórek (w linii 7) dokonujemy inkrementacji wartości zmiennej `mniejsza`, a zatem musimy nadać jej wartość o 1 mniejszą od pierwszej sprawdzanej pozycji. **[linie 2, 3]**
3. Inicjujemy pętlę, w której wartość `większa` będzie odpowiadać kolejnym indeksom tabeli. **[linia 4]**
4. Sprawdzamy w każdej iteracji pętli, czy wartość w tablicy dla indeksu `większa` ma wartość mniejszą niż pivot. Jeżeli tak, inkrementujemy wartość zmiennej `mniejsza`, a następnie zamieniamy miejscami komórki o indeksach `większa` i `mniejsza`. **[linie od 5 do 7]**
5. Wstawiamy pivot w odpowiednie miejsce w tablicy — tak, by wszystkie elementy po lewej stronie były od niego mniejsze, a wszystkie po prawej — większe. **[linia 8]**
6. Zwracamy pozycję pivota i kończymy działanie funkcji. **[linia 9]**

Przejdźmy do właściwej części zadania. Na początku wczytujemy i sortujemy dane. Następnie zapisujemy algorytm, który przeszukuje posortowaną tablicę, by znaleźć rok powstania największej liczby obrazów oraz wskazać liczbę potrzebnych do ich oprawienia złotych ram:

```
1  Lata[0..75] ← wczytaj dane z pliku "lata.txt"
2  sortowanie_szybkie(Lata, 0, długość(Lata) - 1)
3
4  rok ← Lata[0]
5  liczba ← 1
6  maks_rok ← rok
7  maks_liczba ← 1
8
9  dla i = 1, 2, ..., długość(Lata) - 1 wykonuj:
10     jeżeli Lata[i] = rok:
11         liczba ← liczba + 1
12     w przeciwnym razie:
13         rok ← Lata[i]
14         liczba ← 1
15
16     jeżeli liczba >= maks_liczba:
17         maks_liczba ← liczba
18         maks_rok ← rok
19
20 zapisz maks_rok maks_liczba do pliku "ramy.txt"
```

1. Wczytujemy dane z pliku. **[linia 1]**
2. Wywołujemy funkcję, w której został zaimplementowany algorytm sortowania szybkiego. Jako dane wejściowe podajemy: tablicę lat, pozycję, od której należy rozpocząć sortowanie, oraz pozycję końcową. Tablice indeksujemy od zera, zgodnie z ich reprezentacją w komputerze. **[linia 2]**
3. Deklarujemy potrzebne zmienne. **[linie 4–7]**
4. Korzystając z pętli, sprawdzamy poniższe warunki dla każdego roku:
    - jeżeli jest taki sam jak poprzednio rozpatrywany, dodajemy jedno wystąpienie do licznika wystąpień; **[linia 11]**
    - jeżeli jest inny niż poprzednio rozpatrywany, zapisujemy nowy rok jako aktualnie rozpatrywany i ustawiamy licznik wystąpień na 1. **[linie 13–14]**
5. Z każdą iteracją pętli sprawdzamy również, czy dany rok pojawia się najwięcej razy. Jeżeli tak, zapisujemy ten rok i liczbę jego wystąpień w odpowiednich zmiennych. **[linie 16–18]**
6. Na koniec zapisujemy wyniki do odpowiedniego pliku. **[linie 20–21]**

> Odpowiedź do zadania dla danych zapisanych w pliku tekstowym `lata.txt`:
>
> Plik o rozmiarze 7.00 B w języku polskim

Zadanie 1.2

Po zakończeniu wystawy trzeba przygotować eksponaty do zwrócenia muzeom, z których zostały wypożyczone. Sporządzono w tym celu listę — numer każdego jej wiersza oznacza pozycję obrazu na wystawie, zaś w wierszu zapisany jest sześciocyfrowy kod z oznaczeniem danego muzeum i obrazu:

- cyfry na pozycjach od 1. do 3. (od lewej strony) oznaczają numer muzeum,
- cyfry na pozycjach od 4. do 6. to oznaczenie obrazu w danym muzeum.

Im większa jest liczba stanowiąca numer muzeum, tym dalej się ono znajduje. Obrazy należy ułożyć w takiej kolejności, aby najpierw zostały wysłane dzieła należące do najbardziej odległych muzeów. W rezultacie wszystkie eksponaty wrócą na swoje miejsce w podobnym czasie. Przyjmijmy, że z każdego muzeum wypożyczono tylko jeden obraz.

Plik `lista.txt` zawiera 76 liczb (każda zapisana w osobnym wierszu), stanowiących numery poszczególnych dzieł. Numery te się nie powtarzają.

Plik o rozmiarze 531.00 B w języku polskim

Napisz program, który uporządkuje obrazy w kolejności ich wysyłki oraz zapisze numery muzeów, do których wysyłane są obrazy, w pliku `kolejność.txt`.

> **Przykład 1**
>
> Dla danych wejściowych:

```
114165
158168
130185
125172
```

- przykładowy zapis do pliku wynikowego będzie wyglądał następująco:

```
158
130
125
114
```

Do oceny oddajesz:

- plik `kolejność.txt` zawierający odpowiedź (liczby całkowite dodatnie będące numerami muzeów w odpowiedniej kolejności)
- plik(i) z komputerową realizacją zadania (kodem programu)

> **Praca domowa**
>
> Przedstaw rozwiązanie zadania w postaci programu w wybranym języku (C++, Java lub Python). Zadbaj o prawidłowe wczytanie danych z pliku tekstowego do programu. Odpowiedź do zadania znajdziesz w osobnym pliku umieszczonym pod omówieniem pseudokodu.

Rozwiązanie

> Rozwiązanie przedstawimy w postaci pseudokodu, ponieważ na egzaminie maturalnym można korzystać z języków C++, Java lub Python.

Aby rozwiązać zadanie, musimy nieco zmienić kod, który przedstawiliśmy w podpunkcie 1.

W celu zmiany porządku sortowania na nierosnący należy zmodyfikować funkcję `podział()`, ponieważ to ona odpowiada za dzielenie elementów na podtablice mniejszych i większych elementów.

> **Ważne!**
>
> W pseudokodzie tablica `Lata` została zastąpiona tablicą `Kod`.

```
funkcja podział(Kod[], lewy, prawy)
    pivot ← Kod[prawy]
    mniejsza ← lewy - 1
    dla większa = lewy, lewy + 1, ... prawy - 1 wykonuj:
        jeżeli Kod[większa] > pivot:
```

```
6               mniejsza ← mniejsza + 1
7               zamień(Kod[mniejsza], Kod[większa])
8       zamień(Kod[mniejsza + 1], Kod[prawy])
9       zwróć (mniejsza + 1)
```

- Modyfikujemy wyrażenie warunkowe; sprawdzamy, czy liczba na pozycji wskazywanej przez indeks `większa` jest większa niż element rozdzielający. Jeżeli tak, postępujemy dalej tak samo jak w poprzednim zadaniu. **[linia 5]**

Po posortowaniu musimy dodatkowo umieścić trzy pierwsze cyfry każdego kodu w tablicy pomocniczej, którą zapiszemy w pliku `kolejność.txt`.

```
1 Kod[0..75] ← wczytaj dane z pliku "lista.txt"
2 sortowanie_szybkie(Kod, 0, długość(Kod) - 1)
3
4 Pomocnicza[0..długość(Kod)-1]
5 dla i = 0, 1, ..., długość(Kod) - 1 wykonuj:
6     Pomocnicza[i] ← kod_muzeum(Kod[i])
7
8 zapisz Pomocnicza[0..długość(Kod)-1] do pliku "kolejność.txt"
```

- Po posortowaniu danych z pliku `lista.txt` umieszczamy kod każdego muzeum w tablicy pomocniczej. Korzystamy w tym celu z pomocniczej funkcji `kod_muzuem()` zdefiniowanej poniżej. **[linie 4, 5, 6]**
- Zapisujemy kody poszczególnych muzeów w pliku `kolejność.txt` i kończymy działanie programu. **[linia 8]**

```
1 funkcja kod_muzeum(kod)
2     kod ← kod div 1000
3     zwróć kod
```

1. Definiujemy funkcję, która będzie przekształcać sześciocyfrowy kod w liczbę trzycyfrową, złożoną z trzech pierwszych cyfr kodu. **[linia 1]**
2. Kod dzielimy całkowicie przez tysiąc — pozbywamy się w ten sposób trzech ostatnich cyfr (pozostaje liczba trzycyfrowa). **[linia 2]**
3. Zwracamy wartość zmiennej `kod`. **[linia 3]**

> Odpowiedź do zadania dla danych zapisanych w pliku tekstowym `lista.txt`:
>
> Plik o rozmiarze 303.00 B w języku polskim

> **Ważne!**

W pseudokodzie wykorzystaliśmy operator `div`, który jest operatorem dzielenia całkowitego.
Ważne!

W zadaniach maturalnych polegających na pisaniu programu w wybranym języku programowania zdający może stosować funkcje wbudowane wybranego przez siebie języka. Dla funkcji zwracającej posortowaną tablicę liczb całkowitych `tab` o długości `n` są to odpowiednio:

C++

```
sort(tab, tab + n)
```

Java

```
Arrays.sort(tab)
```

Python

```
sorted(tab)
```

Słownik

indeks komórki

numer (pozycja) komórki w tablicy
inkrementacja

zwiększenie wartości zmiennej o jeden
iteracja pętli

inaczej: cykl pętli; pojedyncze wykonanie instrukcji zapisanych w pętli
komórka tablicy

kontener służący do przechowywania pojedynczego elementu tablicy
pivot

element sortowanej tablicy wybierany według określonego schematu (może to być np. element środkowy), ze względu na który tablica rozdzielana jest na podtablice - po lewej stronie tego elementu znajdują się wartości nie większe od pivota, a po prawej nie mniejsze od niego
rekurencja

> technika programowania polegająca na odwoływaniu się procedur lub funkcji do samych siebie, aż do momentu spełnienia warunku podstawowego

Prezentacja multimedialna

Zadanie 2. Firma

W firmie „Hexadecymalni” każdy ze 150 pracowników ma przypisany indywidualny kod zapisany w systemie szesnastkowym, którym posługuje się na terenie przedsiębiorstwa. Identyfikator zbudowany jest w następujący sposób:

- cyfra na pozycji 1. oznacza numer stanowiska w firmie (od 1 do 5),
- cyfra na pozycji 2. określa płeć (od 2 do F; liczby pierwsze są przypisywane kobietom, a złożone mężczyznom),
- cyfry na pozycjach od 3. do 5. to rok urodzenia.

Kody pracowników ułożone są losowo, co powoduje duży chaos w dziale księgowym. Na wniosek szefa tego działu masz za zadanie:

- kody pracowników posortować rosnąco według stanowisk;
- w obrębie każdego stanowiska posortować pracowników niemalejąco względem czasu pozostałego do odejścia danej osoby na emeryturę (w pełnych latach pozostałych do przepracowania).

Ważne, aby uwzględnić różny wiek emerytalny dla kobiet i mężczyzn. Kobiety zatrudnione w firmie „Hexadecymalni” odchodzą na emeryturę wraz z końcem roku poprzedzającego ich 60. rok życia, zaś mężczyźni — wraz z końcem roku poprzedzającego ich 65. rok życia. Przyjmij, że szef działu księgowości zlecił ci zadanie w 2022 roku.

> **Przykład 1**
>
> Pracownik o numerze `167B5` jest mężczyzną (liczba $6_{(16)} = 6_{(10)}$ jest liczbą złożoną). Urodził się on w roku $7B5_{(16)} = 1973_{(10)}$, zatem 65. urodziny będzie obchodzić w roku 2038. Oznacza to, że jego ostatnim przepracowanym rokiem będzie rok 2037, a więc pełne przepracowane lata to lata od roku 2023 do roku 2037. Zgodnie z treścią zadania do emerytury pozostało mu 15 pełnych lat pracy.

W pliku `pracownicy.txt` zapisano kody wszystkich pracowników, każdy w osobnej linii. Każdej osobie w obrębie jednego stanowiska pozostał inny czas do uzyskania emerytury.

Plik o rozmiarze 899.00 B w języku polskim

> **Polecenie 1**
>
> Napisz program, który uporządkuje identyfikatory pracowników według przedstawionych zasad. Wyniki zapisz do pliku `wyniki2.txt`.

Do oceny oddajesz:

- plik `wyniki2.txt` zawierający odpowiedź (kody pracowników pogrupowane rosnąco według stanowiska i posortowane rosnąco względem czasu pozostałego do emerytury w obrębie jednego stanowiska)
- plik(i) z komputerową realizacją zadania

> 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 w osobnym pliku umieszczonym pod omówieniem pseudokodu.

Rozwiązanie

> **Polecenie 2**
>
> Zapoznaj się z rozwiązaniem zadania przedstawionym w postaci pseudokodu.

1

Materiał audio dostępny pod adresem:

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

Rozpoczniemy od wczytania kodów pracowników do tablicy.

2

Materiał audio dostępny pod adresem:

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

Następnie należy przejrzeć wszystkie pozycje w tablicy, aby wydobyć z niej kody pracowników zajmujących te same stanowiska. Utworzymy zatem nową tablicę przechowującą liczbę pracowników na danym stanowisku oraz tablicę dwuwymiarową zawierającą pracowników podzielonych na stanowiska.

Potem należy każdą z podtablic posortować osobno.

3

Materiał audio dostępny pod adresem:

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

Uruchamiamy sortowanie dla każdej pozycji tablicy `Stanowiska` i zapisujemy posortowane podtablice do pliku wynikowego.

4

Materiał audio dostępny pod adresem:

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

Zdefiniujmy funkcję sortowania szybkiego.

5

Materiał audio dostępny pod adresem:

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

Sprawdzamy, czy sortowana tablica nie jest jednoelementowa (jeżeli jest, to `lewy = prawy`).

6

Materiał audio dostępny pod adresem:

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

W funkcji `podział` wyznaczamy wartość rozdzielającą, a dane ustawiane są tak, że na lewo od `punkt_podziału` znajdują się liczby mniejsze od `punkt_podziału`, a na prawo większe. Następnie funkcja sortująca jest wywoływana rekurencyjnie osobno dla danych mniejszych oraz większych.

7

Materiał audio dostępny pod adresem:

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

Definiujemy funkcję podziału, która będzie dzielić tablicę na wartości większe i mniejsze od pivota, a na końcu zwróci jego pozycję.

8

Materiał audio dostępny pod adresem:

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

Ustawiamy początkowe wartości dla zmiennych `pivot` oraz `mniejsza` (wartość `mniejsza` zostaje ustawiona przed pierwszą pozycją w tablicy, ponieważ jest ona inkrementowana przed wykonaniem instrukcji w pętli, zawartych w kolejnym kroku).

9

Materiał audio dostępny pod adresem:

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

Przechodzimy po tablicy, inkrementując wartość `większa`. Przy porównaniu wykorzystamy funkcję `emerytura()`, którą zdefiniujemy w jednym z następnych kroków. Jeżeli okaże się, że osoba z kodem `Stanowisko[większa]` ma wcześniej odejść na emeryturę niż osoba z kodem `pivot`, to zamieniamy kod `Stanowisko[większa]` z wartością pod indeksem `mniejsza`.

10

Materiał audio dostępny pod adresem:

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

Na końcu ustawiamy na właściwym miejscu element będący punktem podziału i zwracamy jego pozycję.

11

Materiał audio dostępny pod adresem:

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

Zdefiniujmy teraz funkcję `emerytura()`. Będzie ona wyznaczać, w jakim wieku dana osoba odchodzi na emeryturę, a następnie obliczać, ile jeszcze pozostało tej osobie pełnych lat pracy w firmie.

Funkcja `heks_do_dziesietny()` konwertuje podaną liczbę jako ciąg znaków w kodzie szesnastkowym na jej reprezentację dziesiętną.

Funkcja `liczba()` konwertuje podany znak w kodzie szesnastkowym na jego reprezentację dziesiętną.

Funkcja `czy_pierwsza()` sprawdza, czy liczba jest pierwsza.

Cały kod wygląda następująco:

> Odpowiedź do zadania dla danych zapisanych w pliku tekstowym `pracownicy.txt`:
>
> Plik o rozmiarze 898.00 B w języku polskim

Sprawdź się

Zadanie 3. Skoki

Drużyna lekkoatletyczna „SportowIT” prowadzi nabór na skoczków w dal. Aby zostać zakwalifikowanym, uczestnik musi wykonać co najmniej jeden skok, który może wielokrotnie poprawiać.

Każdy skok został oznaczony pięciocyfrowym numerem, w którym:

- cyfry na pozycjach 1. i 2. to oznaczenie uczestnika,
- cyfry na pozycjach od 3. do 5. to długość skoku (gdzie np. 521 oznacza skok o długości 5,21 m).

Plik `skoczkowie.txt` zawiera oznaczenia 350 skoków. Korzystając z wybranego języka programowania, napisz program, który znajdzie trzy najlepsze wyniki, posortuje je w kolejności od najwyższego, a następnie wskaże numery zawodników, którzy je uzyskali. Wyniki zapisz w pliku `wyniki.txt`.

Plik o rozmiarze 2.39 KB w języku polskim

> **Przykład 1**
>
> Dla zawodników o następujących danych:
>
> ```
> 03617
> 55445
> 37591
> 76721
> ```
>
> program powinien zapisać do pliku:
>
> ```
> 76
> 3
> 37
> ```

Do oceny oddajesz:

- plik(i) z komputerową realizacją zadania
- plik `wyniki.txt` zawierający odpowiedź do zadania (trzy liczby naturalne będące numerami zawodników, którzy zdobyli trzy najlepsze wyniki, w kolejności od najlepszego wyniku)

> 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.

W przedstawionych ćwiczeniach część danych została umieszczona w tablicach. Wykorzystaj je do rozwiązania zadania.

Pokaż ćwiczenia: 

> C++

Ćwiczenie 1

Twoje zadania

1. Program powinien odpowiednio uporządkować dane dotyczące skoków, a następnie wypisać numery uczestników, którzy osiągnęli trzy najlepsze wyniki, zaczynając od zwycięzcy.

Java

Ćwiczenie 2

Twoje zadania

1. Program powinien odpowiednio uporządkować dane dotyczące skoków, a następnie wypisać numery uczestników, którzy osiągnęli trzy najlepsze wyniki, zaczynając od zwycięzcy.

Python

Ćwiczenie 3

Twoje zadania

1. Program powinien odpowiednio uporządkować dane dotyczące skoków, a następnie wypisać numery uczestników, którzy osiągnęli trzy najlepsze wyniki, zaczynając od zwycięzcy.

Odpowiedź do zadania dla danych zawartych w pliku tekstowym `skoczkowie.txt`:

Plik o rozmiarze 7.00 B w języku polskim

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Sortowanie szybkie – 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:

1) w zależności od problemu rozwiązuje go, stosując metodę wstępującą lub zstępującą;

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;

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

3) objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów, w szczególności:

c) metodę dziel i zwyciężaj (jednoczesne znajdowanie minimum i maksimum, sortowanie przez scalanie i szybkie),

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 przykładowe zadania maturalne wykorzystujące algorytm sortowania szybkiego zapisany za pomocą pseudokodu.
- Zaimplementujesz algorytm sortowania szybkiego przy rozwiązywaniu różnych problemów.
- Rozwiążesz przykładowe zadania maturalne, stosując algorytm *quick sort*.

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: „Sortowanie szybkie – zadania maturalne". Uczniowie zapoznają się z treściami w sekcji „Przeczytaj" w kontekście programowania.

Faza wstępna:

1. Uczniowie metodą burzy mózgów przypominają sobie najważniejsze informacje dotyczące sortowania szybkiego.
2. Wyświetlenie przez nauczyciela tematu i celów lekcji. Określenie wiążących dla uczniów kryteriów sukcesu.
3. **Rozpoznanie wiedzy uczniów.** Nauczyciel zadaje uczniom pytania dotyczące ich aktualnego stanu wiedzy w obszarze poruszanego tematu i programowania, np.
   – na czym polega algorytm sortowania szybkiego?
   – co to jest pivot?
   – co to jest iteracja pętli?
   – jak nazywamy zwiększenie wartości zmiennej o jeden?
   Chętni uczniowie udzielają na nie odpowiedzi.

Faza realizacyjna:

1. **Praca z tekstem.** Jeżeli przygotowanie uczniów do lekcji jest niewystarczające, nauczyciel prosi o indywidualne zapoznanie się z treścią zawartą w sekcji „Przeczytaj". Każdy uczestnik zajęć podczas cichego czytania wynotowuje najważniejsze kwestie poruszane w tekście.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Prezentacja multimedialna". Uczniowie wspólnie zapoznają się z jego treścią. Następnie wykonują zadanie nr 2 w wybranym przez siebie języku programowania.

3. **Ćwiczenia umiejętności.** Uczniowie wykonują zadanie nr 3 z sekcji „Sprawdź się". Ich zadaniem jest napisanie programu, który uporządkuje dane dotyczące skoków, a następnie poda trzy najlepsze wyniki. Na koniec program wypisze oznaczenia trzech skoczków, którzy wykonali te skoki.
   Zadania realizują w jednym z dostępnych języków programowania.

Faza podsumowująca:

1. Nauczyciel ponownie wyświetla na tablicy temat lekcji zawarty w sekcji „Wprowadzenie" i inicjuje krótką rozmowę na temat zrealizowanych celów (czego uczniowie się nauczyli).

Praca domowa:

1. Uczniowie implementują algorytmy z zadań 1.1 i 1.2 z sekcji „Przeczytaj" w wybranym języku programowania.

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:

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