Sortowanie przez wybieranie – zadania maturalne

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

Sortowanie przez wybieranie – zadania maturalne

Źródło: Brienne Hong, domena publiczna.

> W tym e-materiale powtarzamy wiadomości ze szkoły podstawowej.

Z e-materiału Sortowanie przez wybieranie wiesz już, jak działa algorytm sortowania przez wybór. Potrafisz również zaimplementować go w wybranym języku programowania. W tym e-materiale rozwiążesz zadania typu maturalnego dotyczące tego zagadnienia.

Implementacje algorytmu sortowania przez wybieranie w różnych językach programowania zostały omówione w e-materiałach:

- Sortowanie przez wybieranie w języku C++,
- Sortowanie przez wybieranie w języku Java,
- Sortowanie przez wybieranie w języku Python.

> **Twoje cele**
>
> - Przeanalizujesz przykładowe zadania maturalne z zakresu sortowania przez wybór.
> - Napiszesz program, wykorzystując algorytm sortowania przez wybór w danym zadaniu maturalnym.
> - Scharakteryzujesz działanie algorytmu sortowania przez wybieranie.

Przeczytaj

Zadanie 1. Anagramy

Anagram to słowo utworzone z innego słowa poprzez przestawienie kolejności liter. Przez słowo rozumiemy tu dowolny ciąg liter alfabetu łacińskiego.

Przykłady anagramów:

- dla słowa barok – korba, robak, arobk, rokab, orkab...
- dla słowa ranty – tyran, narty, ntyra, natyr, ytnar...

Plik tekstowy `anagram.txt` zawiera 200 wierszy. W każdym wierszu znajduje się po pięć słów o długości od 1 do 20 znaków. Słowa oddzielone są znakiem spacji.

Plik o rozmiarze 6.37 KB w języku polskim

> **Przykład 1**
>
> Przykładowa zawartość pliku:
>
> - abcd cdba dbac cbad dcba
> - barbakan xle ala foto otof
> - smok ayszkm lampa ayszkm bakara
> - skok arabanta oko agnieba dyskietka

Napisz program, który w pliku `anagram.txt` wyszuka wszystkie wiersze, w których wszystkie słowa są anagramami pierwszego słowa w danym wierszu, i zapisze je do pliku `znalezione.txt`.

Do oceny oddajesz:

- plik `znalezione.txt` z odpowiedzią (plik tekstowy zawierający wiersze z pliku `anagramy.txt`, w których wszystkie słowa są anagramami pierwszego słowa z danego wiersza)
- plik(i) z komputerową realizacją programu.

> Zadanie zostało opracowane przez Centralną Komisję Egzaminacyjną i pojawiło się na egzaminie maturalnym z informatyki w maju 2010 roku (poziom rozszerzony). Cały arkusz można znaleźć na stronie internetowej CKE.
> Przedstaw rozwiązanie zadania w postaci programu w języku C++, Java lub Python.

Rozwiązanie

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

Sprawdzenie, czy wyrazy są anagramami, możemy wykonać przez posortowanie znaków łańcuchów w kolejności alfabetycznej i porównanie wyników — jeżeli będą one takie same, to znaczy, że są anagramami.

Zaczniemy od napisania funkcji sortującej, której użyjemy w głównej części kodu. Przyjmie ona jeden argument będący łańcuchem znaków — czyli słowem, które chcemy posortować. Znaki w łańcuchach, podobnie jak tablice, indeksujemy od zera, tak jak jest to reprezentowane w komputerze. W programie skorzystamy z sortowania przez wybieranie.

```
funkcja sortowanie(wyraz)
```

Na początku zadeklarujemy zmienną `n`, która przyjmie wartość długości sortowanego ciągu znaków. Wartość tę pobierzemy za pomocą funkcji `długość()`, której implementacja dostępna jest w każdym języku programowania, jaki możemy wybrać na egzaminie maturalnym.

Następnie zapisujemy pętlę, która przeiteruje po wszystkich znakach słowa `wyraz`.

```
funkcja sortowanie(wyraz)
    n ← długość(wyraz)

    dla i = 0, 1, ..., n - 1 wykonuj:
```

Zainicjujemy teraz zmienną `indeks`, w której będziemy przechowywać indeks najmniejszego znaku z nieposortowanej części ciągu znaków. Nadamy jej wartość `i`.

Następnie zastosujemy kolejną pętlę, w której przeiterujemy po wartościach od `i + 1` do `n - 1`. W tej pętli będziemy szukać indeksu najmniejszego znaku z nieposortowanej części tablicy.

```
funkcja sortowanie(wyraz)
    n ← długość(wyraz)

    dla i = 0, 1, ..., n - 1 wykonuj:
        indeks ← i
        dla j = i + 1, i + 2, ..., n - 1 wykonuj:
```

Wewnątrz funkcji porównujemy znaki pod indeksami `j` oraz `indeks`. W celu porównywania znaków użyjemy funkcji `ascii()` zwracającej liczbę odpowiadającą danemu

znakowi w kodzie ASCII. Funkcja jest możliwa do zaimplementowania w każdym języku dostępnym na egzaminie maturalnym.

```
funkcja sortowanie(wyraz)
    n ← długość(wyraz)

    dla i = 0, 1, ..., n - 1 wykonuj:
        indeks ← i
        dla j = i + 1, i + 2, ..., n - 1 wykonuj:
            jeżeli ascii(wyraz[j]) < ascii(wyraz[indeks]):
```

Jeżeli znak pod indeksem `j` jest mniejszy od znaku o indeksie `indeks`, wówczas zmiennej `indeks` przypisujemy wartość `j`.

Po zakończeniu wewnętrznej pętli zamienimy znak `wyraz[i]` ze znakiem o najmniejszej wartości kodu ASCII z nieposortowanego ciągu znaków, czyli ze znakiem `wyraz[indeks]`.

```
funkcja sortowanie(wyraz)
    n ← długość(wyraz)

    dla i = 0, 1, ..., n - 1 wykonuj:
        indeks ← i
        dla j = i + 1, i + 2, ..., n - 1 wykonuj:
            jeżeli ascii(wyraz[j]) < ascii(wyraz[indeks]):
                indeks ← j
        zamień wyraz[i] z wyraz[indeks]
```

Po wykonaniu się pętli zewnętrznej funkcja zwróci posortowany łańcuch znaków `wyraz`.

```
funkcja sortowanie(wyraz)
    n ← długość(wyraz)

    dla i = 0, 1, ..., n - 1 wykonuj:
        indeks ← i
        dla j = i + 1, i + 2, ..., n - 1 wykonuj:
            jeżeli ascii(wyraz[j]) < ascii(wyraz[indeks]):
                indeks ← j
        zamień wyraz[i] z wyraz[indeks]
    zwróć wyraz
```

Przejdźmy teraz do właściwej części rozwiązania. Zaczynamy od wczytania danych z pliku.

```
słowa[0...199][0...4] ← wczytaj dane z pliku "anagramy.txt"
```

Następnie tworzymy pętlę, która przejdzie po wszystkich wierszach. Wewnątrz deklarujemy zmienną `anagram`, która będzie przechowywać informację, czy wszystkie słowa z wiersza są anagramami. Ustawiamy jej wartość na prawdę. Deklarujemy również zmienną `pierwsze` przechowującą posortowane pierwsze słowo z wiersza.

```
słowa[0...199][0...4] ← wczytaj dane z pliku "anagramy.txt"

dla i = 0, 1, ..., 199 wykonuj
    anagram ← prawda
    pierwsze ← sortowanie(słowa[i][0])
```

W kolejnej pętli przechodzimy po wszystkich słowach z wiersza oprócz pierwszego. Jeżeli wynik sortowania danego słowa będzie różny od zmiennej `pierwsze`, oznacza to, że w wierszu tym na pewno nie występują same anagramy, więc ustawiamy wartość zmiennej `anagram` na fałsz i przerywamy pętlę.

```
słowa[0...199][0...4] ← wczytaj dane z pliku "anagramy.txt"

dla i = 0, 1, ..., 199 wykonuj:
    anagram ← prawda
    pierwsze ← sortowanie(słowa[i][0])

    dla j = 1, 2, ..., 4 wykonuj:
        jeżeli sortowanie(słowa[i][j]) != pierwsze:
            anagram ← fałsz
            przerwij pętlę
```

Po zakończeniu wewnętrznej pętli sprawdzamy wartość zmiennej `anagram`. Jeśli jest ona równa prawdzie, przepisujemy cały analizowany wiersz do pliku wynikowego.

```
słowa[0...199][0...4] ← wczytaj dane z pliku "anagramy.txt"

dla i = 0, 1, ..., 199 wykonuj:
    anagram ← prawda
```

```
    pierwsze ← sortowanie(słowa[i][0])
    dla j = 1, 2, ..., 4 wykonuj:
        jeżeli sortowanie(słowa[i][j]) != pierwsze:
            anagram ← fałsz
            przerwij pętlę
    jeżeli anagram = prawda:
        dopisz słowa[i] do pliku "znalezione".txt
```

Schemat oceniania

- **6 pkt** – za poprawną zawartość pliku `znalezione.txt` zawierającego wszystkie wiersze z anagramami
- **2 pkt** – za zawartość pliku `znalezione.txt` z jednym błędem (błędny jeden wiersz lub brak jednego wiersza)
- **0 pkt** – za zawartość pliku `znalezione.txt` z dwoma błędami lub więcej albo brak odpowiedzi

Odpowiedź

Opowiedź do zadania dla danych z pliku `anagramy.txt` znajduje się w pliku `znalezione.txt`:

Plik o rozmiarze 229.00 B w języku polskim

Słownik

> **anagram**
>
> wyraz, wyrażenie lub całe zdanie, które powstało przez przestawienie liter bądź sylab innego wyrazu lub zdania, wykorzystujące wszystkie litery (głoski bądź sylaby) materiału wyjściowego

Prezentacja multimedialna

Zadanie 2. Czas na emeryturę

Piotr zarządza firmą Bitownicy Pracy, specjalizującą się w pozyskiwaniu pracowników fizycznych dla firm, które zajmują się m.in. budowami, pracami porządkowymi czy organizacją imprez masowych. Firma składa się z 50 oddziałów rozsianych po całym świecie, w których pracuje po 30 osób. W związku z niechęcią Piotra do zatrudniania młodych i niedoświadczonych osób, kadra przedsiębiorstwa staje się coraz starsza.

Na polecenie Piotra został sporządzony raport o wieku każdego z pracowników ze wszystkich oddziałów. Dane zostały zawarte w pliku `pracownicy.txt`, każdy oddział w osobnej linii, liczby oznaczające liczbę lat życia pracowników w ramach jednego oddziału oddzielone są pojedynczym znakiem odstępu. Lata życia wszystkich pracowników firmy należą do przedziału $\langle 18, 65 \rangle$.

Plik o rozmiarze 4.39 KB w języku polskim

Piotr chce się dowiedzieć, w jakim przedziale wiekowym są pracownicy w każdym z oddziałów. Nie interesują go jednak obserwacje odstające, a dokładnie pięć najmniejszych i pięć największych liczb.

Napisz program, który dla każdego wiersza z pliku `pracownicy.txt` wyznaczy parę liczb, pomiędzy którymi znajdują się lata życia pracowników z danego oddziału, nie uwzględniając pięciu najmniejszych i pięciu największych wartości. Obie liczby zapisz do pliku `przedziały.txt`, każda para w osobnej linii, oddzielone pojedynczym znakiem odstępu.

Do oceny oddajesz:

- plik `przedziały.txt` z odpowiedzią (plik tekstowy zawierający 50 par liczb, oznaczających przedziały wiekowe dla odpowiadających danych z pliku `pracownicy.txt`, bez uwzględniania pięciu najmniejszych i pięciu największych wartości z danego oddziału)
- plik(i) z komputerową realizacją programu (kodem źródłowym)

> **Polecenie 1**
>
> Przedstaw rozwiązanie zadania w postaci programu napisanego 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 rozwiązania.

Rozwiązanie

Polecenie 2

Zapoznaj się z rozwiązaniem zadania. Zostało ono przedstawione w postaci pseudokodu, ponieważ na egzaminie maturalnym można korzystać z wybranego języka programowania: C++, Java lub Python.

1

Na egzaminie maturalnym możesz spotkać się z wieloma typami zadań, np. zadaniami z luką, zadaniami krótkiej odpowiedzi oraz zadaniami rozszerzonej odpowiedzi. Są to zadania otwarte. Na egzaminie pojawiają się jednak również zadania zamknięte. Prezentowane zadanie jest zadaniem rozszerzonej odpowiedzi i wymaga napisania programu.
Źródło: Pixabay, CC0.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PD5nxw9Vf

W celu znalezienia przedziałów wiekowych pracowników bez uwzględniania pięciu najmniejszych i pięciu największych wartości wystarczy znaleźć odpowiednio szósty najmniejszy i szósty największy wiek dla każdego oddziału. Możemy w tym celu użyć sortowania przez wybieranie, które w każdy przebiegu zewnętrznej pętli znajduje kolejny najmniejszy element z tablicy danych i wstawia go na odpowiednie miejsce. Na początku wczytajmy dane z pliku:

```
pracownicy[0..49][0..29] ←
wczytaj dane z pliku
"pracownicy.txt"
```

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PD5nxw9Vf

Zapisujemy zewnętrzną pętlę przechodzącą po wszystkich wierszach z pliku. Wewnątrz użyjemy zmodyfikowanego algorytmu sortowania przez wybieranie. Nie musimy porządkować całej tablicy — wystarczy zrobić to dla pierwszych sześciu i ostatnich sześciu wartości:

```
pracownicy[0..49][0..29] ←
wczytaj dane z pliku
"pracownicy.txt"

dla i = 0, 1, ..., 49
wykonuj:
    dla j = 0, 1, ..., 5
wykonuj:
```

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PD5nxw9Vf

Deklarujemy zmienną, która będzie przechowywać indeks z najmniejszym elementem nieposortowanej części tablicy:

```
pracownicy[0..49][0..29] ←
wczytaj dane z pliku
"pracownicy.txt"

```

```
dla i = 0, 1, ..., 49
wykonuj:
    dla j = 0, 1, ..., 5
wykonuj:

        minimum ← j
```

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PD5nxw9Vf

Następnie zgodnie z zasadą sortowania przez wybieranie przechodzimy po wszystkich elementach nieposortowanej części tablicy – iterujemy po indeksach od `j + 1` do `29 - j`, ponieważ będziemy wstawiać na odpowiedni koniec zarówno najmniejszy, jak i największy element:

```
pracownicy[0..49][0..29] ←
wczytaj dane z pliku
"pracownicy.txt"

dla i = 0, 1, ..., 49
wykonuj:
    dla j = 0, 1, ..., 5
wykonuj:

        minimum ← j
        dla k = j + 1, j +
2, ..., 29 - j wykonuj:
```

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PD5nxw9Vf

5

Wewnątrz pętli porównujemy analizowany element z wartością pod indeksem `minimum`. Jeśli jest od niego mniejszy, to wstawiamy jego indeks pod zmienną `minimum`:

```
pracownicy[0..49][0..29] ← wczytaj dane z pliku "pracownicy.txt"

dla i = 0, 1, ..., 49 wykonuj:
    dla j = 0, 1, ..., 5 wykonuj:

        minimum ← j
        dla k = j + 1, j + 2, ..., 29 - j wykonuj:

            jeżeli pracownicy[i][k] < pracownicy[i][minimum]:
                minimum ← k
```

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PD5nxw9Vf

Po zakończeniu wewnętrznej pętli zamieniamy elementy pod indeksami `j` oraz `minimum`, dzięki czemu wstawiamy na odpowiednie miejsce najmniejszy element z nieposortowanej części tablicy:

```
pracownicy[0..49][0..29] ← wczytaj dane z pliku "pracownicy.txt"

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

```
4       dla j = 0, 1, ..., 5
wykonuj:
5
6           minimum ← j
7           dla k = j + 1, j +
2, ..., 29 - j wykonuj:
8
9               jeżeli
pracownicy[i][k] <
pracownicy[i][minimum]:
10                  minimum ←
k
11
12          zamień
pracownicy[i][j] z
pracownicy[i][minimum]
```

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PD5nxw9Vf

7

Analogiczne postępujemy z wyszukiwaniem największego elementu – z tą różnicą, że wstawiliśmy już na odpowiednie miejsce najmniejszy element, więc zmiennej `maximum` przypisujemy wartość `j + 1`. Podobnie pętlę rozpoczynamy od indeksu o 1 większego niż w pętli wyszukującej element najmniejszy:

```
1 pracownicy[0..49][0..29] ←
wczytaj dane z pliku
"pracownicy.txt"
2
3 dla i = 0, 1, ..., 49
wykonuj:
4       dla j = 0, 1, ..., 5
wykonuj:
5
6           minimum ← j
7           dla k = j + 1, j +
2, ..., 29 - j wykonuj:
```

```
8
9               jeżeli
    pracownicy[i][k] <
    pracownicy[i][minimum]:
10                  minimum ←
    k
11
12          zamień
    pracownicy[i][j] z
    pracownicy[i][minimum]
13
14          maximum ← j + 1
15          dla k = j + 2, j +
    3, ..., 29 - j wykonuj:
16
17              jeżeli
    pracownicy[i][k] >
    pracownicy[i][maximum]:
18                  maximum ←
    k
19
20          zamień
    pracownicy[i][29 - j] z
    pracownicy[i][maximum]
```


Więcej zadań znajdziesz w zbiorze dostępnym na stronie internetowej Okręgowej Komisji Egzaminacyjnej w Warszawie.
Źródło: OKE, oke.waw.pl, tylko do użytku edukacyjnego.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PD5nxw9Vf

Po zakończeniu wybierania elementów w danym wierszu jesteśmy pewni, że na odpowiednich miejscach znajdują się szósty

najmniejszy oraz szósty największy element z tablicy. Zapisujemy je zatem do pliku wynikowego i przechodzimy do kolejnego wiersza z danymi.

```
pracownicy[0..49][0..29] ← wczytaj dane z pliku "pracownicy.txt"

dla i = 0, 1, ..., 49 wykonuj:
    dla j = 0, 1, ..., 5 wykonuj:

        minimum ← j
        dla k = j + 1, j + 2, ..., 29 - j wykonuj:

            jeżeli pracownicy[i][k] < pracownicy[i][minimum]:
                minimum ← k

        zamień pracownicy[i][j] z pracownicy[i][minimum]

        maximum ← j + 1
        dla k = j + 2, j + 3, ..., 29 - j wykonuj:

            jeżeli pracownicy[i][k] > pracownicy[i][maximum]:
                maximum ← k

        zamień pracownicy[i][29 - j] z pracownicy[i][maximum]

    zapisz pracownicy[i][5], pracownicy[i][24] do pliku "przedziały.txt"
```

Schemat oceniania

- **2 pkt** – za prawidłową odpowiedź
- **1 pkt** – za odpowiedź wynikającą ze zwrócenia piątego najmniejszego i piątego największego elementu dla każdego wiersza pliku
- **0 pkt** – za niepoprawną odpowiedź lub brak odpowiedzi

Odpowiedź

Odpowiedź do zadania znajduje się w pliku `przedziały.txt`:

Plik o rozmiarze 299.00 B w języku polskim

Sprawdź się

Zadanie 3. Piekarnia w Unixlandii

Kasia i Tomek prowadzą rodzinną piekarnię w Unixlandii. Codziennie rano wypiekają różną liczbę bułek, chlebów, bagietek oraz ciast, które następnie są rozwożone po całym kraju.

W sumie wypiekanych jest dziewięć rodzajów pieczywa, a ich liczby zapisywane są każdego dnia do pliku `pieczywo.txt`, oddzielone pojedynczym znakiem odstępu, każdy dzień w osobnej linii. W żadnym dniu nie ma dwóch rodzajów pieczywa o tej samej liczbie wypieków.

Plik o rozmiarze 2.64 KB w języku polskim

Napisz program, który dla każdego wiersza z pliku `pieczywo.txt` wyznaczy indeks pieczywa, które ma dokładnie środkową liczbę wypieków wśród pozostałych rodzajów pieczywa danego dnia. Wyniki zapisz do pliku `środkowe.txt`, każdy indeks w osobnej linii.

Uwaga: pieczywa w pliku, podobnie jak tablice w komputerze, indeksujemy od zera.

Do oceny oddajesz:

- plik `środkowe.txt` z odpowiedzią (plik tekstowy zawierający indeksy rodzajów pieczywa z odpowiadających wierszy pliku `pieczywo.txt`, które danego dnia miały dokładnie środkową liczbę wypieków)
- plik(i) z komputerową realizacją programu (kodem źródłowym)

> **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:

> JĘZYK C++

Ćwiczenie 1

Twoje zadania

1. Program dla każdej podtablicy tablicy `pieczywo` wypisuje indeks elementu ze środkową wartością.

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

int main() {
    int pieczywo[5][9] = {
        {23, 35, 18, 12, 15, 41, 13, 36, 46},
        {28, 38, 42, 25, 37, 20, 30, 10, 13},
        {20, 22, 41, 21, 33, 49, 37, 18, 43},
        {48, 44, 31, 40, 33, 35, 34, 19, 13},
        {34, 46, 23, 28, 14, 45, 18, 41, 26}
    };

    // Tutaj dodaj kod

```

```
1
```

Ćwiczenie 2

Twoje zadania

1. Program dla każdej podtablicy tablicy `pieczywo` wypisuje indeks elementu ze środkową wartością.

```
public class Main {
    public static void main(String [] args) {
        int[][] pieczywo = {
            {23, 35, 18, 12, 15, 41, 13, 36, 46},
            {28, 38, 42, 25, 37, 20, 30, 10, 13},
            {20, 22, 41, 21, 33, 49, 37, 18, 43},
            {48, 44, 31, 40, 33, 35, 34, 19, 13},
            {34, 46, 23, 28, 14, 45, 18, 41, 26}
        };

        // Tutaj dodaj kod

    }
}
```

```
1
```

Ćwiczenie 3

Twoje zadania

1. Program dla każdej podtablicy tablicy `pieczywo` wypisuje indeks elementu ze środkową wartością.

```
pieczywo = [
    [23, 35, 18, 12, 15, 41, 13, 36, 46],
    [28, 38, 42, 25, 37, 20, 30, 10, 13],
    [20, 22, 41, 21, 33, 49, 37, 18, 43],
    [48, 44, 31, 40, 33, 35, 34, 19, 13],
    [34, 46, 23, 28, 14, 45, 18, 41, 26]
]
# Tutaj dodaj kod
```

```

```

Schemat oceniania

- **2 pkt** – za poprawną odpowiedź
- **0 pkt** – za niepoprawną odpowiedź lub brak odpowiedzi

Odpowiedź

Odpowiedź do zadania znajduje się w pliku `środkowe.txt`:

Plik o rozmiarze 199.00 B w języku polskim

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Sortowanie przez wybieranie – zadania maturalne

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres podstawowy i 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 podstawowy. Uczeń:

1) planuje kolejne kroki rozwiązywania problemu, z uwzględnieniem podstawowych etapów myślenia komputacyjnego (określenie problemu, definicja modeli i pojęć, znalezienie rozwiązania, zaprogramowanie i testowanie rozwiązania).

2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy:

c) porządkowania ciągu liczb: przez wstawianie i metodą bąbelkową,

4) porównuje działanie różnych algorytmów dla wybranego problemu, analizuje algorytmy na podstawie ich gotowych implementacji;

5) sprawdza poprawność działania algorytmów dla przykładowych danych.

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

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

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):

- Przeanalizujesz przykładowe zadania maturalne z zakresu sortowania przez wybór.
- Napiszesz program, wykorzystując algorytm sortowania przez wybór w danym zadaniu maturalnym.
- Scharakteryzujesz działanie algorytmu sortowania przez wybieranie.

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 informacje dotyczące sortowania przez wybieranie.

Faza wstępna:

1. Wyświetlenie przez nauczyciela tematu i celów zajęć, przejście do wspólnego ustalenia 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. **Praca z tekstem.** Nauczyciel wyświetla zawartość sekcji „Przeczytaj". Uczniowie zapoznają się z treścią zadania 1, analizują jego rozwiązanie, a następnie implementują je w wybranym języku programowania. W kolejnym kroku porównują wyniki swojej pracy z kodem innej osoby programującej w tym samym języku.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Prezentacja multimedialna". Uczniowie zapoznają się z zadaniem 2 , a następnie na forum klasy analizują rozwiązanie przedstawione w prezentacji. Nauczyciel wyjaśnia niezrozumiałe kroki procedury.
3. **Ćwiczenie umiejętności.** Nauczyciel przechodzi do sekcji „Sprawdź się". Uczniowie indywidualnie rozwiązują zadanie 3. Następnie omawiają swoje rozwiązania na forum klasy.

Faza podsumowująca:

1. Nauczyciel prosi uczniów o podsumowanie zgromadzonej wiedzy w zakresie programowania w języku Java.

Praca domowa:

1. Uczniowie zapisują program z sekcji „Prezentacja multimedialna” za pomocą wybranego języka 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 „Prezentacja multimedialna” można wykorzystać na lekcji jako podsumowanie i utrwalenie wiedzy uczniów.