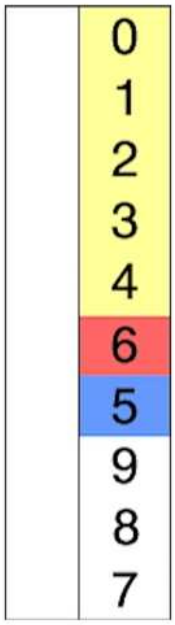
Sortowanie przez wybieranie w języku C++

Źródło: Brienne Hong, domena publiczna.

Potrafimy już scharakteryzować algorytm sortowania przez wybieranie. Za jego pomocą możemy sortować np. karty, ale też oceny z kartkówek, czy wyniki uczestników zawodów sportowych. W tym e-materiale poznamy implementację tego algorytmu w języku C++.



Film dostępny pod adresem /preview/resource/R4Zh6fgEI5GG7

Przykład działania sortowania przez wybieranie, licencja: CC BY-SA 3.0, źródło: wikipedia.org.

Film nawiązujący do treści materiału

Implementację algorytmu sortowania przez wybieranie w innych językach programowania przedstawiamy w e-materiałach:

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

Więcej zadań? Sortowanie przez wybieranie – zadania maturalne.

> **Twoje cele**
>
> - Przeanalizujesz symulację sortowania przez wybieranie.
> - Zaimplementujesz program, w którym wykorzystasz technikę sortowania przez wybieranie w języku C++.
> - Określisz stabilność oraz złożoność obliczeniową omawianego algorytmu.

Przeczytaj

Implementacja algorytmu sortowania przez wybieranie w porządku niemalejącym w języku C++

Przykład 1

Pracujesz w firmie, w której zajmujesz się rozkładaniem towarów na półki. W ostatniej dostawie przyszło sporo nowych długopisów w różnych cenach. Kierownik kazał ci je posortować i poukładać na półkach zgodnie z cenami – od najmniejszych do największych.

Napisz program sortujący w porządku niemalejącym n-elementową tablicę zawierającą ceny długopisów. Cena jest równocześnie identyfikatorem towaru. Program przetestuj dla tablicy `zbior` = [2, 65, 83, 13, 61, 89, 13, 27, 1, 36, 89, 20, 19, 10, 149, 64, 27, 91, 13, 8].

Specyfikacja:

Dane:

- `n` – liczba elementów w tablicy; liczba naturalna
- `zbior` – n-elementowa tablica zawierająca ceny długopisów; tablica liczb naturalnych

Wynik:

- `zbior` – posortowana niemalejąco tablica

1

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P4r9ogXWY

Pisanie programu realizującego omawiane zadanie z wykorzystaniem algorytmu sortowania przez wybieranie zaczynamy od zdefiniowania zmiennych.

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P4r9ogXWY

Zmienna `zbior` jest tablicą, w której znajdują się ceny długopisów. Z kolei zmienna n przechowuje informację o rozmiarze tablicy `zbior`. W zmiennej `pomoc` przechowujemy wartość pomocniczą, która na początku jest równa zero.

```
int zbior[20] = {2, 65, 83, 13, 61, 89, 13, 27, 1, 36, 89, 20, 19, 10, 149, 64, 27, 91, 13, 8};
int n = 20;
int pomoc = 0;
```

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P4r9ogXWY

Teraz definiujemy główną pętlę `for`, za pomocą której będziemy wskazywać kolejne elementy tablicy `zbior`. Zawsze gdy przeszukujemy tablicę o rozmiarze n, jej ostatni element ma indeks `n - 1`, ze względu na to, że pierwszy element w tablicy znajduje się na pozycji o indeksie 0. Warunek końcowy w poniższej pętli `for` wskazuje nam jednak, że indeks n - 1 pomijamy. Możemy tak zrobić, ponieważ gdy ustawimy na odpowiednich pozycjach n - 1 elementów minimalnych, na ostatniej pozycji pozostanie największy element. W związku z tym, że sortujemy nasz zbiór niemalejąco, oznacza to, że element ten znalazł się na odpowiedniej pozycji i nie wymaga dalszego sortowania.

```
for (int i = 0; i < n - 1; i++) {
    // Sortowanie tablicy
}
```

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P4r9ogXWY

Aby posortować tablicę zgodnie z algorytmem sortowania przez wybieranie, należy wyszukać najmniejszy element w nieposortowanej części tablicy, której początkowy indeks wyznacza zmienna `i`. Następnie znaleziony najmniejszy element wymieniamy z elementem na pierwszej pozycji `i`. W ten sposób na początku tablicy otrzymamy wartość najmniejszą. Powyższe kroki będziemy wykonywać ponownie dla nieposortowanej części tablicy (zmniejszonej o pierwszy element), to znaczy rozpoczynającej się indeksem `i + 1` dotąd, aż cała tablica będzie posortowana.

Indeks najmniejszego elementu będziemy przechowywać dzięki dodatkowej zmiennej `indeksNajmniejszegoElementu`:

```
1 for (int i = 0; i < n - 1;
  i++) {
2     int
  indeksNajmniejszegoElement
  u = i;
3
4     // Sortowanie tablicy
5 }
```

Zmiennej `indeksNajmniejszegoElementu` przypisaliśmy wstępnie wartość zmiennej sterującej `i` (zdarza się bowiem, że najmniejsza liczba jest od razu na właściwym miejscu).

Materiał audio dostępny pod adresem:

5

https://zpe.gov.pl/b/P4r9ogXWY

Pora wyszukać najmniejszą liczbę w nieposortowanej jeszcze części zbioru. Potrzebna jest kolejna pętla `for`:

```
for (int i = 0; i < n - 1; i++) {
    int indeksNajmniejszegoElementu = i;

    for (int j = i + 1; j < n; j++) {
        // Znajdz najmniejszy element w nieposortowanym obszarze
    }

    // Sortowanie tablicy
}
```

Ponieważ przeszukujemy obszar nieposortowany, a zmienna `indeksNajmniejszegoElementu` ma początkowo wartość równą `i`, to przeglądanie tablicy rozpoczynamy od elementu o indeksie `j = i + 1`.

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P4r9ogXWY

Jeżeli sprawdzany właśnie element jest mniejszy niż najmniejsza dotychczas liczba, przypisujemy jego indeks zmiennej `indeksNajmniejszegoElementu`.

```
for (int i = 0; i < n - 1; i++) {
    int indeksNajmniejszegoElement
```

```
u = i;

    for (int j = i + 1; j < n; j++) {
        if (zbior[j] < zbior[indeksNajmniejszegoElementu]) {
indeksNajmniejszegoElement u = j;
        }
    }

    // Sortowanie tablicy
}
```

7

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P4r9ogXWY

Po znalezieniu najmniejszej wartości w nieposortowanym zbiorze zamieniamy ją miejscami z elementem na pierwszej pozycji tego zbioru, to znaczy wskazywanym przez `i`. Zmienna `pomoc` przechowuje liczbę, która wcześniej była przechowywana w tablicy na pozycji `indeksNajmniejszegoElementu`. Gdybyśmy nie korzystali ze zmiennej pomocniczej, ta liczba zostałaby zagubiona podczas zamian elementów.

```

    for (int i = 0; i < n - 1; i++) {
        int indeksNajmniejszegoElement u = i;

        for (int j = i + 1; j < n; j++) {
```

```
            if (zbior[j] < zbior[indeksNajmniejszegoElementu]) {
                indeksNajmniejszegoElementu = j;
            }
        }

        pomoc = zbior[indeksNajmniejszegoElementu];
        zbior[indeksNajmniejszegoElementu] = zbior[i];
        zbior[i] = pomoc;
    }

```

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P4r9ogXWY

Na końcu wyświetlamy posortowane elementy tablicy:

```
    for (int i = 0; i < n - 1; i++) {
        int indeksNajmniejszegoElementu = i;

        for (int j = i + 1; j < n; j++) {
            if (zbior[j] < zbior[indeksNajmniejszegoElementu]) {
                indeksNajmniejszegoElementu = j;
            }
```

```
        }

        pomoc = zbior[indeksNajmniejszegoElementu];
        zbior[indeksNajmniejszegoElementu] = zbior[i];
        zbior[i] = pomoc;
    }

    for (int i = 0; i < n; i++) {
        cout << zbior[i] << " " << endl;
    }
```

9

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P4r9ogXWY

A oto cały program:

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

int main() {
    int zbior[20] = {2, 65, 83, 13, 61, 89, 13, 27, 1, 36, 89, 20, 19, 10, 149, 64, 27, 91, 13, 8};
    int n = 20;
    int pomoc = 0;


    for (int i = 0; i < n - 1; i++) {
        int indeksNajmniejszegoElementu = i;

```

```
13          for (int j = i + 1; j < n; j++) {
14              if (zbior[j] < zbior[indeksNajmniejszegoElementu]) {
15                  indeksNajmniejszegoElementu = j;
16              }
17          }
18
19          pomoc = zbior[indeksNajmniejszegoElementu];
20          zbior[indeksNajmniejszegoElementu] = zbior[i];
21          zbior[i] = pomoc;
22      }
23
24      for (int i = 0; i < n; i++) {
25          cout << zbior[i] << " " << endl;
26      }
27
28      return 0;
29  }
```


Polecenie 1

Na podstawie zaprezentowanego przykładu odpowiedz na pytanie, do jakiej grupy algorytmów należy algorytm sortowania przez wybieranie – stabilnych czy niestabilnych?

Jaka jest jego złożoność czasowa?

Ważne!

Algorytm sortowania przez wybieranie należy do grupy algorytmów niestabilnych.

Oznacza to, że w przypadku, kiedy sortujemy elementy o takiej samej wartości, mogą one zmienić pozycję względem siebie.

Jego złożoność czasowa wynosi $O(n^2)$ i rośnie wykładniczo wraz ze wzrostem elementów w nieposortowanym zbiorze. Sortowanie odbywa się w miejscu.

Dla zainteresowanych

W celu zamiany miejscami dwóch zmiennych można także wykorzystać funkcję `swap()`. Wywołanie jej przyjęłoby następującą postać: `swap(zbior[i], zbior[minimalnyIndeks]);`. Gdy używamy omawianej funkcji, nie musimy korzystać ze zmiennej pomocniczej.

Problem 1

Zadania:

Pracujesz w banku. Twoim przełożonym zależy na usprawnieniu procesu przeszukiwania bazy klientów; szef działu technicznego zasugerował, by zastosować algorytm wyszukiwania binarnego, który omówiono w e-materiale Wstęp do algorytmów sortowania. Zanim będzie można to zrobić, trzeba posortować dane klientów.

- Napisz program sortujący w porządku niemalejącym n-elementową tablicę zawierającą dane klientów. Każdy z klientów ma przyporządkowany do siebie numer identyfikatora.
- Swój program przetestuj dla tablicy `zbior = [5, 7, 86, 32, 64, 54, 2].`

Specyfikacja:

Dane:

- `zbior` – tablica n - elementowa zawierająca dane klientów; tablica liczb naturalnych
- n – liczba elementów w tablicy `zbior`; dodatnia liczba naturalna

Wynik:

- posortowana niemalejąco tablica `zbior`

```
#include <iostream>

using namespace std;

int main () {

    // Tutaj dodaj kod. Żeby coś wypisać, użyj polecenia: cout
}
```

```
1
```

Słownik

algorytmy niestabilne

algorytmy, w których elementy o równej wartości nie występują po posortowaniu w tej samej kolejności, jaką miały w zbiorze nieposortowanym

algorytmy stabilne

w przypadku algorytmów stabilnych podczas sortowania w uporządkowanej tablicy nie zmienia się kolejność elementów o tych samych wartościach klucza – przykładowo, gdy w tablicy znajdą się obok siebie dwie liczby o wartości 6, to nie zostaną one zamienione miejscami

wyszukiwanie binarne

metoda odnajdowania elementów w uporządkowanych zbiorach; polega ona na wielokrotnym dzieleniu przeszukiwanego zbioru na dwie części i porównywaniu wartości szukanej z wyznaczonym elementem środkowym; gdy zbiór ma parzystą liczbę elementów, za element środkowy uznaje się jeden z dwóch elementów będących najbliżej środka

zagnieżdżona pętla

pętla działająca wewnątrz innej pętli

Symulacja interaktywna

Polecenie 1

Symulacja interaktywna przedstawia wersję algorytmu sortowania przez wybieranie, w którym w ciągu jeszcze nieposortowanym szukany jest element największy. Przeanalizuj działanie tego algorytmu.

Polecenie 2

Napisz program w języku C++ realizujący algorytm prezentowany za pomocą symulacji.

Specyfikacja:

Dane:

- `n` – liczba elementów w tablicy; liczba naturalna
- `zbior` – n-elementowa tablica liczb rzeczywistych

Wynik:

- `zbior` – posortowana niemalejąco tablica

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Napisz program, który wypisze na standardowe wyjście indeks najmniejszego elementu tablicy.

Program przetestuj dla tablicy `dane` = [423, 654, 423, 659, 345, 432, 765, 534, 469, 421, 6457, 856, 543, 645, 523, 576, 7645].

Specyfikacja:

Dane:

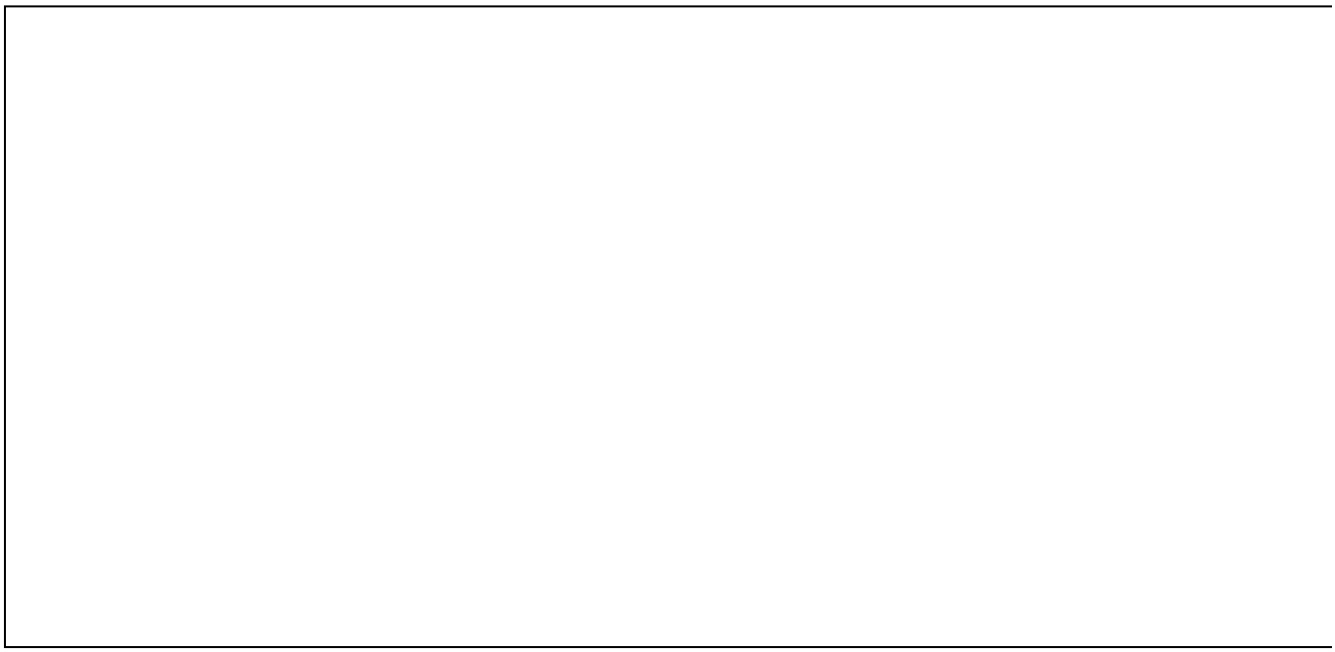
- `dane` – tablica liczb całkowitych w której należy znaleźć indeks najmniejszego elementu
- `rozmiarZbioru` – rozmiar tablicy `dane`; liczba naturalna

Wynik:

- indeks najmniejszej liczby w tablicy `dane`

Twoje zadania

1. Wypisz na standardowe wyjście indeks najmniejszego element tablicy `dane`.



Ćwiczenie 2

Uzupełnij podany kod, aby otrzymać działającą implementację algorytmu sortowania przez wybieranie. Zbiór należy posortować nierosnąco.

Program przetestuj dla tablicy `dane` = [3, 5, 6, 7, 64, 745, 534, 765, 543, 7654, 543, 65, 543, 234, 654, 76, 65]

Specyfikacja:

Dane:

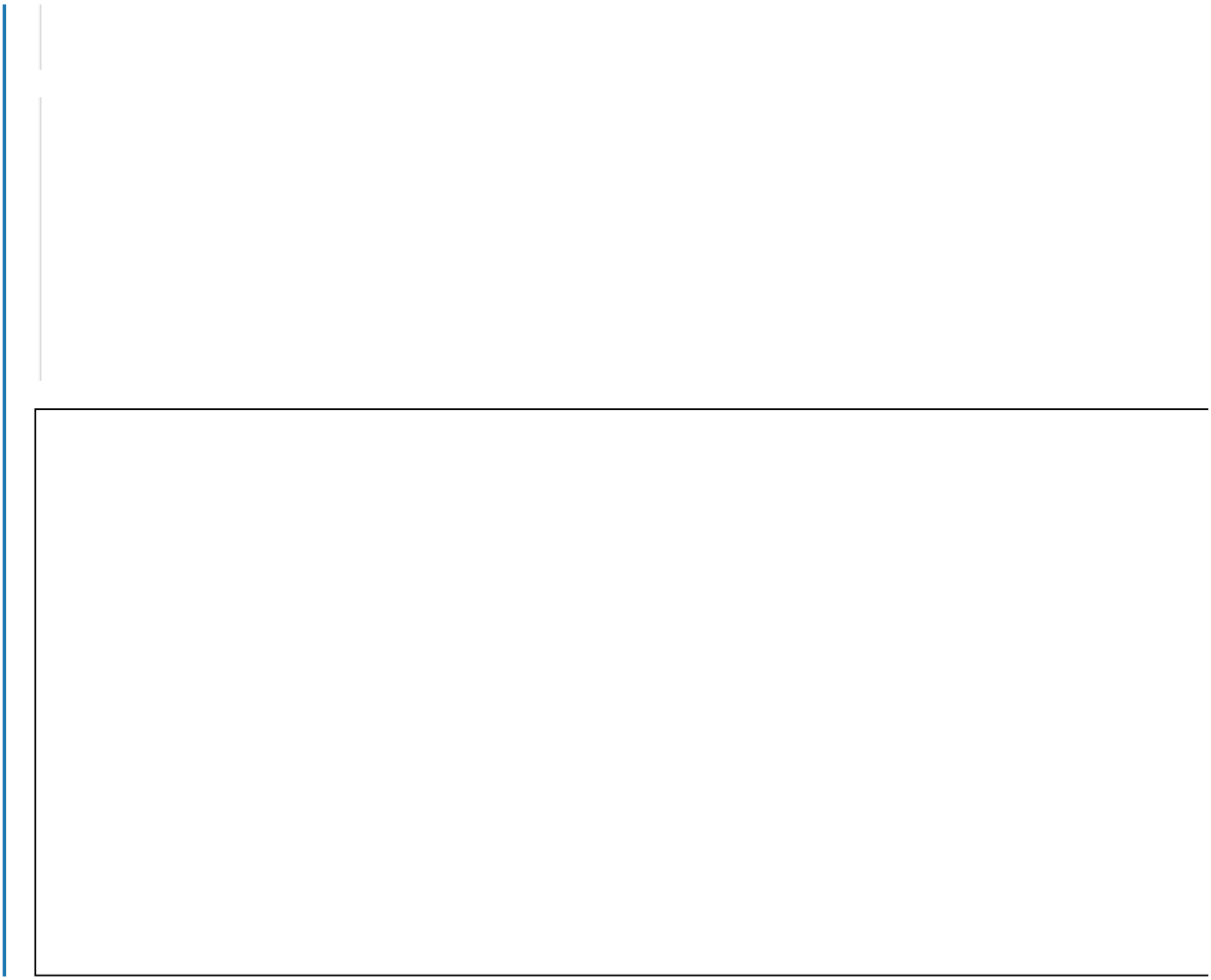
- `dane` – tablica liczb całkowitych
- `rozmiarZbioru` – rozmiar tablicy `dane`; liczba naturalna

Wynik:

- posortowany nierosnąco ciąg liczb z tablicy `dane`; każda liczba wydrukowana jest w nowej linii

Twoje zadania

1. Program sortuje nierosnąco tablicę `dane` z użyciem algorytmu sortowania przez wybieranie i drukuje na standardowe wyjście posortowane elementy tablicy.



Ćwiczenie 3

Napisz program, który posortuje niemalejąco podaną tablicę `dane` z użyciem algorytmu sortowania przez wybieranie. Policz liczbę wykonanych porównań elementów tablicy oraz przestawień.

Program przetestuj dla tablicy `dane` = [654, 435, 543, 764, 432, 364, 765, 342, 746, 845, 541, 111, 325, 257].

Specyfikacja:

Dane:

- `dane` – tablica liczb całkowitych, którą należy posortować
- `rozmiarDanych` – rozmiar tablicy `dane`; liczba naturalna

Wynik:

- `liczbaPorownan` – liczba porównań elementów tablicy `dane`; liczba naturalna
- `liczbaPrzestawien` – liczba przestawień elementów tablicy `dane`; liczba naturalna
- posortowana niemalejąco tablica `dane`

Twoje zadania

1. Program sortuje podaną tablicę, zlicza liczbę porównań elementów tablicy i przestawień wykonanych w trakcie sortowania oraz drukuje te informacje na standardowe wyjście wraz z posortowanymi danymi.

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Sortowanie przez wybieranie w języku C++

Grupa docelowa:

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

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).

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

5) sprawdza poprawność działania algorytmów dla przykładowych danych.

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 symulację sortowania przez wybieranie.
- Zaimplementujesz program, w którym wykorzystasz technikę sortowania przez wybieranie w języku C++.
- Określisz stabilność oraz złożoność obliczeniową omawianego algorytmu.

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.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Sortowanie przez wybieranie w języku C++”. Uczniowie zapoznają się z treściami w sekcji „Przeczytaj”.

Faza wstępna:

1. Nauczyciel wyświetla uczniom temat, wskazuje cele zajęć oraz ustala z uczestnikami zajęć kryteria 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.** 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. Następnie uczniowie analizują przedstawione w prezentacji z sekcji „Przeczytaj" rozwiązanie przykładu 1 i powtarzają je na swoim komputerze. W kolejnym kroku uczniowie w parach rozwiązują problem 1. Chętne osoby przedstawiają swoje odpowiedzi, pozostali uczniowie weryfikują poprawność metod lub przedstawiają alternatywne sposoby rozwiązania zadania.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Symulacja interaktywna" i czyta treść polecenia 1. Uczniowie analizują symulację interaktywną i omawiają swoje spostrzeżenia. W kolejnym kroku wykonują polecenie 2, następnie w parach porównują swoje rozwiązania i dzielą się spostrzeżeniami na forum klasy.
3. **Ćwiczenie umiejętności.** Uczniowie, pracując w parach, wykonują ćwiczenie nr 1 z sekcji „Sprawdź się". Nauczyciel sprawdza poprawność pisanych kodów, porównuje je i omawia wraz z uczniami. Wskazuje najbardziej efektywne rozwiązanie.
4. Nauczyciel dzieli uczniów na grupy czteroosobowe. Uczniowie wykonują ćwiczenie nr 2 z sekcji „Sprawdź się" a następnie każda grupa wyznacza jedną osobę, którą wymienia się z inną grupą. W nowych zespołach uczniowie porównują swój kod i wybierają najbardziej efektywne rozwiązanie.

Faza podsumowująca:

1. Nauczyciel zadaje pytania podsumowujące, np.
    - na czym polega sortowanie przez wybieranie?
    - co oznacza, że algorytm jest stabilny bądź nie i do której grupy należy algorytm sortowania przez wybieranie?
    - jaka jest złożoność czasowa omawianego algorytmu?

   Nauczyciel ponownie wyświetla na tablicy temat lekcji zawarty w sekcji „Wprowadzenie" i inicjuje krótką rozmowę na temat zrealizowanych celów (czego uczniowie się nauczyli).
2. Wybrany uczeń podsumowuje zajęcia z programowania w C++, zwracając uwagę na nabyte umiejętności.

Praca domowa:

1. Uczniowie wykonują ćwiczenie nr 3 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.

Wskazówki metodyczne:

- Uczniowie mogą wykorzystać treści w sekcjach: „Przeczytaj", „Symulacja interaktywna", „Sprawdź się" jako materiał do lekcji powtórkowej.