C++ - WYSZUKIWANIE I SORTOWANIE

Każda linia kodu z wyjaśnieniem w komentarzu

Opracowanie na podstawie plików „Cpp4U-Sortowanie.pdf” oraz „Cpp4U-Sortowanie-Ćwiczenia.pdf”. Zachowano strukturę i rozwiązania z materiałów źródłowych, a do każdej linii kodu dopisano komentarz wyjaśniający jej działanie.

CZĘŚĆ I. Wyszukiwanie liniowe

Wyszukiwanie liniowe sprawdza kolejne elementy tablicy po kolei, aż znajdzie szukaną wartość albo dojdzie do końca tablicy.

Przykład 1 - sprawdzenie, czy liczba istnieje

```
#include <iostream>                     // Dołączamy bibliotekę wejścia i wyjścia, aby używać cout.
using namespace std;                    // Umożliwiamy używanie nazw z przestrzeni std bez przedrostka std::.
int main() {                            // Rozpoczynamy główną funkcję programu.
    int tab[5] = {2, 4, 6, 8, 10};      // Tworzymy tablicę 5 liczb całkowitych i od razu nadajemy jej wartości.
    int x = 6;                          // Tworzymy zmienną x i ustawiamy szukaną wartość na 6.
    bool znaleziono = false;            // Tworzymy flagę logiczną; na początku zakładamy, że liczby nie znaleziono.
    for (int i = 0; i < 5; i++) {       // Przechodzimy kolejno przez indeksy tablicy od 0 do 4.
        if (tab[i] == x) {              // Sprawdzamy, czy aktualny element tablicy jest równy szukanej wartości x.
            znaleziono = true;          // Jeśli tak, ustawiamy flagę znaleziono na true.
        }                               // Kończymy blok instrukcji if.
    }                                   // Kończymy pętlę przeszukującą tablicę.
    if (znaleziono) cout << "TAK\n";   // Jeśli znaleziono liczbę, wypisujemy komunikat TAK.
    else cout << "NIE\n";               // W przeciwnym razie wypisujemy komunikat NIE.
    return 0;                           // Kończymy program poprawnie i zwracamy kod 0.
}                                       // Kończymy funkcję main.
```

Przykład 2 - znajdź indeks

```
#include <iostream>                     // Dołączamy bibliotekę potrzebną do wypisywania danych.
using namespace std;                    // Używamy standardowej przestrzeni nazw.
int main() {                            // Rozpoczynamy funkcję główną programu.
    int tab[5] = {1, 3, 7, 9, 5};       // Tworzymy tablicę 5 liczb całkowitych z podanymi wartościami.
    int x = 9;                          // Ustawiamy szukaną wartość na 9.
    for (int i = 0; i < 5; i++) {       // Przechodzimy przez wszystkie indeksy tablicy od 0 do 4.
        if (tab[i] == x) {              // Sprawdzamy, czy aktualny element jest równy x.
            cout << "Indeks: " << i << endl; // Jeśli znaleźliśmy x, wypisujemy jego indeks.
        }                               // Kończymy instrukcję if.
    }                                   // Kończymy pętlę przeszukującą.
    return 0;                           // Kończymy program poprawnie.
}                                       // Kończymy funkcję main.
```

CZĘŚĆ II. Sortowanie bąbelkowe - Bubble Sort

Sortowanie bąbelkowe porównuje sąsiednie elementy i zamienia je miejscami, gdy są w złej kolejności. Po kolejnych przejściach większe wartości przesuwają się na koniec tablicy.

Przykład - sortowanie rosnące

```
#include <iostream>                     // Dołączamy bibliotekę wejścia i wyjścia.
using namespace std;                    // Używamy standardowej przestrzeni nazw.
int main() {                            // Rozpoczynamy funkcję main.
    int tab[5] = {5, 2, 8, 1, 3};       // Tworzymy pięcioelementową tablicę z nieuporządkowanymi wartościami.
    for (int i = 0; i < 5 - 1; i++) {   // Wykonujemy kolejne przejścia sortowania; maksymalnie 4 przejścia.
        for (int j = 0; j < 5 - i - 1; j++) { // Porównujemy sąsiednie elementy, pomijając już uporządkowany koniec.
            if (tab[j] > tab[j + 1]) {  // Jeśli lewy element jest większy od prawego, są w złej kolejności rosnącej.
                int temp = tab[j];      // Zapamiętujemy lewy element w zmiennej pomocniczej temp.
                tab[j] = tab[j + 1];    // Przenosimy prawy, mniejszy element na lewą pozycję.
                tab[j + 1] = temp;      // Wstawiamy zapamiętany element na prawą pozycję - wykonujemy zamianę.
            }                           // Kończymy warunek odpowiedzialny za zamianę elementów.
        }                               // Kończymy wewnętrzną pętlę porównującą sąsiadów.
    }                                   // Kończymy zewnętrzną pętlę sortowania.
    for (int i = 0; i < 5; i++) {       // Przechodzimy przez posortowaną tablicę.
        cout << tab[i] << " ";           // Wypisujemy kolejny element i spację.
    }                                   // Kończymy pętlę wypisującą.
    return 0;                           // Kończymy program poprawnie.
}                                       // Kończymy funkcję main.
```

CZĘŚĆ III. Ćwiczenia - wyszukiwanie i sortowanie

Zadanie 1 - sprawdź, czy podana liczba występuje w tablicy

```
#include <iostream>                    // Dołączamy bibliotekę wejścia i wyjścia.
using namespace std;                   // Używamy standardowej przestrzeni nazw.
int main() {                           // Rozpoczynamy program.
    int tab[10], x;                    // Tworzymy tablicę 10 liczb oraz zmienną x na wartość szukaną.
    bool znaleziono = false;           // Ustawiamy flagę znaleziono na false, bo jeszcze niczego nie sprawdziliśmy.
    for (int i = 0; i < 10; i++) cin >> tab[i]; // Wczytujemy 10 liczb do kolejnych elementów tablicy.
    cin >> x;                          // Wczytujemy liczbę, której będziemy szukać.
    for (int i = 0; i < 10; i++) {     // Przechodzimy przez wszystkie elementy tablicy.
        if (tab[i] == x) znaleziono = true; // Jeśli znajdziemy x, ustawiamy flagę znaleziono na true.
    }                                  // Kończymy pętlę wyszukującą.
    if (znaleziono) cout << "TAK\n";   // Jeśli flaga ma wartość true, wypisujemy TAK.
    else cout << "NIE\n";              // Jeśli nie znaleziono x, wypisujemy NIE.
    return 0;                          // Kończymy program poprawnie.
}                                      // Kończymy funkcję main.
```

Zadanie 2 - wypisz indeks pierwszego wystąpienia

```
#include <iostream>                    // Dołączamy bibliotekę wejścia i wyjścia.
using namespace std;                   // Używamy standardowej przestrzeni nazw.
int main() {                           // Rozpoczynamy program.
    int tab[10], x;                    // Tworzymy tablicę 10 liczb i zmienną x na szukaną wartość.
    for (int i = 0; i < 10; i++) cin >> tab[i]; // Wczytujemy 10 elementów tablicy.
    cin >> x;                          // Wczytujemy liczbę, której indeks chcemy znaleźć.
    for (int i = 0; i < 10; i++) {     // Przeszukujemy tablicę od początku.
        if (tab[i] == x) {             // Sprawdzamy, czy aktualny element jest równy x.
            cout << i << endl;         // Wypisujemy indeks pierwszego znalezionego wystąpienia.
            break;                     // Przerywamy pętlę, aby nie szukać dalszych wystąpień.
        }                              // Kończymy instrukcję if.
    }                                  // Kończymy pętlę wyszukującą.
    return 0;                          // Kończymy program poprawnie.
}                                      // Kończymy funkcję main.
```

Zadanie 3 - policz, ile razy występuje dana liczba

```
#include <iostream>                    // Dołączamy bibliotekę iostream.
using namespace std;                   // Używamy standardowej przestrzeni nazw.
int main() {                           // Rozpoczynamy program.
    int tab[10], x, licznik = 0;       // Tworzymy tablicę, szukaną wartość x i licznik wystąpień równy 0.
    for (int i = 0; i < 10; i++) cin >> tab[i]; // Wczytujemy 10 elementów tablicy.
    cin >> x;                          // Wczytujemy liczbę, której wystąpienia chcemy policzyć.
    for (int i = 0; i < 10; i++) {     // Przechodzimy przez wszystkie elementy tablicy.
        if (tab[i] == x) licznik++;     // Jeśli element jest równy x, zwiększamy licznik o 1.
    }                                  // Kończymy pętlę zliczającą.
    cout << licznik << endl;            // Wypisujemy liczbę wystąpień wartości x.
    return 0;                          // Kończymy program poprawnie.
}                                      // Kończymy funkcję main.
```

Zadanie 4 - znajdź najmniejszy element bez sortowania

```
#include <iostream>                    // Dołączamy bibliotekę wejścia i wyjścia.
using namespace std;                   // Używamy standardowej przestrzeni nazw.
int main() {                           // Rozpoczynamy program.
    int tab[10];                       // Tworzymy tablicę 10 liczb całkowitych.
    for (int i = 0; i < 10; i++) cin >> tab[i]; // Wczytujemy wszystkie elementy tablicy.
    int minL = tab[0];                 // Przyjmujemy pierwszy element jako początkowe minimum.
    for (int i = 1; i < 10; i++) {     // Sprawdzamy pozostałe elementy, zaczynając od indeksu 1.
        if (tab[i] < minL) minL = tab[i]; // Jeśli aktualny element jest mniejszy, zapisujemy go jako nowe minimum.
    }                                  // Kończymy pętlę wyszukującą minimum.
    cout << minL << endl;               // Wypisujemy najmniejszą znalezioną wartość.
    return 0;                          // Kończymy program poprawnie.
}                                      // Kończymy funkcję main.
```

Zadanie 5 - posortuj 10 liczb rosnąco

```
#include <iostream>                    // Dołączamy bibliotekę wejścia i wyjścia.
using namespace std;                   // Używamy standardowej przestrzeni nazw.
int main() {                           // Rozpoczynamy program.
    int tab[10];                       // Tworzymy tablicę 10 liczb.
    for (int i = 0; i < 10; i++) cin >> tab[i]; // Wczytujemy 10 wartości do tablicy.
    for (int i = 0; i < 9; i++) {      // Wykonujemy kolejne przejścia sortowania bąbelkowego.
        for (int j = 0; j < 9; j++) {  // Porównujemy sąsiednie elementy tablicy.
            if (tab[j] > tab[j + 1]) { // Jeśli lewy element jest większy, zamieniamy elementy miejscami.
                int temp = tab[j];     // Zapamiętujemy lewy element w zmiennej temp.
                tab[j] = tab[j + 1];   // Przenosimy prawy element na lewą pozycję.
                tab[j + 1] = temp;     // Wstawiamy zapamiętany element na prawą pozycję.
            }                          // Kończymy instrukcję if.
        }                              // Kończymy pętlę wewnętrzną.
    }                                  // Kończymy pętlę zewnętrzną.
    for (int i = 0; i < 10; i++) cout << tab[i] << " "; // Wypisujemy posortowaną tablicę rosnąco.
    return 0;                          // Kończymy program poprawnie.
}                                      // Kończymy funkcję main.
```

Zadanie 6 - posortuj 10 liczb malejąco

```
#include <iostream>                    // Dołączamy bibliotekę wejścia i wyjścia.
using namespace std;                   // Używamy standardowej przestrzeni nazw.
int main() {                           // Rozpoczynamy program.
    int tab[10];                       // Tworzymy tablicę 10 liczb całkowitych.
    for (int i = 0; i < 10; i++) cin >> tab[i]; // Wczytujemy 10 elementów tablicy.
    for (int i = 0; i < 9; i++) {      // Wykonujemy serię przejść potrzebnych do sortowania.
        for (int j = 0; j < 9; j++) {  // Porównujemy kolejne pary sąsiednich elementów.
            if (tab[j] < tab[j + 1]) { // Jeśli lewy element jest mniejszy, zamieniamy je dla kolejności malejącej.
                int temp = tab[j];     // Zapamiętujemy lewy element w zmiennej pomocniczej.
                tab[j] = tab[j + 1];   // Przenosimy większy prawy element na lewą pozycję.
                tab[j + 1] = temp;     // Wstawiamy zapamiętany element na prawą pozycję.
            }                          // Kończymy instrukcję if.
        }                              // Kończymy wewnętrzną pętlę.
    }                                  // Kończymy zewnętrzną pętlę sortowania.
    for (int i = 0; i < 10; i++) cout << tab[i] << " "; // Wypisujemy tablicę w kolejności malejącej.
    return 0;                          // Kończymy program poprawnie.
}                                      // Kończymy funkcję main.
```

Zadanie 7 - wypisz najmniejszą i największą po posortowaniu

```
#include <iostream>                    // Dołączamy bibliotekę wejścia i wyjścia.
using namespace std;                   // Używamy standardowej przestrzeni nazw.
int main() {                           // Rozpoczynamy program.
    int tab[10];                       // Tworzymy tablicę 10 liczb.
    for (int i = 0; i < 10; i++) cin >> tab[i]; // Wczytujemy 10 elementów.
    // sort                             // Rozpoczynamy fragment kodu odpowiedzialny za sortowanie tablicy.
    for (int i = 0; i < 9; i++) {      // Wykonujemy kolejne przejścia sortowania bąbelkowego.
        for (int j = 0; j < 9; j++) {  // Porównujemy sąsiednie elementy.
            if (tab[j] > tab[j + 1]) { // Jeśli są w złej kolejności rosnącej, wykonujemy zamianę.
                int temp = tab[j];     // Zapamiętujemy lewy element.
                tab[j] = tab[j + 1];   // Przenosimy mniejszy element w lewo.
                tab[j + 1] = temp;     // Umieszczamy zapamiętany element po prawej stronie.
            }                          // Kończymy instrukcję if.
        }                              // Kończymy pętlę wewnętrzną.
    }                                  // Kończymy pętlę zewnętrzną.
    cout << "Min: " << tab[0] << endl; // Po sortowaniu rosnącym pierwszy element jest najmniejszy.
    cout << "Max: " << tab[9] << endl; // Po sortowaniu rosnącym ostatni element jest największy.
    return 0;                          // Kończymy program poprawnie.
}                                      // Kończymy funkcję main.
```

Zadanie 8 - sprawdź, czy tablica jest już posortowana rosnąco

```
#include <iostream>                    // Dołączamy bibliotekę wejścia i wyjścia.
using namespace std;                   // Używamy standardowej przestrzeni nazw.
int main() {                           // Rozpoczynamy program.
    int tab[10];                       // Tworzymy tablicę 10 liczb całkowitych.
    bool rosnaca = true;               // Zakładamy początkowo, że tablica jest rosnąca.
    for (int i = 0; i < 10; i++) cin >> tab[i]; // Wczytujemy wszystkie elementy tablicy.
    for (int i = 1; i < 10; i++) {     // Porównujemy każdy element z elementem poprzednim.
        if (tab[i] < tab[i - 1]) rosnaca = false; // Jeśli kolejny element jest mniejszy, tablica nie jest rosnąca.
    }                                  // Kończymy pętlę sprawdzającą.
    if (rosnaca) cout << "TAK\n";      // Jeśli nie znaleziono spadku wartości, wypisujemy TAK.
    else cout << "NIE\n";               // Jeśli znaleziono spadek wartości, wypisujemy NIE.
    return 0;                          // Kończymy program poprawnie.
}                                      // Kończymy funkcję main.
```

Zadanie 9 - znajdź drugą najmniejszą liczbę

```
#include <iostream>                    // Dołączamy bibliotekę wejścia i wyjścia.
using namespace std;                   // Używamy standardowej przestrzeni nazw.
int main() {                           // Rozpoczynamy program.
    int tab[10];                       // Tworzymy tablicę 10 liczb całkowitych.
    for (int i = 0; i < 10; i++) cin >> tab[i]; // Wczytujemy wszystkie wartości.
    // sort                             // Rozpoczynamy fragment sortowania tablicy.
    for (int i = 0; i < 9; i++) {      // Wykonujemy kolejne przejścia Bubble Sort.
        for (int j = 0; j < 9; j++) {  // Porównujemy sąsiednie elementy.
            if (tab[j] > tab[j + 1]) { // Jeśli lewy element jest większy, zamieniamy elementy miejscami.
                int temp = tab[j];     // Zapamiętujemy lewy element.
                tab[j] = tab[j + 1];   // Przenosimy prawy element na lewą pozycję.
                tab[j + 1] = temp;     // Wstawiamy zapamiętany element po prawej stronie.
            }                          // Kończymy warunek zamiany.
        }                              // Kończymy pętlę wewnętrzną.
    }                                  // Kończymy pętlę zewnętrzną.
    cout << tab[1] << endl;            // Po sortowaniu rosnącym wypisujemy element o indeksie 1 jako drugą pozycję.
    return 0;                          // Kończymy program poprawnie.
}                                      // Kończymy funkcję main.
```

Zadanie 10 - kod z materiału: zamiana wszystkich wystąpień x na 0

```
#include <iostream>                    // Dołączamy bibliotekę wejścia i wyjścia.
using namespace std;                   // Używamy standardowej przestrzeni nazw.
int main() {                           // Rozpoczynamy program.
    int tab[10], x;                    // Tworzymy tablicę 10 liczb oraz zmienną x.
    for (int i = 0; i < 10; i++) cin >> tab[i]; // Wczytujemy 10 liczb do tablicy.
    cin >> x;                          // Wczytujemy wartość x, której wystąpienia mają zostać zastąpione.
    for (int i = 0; i < 10; i++) {     // Przechodzimy przez wszystkie elementy tablicy.
        if (tab[i] == x) tab[i] = 0;   // Jeśli element jest równy x, zastępujemy go wartością 0.
    }                                  // Kończymy pętlę modyfikującą.
    for (int i = 0; i < 10; i++) cout << tab[i] << " "; // Wypisujemy tablicę po wykonaniu zamian.
    return 0;                          // Kończymy program poprawnie.
}                                      // Kończymy funkcję main.
```

Uwaga: w źródłowym pliku przy Zadaniu 10 widoczny jest kod, natomiast pełna treść polecenia nie została pokazana w warstwie tekstowej. Opis zadania powyżej wynika bezpośrednio z działania kodu widocznego na stronie 5 materiału ćwiczeniowego.