PROJEKT ALGORYTMY I STRUKTURY DANYCH

Tytuł Projektu:

Dla zadanego ciągu liczb całkowitych (w postaci tablicy) znajdź liczbę wszystkich podciągów malejących (podciąg musi składać się z przynajmniej dwóch wartości).

Autor: Joanna Soroka

Numer albumu: 173216

Spis treści

1. Temat

Dla zadanego ciągu liczb całkowitych (w postaci tablicy) znajdź liczbę wszystkich podciągów malejących (podciąg musi składać się z przynajmniej dwóch wartości).

Przykład:

Wejście: A[] = [5, 4, 2, 2, 1]

Wyjście: Liczba wszystkich podciągów malejących to 4

[5,4], [5, 4, 2], [4, 2], [2, 1]

Wejście: A[] = [2, 5, 3]

Wyjście: Liczba wszystkich podciągów malejących to 1

[5, 3]

Główny kod programu powinien być zaimplementowany w oddzielnej funkcji, która powinna być wywołana wewnątrz programu.

Kod powinien być opatrzony stosownymi komentarzami.

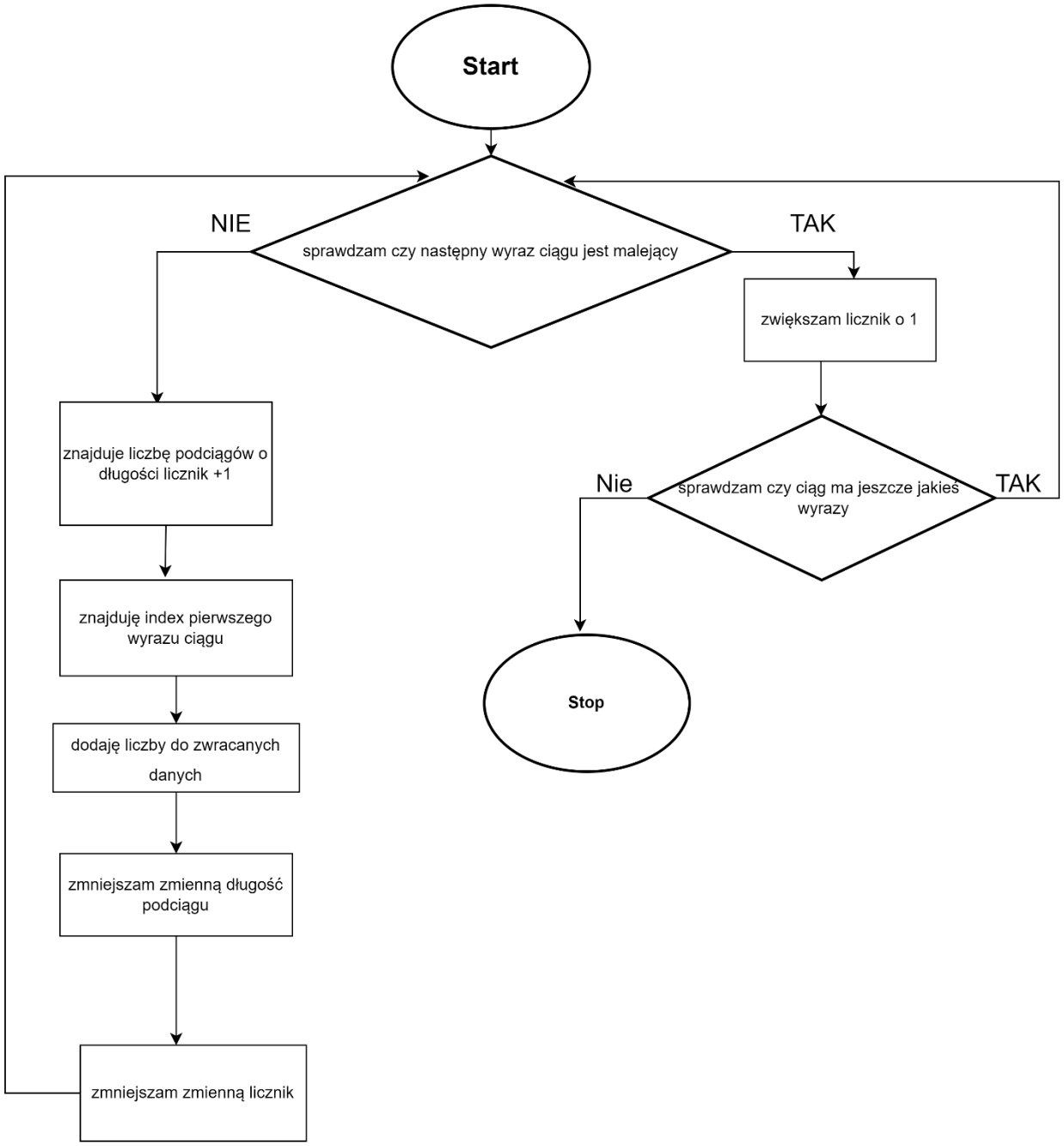
Wykonano kilka testów mających na celu sprawdzenie działania programu.

2. Projektowanie

a) Problem

W zadaniu trzeba znaleźć, wypisać podciągi malejące oraz podać ich liczbę.
Sprawdzamy wszystkie wyrazy ciągu i jeśli tworzą podciąg malejący wypisujemy je w tablicy.
W zależności od długości podciągu szukamy w nim następnych podciągów malejących lub jeśli takie nie występują przechodzimy do wypisania ich.

b) Schemat blokowy algorytmu



c) Pseudokod algorytmu

K0: sprawdzam czy następny wyraz ciągu jest malejący
K1: jeśli tak zwiększam licznik o 1
K2: index != dane_wejsciowe.size() - 1; sprawdzam czy istnieje następny wyraz ciągu
K3:jeśli tak powracam do sprawdzenia czy następny wyraz ciągu jest malejący
K4:dlugosc_podciagu = maxCounter - licznik + 1; jeśli nie wykonaj
K5:firstIndex = index - licznik - dlugosc_podciagu + 1; znajduję index pierwszego wyrazu podciągu malejącego
K6:dodaję liczby do zwracanych danych
K7:dlugosc_podciągu--; zmniejszam zmienną
K08:licznik--;zmniejszam zmienną licznik

3. Kod Programu

```cpp
#include <iostream>
#include <vector>
#include <bits/stdc++.h>


using namespace std;

//funkcja sprawdza czy nastepna liczba w ciagu jest malejaca czy nie
bool czy_nastepna_liczba_w_ciagu_jest_malejaca(int index, vector<int> dane_wejsciowe)
{
    //sprawdzam czy istnieje nastepny wyraz ciagu
    if (index != dane_wejsciowe.size() - 1)
    {
        //jesli istnieje nastepny wyraz ciagu, to sprawdzam czy jest mniejszy niz poprzedni
        return (dane_wejsciowe[index]) > (dane_wejsciowe[index + 1]);
    }
//jesli nie istnieje nastepny wyraz to zwracam falsz
    return false;
}

//funkcja znajduje i dodaje do zwracanych danych podciagi malejace w zakresie do indexu
vector<vector<int>>znajdz_wszystkie_podciagi_malejace_w_zakresie_od_index(int licznik, int index, vector<int> dane_wejsciowe)
{

    vector<vector<int>> dane_zwracane;
    //maxcounter zapisuje poczatkowy stan licznika
    int maxCounter = licznik;
```

```cpp
    vector<vector<int>> dane_zwracane;
    //maxcounter zapisuje poczatkowy stan licznika
    int maxCounter = licznik;
    //petla wykonuje sie tyle razy, ile wynosi licznik, dodajac coraz krotsze podciagi
    while (licznik > 0)
    {
        //znajduje liczbe podciagow o dlugosci licznik+1
        int dlugosc_podciagu = maxCounter - licznik + 1;
        while (dlugosc_podciagu > 0)
        {
            //znajduje index liczby rozpoczynajacej podciag malejacy
            int firstIndex = index - licznik - dlugosc_podciagu + 1;
            //dodaje liczby od indexu firstindex do licznik
            //z danych wejsciowych, do zwracanych danych
            dane_zwracane.push_back(vector<int> {dane_wejsciowe.begin()+firstIndex, dane_wejsciowe.begin()+firstIndex+licznik+1});
            //zmniejszam zmienna dlugosc_podciagu, dzieki czemu
            //przy nastepnej iteracji funkcji bede dodawac liczby bedace
            //w tablicy wejsciowej o jeden index dalej
            dlugosc_podciagu--;
        }
        //zmniejszam zmienna licznik dzieki czemu przy nastepnej
        //iteracji funkcji bede szukac wyrazow krotszych o jeden
        licznik--;
    }
    return dane_zwracane;
}
```

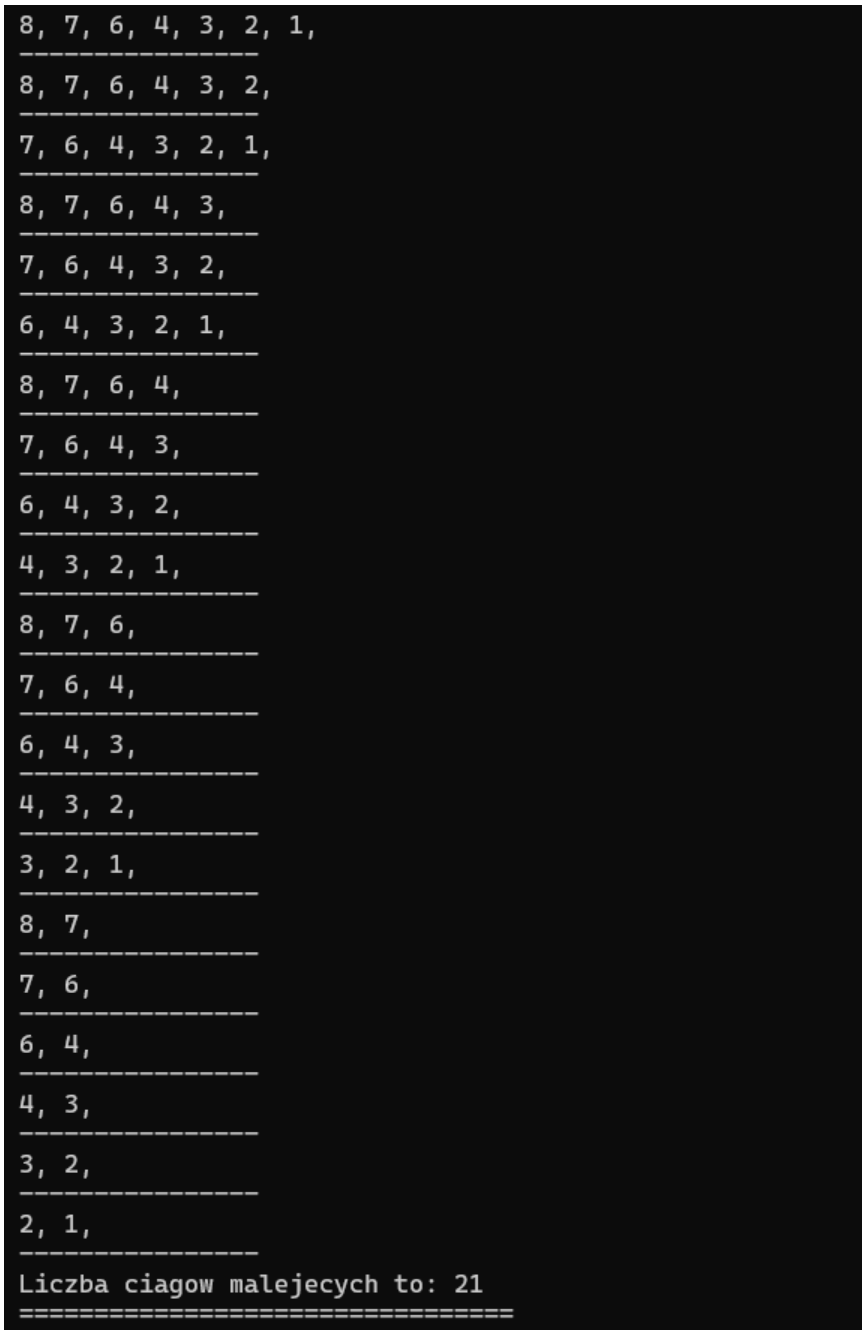
```cpp
//znajduje dlugosc kolejnych podciagow malejacych
vector<vector<int>> znajdz_ciagi_malejace(vector<int> dane_wejsciowe)
{
    vector<vector<int>> dane_zwracane;

    int licznik = 0;
    //przechodze przez kolejne wyrazy ciagu
    for (int i = 0; i < dane_wejsciowe.size(); i++)
    {
        //jesli nastepna liczba jest malejaca zwiekszam licznik
        if (czy_nastepna_liczba_w_ciagu_jest_malejaca(i, dane_wejsciowe))
        {
            licznik++;
        }
        //jesli nastepna liczba nie jest mniejsza to podaje dane wejsciowe, licznik i index
        //do funkcji znajdujacej wszystkie podciagi malejace w zakresie:licznik, od indexu:index
        //dane zwrocone z funkcji dodaje do tablicy dane_zwracane
        //na koncu zeruje licznik
        else
        {
            vector<vector<int>> pociagi_malejace_w_zakresie_od_index =znajdz_wszystkie_podciagi_malejace_w_zakresie_od_index(licznik, i, dane_wejsciowe);
            dane_zwracane.insert(vector<vector<int>> znajdz_ciagi_malejace::pociagi_malejace_w_zakresie_od_index.begin(), pociagi_malejace_w_zakresie_od_index.end());
            licznik = 0;
        }
    }

    return dane_zwracane;
```

```
        //na koncu zeruje licznik
        else
        {
            vector<vector<int>> pociagi_malejace_w_zakresie_od_index =znajdz_wszystkie_podciagi_malejace_w_zakresie_od_index(licznik, i, dane_wejsciowe);
            dane_zwracane.insert (dane_zwracane.end(),pociagi_malejace_w_zakresie_od_index.begin(),pociagi_malejace_w_zakresie_od_index.end());
            licznik = 0;
        }
    }

    return dane_zwracane;
}
void wypisz_ciagi(vector<vector<int>>wszystkie_dane)
{
    for (auto dane: wszystkie_dane)
    {
        for (auto item: dane)
        {
            cout << item << ", ";
        }
        cout << endl;
        cout << "----------------" << endl;

    }
    cout << "Liczba ciagow malejecych to: " << wszystkie_dane.size()<<endl;
    cout<< "================================="<<endl;
}
```

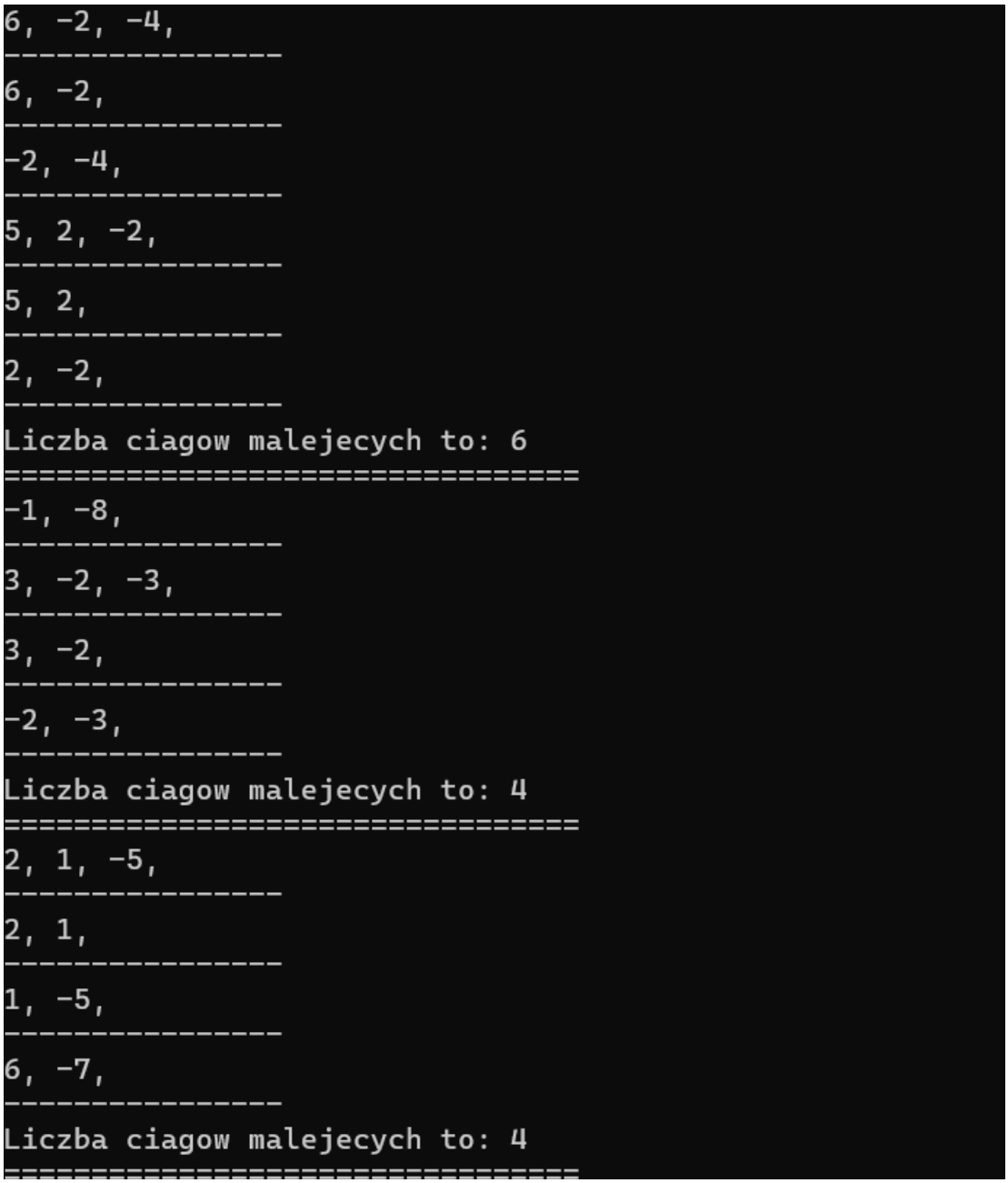
```
int main()
{
    clock_t start, end;
    start = clock();

    vector<int> dane_wejsciowe_1 = {7,5,-2,5,1,0};
    vector<int> dane_wejsciowe_2 = {-1,-8,3,-2,-3};
    vector<int> dane_wejsciowe_3 = {-2,1,2,1,-5,-4,6,-7};
    //podaje dane do funkcji wyszukujacej wszystkie podciagi malejace
    vector<vector<int>> wszystkie_ciagi_malejace = znajdz_ciagi_malejace(dane_wejsciowe_1);
    vector<vector<int>> wszystkie_ciagi_malejace_2 = znajdz_ciagi_malejace(dane_wejsciowe_2);
    vector<vector<int>> wszystkie_ciagi_malejace_3 = znajdz_ciagi_malejace(dane_wejsciowe_3);
    wypisz_ciagi(wszystkie_ciagi_malejace);
    wypisz_ciagi(wszystkie_ciagi_malejace_2);
    wypisz_ciagi(wszystkie_ciagi_malejace_3);

    end = clock();
    double time_taken = double(end - start) / double(CLOCKS_PER_SEC);
    cout << "Program wykonuje sie w : " << fixed
         << time_taken << setprecision(5);
    cout << " sec " << endl;

    return 0;
}
```

4. Działanie programu

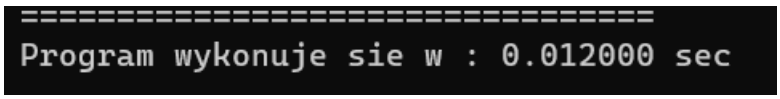
Program dla podanego ciągu liczb znajduje podciągi malejące, wypisuje je oraz podaje ich liczbę. Poniżej znajdują się wyniki działania programu dla kolejno: dłuższego ciągu(**Błąd! Nieprawidłowy odsyłacz do zakładki: wskazuje na nią samą.**) oraz kilku ciągów jednocześnie(Screen 2). Program podaje także czas, w którym procesor wykonuje obliczenia(Screen 3).



Screen 1



Screen 2



Screen 3

5. Wnioski

Wyniki wypisywane są w konsoli.
Program działa prawidłowo nawet dla bardzo długich ciągów.
Udało się pokazać działanie programu na liczbach dodatnich oraz ujemnych.
Program posiada kilka funkcji.
Zostało wykonanych kilka testów sprawdzających działanie programu.
W kodzie zostały zamieszczone komentarze, aby pomóc zrozumieć kod.
Został sporządzony schemat blokowy algorytmu oraz jego pseudokod.
Dodatkowo wykonane zostały pomiary złożoności czasowej programu.