



**WYDZIAŁ
MATEMATYKI
I FIZYKI STOSOWANEJ**
POLITECHNIKI RZESZOWSKIEJ

Marek Socha

Wyszukiwanie malejących podciągów z ciągu

Praca projektowa

Opiekun pracy:

dr. inż. Mariusz Borkowski, prof. PRz

Rzeszów, 2024

Spis treści

1.	Temat zadania projektowego	3
1.1.	Problematyka.....	3
1.2.	Przykładowe działanie programu.....	3
1.2.1.	Schemat blokowy przykładowego działania programu.....	4
1.2.2.	Pseudokod przykładowego działania programu	5
2.	Tworzenie kodu.....	6
2.1.	Kod programu	6
2.1.1	Używane biblioteki	8
2.1.2	Czas potrzebny do wykonania programu	8
2.1.3	Test działania programu.....	9
3.	Wnioski i Podsumowanie	10

1. Temat zadania projektowego

Dla ciągu (w postaci tablicy) zawierającego wartości całkowite, znajdź malejący podciąg o największej długości.

Przykład

Wejście: $A = [-10, 5, 8, 1, -4, -4, 10, 3, -1, 1]$

Wyjście: Najdłuższy malejący podciąg to $\{8, 1, -4\}$ oraz $\{10, 3, -1\}$

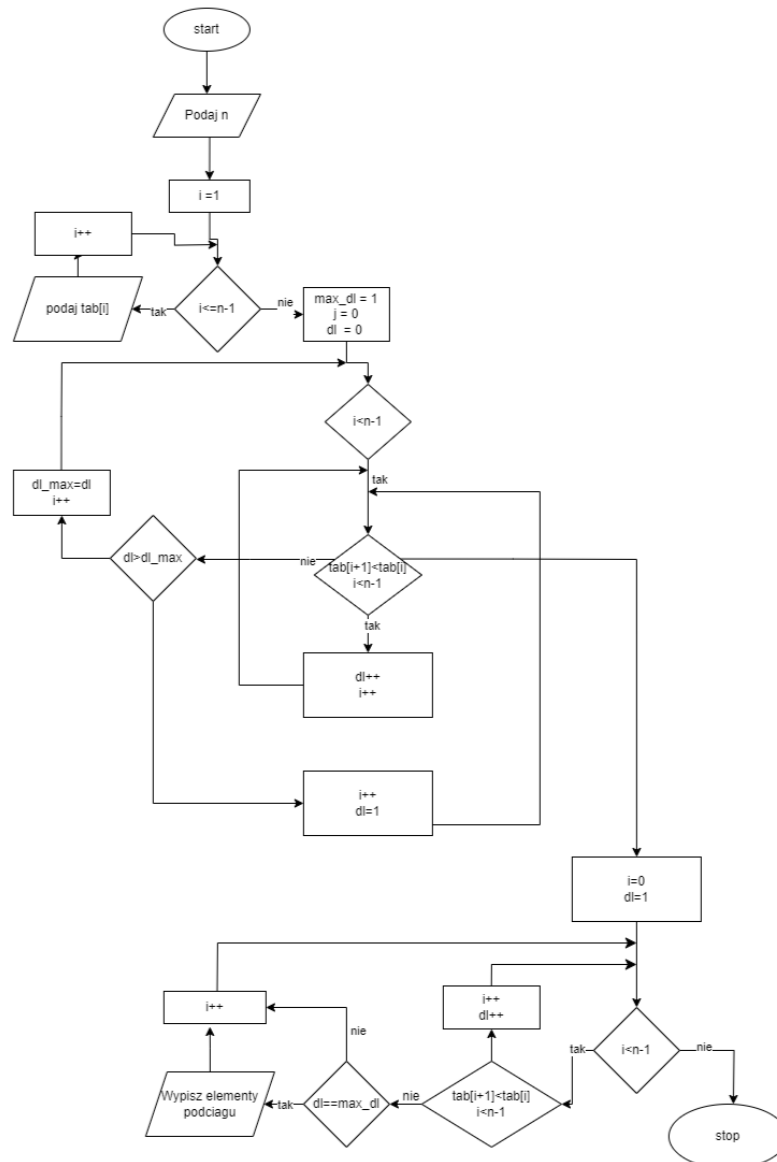
1.1. Problematyka

Program ma wyznaczyć najdłuższy malejący podciąg z ciągu w postaci tablicy, a następnie wypisać wszystkie malejące podciągi o największej długości. Program porównuje każdy element tablicy do następnego sprawdzając czy spełnia warunek $A[i] < A[i+1]$ i wypisuje najdłuższe podciągi spełniające ten warunek.

1.2. Przykładowe działanie programu

Program ma za zadanie wczytanie do tablicy dane wprowadzone przez użytkownika, następnie przeszukać ją pod kątem warunku $A[i] < A[i+1]$, zapisując ilość cyfr w najdłuższym podciągu spełniający ten warunek w zmiennej i dalej przeszukując tablicę za następnym większym podciągiem, aż do końca tablicy. Następnie mając zapisaną największą długość podciągu przeczesuje tablicę jeszcze raz, wypisując podciągi o największej długości.

1.2.1. Schemat blokowy przykładowego działania programu



Rysunek 1.2.1.1. Schemat blokowy algorytmu

1.2.2. Pseudokod przykładowego działania programu

```
1  max_dl <- 1
2  ciąg[] <- {-10, 5, 8, 1, -4, -4, 10, 3, -1, 1}
3  najdl_ciągi = []
4  i <- 1
5  j <- 0
6  dl <- 0
7  n <- 10
8  wypisz("Najdłuższy malejący podciąg to:")
9  dla i < n-1 wykonuj
10     i <- i+1
11     dl <- 1
12     dopóki ciąg[i+1]<ciąg[i] i i<n-1 wykonuj
13         dl <- dl+1
14         i <- i+1
15     jeśli (dl>max_dl) wykonaj
16         max_dl <- dl
17 jeśli (max_dl>1) wykonaj
18     i <- 1
19     dla i>n-1 wykonuj
20         dl <- 1
21         i <- i+1
22         dopóki ciąg[i+1]<ciąg[i] i i<n-1 wykonuj
23             i <- i+1
24             dl <- dl+1
25         jeśli (dl=max_dl) wykonaj
26             j <- i-dl+1
27             dla j <= wykonaj
28                 wypisz(ciąg[j])
29     wypisz("\n")
30
```

Rysunek 2.2.2.1 Pseudokod algorytmu

2. Tworzenie kodu

Program zostanie napisany tak by zaczytać dane wprowadzane przez użytkownika a następnie wypisać wynik spełniający założenia zadania i zapisać go w pliku tekstowym.

2.1. Kod programu

Kod programu umieszczam zarówno w sprawozdaniu jak i w repozytorium github (<https://github.com/Roy4ke/Projekt-AiSD-M.S>).

```
#include <iostream>
#include <vector>
#include <fstream>
#include <chrono>
using namespace std;
using namespace std::chrono;
// funkcja znajdująca maksymalną długość szukanego podciągu
int znajdzMaxDlugosc(const vector<int>& tablica) {
    // rozpoczęcie pomiaru czasu działania programu
    auto start = high_resolution_clock::now();
    // deklaracja zmiennej ciąg
    vector<int> ciag;
    // początkowa najdluzsza dlugosc ciagu
    int maxDlugosc = 0;
    // wyszukiwanie najdluzszych podciagow
    for (size_t i = 0; i < tablica.size(); i++) {
        if (ciag.empty() || tablica[i] < ciag.back()) {
            ciag.push_back(tablica[i]);
        } else {
            maxDlugosc = max(maxDlugosc, static_cast<int>(ciag.size()));
            ciag = {tablica[i]};
        }
    }
    maxDlugosc = max(maxDlugosc, static_cast<int>(ciag.size()));
    // zakończenie oraz wypisanie pomiaru czasu działania programu
    auto end = high_resolution_clock::now();
    auto duration = duration_cast<nanoseconds>(end - start);
    cout << "Czas wyznaczania najdluzszych malejacych podciagow: " << duration.count() << " nanosekund" <<
endl;
    return maxDlugosc;
}
// funkcja wypisująca najdluzszy znaleziony podciąg
void wypiszNajdluzszeCiagi(const vector<int>& tablica, int maxDlugosc, ofstream& plik) {
    vector<int> ciag;
```

```

// wypisanie długości nadszerego podciagu i zapisanie jej w pliku
cout << "Najdluzsze ciagi malejace (dlugosc = " << maxDlugosc << "):" << endl;
plik << "Najdluzsze ciagi malejace (dlugosc = " << maxDlugosc << "):" << endl;
// obliczanie maksymalnej dlugosci podciagu
for (size_t i = 0; i < tablica.size(); i++) {
    if (ciag.empty() || tablica[i] < ciag.back()) {
        ciag.push_back(tablica[i]);
    } else {
        if (ciag.size() == maxDlugosc) {
            for (int num : ciag) cout << num << " ";
            cout << endl;
            for (int num : ciag) plik << num << " ";
            plik << endl;
        }
        ciag = {tablica[i]};
    }
}
// ostatni ciag malejacy
if (ciag.size() == maxDlugosc) {
    for (int num : ciag) cout << num << " ";
    for (int num : ciag) plik << num << " ";
    // wpisanie wyniku dzialania do pliku oraz jego wyswietlenie w terminalu
    plik << endl;
    cout << endl;
}
}

int main() {
    // zmienna okreslajaca rozmiar tablicy
    int N;
    cout << "Podaj liczbe elementow tablicy" << endl;
    cin >> N;
    // deklaracja tablicy przechowujacej N elementow wprowadzanych przez uzytkownika
    vector<int> tablica(N);
    for (int i = 0; i < N; i++) {
        cout << "Podaj liczbe: " << endl;
        cin >> tablica[i];
    }
    // stworzenie pliku wynik.txt w ktorym beda umieszczane wyniki programu
    ofstream plik("wynik.txt");
    // sprawdzanie czy plik zostaje poprawnie odczytywany
    if (!plik) {
        cout << "Nie mozna otworzyc pliku do zapisu!" << endl;
        return 1; // Kod bledu
    }

    // znajdz maksymalna dlugosc ciagu malejacego
    int maxDlugosc = znajdzMaxDlugosc(tablica);
    // wypisz tylko te ciagi, ktore maja maksymalna dlugosc
    wypiszNajdluzszeCiagi(tablica, maxDlugosc, plik);
    // zakonczenie pracy na pliku tekstowym
    plik.close();
    return 0;
}

```

2.1.1 Używane biblioteki

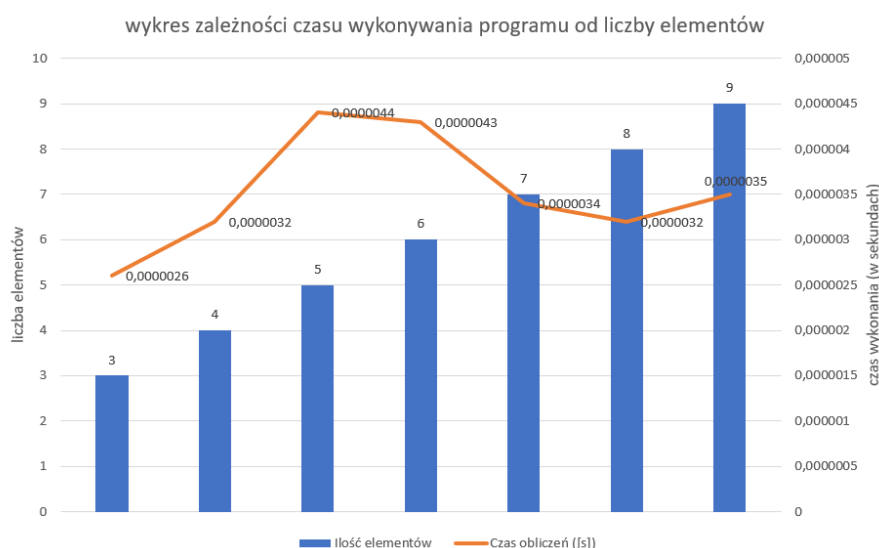
Program korzysta z biblioteki `iostream`, która służyła do zbudowania menu, zaczytywania z klawiatury danych, wykonywania obliczeń oraz do tworzenia pętli potrzebnych do rozwiązania problemu. Do zapisu wyniku działania programu wykorzystuję bibliotekę `fstream`, która umożliwia takie działania w języku C++. Aby obliczyć czas potrzebny do wykonania programu stosuję bibliotekę `chrono` umożliwiającą bieżący pomiar czasu.

2.1.2 Czas potrzebny do wykonania programu

Program jest liniowy względem liczby elementów w tablicy, posiadając złożoność obliczeniową $O(n)$. Czas liczony jest dla samego wykonania programu nie biorąc pod uwagę czasu, w którym użytkownik wprowadzał dane.

Ilość elementów	3 elementy	4 elementy	5 elementów	6 elementów	7 elementów	8 elementów	9 elementów
Czas obliczeń ([s])	0,0000026	0,0000032	0,0000044	0,0000043	0,0000034	0,0000032	0,0000035

Tabela 2.1.2.1. Czas obliczeń



Rysunek 3.1.2.1. Wykres potrzebnego czasu w zależności od liczby elementów

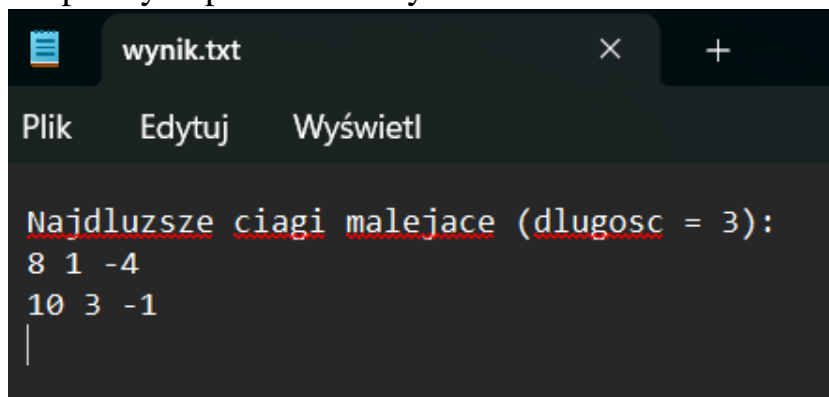
2.1.3 Test działania programu

Jako dane wprowadzane do programu posłużę się danymi zawartymi w treści zadania.

```
Podaj liczbe elementow tablicy
10
Podaj liczbe:
-10
Podaj liczbe:
5
Podaj liczbe:
8
Podaj liczbe:
1
Podaj liczbe:
-4
Podaj liczbe:
-4
Podaj liczbe:
10
Podaj liczbe:
3
Podaj liczbe:
-1
Podaj liczbe:
1
Czas wyznaczania najdluzszych malejacych podciagow: 3200 nanosekund
Najdluzsze ciagi malejace (dlugosc = 3):
8 1 -4
10 3 -1
```

Rysunek 2.1.3.1. Przykładowy wynik działania programu

Wynik zapisany w pliku tekstowym:



The screenshot shows a text editor window with a single tab titled 'wynik.txt'. The menu bar contains 'Plik', 'Edytuj', and 'Wyświetl'. The text area displays the following content:

```
Najdluzsze ciagi malejace (dlugosc = 3):
8 1 -4
10 3 -1
|
```

Rysunek 2.1.3.2 Zapis wyniku do pliku

3. Wnioski i Podsumowanie

Program efektywnie analizuje tablicę danych, wykonuje obliczenia i zwraca poprawny wynik. Przez niską złożoność czasową ($O(n)$) program nie potrzebuje dużo czasu by wykonać obliczenia.

Program jest przykładem prostego, lecz dobrze zaprojektowanego rozwiązania problemu analizy ciągów w tablicy liczb całkowitych.