



**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 .....	4
1.1.	Problematyka.....	4
1.2.	Przykładowe działanie programu.....	4
1.2.1.	Schemat blokowy przykładowego działania programu.....	5
1.2.2.	Pseudokod przykładowego działania programu .....	6
2.	Tworzenie kodu.....	7
2.1.	Kod programu .....	8
2.1.1	Używane biblioteki .....	10
2.1.2	Czas potrzebny do wykonania programu .....	11
2.1.3	Test działania programu.....	12
3.	Wnioski i Podsumowanie .....	13

# 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  $\{10, 3, -1\}$

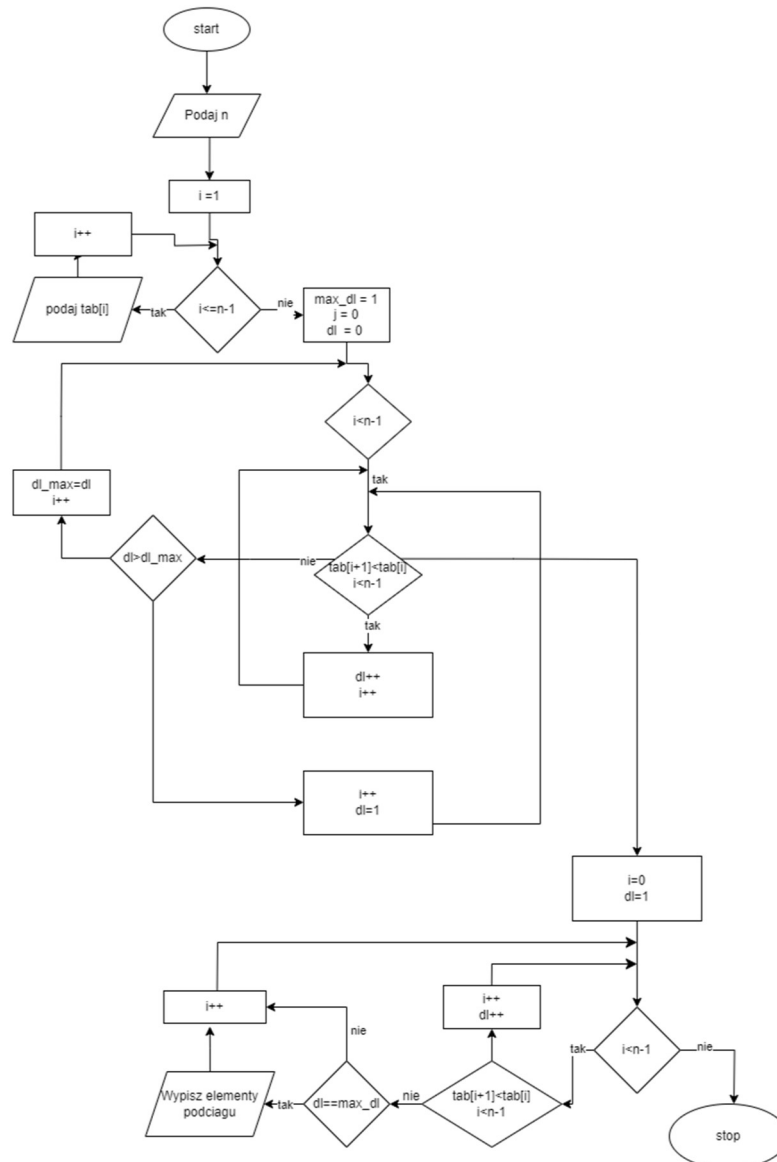
## 1.1. Problematyka

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 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>).

```
1  #include <iostream>
2  #include <vector>
3  #include <fstream>
4  #include <chrono>
5  using namespace std;
6  using namespace std::chrono;
7  // funkcja znajdująca maksymalną długość szukanego podciągu
8  int znajdzMaxDlugosc(const vector<int>& tablica) {
9      // rozpoczęcie pomiaru czasu działania programu
10     auto start = high_resolution_clock::now();
11     // deklaracja zmiennej ciąg
12     vector<int> ciag;
13     // początkowa najdłuższa długość ciągu
14     int maxDlugosc = 0;
15     // wyszukiwanie najdłuższych podciągów
16     for (size_t i = 0; i < tablica.size(); i++) {
17         if (ciag.empty() || tablica[i] < ciag.back()) {
18             ciag.push_back(tablica[i]);
19         } else {
20             maxDlugosc = max(maxDlugosc, static_cast<int>(ciag.size()));
21             ciag = {tablica[i]};
22         }
23     }
24     maxDlugosc = max(maxDlugosc, static_cast<int>(ciag.size()));
25     // zakończenie oraz wypisanie pomiaru czasu działania programu
26     auto end = high_resolution_clock::now();
27     auto duration = duration_cast<nanoseconds>(end - start);
28     cout << "Czas wyznaczania najdłuższych malejących podciągów: " << duration.count() << " nanosekund" << endl;
29     return maxDlugosc;
```



```

30 }
31 // funkcja wypisująca najdłuższy znaleziony podciąg
32 void wypiszNajdluzszeCiagi(const vector<int>& tablica, int maxDlugosc, ofstream& plik) {
33     vector<int> ciag;
34     // wypisanie długości nadszyszego podciagu i zapisanie jej w pliku
35     cout << "Najdluzsze ciagi malejace (dlugosc = " << maxDlugosc << "):" << endl;
36     plik << "Najdluzsze ciagi malejace (dlugosc = " << maxDlugosc << "):" << endl;
37     // obliczanie maksymalnej dlugosci podciagu
38     for (size_t i = 0; i < tablica.size(); i++) {
39         if (ciag.empty() || tablica[i] < ciag.back()) {
40             ciag.push_back(tablica[i]);
41         } else {
42             if (ciag.size() == maxDlugosc) {
43                 for (int num : ciag) cout << num << " ";
44                 cout << endl;
45                 for (int num : ciag) plik << num << " ";
46                 plik << endl;
47             }
48             ciag = {tablica[i]};
49         }
50     }
51     // ostatni ciag malejacy
52     if (ciag.size() == maxDlugosc) {
53         for (int num : ciag) cout << num << " ";
54         for (int num : ciag) plik << num << " ";
55         // wpisanie wyniku dzialania do pliku oraz jego wyswietlenie w terminalu
56         plik << endl;
57         cout << endl;
58     }
59 }
60 int main() {
61     // zmienna okreslajaca rozmiar tablicy
62     int N;
63     cout << "Podaj liczbe elementow tablicy" << endl;
64     cin >> N;
65     // deklaracja tablicy przechowujacej N elementow wprowadzanych przez uzytkownika
66     vector<int> tablica(N);
67     for (int i = 0; i < N; i++) {
68         cout << "Podaj liczbe: " << endl;
69         cin >> tablica[i];
70     }
71     // stworzenie pliku wynik.txt w ktorym beda umieszczane wyniki programu
72     ofstream plik("wynik.txt");
73     // sprawdzanie czy plik zostaje poprawnie odczytywany
74     if (!plik) {
75         cout << "Nie mozna otworzyc pliku do zapisu!" << endl;
76         return 1; // Kod bledu
77     }
78
79     // znajdz maksymalna dlugosc ciagu malejacego
80     int maxDlugosc = znajdzMaxDlugosc(tablica);
81     // wypisz tylko te ciagi, ktore maja maksymalna dlugosc
82     wypiszNajdluzszeCiagi(tablica, maxDlugosc, plik);
83     // zakonczenie pracy na pliku tekstowym
84     plik.close();
85     return 0;
86 }

```

Rysunek 2.3.1. Kod programu

### **2.1.1    Używane biblioteki**

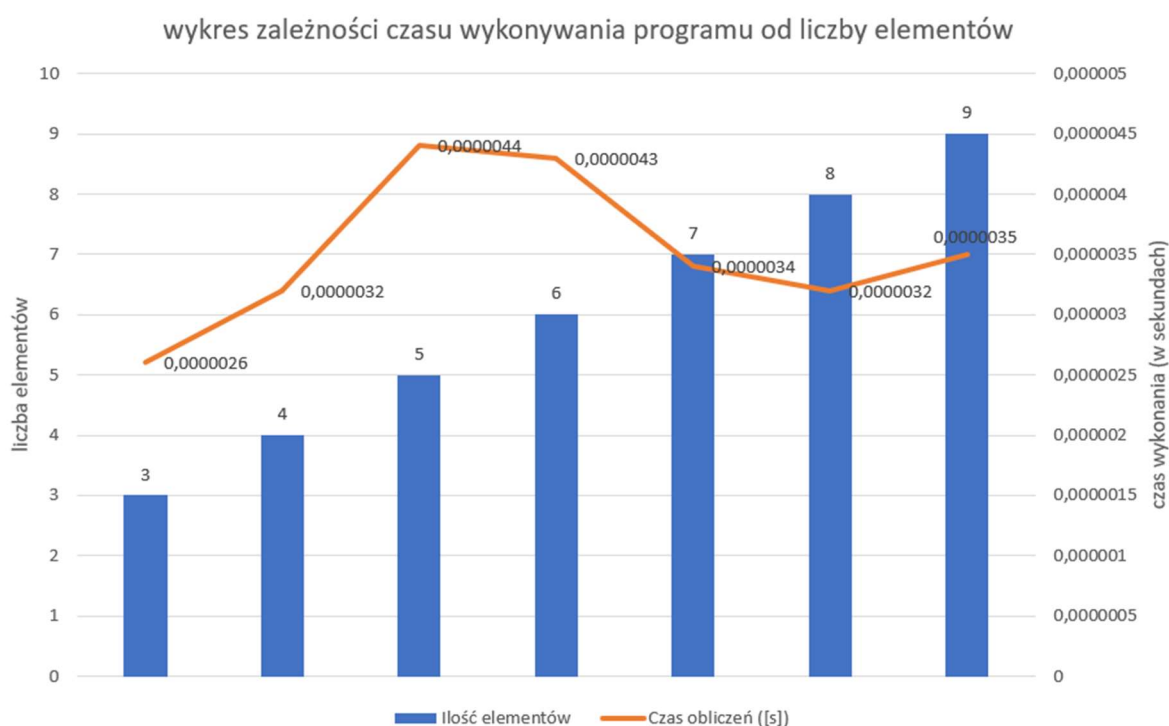
Program korzysta z biblioteki `iostream`, która służyła do zbudowania menu, zaczytywania z klawiatury danych, obliczeń oraz do tworzenia pętli potrzebnych do rozwiązania problemu. Do zapisu wyniku działania programu wykorzystuję bibliotekę `fstream`, która umożliwia takie działania w języku C++. Aby 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 bez 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 4.1.2.1. Wykres potrzebnego czasu w zależności od liczby elementów

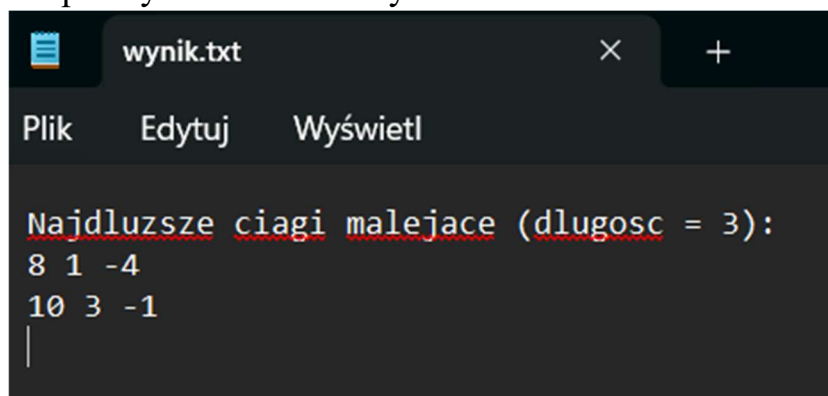
### 2.1.3 Test działania programu

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 liku tekstowym:



```
wynik.txt
Plik  Edytuj  Wyświetl
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.