



**POLITECHNIKA  
RZESZOWSKA**  
im. IGNACEGO ŁUKASIEWICZA

**Filip Kosiorowski**  
**173161**

**Projekt inżynierski**

Opiekun pracy:  
dr inż. Mariusz Borkowski , prof. PRz

Rzeszów, 2022



# 1.Spis treści

2.	Wstęp.....	4
3.	Pseudokod.....	4
4.	Kod Blokowy.....	5
	.....	5
5.	Testy działania algorytmu.....	6
6.	Złożoność obliczeniowa algorytmu .....	9
7.	Złożoność czasowa algorytmu .....	9
8.	Podsumowanie .....	10
9.	Kod .....	10

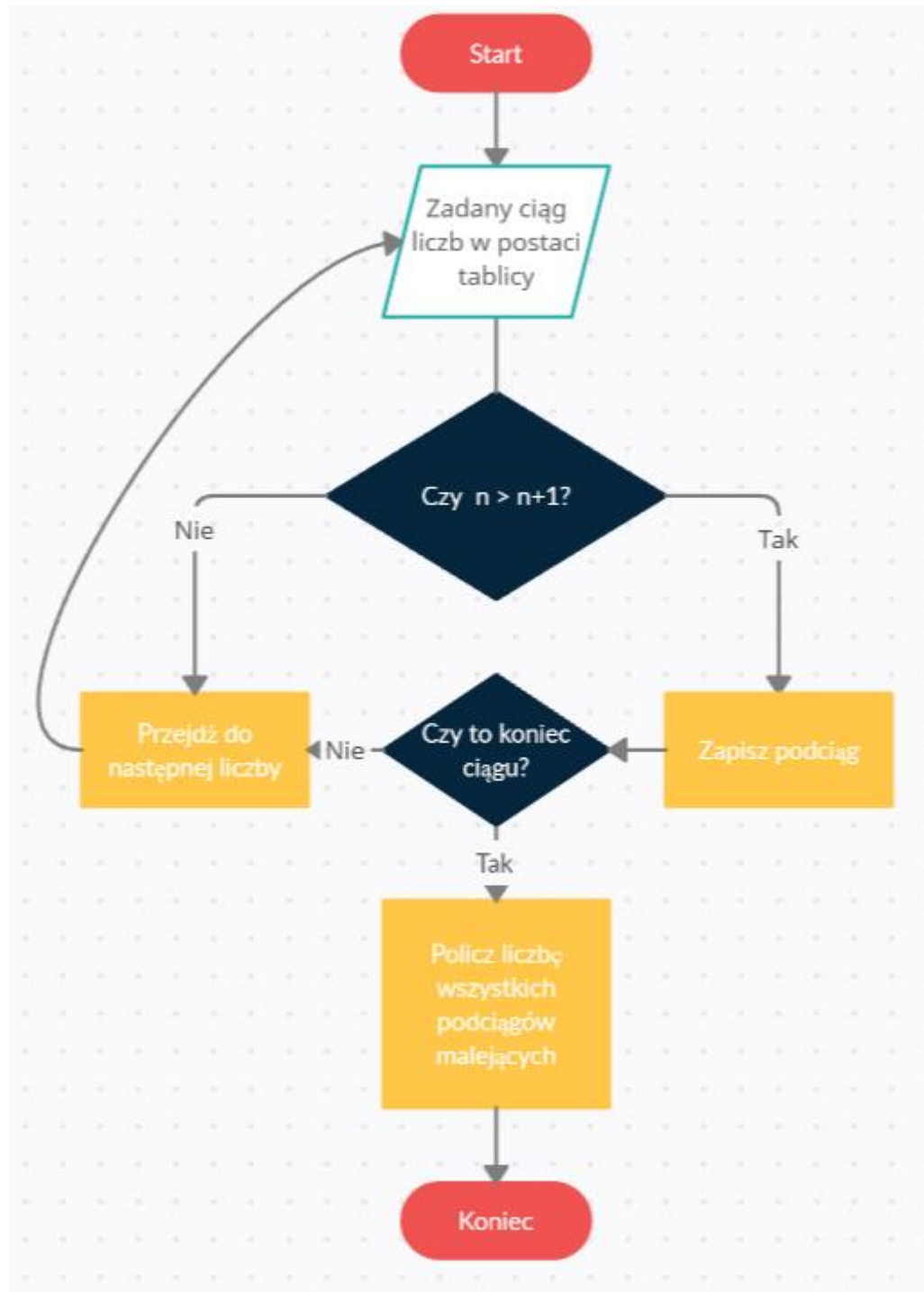
## 2.Wstęp

Zadanie polega na znalezieniu liczby wszystkich podciągów malejących dla zadanego ciągu liczb całkowitych.

## 3.Pseudokod

- Wczytaj wprowadzoną tablicę
- Jeśli  $n > n+1$ , wypisz  $[n, n+1]$
- Powtarzaj póki  $n < n+1$
- Jeśli  $n < n+1$ , przejdź do kolejnej liczby
- Wypisz wszystkie podciągi
- Wypisz liczbę podciągów

## 4.Kod Blokowy



Rysunek 1 Kod Blokowy

## 5. Testy działania algorytmu

```
Podaj rozmiar tablicy: 5
-----
Podaj element [1] tablicy: 5
Podaj element [2] tablicy: 4
Podaj element [3] tablicy: 2
Podaj element [4] tablicy: 2
Podaj element [5] tablicy: 1
-----
Element [1] tablicy = 5
Element [2] tablicy = 4
Element [3] tablicy = 2
Element [4] tablicy = 2
Element [5] tablicy = 1
-----
[5,4]
[4,2]
[2,1]
Liczba wszystkich podciągów malejących to 3
-----
Czas wykonywania programu: 6.11951s
Process returned 0 (0x0)   execution time : 6.146 s
Press any key to continue.
```

*Rysunek 3 Test 1*

```
Podaj rozmiar tablicy: 3
-----
Podaj element [1] tablicy: 2
Podaj element [2] tablicy: 5
Podaj element [3] tablicy: 3
-----
Element [1] tablicy = 2
Element [2] tablicy = 5
Element [3] tablicy = 3
-----
[5,3]
Liczba wszystkich podciągów malejących to 1
-----
Czas wykonywania programu: 4.99474s
Process returned 0 (0x0)   execution time : 5.018 s
Press any key to continue.
```

*Rysunek 2 Test 2*

```
Podaj rozmiar tablicy: 5
-----
Podaj element [1] tablicy: 1
Podaj element [2] tablicy: 2
Podaj element [3] tablicy: 4
Podaj element [4] tablicy: 6
Podaj element [5] tablicy: 7
-----
Element [1] tablicy = 1
Element [2] tablicy = 2
Element [3] tablicy = 4
Element [4] tablicy = 6
Element [5] tablicy = 7
-----
Liczba wszystkich podciągów malejących to 0
-----
Czas wykonywania programu: 6.65808s
Process returned 0 (0x0)   execution time : 6.669 s
Press any key to continue.
```

*Rysunek 4 Test 3*

```

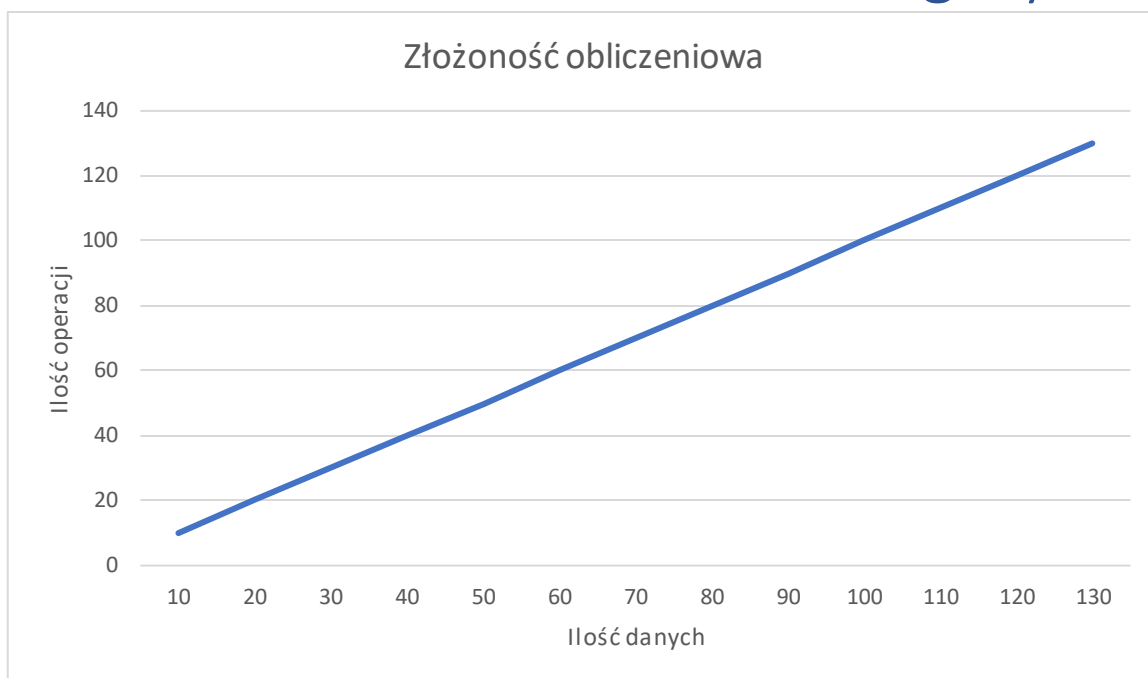
Podaj rozmiar tablicy: 20
-----
Podaj element [1] tablicy: 1
Podaj element [2] tablicy: 2
Podaj element [3] tablicy: 3
Podaj element [4] tablicy: 4
Podaj element [5] tablicy: 99
Podaj element [6] tablicy: 87
Podaj element [7] tablicy: 76
Podaj element [8] tablicy: 65
Podaj element [9] tablicy: 54
Podaj element [10] tablicy: 5
Podaj element [11] tablicy: 4
Podaj element [12] tablicy: 3
Podaj element [13] tablicy: 6
Podaj element [14] tablicy: 7
Podaj element [15] tablicy: 43
Podaj element [16] tablicy: 101
Podaj element [17] tablicy: 202
Podaj element [18] tablicy: 69
Podaj element [19] tablicy: 42
Podaj element [20] tablicy: 9
-----
Element [1] tablicy = 1
Element [2] tablicy = 2
Element [3] tablicy = 3
Element [4] tablicy = 4
Element [5] tablicy = 99
Element [6] tablicy = 87
Element [7] tablicy = 76
Element [8] tablicy = 65
Element [9] tablicy = 54
Element [10] tablicy = 5
Element [11] tablicy = 4
Element [12] tablicy = 3
Element [13] tablicy = 6
Element [14] tablicy = 7
Element [15] tablicy = 43
Element [16] tablicy = 101
Element [17] tablicy = 202
Element [18] tablicy = 69
Element [19] tablicy = 42
Element [20] tablicy = 9
-----
[99,87]
[87,76]
[76,65]
[65,54]
[54,5]
[5,4]
[4,3]
[202,69]
[69,42]
[42,9]
Liczba wszystkich podciagow malejacych to 10
-----
Czas wykonywania programu: 34.6758s
Process returned 0 (0x0)   execution time : 34.685 s
Press any key to continue.

```

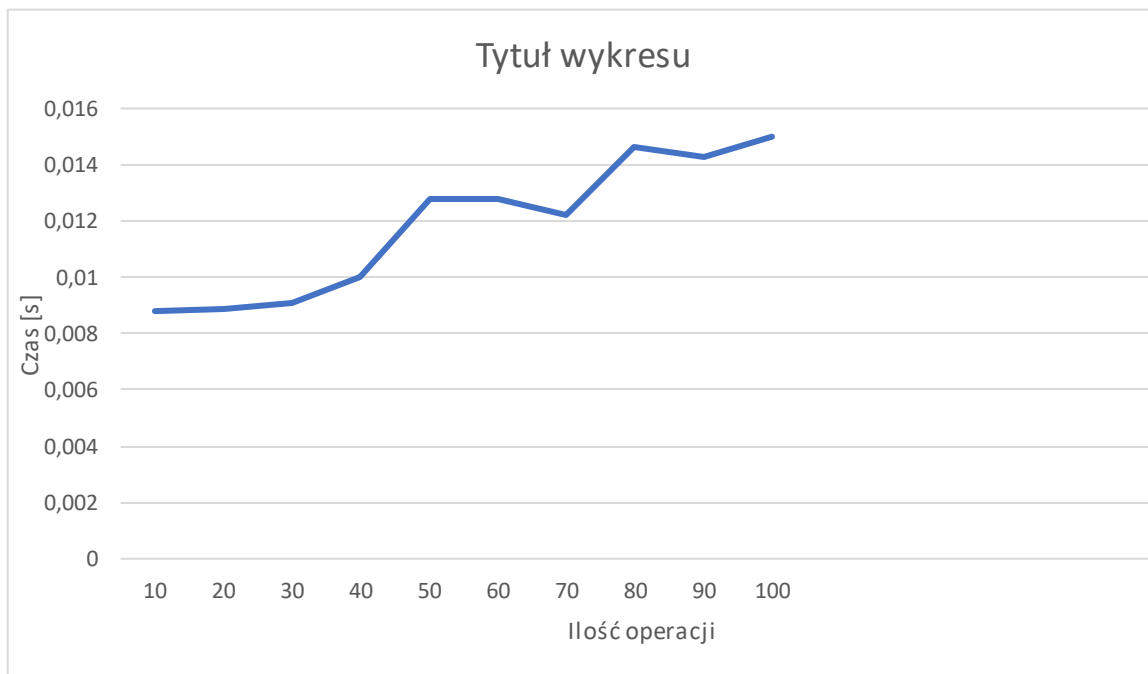
Rysunek 5Test 4



## 6. Złożoność obliczeniowa algorytmu



## 7. Złożoność czasowa algorytmu



## 8.Podsumowanie

Algorytm działa dla bardzo długich ciągów. Ma niską złożoność i działa sprawnie. Program jest przejrzysty.

## 9.Kod

```
#include <iostream>
#include <vector>
#include <chrono>

using namespace std;

void tablica(){

    vector<float> tab; //deklaracja wektora typu float
    int r,temp; // deklaracja zmiennej typu int
    cout << "Podaj rozmiar tablicy: ";
    cin >> r;
    cout << "-----"
<< endl;
    for(int i = 0; i < r; i++)
    {
        cout << "Podaj element "<< "["<<i+1<<"]"<< "
tablicy: ";
        cin >> temp; //zmienna tymczasowa
        tab.push_back(temp); //dodaje element na sam
koniec tablicy
    }
```

```

        cout << "-----"
<< endl;

    for(int i = 0; i < r; i++)
    {
        cout << "Element " << "[" << i+1 << "]" << " tablicy
= " << tab[i] << endl;
    }
    cout << "-----"
<< endl;
int n=0;
int liczenie=0;
    for(int i = 0; i + 1 < tab.size(); i++)
    { n=0;
      if(tab[i] > tab[i + 1])
      {
          cout << "[" << tab[i] << ", " << tab[i+1] << "]" <<
endl;
          n++; //Jeśli if znajdzie podciąg, n się zwiększy
      }
      if(n !=0){ //jeśli n !=0 tzn, że znaleźliśmy podciąg
          liczenie++; // zliczanie ilości ciągów
      }
    }
    //jesli nie wyzerujemy n to gdy nie znajdziemy
    podciągu to liczenie i tak sie zwiększy o 1
    cout << "Liczba wszystkich podciagow malejacych
to " << liczenie << endl;
}

```

```

int main()
{
    auto start = chrono::steady_clock::now();
    tablica();
    auto finish = chrono::steady_clock::now();
    cout << "-----" <<
    endl;
    cout << "Czas wykonywania programu: " <<
    chrono::duration_cast<chrono::microseconds>(finish-
start).count()/1000000.0 << "s";
    return 0;
}

```