WYDZIAŁ
MATEMATYKI
I FIZYKI STOSOWANEJ
POLITECHNIKI RZESZOWSKIEJ

Szymon Sulisz, numer albumu 173221

Uporządkuj zadaną tablicę wielu powtarzających się elementów tak aby zgromadzić te same elementy zachowując kolejność ich pierwszego wystąpienia

Projekt zaliczeniowy nr 1

Rzeszów, 2022

1. Spis treści

2. Spis ilustracji

3. Temat

Uporządkowanie tablicy składającej się z wielu powtarzających się elementów, w taki sposób aby zgromadzić te same dane zachowując kolejność ich pierwszego wystąpienia
Przykład
Wejście : [1, 2, 3, 1, 2, 1]
Wyjście : [1, 1, 1, 2, 2, 3]

Wejście : [5, 4, 5, 5, 3, 1, 2, 2, 4]
Wyjście : [5, 5, 5, 4, 4, 3, 1, 2, 2]

4. Analiza, projektowanie.

Zasada działania programu

Program ten ma za zadanie uporządkowanie liczb znajdujących się w tabeli zgodnie z kolejnością ich występowania w danej tabeli. Dane te muszą zostać wylosowane do tabeli w związku z czym musimy wprowadzić dane odpowiedzialne za rozmiar naszej tablicy oraz jej dolny i górny zakres.

Struktury danych

Dane przechowywyane będą w tabeli składającej się z n – elementów, gdzie n będzie liczbą dodatnią, całkowitą, większą od zera. Zakres danych, które będą losowane do naszej tabeli będzie również należał do liczb dodatnich całkowitych z wyjątkiem zera. Użycie ograniczonej n – elementowej tablicy pozwoli nam na zwiększenie wydajności działania naszego programu oraz ograniczy nam możliwość popełnienia błędów mogących pojawić się podczas pracy na tablicach dynamicznych.

Metodyka

1) Funkcja wprowadzająca dane

Pierwszą rzeczą, która musimy stworzyć podczas pracy nad naszym programem jest funkcja, dzięki której będziemy mogli wprowadzić dane, na których to w późniejszym czasie będziemy mogli pracować.

Funkcje tą musimy rozpocząć od zadeklarowania rozmiaru, który w późniejszym czasie będzie miała nasza tablica. Po wprowadzeniu danych musimy sprawdzić czy rozmiar naszej tablicy jest liczba całkowitą dodatnią.

Etapem kolejnym jest podanie zakresów naszej tabeli oraz skontrolowanie czy na pewno zmienna odpowiadająca za wartość dolnego zakresu jest mniejsza od górnego zakresu i czy obie liczby są dodatnie oraz całkowite.

Następnie musimy do naszej tabeli wprowadzić dane za pomocą funkcji losującej. Dane te muszą pochodzić z podanego wcześniej zakresu oraz ich ilość nie może przekraczać rozmiaru tablicy podanego wcześniej przez użytkownika.

Funkcja ta powinna zakończyć się zwróceniem wylosowanej tablicy do funkcji głównej

Złożoność tej funkcji jest równa n.

2) Funkcja sortująca dane

W tej funkcji musimy stworzyć pętlem która odpowiadała będzie za odwołanie do kolejnego elementu znajdującego się w naszej wylosowanej wcześniej tabeli.

W tej pętli powinna zostać napisana kolejna pętla, która odpowiadała będzie za odwołanie do kolejnych elementów tabeli. Pętla ta ma zaczynać odwołania do elementów tabeli od końca do elementu, który jest aktualnie analizowany przez pętlę zewnętrzną.

W pętli wewnętrznej musi znajdować się operator warunkowy, który będzie sprawdzał czy elementy tablicy, które są akurat analizowane przez operatory z pętli są sobie równe.

Jeżeli wartości elementów są sobie równe to element, analizowany przez pętle wewnętrzną powinien zostać zamieniony miejscem z elementem o jeden mniejszym.

Pętle te powinny zwiększać swoje zmienne licznikowe o 1 w każdym jej kolejnym powtórzeniu.

Funkcja ta kończy się zwróceniem posortowanej tablicy zgodnie z poleceniem.

Złożoność tej funkcji jest równa n^2.

3) Funkcja zapisująca dane do pliku

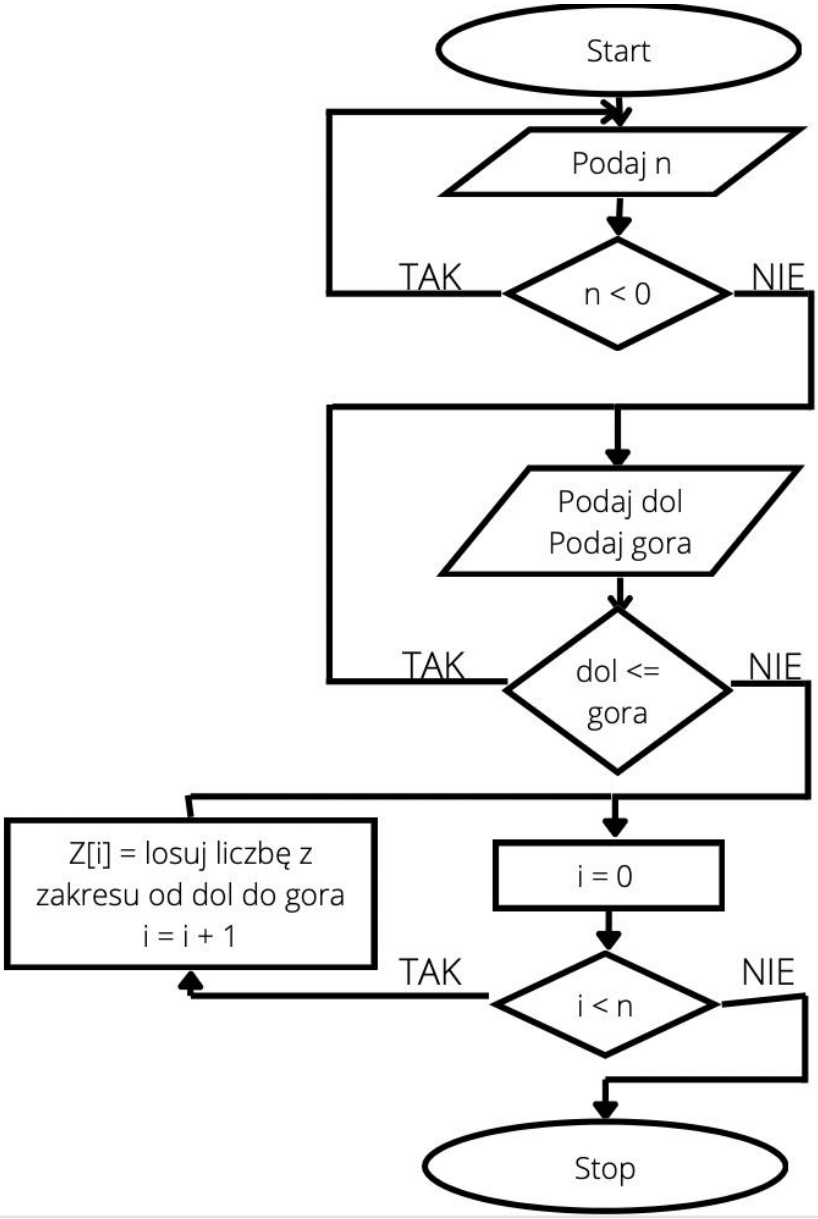
Funkcja ta odpowiadała będzie za zapis danych do pliku.

Po pierwsze będziemy musieli otworzyć plik o nazwie dane.txt oraz włączyć możliwość zapisywania danych do tego pliku oraz dopisywania tych danych do tego pliku bez usuwania poprzednich danych.

Następnie tworzymy pętle w której będzie znajdowała się komenda zapisująca kolejne elementy do tablicy zgodnie ze zmienną, która z każdym powtórzeniem pętli będzie zwiększała swoją wartość o 1 aż dopóki nie osiągnie wartości o jeden mniejszej od n.

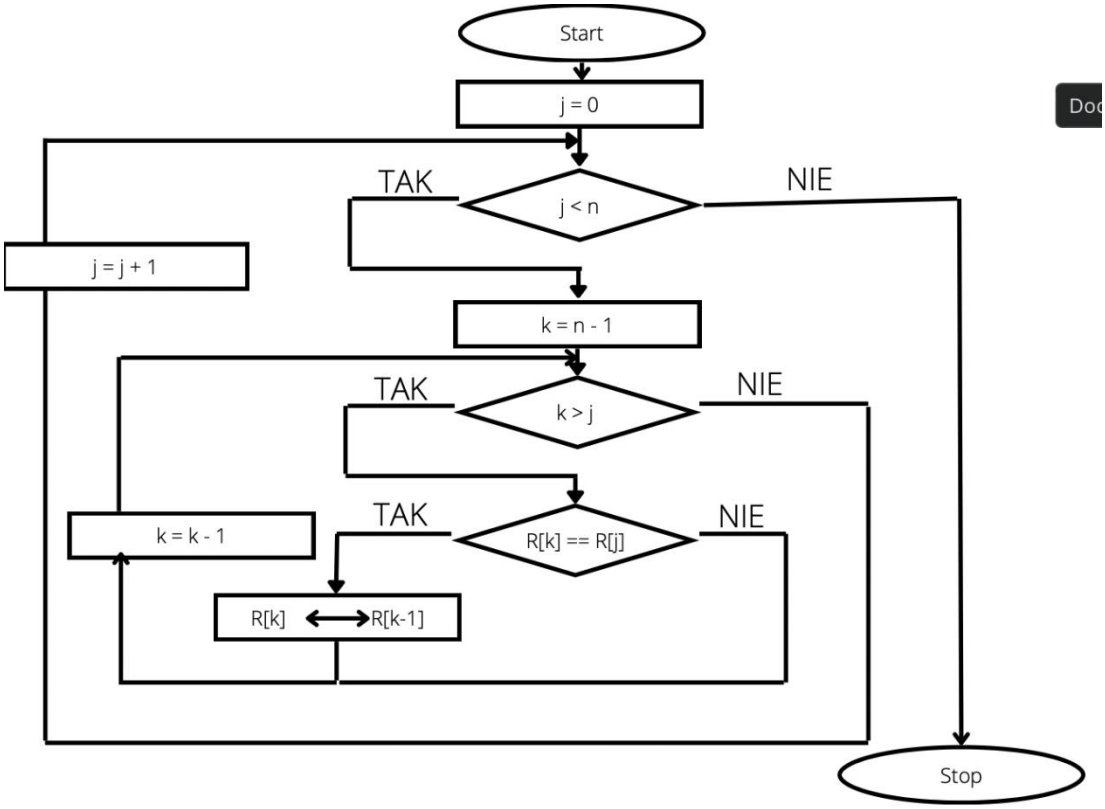
5. Schemat blokowy

Wprowadzanie danych



Rysunek 1

Sortowanie danych



Rysunek 2

7. Pseudokod

Wprowadzanie danych

```
Wczytaj (n)
Dopóki ( a = 1 )
        Jeżeli ( n < 0 )
                Wczytaj (n)
        W przeciwnym razie a = 0
Wczytaj (dol)
Wczytaj (gora)
a = 1
Dopóki ( a = 1 )
        Jeżeli ( dol <= gora )
                Wczytaj (dol)
                Wczytaj (gora)
        W przeciwnym razie a = 0
Z[n]
Dla i = 0 do n wykonuj
        Z[i] = losuj liczby z zakresu od dol do gora
        i = i + 1
Zwróć wartość Z[n]
```

Sortowanie danych

```
Dla j = 0 do n wykonuj
        Dla k = n – 1 do j wykonuj
                Jeżeli ( Z[k] = Z[j] )
                        Zamień R[k] z R[k-1]
                k = k – 1
        j = j + 1
Zwróć wartość Z[n]
```

8. Analizy

Tabela pokazująca czas trwania pliku w zależności od rozmiaru tablicy.

Rozmiar tabeli	Czas trwania bez zapisywania danych do pliku(s)	Czas trwania z zapisywaniem danych do pliku (s)
10	0.000219	0.001083
50	0.000371	0.000948
100	0.000455	0.001233
200	0.000824	0.001855
300	0.001106	0.002257
400	0.001812	0.003033
500	0.002576	0.003278
650	0.002965	0.004562
800	0.004368	0.005035
1000	0.005995	0.006809

Tabela 1

9. Kod programu

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include <fstream>

using namespace std;

int n, dol, gora; // zmienne do rozmiaru i zakresu tablicy
fstream plik; // zmienna plikowa

int zakres_gora() // funkcja podajaca gorny zakres
{
   cout << "Podaj gorny zakres losowania liczb: ";
   cin >> gora;
   return gora;
}
int zakres_dol() // funkcja podajaca dolny zakres
{
   cout << "Podaj dolny zakres losowanych liczb: ";
   cin >> dol;
   return dol;
}
int rozmiar_tablicy() // funkcja podajaca rozmiar tablicy i sprawdzajaca jej poprawnosc
{
   cout << "Podaj rozmiar tablicy: ";
   int a = 1;
   cin >> n;
   while( a == 1 )
   {

      if( n < 0 )
      {
         cout << "Podales zly rozmiar tablicy. Podaj go jeszcze raz: ";
         cin >> n;
      } else a = 0;
   } // instrukcja warunkowa sprawdzajaca poprawnosc rozmiaru tablicy
   return n;
}
void sprawdzenie() // funkcja sprawdzajaca czy sa dobre zakresy
{
   int a = 1;
   while( a == 1)
   {
      if( dol >= gora)
      {
         cout << "Podales zle zakresy" << endl;
         zakres_dol();
         zakres_gora();
```

```
        }else a = 0; // instrukcja warunkowa sprawdzajaca poprawnosc zakresow i ich
     ewentualna korekta
    }
}
void zapis(int Z[]) // funkcja zapisujaca dane do pliku dane.txt
{
    plik.open("Dane.txt", ios::out | ios::app);
    if(plik.good() == true ) // instrukcja sprawdzajaca poprawnosc otwarcia pliku
    {
        for( int i = 0 ; i < n ; i++)
        {
            plik << Z[i] << " ";
        }
        plik << endl;
        plik.close();
    }else cout << "Nie udalo sie otworzyc pliku";
}
int losowanie_liczb(int Z[]) // funkcja losujaca dane do tabeli
{
    srand(time(NULL));
    cout << "Wylosowane dla ciebie liczby to: " << endl;
    for ( int i = 0 ; i < n ; i++)
    {
        Z[i] = rand() % (gora - dol + 1) + dol;
        cout << Z[i] << " ";
    }
    cout << endl;
    return Z[n];
}
int sortowanie(int R[]) // funkcja zawierajaca algorytm sortowania majacy na celu posortowac
     dane wzgledem ich wystepowania w tablicy
{
    for( int j = 0 ; j < n ; j++)
    {
        for( int k = n-1 ; k > j ; k--)
        {
            if( R[k] == R[j] )
            {
                swap( R[k], R[k-1]);
            }
        }
    }
    int i = 0;
    cout << "Tablica posortowana wyglada nastepujaco: " << endl;
    while( i < n )
    {
        cout << R[i] << " ";
        i++;
    }
    return R[n];
```

```
}
int main()
{
   clock_t start = clock(); // zmienna pobierajaca czas rozpoczecia dzialania programu
   rozmiar_tablicy();
   zakres_dol();
   zakres_gora();
   sprawdzenie();
   cout << "Rozmiar twojej tablicy wynosi " << n << " dolny zakres wynosi " << dol << " a
    gorny " << gora << endl;
   int Z[n]; // deklaracja tablicy o rozmiarze n podanym za pomoca funkcji rozmiar_tablicy()
   losowanie_liczb(Z);
   zapis(Z);
   sortowanie(Z);
   zapis(Z);
   clock_t end = clock(); //zmienna pobierajaca czas zakonczenia dzialania programu
   double x = end - start;
   double czas = x / CLOCKS_PER_SEC; // obliczenie czasu dzialania programu
   cout << endl << "Czas trwania programu wynosi " << czas;
   plik.open("Czasy.txt", ios::out | ios::app); // zapis danych o dlugosci trwania programu do
    pliku czasy.txt
   if( plik.good() == true ) // funkcja sprawdzajaco poprawnosc otwarcia pliku
   {
      plik << czas << endl;
      plik.close();
   }else cout << "Nie udalo sie otworzyc pliku";
   return 0;
}
```

10. Podsumowanie

Program poprawnie rozwiązuje problem zadeklarowany na rozpoczęciu prac. Czasy trwania programu są zależne od rozmiaru tablicy i wzrastają w powolnym tempie w stosunku do wzrostu tablicy. Czas działania programu, który zapisuje dane do pliku jest większy od programu, który tego nie robi dla małych rozmiarów tablicy. Lecz przy większych rozmiarach tablic ta różnica jest znikoma.