Politechnika Rzeszowska, 03.12.2020r.

Zadanie projektowe nr 1 Algorytmy i Struktury Danych

Inżynieria i analiza danych, I rok

Wiktor Barć
Numer indeksu: 166632
Grupa: P01

1. Opis problemu

1.1. Treść zadania

Dla ciągu (w postaci tablicy) zawierającego wyłącznie wartości 0 lub 1, znajdź podciąg zawierający równą liczbę zer i jedynek, którego długość jest największa.
Przykład:
Wejście: 0, 0, 1, 0, 1, 0, 0
Wyjście: Najdłuższym podciągiem są 0, 1, 0, 1 oraz 1, 0, 1, 0

1.2. Przeznaczenie programu

Program ma za zadanie znaleźć w ciągu (w postaci tablicy) zawierającego wyłącznie wartości 0 lub 1, podciąg o możliwie największej długości.

1.3. Tablice w C++

[1]**Tablica** (ang. array) lub **wektor** (ang. vector) jest **złożoną strukturą danych** (ang. compound data structure) zbudowaną z ciągu elementów tego samego typu. W pamięci komputera elementy tablicy są ułożone kolejno jeden obok drugiego. Dostęp do elementu odbywa się poprzez numer zwany indeksem. Na podstawie indeksu, rozmiaru elementu oraz adresu początku tablicy komputer oblicza adres elementu i w ten sposób uzyskujemy do niego dostęp.

We współczesnych językach programowania tablice są stosowane powszechnie do przechowywania danych podobnego rodzaju. Przy ich pomocy można zapisywać ciągi liczbowe, wyniki pomiarów różnych wielkości oraz tworzyć złożone bazy danych. Liczba zastosowań tablic jest w zasadzie ograniczona naszą wyobraźnią. Podstawową zaletą tablic jest prostota przetwarzania ich elementów. Dzięki dostępowi poprzez indeksy, elementy tablic daje się łatwo przetwarzać w pętlach iteracyjnych.

1.4. Krótka charakterystyka programu

Program ma na celu odnalezienie najdłuższego podciągu dla ciągu (w postaci tablicy) zawierającego wyłącznie wartości 0 i 1. Na początku definiujemy naszą tablicę wejściową.

Po wykonaniu tego zadania, obliczamy ilość zer i jedynek w tablicy. Następnie wyliczamy maksymalną długość ciągu. W późniejszych krokach przeszukujemy tablicę wejściową.

Po uruchomieniu programu na konsoli wyświetlają się wyniki oraz tworzy się plik tekstowy o nazwie „WYNIKI.txt”, w którym zapisują się najdłuższe podciągi.

[1] Źródło: https://eduinf.waw.pl/inf/alg/001_search/0028.php

2. Rozwiązanie zadania

2.1. Kod programu

Program wykonany w środowisku Code::Blocks IDE w języku C++

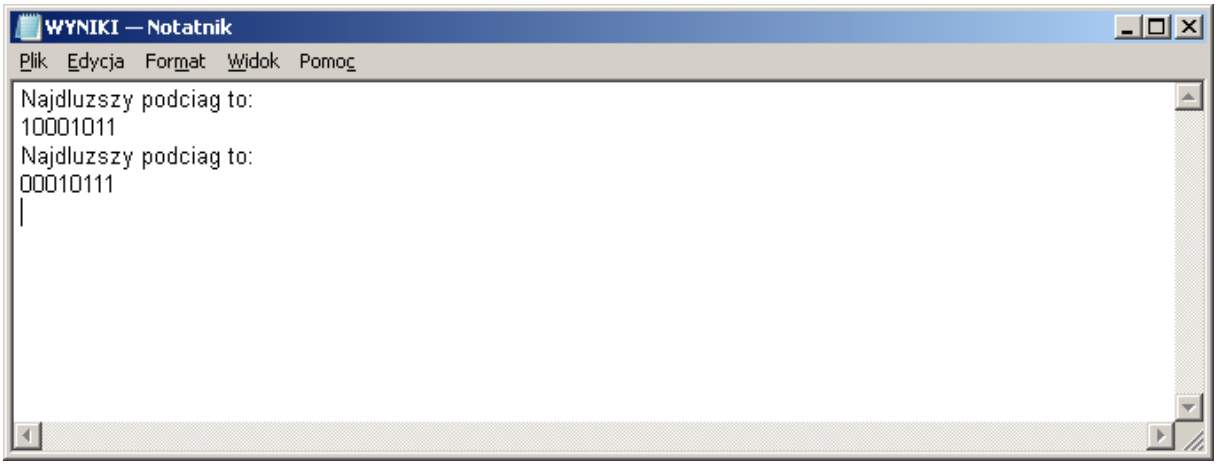
Program.cpp

```
1   #include <iostream>
2   #include <cstdlib>
3   #include <time.h>
4   #include <fstream>
5   using namespace std;
6
7   void szukaj_podciagu(int t[], int n)
8   {
9   //deklaracja zmiennych i przypisanie wartosci
10      int* tempArray;
11      int zero = 0;
12      int jeden = 0;
13      int patternzero = 0;
14      int patternjeden = 0;
15      int max = 0;
16      int znalezione = 0;
17      int i, j;
18
19  fstream z;
20  //otwarcie pliku,w ktorym zapisywane sa podciagi
21  z.open("WYNIKI.txt",ios::out);
22
23  //obliczamy ilosc zer i jedynek w tablicy
24      for (i = 0; i < n; i++)
25      {
26          if (t[i] == 0)
27              zero++;
28          else
29              jeden++;
30      }
31
32  //obliczamy maksymalna dlugosc ciagu
33      if (zero < jeden)
34          max = zero * 2;
35      if (jeden < zero)
36          max = jeden * 2;
37      if (jeden == zero)
38          max = jeden * 2;
39
```

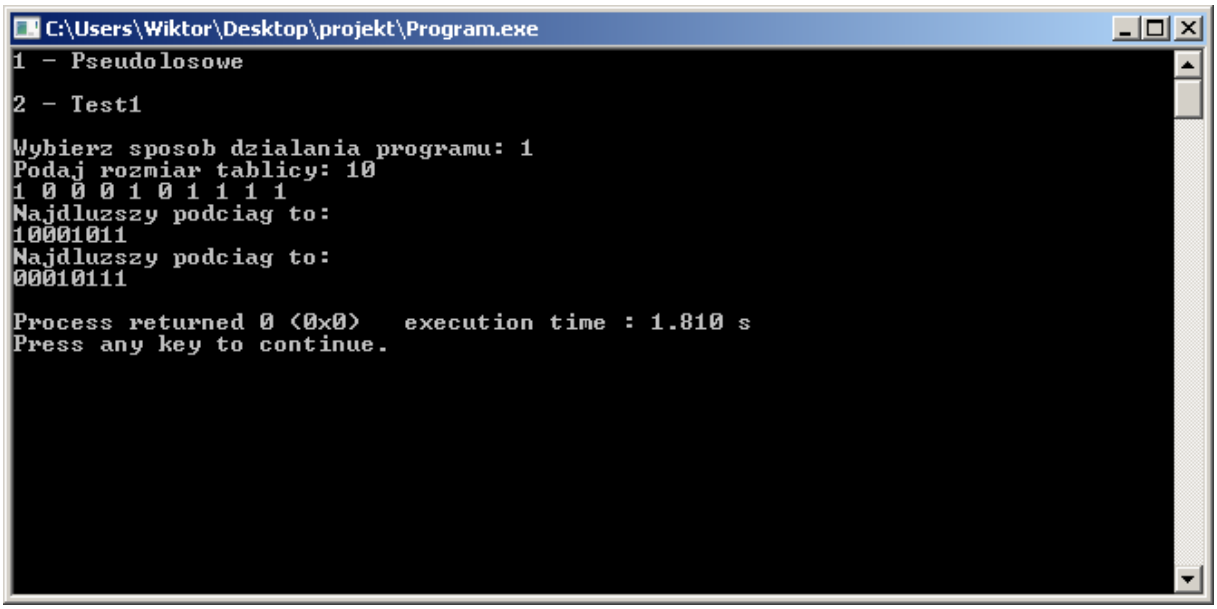
```
39
40      //tworzenie tablicy
41          tempArray = new int[max];
42
43      //przeszukujemy tablice wejsciowa
44          while (max > 0)
45          {
46              for (i = 0; i <= n - max; i++)
47              {
48                  patternzero = 0;
49                  patternjeden = 0;
50                  for (j = 0 + i; j <max + i; j++)
51                  {
52                      if (t[j] == 0)
53                      {
54                          patternzero++;
55                          tempArray[j - i] = 0;
56                      }
57                      else
58                      {
59                          patternjeden++;
60                          tempArray[j - i] = 1;
61                      }
62                  }
63                  if (patternjeden == patternzero)
64                  {
65                      cout<<"Najdluzszy podciag to: "<<endl;
66                      for (j = 0; j < max; j++)
67                      {
68                      cout<<tempArray[j];
69                      }
70                      cout<<"\n";
71                      for (j = 0; j < max; j++)
72                      {
73                          z <<tempArray[j];
74                          znalezione++;
75                      }
76                      z << "\n";
77                  }
78
79                  patternzero = 0;
80                  patternjeden = 0;
81              }
82              if (znalezione > 0)
83              {
84                  delete[] tempArray;
85                  break;
86              }
87              max--;
88          }
89          if (znalezione == 0)
90          delete[] tempArray;
91          z.close();
92
93
94      }
95
96      void test_1()
97      {
98          cout<<"Test 1"<<endl<<endl;
99      //deklaracja stalej tablicy
100         int a[2]={0,1};
101         cout<<"A[] = ";
102         for(int i = 0; i<2;i++) cout<<a[i]<<" ";
103         cout<<endl;
104         szukaj_podciagu(a,2);
105         cout<<endl;
106
107     }
108
109
```

```
110  int main ()
111  {
112      cout<<"1 - Pseudolosowe"<<endl<<endl;
113      cout<<"2 - Test1"<<endl<<endl;
114
115       int sposob;
116       cout<<"Wybierz sposob dzialania programu: ";
117       cin>>sposob;
118
119  //stworzenie  menu
120       switch (sposob)
121      {
122
123      case 1:
124      {
125
126      int i,n;
127      int *t;
128      cout<<"Podaj rozmiar tablicy: "; //podajemy rozmiar tablicy dynamicznej
129      cin>>n;
130      t=new int[n];
131          srand((unsigned)time(NULL));
132
133  //Wypelnienie tablicy 0 lub 1
134          for(int i=0;i<n;i++)
135          {
136              t[i]= (rand()%2);
137              cout<<t[i]<<" ";
138          }
139              cout<<endl;
140
141      szukaj_podciagu(t,n);
142      delete [] t;
143      break;
144      }
145
146      case 2: test_1();
147      break;
148
149      }
150
151      return 0;
152  }
```

2.2. Plik tekstowy z rozwiązaniem



2.3. Rozwiązania wyświetlone na konsoli



2.4. Pseudokod

*tempArray
zero←0
jeden←0
patternzero←0
patternjeden←0
max←0
znalezione←0
i
j

fstream z;
otwórz plik z o nazwie „WYNIKI.txt”

dla i←0, i<n, i++ wykonuj
    jeżeli t[i]==0
        inkrementuj zero
    w przeciwnym razie
        inkrementuj jeden (koniec warunku)

jeżeli zero<jeden to
    max←zero*2 (koniec warunku)
jeżeli jeden<zero to
    max←jeden*2 (koniec warunku)
jeżeli jeden==zero to
    max←jeden*2 (koniec warunku)

```
tempArray←nowy[max]

kiedy max>0
    dla i=0, i<=n, i++ wykonuj
        patternzero←0
        patternjeden←0

dla j=0+i, j<max, j++ wykonuj
    jeżeli t[j]==0
    to
        inkrementuj patternzero
        tempArray[j-i] ←0

    w przeciwnym razie
        inkrementuj patternjeden
        tempArray[j-i] ←1 (koniec warunku)

jeżeli patternjeden==patternzero
to
    wypisz do konsoli „Najdluzszy podciag to: ” koniec linii, przejdź do następnej

    dla j←0, j<max, j++ wykonuj
        wypisz do konsoli tempArray[j]

    wypisz do konsoli „\n” (przejdź do następnej linii)

    wypisz do pliku „Najdluzszy podciag to: ” koniec linii, przejdź do nastepnej

    dla j←0, j<max; j++ wykonuj
        wypisz do pliku tempArray[j]
        inkrementuj znalezione

    wypisz do pliku „\n” (przejdź do następnej linii) (koniec warunku)

patternzero←0
patternjeden←0

jeżeli znalezione>0 to
    usun[]tempArray
    przerwij wykonywanie pętli (koniec warunku)

dekrementuj max

jeżeli znalezione==0
    usun[]tempArray
    zamknij plik z (koniec warunku)
```

2.5. Schemat blokowy

Ze względu na długość schematu blokowego, został on dołączony przeze mnie jako osobny plik o nazwie „schemat blokowy.jpg”.

3. Opis doświadczenia i dokumentacja

3.1. Zawartość programu

W programie zostały zawarte elementy takie jak:

- Menu programu, pozwalające na wybór jednej z dwóch dostępnych opcji: 1-Pseudolosowe (pozwalająca na określenie długości ciągu wejściowego), oraz 2-Test1 (mająca na celu sprawdzić poprawność działania programu).
- Struktura wypełniająca tablicę w sposób losowy zerami lub jedynkami.
- Pętla while i zawarty w niej szereg operacji, mający na celu przeszukanie ciągu wejściowego (w postaci tablicy).
- Pętla for obliczająca ilość zer i jedynek w tablicy.
- Operacja tworząca plik tekstowy, w którym po zakończonym poprawnie wykonaniu programu znajdują się wyniki

3.2. Problemy i błędy

Podczas tworzenia programu pojawiały się pewne trudności, głównie związane z: niepoprawną implementacją pętli, przypisanymi im warunkami oraz poprawnym zawarciu kilku z nich w sobie. O ile wypisanie wyników do pliku tekstowego czy wypełnienie tablicy zerami i jedynkami nie sprawiało większych trudności, tak przeszukanie tablicy wejściowej i obliczenie maksymalnej długości ciągu wymagało kilkudniowego zastanowienia i dokładnego przeanalizowania zaistniałego problemu. Po dokładnym przeanalizowaniu programu i schematu jego działania, oraz wprowadzeniu wielu znaczących poprawek, program działa poprawnie.

3.3. Testowanie poprawności działania

Na potrzeby sprawdzenia poprawności działania programu, został zaimplementowana struktura, w której zawiera się: deklaracja stałego ciągu w postaci tablicy, którego wartości wypisywane są do konsoli za pomocą prostej pętli for, oraz operacja szukająca wyszukująca w tym ciągu najdłuższego podciągu.

4. Wnioski

Zadanie z pewnością nie należało do najprostszych, pochłonęło ono wiele czasu i wysiłku, aby w efekcie otrzymać poprawnie działający program. Rozwiązanie tego zagadnienia zajęło wiele dni nieustannego myślenia, analizy i testów, ale także przyczyniło się do poszerzenia swojej wiedzy z zakresu programowania. Konieczne było zasięgnięcie po dokładne opisy i wyjaśnienia dotyczące działania niektórych struktur języka C++. Aby mieć pewność co do poprawności działania całego programu, należało dokładnie przeanalizować zapisany kod, a także przeprowadzić odpowiednią ilość testów.