LUCRAREA DE LABORATOR NR. 10

Tema: Prelucrarea tablourilor prin intermediul pointerilor

Obiective:

- să definească noțiunea de pointer;
- să aplice alocarea dinamică a memoriei;
- să elaboreze expresii de diferite complexități cu pointeri;
- să descrie relaţii dintre pointeri şi tablouri;

10.1. Sarcină pentru soluţionare

De realizat sarcina lucrării de laborator nr.9 cu astfel de condiţii adăugătoare:

- mărimea tabloului se determină în momentul execuției programului ca număr aliator din domeniul 50-200;
- în textul programului se interzice utilizarea operației de indexare.

10.2. Exemplu de soluţionare a sarcinii

Elaborarea algoritmului soluţionării

Algoritmul de soluţionare este acelaşi, ca şi în lucrarea nr.9. Astfel, schema algoritmului aici nu este prezentată. Diferenţă constă în aceea, că la începutul execuției, după inițializarea generatorului de valori aliatorii, este necesar de obținut un număr aliator din intervalul 50-200 (se atribuie variabilei *ssize*) şi se alocă memorie pentru tabloul de numere întregi *Ta* cu mărimea *ssize*. La ieşirea din program, memoria alocată se eliberează.

Deosebiri esenţiale apar în realizarea algoritmului. Cel mai simplu ar fi fost îndeplinirea sarcinii cu un simplu schimb: în toate locurile de apariție a operației de indexare cu operația de adresă, având în vedere identitatea: Ta[i] ≡ *(Ta+i)

Aceasta ar fi corespuns sarcinii, dar nu cu sufletul limbajului C/C++. Trecerea de la indexare la adresare duce la aplicarea altei metode de realizare a ciclurilor. Dacă variabila *Ta* este pointer la începutul tabloului, atunci în ciclu indicii *i* şi *j* se substituie cu un oarecare pointer *Pr* ce variază de la *Ta* până la *Ta+size*. Astfel, când este nevoie să se memorizeze adresa elementului maxim şi a celui minim din consecutivitate, se memorizează adresa pointerului care reprezintă poziția elementului curent.

Determinarea variabilelor programului

Variabilele programului se descriu în corespundere cu lucrarea nr.9. Memoria pentru tabloul de numere întregi se alocă la momentul compilării, astfel este de ajuns de declarat în program numai variabila pointer la început de tablou:

```
int *Ta;
```

Mărimea tabloului se determină la îndeplinirea programului, astfel pentru păstrare este nevoie de o variabilă separată:

```
int ssize;
```

În locul variabilelor, care în lucrarea nr.9 sunt indicii elementelor tabloului, se utilizează un pointer ce se referă la elementul curent în tablou:

```
int *Pr;
```

şi pointerii ce vor păstra adresele elementelor minim şi maxim din consecutivitate.

```
int *maxi, *mini;
```

Variabilele de păstrare a valorilor minime, maxime, elementului intermediar şi numărul de elemente în consecutivitate vor rămâne aceleași:

```
int mmin, mmax;
int inter;
int nn;
```

Elaborarea textului programului

Funcţia principală începe cu declararea variabilelor programului. Partea codată a programului începe cu inițializarea generatorului de numere aliatorii şi obţinerea numărului aliator pentru mărimea tabloului:

```
srand(time(NULL));
ssize=rand()%151+50;
```

Funcţia *rand* returnează un număr aliator din domeniul 0-150, la acest domeniu se adună 50 pentru a fi transformat în 50-200.

Codul de program continuă cu instrucțiunea de alocare a memoriei dinamice:

```
Ta=new int[ssize];
```

Operatorul *new* solicită un parametru – numărul de poziții în memorie. Operatorul *new* returnează un pointer de tipul *int* care se înscrie în variabila *Ta*.

În continuare, se organizează ciclul de traversare a tabloului. Spre deosebire de lucrarea nr.9, valorile aliatorii se obțin în ciclu şi tot aici are loc afișarea tabloului la ecran. Antetul acestui ciclu se deosebește de cel din lucrarea nr.9:

```
for(nn=0, Pr=Ta; Pr<Ta+ssize; Pr++){
```

În condiţiile inițiale ale ciclului, variabilei *Pr* i se atribuie adresa de început al tabloului, în caz contrar *Pr* nu este inițializat și poate provoca diferite erori la execuția programului. La sfârșitul fiecărei iterații, *Pr* creşte cu 1 indicând către următorul element din tablou. Ultima iterație se execută la valoarea *Pr = Ta+ssize-1*. *Pr* indică adresa, iar **Pr* – valoarea elementului.

Codul de program continuă cu corpul ciclului, în general, este similar cu cel prezentat în lucrarea nr.9, cu excepția că adresarea către elementul curent al tabloului se realizează prin intermediul pointerului **Pr*. Pozițiile elementelor minime şi maxime sunt memorizate în pointerii **mini* şi **maxi*.

Penultimul operator – adresarea către operatorul *delete* eliberează zona de memorie alocată de către operaturl new:

```
delete Ta;
```

Textul integral al programului este adus mai jos.

lab12.cpp

```cpp
#include <iostream>
#include <time.h>
#include <stdlib.h>
using namespace std;

int main(){
    int ssize;          //marimea tabloului
    int *Ta;            //pointer catre primul element din tablou
    int *Pr;            //pointer catre elementul curent din tablou
    int *maxi, *mini;       //pozitiile valorilor min si max in tablou
    int mmin, mmax;     //valoarea minima si maxima
    int inter;          //variabila intermediara
    int nn;             //numarul de elemente in consecutivitate

//initierea generatorului de valori aleatorii
    srand(time(NULL));

//stabilirea numarului de elemente in tablou
    ssize=rand()%151+50;

//alocarea memoriei dinamice
    Ta=new int[ssize];

//popularea tabloului cu valori aliatorii si afisarea lui
    cout<<"\n Tabloul sursa"<<endl;
    for(Pr=Ta; Pr<Ta+ssize; Pr++){
    *Pr=rand()%101-50;
    cout<<"\t"<<*Pr;
    }
    cout<<"\n\n\n";
```

```
31
32  //prelucrarea tabloului
33      for(nn=0, Pr=Ta; Pr<Ta+ssize; Pr++){
34  //prelucrarea elementului negativ
35      if(*Pr<0)
36          if(!nn){
37  //inceputul consecutivitatii
38          mmax=*Pr; mmin=*Pr; maxi=Pr; mini=Pr; nn=1;
39          }
40          else{
41  //se verifica elementele din consecutivitate
42              if(*Pr>mmin) {mmin=*Pr; mini=Pr;}
43              if(*Pr<mmax) {mmax=*Pr; maxi=Pr;}
44                  nn++;
45              }
46  //prelucrarea elementelor pozitive
47      else
48          if(nn>0){
49  /*se inlocuieste elementul minim cu elementul maxim din
50  consecutivitatea negativa*/
51              inter=*mini; *mini=*maxi; *maxi=inter;
52  //indicam ca consecutivitatea a fost prelucrata
53              nn=0;
54          }
55      } //sfarsitul algoritmului de prelucrare a tabloului
56
57      //se prelucreaza ultima consecutivitate negative
58       if(nn>0){
59              inter=*mini; *mini=*maxi; *maxi=inter;
60          }
61      cout<<"\n Tabloul prelucrat\n";
62      for(Pr=Ta; Pr<Ta+ssize; Pr++)
63          cout<<"\t"<<*Pr;
64
65  //lichidarea memoriei alocate dinamic
66      delete Ta;
67      return 0;
68  }
```

Figura 10.1. Textul programului

Ajustarea programului

Ajustarea programului se realizează conform planului prezentat în lucrarea nr.9. Dar, trebuie de menționat, că urmărirea valorilor în regimul pas cu pas este ceva mai anevoios. Dacă în lucrarea nr.9 era posibil de observat valori inteligibile ale indicilor de poziție, aici, în locul lor, noi vom vedea valorile pointerilor, mai complicate pentru a fi înțelese. Recomandăm de bazat mai mult pe depistarea erorilor la analiza rezultatelor programului.

Rezultatele lucrului programului

Exemplu de rezultate a programului este prezentat mai jos

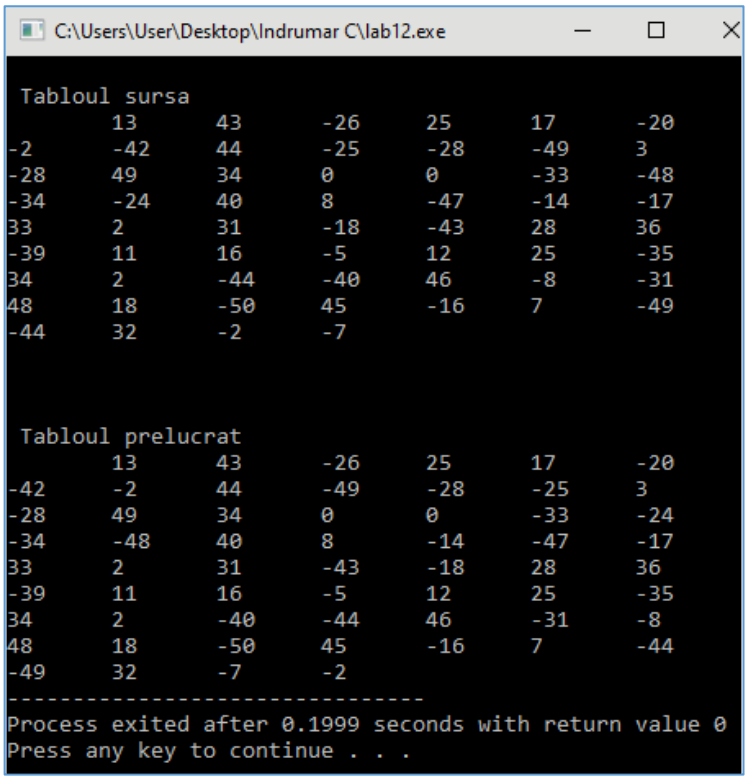

Figura 10.2. Rezultatele furnizate de program

10.3. Subiecte de evaluare

Exerciții:

1. Să se implementeze programul din exemplul prezentat.
2. Folosind operaţiile cu pointeri, să se citească de la tastatură elementele unui tablou unidimensional. Să se înlocuiască toate valorile pozitive cu zero. Să se afișeze tabloul până la prelucrare şi după prelucrare.
3. Folosind operaţiile cu pointeri, să se citească de la tastatură elementele a două tablouri unidimensionale de aceeași dimensiune. Să se calculeze elementele unui al treilea tablou ca sumă a elementelor de același indice al primelor două tablouri. Să se afișeze tabloul trei.
4. Folosind operaţiile cu pointeri, să se citească de la tastatură două șiruri de caractere. Să se determine un al treilea șir prin concatenarea primelor două şi să se afișeze la ecran.

Verificarea cunoștințelor:

1. Definiți noțiunea de pointer.
2. Prezentați exemple de adresare către pointeri.
3. Relatați despre metodele de inițializare a pointerilor.
4. Descrieți prioritatea prezentării informaţiei prin intermediul pointerilor.
5. Descrieți modul de amplasare în memorie a pointerilor.
6. Identificați operatorii ce pot fi aplicați asupra pointerilor. Prezentați exemple.
7. Identificați destinația operatorului adresă & în lucru cu pointerii.
8. Definiți noțiunea de tablou dinamic.
9. Relatași despre funcțiile utilizate la declararea/lichidarea tabloului dinamic,

10.4. Bibliografie

1. Kris Jamsa & Lars Klander, *Totul despre C şi C++*, Editura Teora, Bucureşti, 2006.
2. Herbert Schildt, *C++ manual complet*, Editura Teora, Bucureşti, 1999. Disponibil: https://drive.google.com/file/d/1BrQtITgykcWk03xxtl3Q0-nvcRRFAsy7/view?usp=sharing
3. Tutorials C++ Language. Operators. Disponibil: https://m.cplusplus.com/doc/tutorial-ro/pointers/