

Tema: Prelucrarea tablourilor unidimensionale în C/C++

Obiective:

- să definească noțiunea de tablou unidimensional;
- să cunoască diferite moduri de adresare către elementele tabloului;
- să aplice instrucțiunile repetitive pentru prelucrarea componentelor tablourilor;
- să genereze tablouri cu valori aleatorii;
- să aplice algoritmi fundamentali de prelucrare a tablourilor unidimensionale;
- să propună algoritmi de prelucrare a tablourilor unidimensionale.

7.1. Sarcina pentru soluționare

Să se declare un tablou de numere întregi și să se completeze cu valori aleatorii. Mărimea tabloului – 100 elemente, domeniul de valori – $-50 \div +50$. În toate conectivitățile de numere negative să se schimbe cu locul elementele cu valoare maximă și minimă.

Notă:

- 0 este considerat număr pozitiv;
- consecutivitatea se consideră șirul ce include nu mai puțin de două elemente;
- precizia admisă la executarea calculelor este cea utilizată în operațiile aritmetice asupra întregilor.

7.2. Exemplu de soluționare a sarcinii

Elaborarea algoritmului

Schema algoritmului este prezentată pe Figura 7.1.

La prima fază de realizare a programului este necesar de creat un tablou de valori aleatorii. Generatorul de valori aleatorii se inițializează în blocul 2. În continuare, se organizează ciclu cu contor (blocul 3), unde la fiecare iterație se generează un număr aliator ce se înscrie în următorul element al tabloului (blocul 4). După finisarea ciclului de umplere a tabloului, acesta se afișează la ecran (blocul 5).

În continuare se determină elementele minim, maxim și poziția lor în consecutivitate. În acest scop includem variabila *nn* – contorul elementelor consecutivității, care inițial se stabilește nulă (blocul 6). În cazul că valoarea acesteia nu se modifică, rezultă că nu există consecutivitate pentru prelucrare.

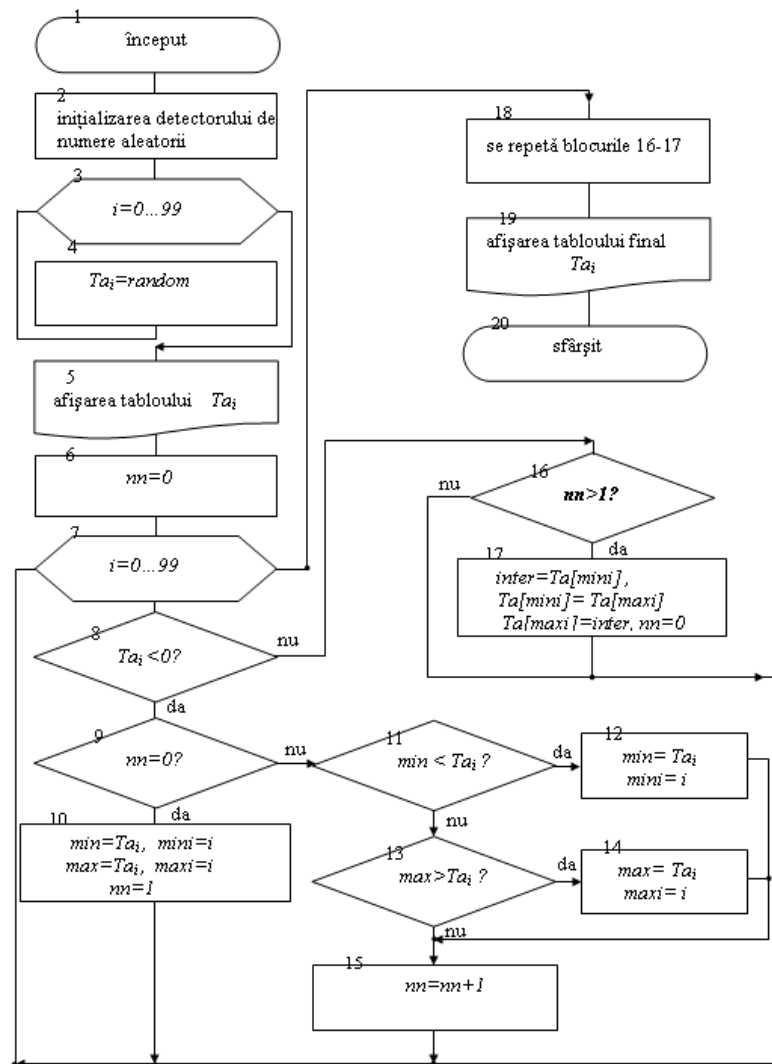


Figura 8.1. Schema bloc

La următoarea etapă se organizează ciclul cu contor (blocul 7), în care se verifică elementele tabloului. Dacă elementul este negativ, atunci el poate fi sau nu primul element al consecutivității. Aceasta se poate determina prin verificarea valorii variabilei *nn*: dacă ea este 0 – primul element din consecutivitate (blocul 9), este un indicator de inițializare a variabilelor *min* și *max* și poziția acestora, a variabilelor *maxi* și *mini*. Fixăm contorul elementelor *nn* în 1. Pentru următoarele elemente negative, vom mări contorul cu 1 și vom efectua două comparații: pentru a determina elementul *min*, *max* și pozițiile lor în consecutivitate (blocurile 11-14).

Dacă elementul curent din tablou este pozitiv, rezultă că consecutivitatea negativă s-a finisat și se efectuează prelucrarea. La prelucrare se schimbă cu locurile elementele *min* și *max* (blocul 16). La ieșirea din ciclu se inițializează contorul *nn* cu 0 (blocul 17), ca semn, că nu există o consecutivitate neprelucrată. Pentru elementele pozitive, ce urmează după consecutivitatea negativă, nu este necesar a îndeplini ceva.

După finisarea ciclului care începe în blocul 7, este necesar de verificat dacă ultima consecutivitate de numere nu a fost negativă, dacă da, se prelucerează. În schema algoritmului (blocul 18) sunt prezentate acțiunile care se îndeplinesc, ele sunt identice cu acțiunile care sunt arătate detaliat în blocurile 16-17.

După finisarea prelucrării se afișează tabloul-rezultat (blocul 19) și se finisează programul.

Determinarea variabilelor programului

Pentru realizarea algoritmului programului sunt necesare următoarele variabile.

Tabloul de numere întregi se amplasează în memoria statică:

```
int Ta[100];
```

Indicii elementelor tabloului:

```
int i, j;
```

Variabilele, în care se vor păstra parametrii consecutivității curente: valorile elementelor minim – *min* și maxim – *max*, poziția corespunzătoare acestor elemente în tablou – *mini* și *maxi*, variabila intermediară – *inter*, numărul de elemente în consecutivitate – *nn*:

```
int max, min;
```

```
int maxi, mini;
```

```
int inter;
```

```
int nn;
```

Menționăm că pentru majoritatea datelor, pentru care se acordă variabilele programului, ar fi fost de ajuns și tipul *short* sau *char*, așa cum valorile lor se includ în domeniul: -128 – +128. Însă alegem tipul *int* în conformitate cu stilul de programare în limbajul C.

Elaborarea textului programului

Începutul programului include fișierele antet ce conțin descrierea funcțiilor predefinite la care ne vom adresa:

```
#include <iostream>
```

```
#include <stdlib.h>
```

```
#include <time.h>
```

```
using namespace std;
```

Unde fișierul *iostream* include fluxurile standard de intrare/ieșire definite pentru limbajul C++; fișierul *stdlib.h* – funcția de generare a numerelor aleatoare. Funcția *rand()* reprezintă un macros, care se adresează funcției *time()*, din care motiv în program se include fișierul *time.h*.

Pentru a amplasa tabloul în memoria statică, el se declară ca o variabilă globală.

```
int Ta[100];
```

În continuare, vine antetul funcției principale *main* și deschiderea corpului ei:

```
int main(void){
```

Funcția principală începe cu declararea variabilelor și continuă cu adresarea către funcția de inițializare a generatorului de numere aleatoare (blocul 2):

```
srand(time(NULL));
```

Urmează organizarea ciclului (blocul 3), la fiecare iterație va fi înscris numărul aliator în următorul element al tabloului (blocul 4). Adresarea către funcția *rand()* returnează un număr în limitele 0 - 100; scăzând din el 50, îl aducem la domeniul -50 - +50.

```
for(i=0; i<100; Ta[i++]=rand()%101-50);
```

Tabloul completat cu numere aleatorii se afișează la ecran (blocul 5).

```
cout<<"\n Tabloul sursa"<<endl;
```

```
for(i=0; i<100; cout<<Ta[i++]<<" ");
```

```
cout<<"\n\n";
```

Programul continuă cu antetul ciclului de prelucrare a tabloului (blocul 7), în care atribuim valoarea inițială contorului *nn* (blocul 6):

```
for (nn=i=0; i<100; i++) {
```

Corpul ciclului este alcătuit dintr-o singură instrucțiune condițională. Acest operator verifică (blocul 8) semnul elementului i din tablou:

```
if (Ta[i]<0)
```

Dacă această condiție – TRUE, se verifică contorul elementelor consecutivității dacă este sau nu egal cu 0 (blocul 9):

```
if (!nn)
```

La executarea acestei condiții, se îndeplinesc un șir de acțiuni (blocul 10), pe care le luăm în acolade:

```
{max=Ta[i]; min=Ta[i]; maxi=i; mini=i; nn=1;}
```

Pentru cazul condiției – FALSE, se îndeplinește operatorul compus (blocurile 11-14):

```
else{  
if(Ta[i]<min) {min=Ta[i]; mini=i;}  
if(Ta[i]>max) {max=Ta[i]; maxi=i;}  
nn++;}
```

Dacă elementul este pozitiv, atunci se execută partea *else* a primei instrucțiuni condiționale, în care se analizează nn – dacă nu este vreo consecutivitate negativă neprelucrată (blocul 16):

```
else if (nn) {
```

Dacă consecutivitatea este neprelucrată, are loc interschimbarea valorii minime și valorii maxime (blocul 17):

```
inter=Ta[mini]; Ta[mini]=Ta[maxi]; Ta[maxi]=inter;
```

La ieșire se înscrie 0 în contorul nn (blocul 17):

```
nn=0;
```

Cu aceasta se finisează ciclul.

Instrucțiunile ce urmează după ieșirea din ciclu (blocul 18), asigură prelucrarea ultimei consecutivități și este copia instrucțiunilor, care se realizează în blocurile 16-17. Diferența constă în aceea, că s-a exclus atribuirea $nn = 0$, așa cum valoarea nn nu ne mai este necesară.

Instrucțiunile de afișare a tabloului final – copia afișării tabloului inițial.

Textul integral al programului este prezentat în Figura 7.2.

```

[*] Lab9.cpp
1  #include <iostream>
2  #include <stdlib.h>
3  #include <time.h>
4  using namespace std;
5  int Ta[100];    //declararea tabloului - variabila globala
6
7  int main(){
8      int i;        //variabila contor
9      int max, min;  //valoarea maxima si minima
10     int mini, maxi; //pozitiile valorilor minime si maxime
11     int inter;      //variabila intermediara
12     int nn;         //numarul de elemente in consecutivitate
13     srand(time(NULL)); //initializarea generatorului de numere aleatorii
14     //popularea tabloului cu valori aleatorii
15     for(i=0; i<100; Ta[i++]=rand()%101-50);
16     //afisarea tabloului
17     cout<<"\n Tabloul sursa"<<endl;
18     for(i=0; i<100; cout<<Ta[i++]<<" ", );
19     cout<<"\n\n";
20     //prelucrarea tabloului
21     for(nn=i=0; i<100; i++){
22         if(Ta[i]<0)
23             //se prelucreaza elementul negativ
24             if(!nn){
25                 max=Ta[i]; min=Ta[i]; maxi=i; mini=i; nn=1;
26             }
27             else{
28                 if(Ta[i]<min) {min=Ta[i]; mini=i;}
29                 if(Ta[i]>max) {max=Ta[i]; maxi=i;}
30                 nn++;
31             }
32         else
33             if(nn>=0){
34                 inter=Ta[mini]; Ta[mini]=Ta[maxi]; Ta[maxi]=inter;
35                 nn=0;
36             }
37     }
38     //prelucrarea ultimei consecutivitati
39     if(nn>=0){
40         inter=Ta[mini]; Ta[mini]=Ta[maxi]; Ta[maxi]=inter;
41         nn=0;
42     }
43     //afisarea tabloului prelucrat
44     cout<<"\n Tabloul prelucrat"<<endl;
45     for(i=0; i<100; cout<<Ta[i++]<<" ", );
46     return 0;
47 }

```

Figura 7.2. Textul programului

Ajustarea programului

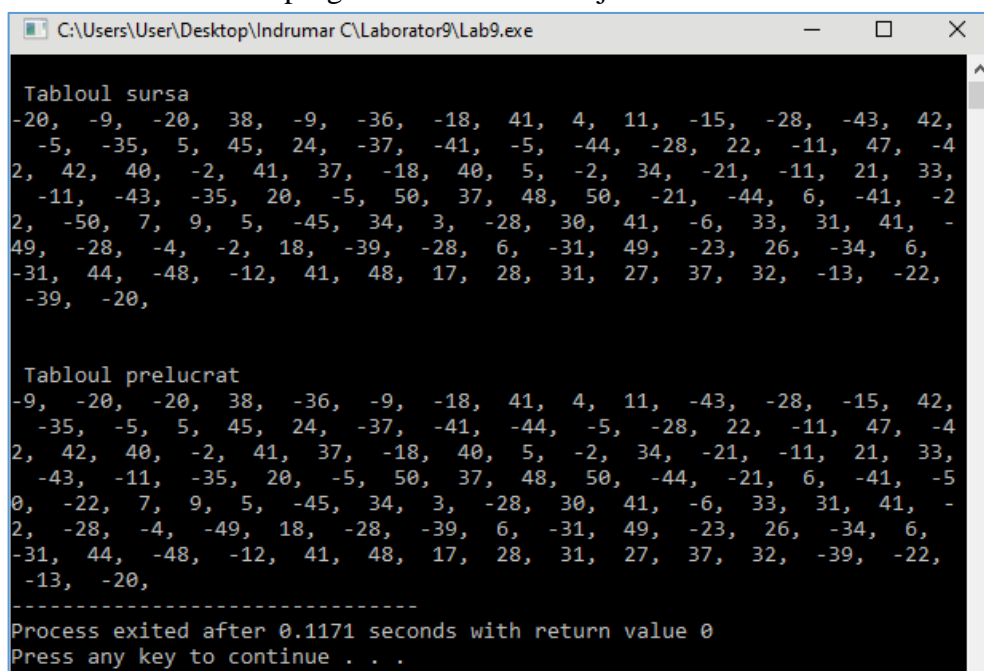
Ajustarea programului include verificarea rezultatelor, care sunt furnizate de program. Mărimea tabloului și valorile elementelor, ce nu prezintă nici o greutate de verificare care se execută manual sau cu ajutorul calculatorului.

În caz de depistare a erorilor în calcul, urmează să se verifice lucrul algoritmului, unde se va urmări în primul rând valorile variabilelor *max*, *maxi*, *min*, *mini*.

Separat se verifică acele cazuri, când tabloul începe cu o consecutivitate negativă sau se finalizează cu o consecutivitate negativă.

Rezultatele derulării programului

Rezultatele furnizate de program sunt aduse mai jos.



```
C:\Users\User\Desktop\Indrumar C\Laborator9\Lab9.exe

Tabloul sursa
-20, -9, -20, 38, -9, -36, -18, 41, 4, 11, -15, -28, -43, 42,
-5, -35, 5, 45, 24, -37, -41, -5, -44, -28, 22, -11, 47, -4
2, 42, 40, -2, 41, 37, -18, 40, 5, -2, 34, -21, -11, 21, 33,
-11, -43, -35, 20, -5, 50, 37, 48, 50, -21, -44, 6, -41, -2
2, -50, 7, 9, 5, -45, 34, 3, -28, 30, 41, -6, 33, 31, 41, -
49, -28, -4, -2, 18, -39, -28, 6, -31, 49, -23, 26, -34, 6,
-31, 44, -48, -12, 41, 48, 17, 28, 31, 27, 37, 32, -13, -22,
-39, -20,

Tabloul prelucrat
-9, -20, -20, 38, -36, -9, -18, 41, 4, 11, -43, -28, -15, 42,
-35, -5, 5, 45, 24, -37, -41, -44, -5, -28, 22, -11, 47, -4
2, 42, 40, -2, 41, 37, -18, 40, 5, -2, 34, -21, -11, 21, 33,
-43, -11, -35, 20, -5, 50, 37, 48, 50, -44, -21, 6, -41, -5
0, -22, 7, 9, 5, -45, 34, 3, -28, 30, 41, -6, 33, 31, 41, -
2, -28, -4, -49, 18, -28, -39, 6, -31, 49, -23, 26, -34, 6,
-31, 44, -48, -12, 41, 48, 17, 28, 31, 27, 37, 32, -39, -22,
-13, -20,

-----
Process exited after 0.1171 seconds with return value 0
Press any key to continue . . .
```

Figura 7.3. Rezultatele furnizate de program

7.3. Subiecte de evaluare

Exerciții:

1. Să se implementeze programul din exemplul prezentat.
2. Elaborați un program care ordonează un șir de 100 de valori (întregi sau reale), generate aliator.
3. Scrieți un program care determină dacă elementele unui vector de 20 de valori reale sunt ordonate (crescător sau descrescător).
4. Scrieți un program, care determină cel mai mic număr par dintr-un vector de 20 de elemente întregi. În cazul în care nu există nici un element par, se va afișa un mesaj corespunzător.

Verificarea cunoștințelor:

1. Definiți noțiunea de tablou unidimensional.
2. Prezentați exemple de adresare către elementele tabloului unidimensional.
3. Relatați despre operatorii ce pot fi aplicați asupra elementelor unui tablou.
4. Relatați despre tipul variabilei indice a elementelor în tablou.
5. Relatați despre tipul elementelor unui tablou.

7.4. Bibliografie

1. Kris Jamsa & Lars Klander, *Totul despre C și C++*, Editura Teora, București, 2006.
2. Herbert Schildt, *C++ manual complet*, Editura Teora, București, 1997. Disponibil:

<https://drive.google.com/file/d/1BrQtITgykcWk03xxtl3Q0-nvcRRFAsy7/view?usp=sharing>
3. Tutorials C++ Language. Operators. Disponibil: <https://cplusplus.com/doc/tutorial-ro/arrays/>