SCRIEREA MODULARĂ A PROGRAMELOR

1. SCOPUL LUCRĂRII

În această lucrare se va studia modul in care se definesc funcţiile, în limbajul C.

2. BREVIAR TEORETIC

2.1. Definirea funcţiilor

Orice program C conţine cel puţin o funcţie şi aceasta este *main* (funcţie a cărei prezenţă este obligatorie). O funcţie poate fi apelată fie din programul principal, fie dintr-o altă funcţie.

În limbajul C, o funcţie nu poate fi definită în interiorul altei funcţii.

Definiţia unei funcţii trebuie să fie făcută o singură dată în program şi are sintaxa:

tipul_rezultatului nume_funcţie(listă_de_parametrii)
{
declaraţii interne
instrucţiuni
}

Lista de parametrii cuprinde parametrii funcţiei, ce pot fi:

- de intrare. Valorile lor sunt transmise de către programul apelant . Nu sunt modificaţi în funcţie.
- de ieşire. Sunt creaţi de funcţie (valorile lor sunt calculate în interiorul funcţiei).
- de intrare / ieşire. Sunt şi folosiţi în funcţie (recepţionaţi de la programul ce a apelat funcţia) dar şi modificaţi în corpul funcţiei.

Alegerea antetului funcţiei depinde de programator.

În momentul apelării, toţi parametrii de intrare trebuie să aibă valori (sa fie iniţializaţi).

În cazul funcţiilor ce au parametrii de ieşire sau parametrii de intrare / ieşire, pentru aceştia trebuie transmisă funcţiei adresele lor. Pentru a evita lucrul direct cu pointeri şi adresarea indirectă, care este ceva mai greoi, lăsând aceasta să fie făcută de compilator, vom folosi pentru parametrii de ieşire sau de intrare / ieşire ai unei funcţii (deci, pentru parametrii care trebuiau daţi ca pointeri) notaţia de variabile referinţă (se foloseşte pentru această notaţie simbolul & plasat între tipul parametrului şi numele său).

2.2. Apelul funcţiilor

Un program C, ce constă doar dintr-o definiţie de funcţie (alta decât *main*), nu poate fi rulat. Pentru testarea unei funcţii scrise, trebuie să fie prezentă cel puţin şi funcţia *main*. În general, orice program de test trebuie :

- să iniţializeze parametrii de intrare
- să apeleze funcţia
- să afişeze rezultatul

Un apel de funcţie constă din numele funcţiei, urmat de parametrii actuali ai funcţiei. Dacă funcţia returnează o valoare, aceasta trebuie atribuită unei variabile.

Urmărim:

- ca toţi parametrii de intrare să fie iniţializaţi
- pentru fiecare parametru în parte, trebuie să se respecte tipul lui
- să se respecte numărul şi ordinea parametrilor

Exemplu:

Se dă următorul prototip de funcţie:

```
int f1(int a,double b);
```

Fie următoarea secvenţă de program:

```
void main(void)
{
 int a;
 double b;
 int rezultat;
 a=3;
 //se apeleaza functia f1():
 …..
}
```

Vom da mai multe exemple de mod de apelare a funcţiei f1(), din programul principal. Unele dintre acestea vor fi greşite şi vom explica unde sunt greşelile, altele vor fi corecte.

Apelul 1:

```
f1(a,b);
```

Acest apel nu este corect pentru că argumentul b nu este iniţializat.

Apelul 2:

```
f1(a,7.5);
```

Este corect , dar valoarea returnată de funcţie nu este folosită.

Apelul 3:

rezultat=f1(2);

Nu este corect, pentru ca funcţia f1() are doi parametrii.

Apelul 4:

rezultat=f1(a,3.14);

Este corect, valoarea returnată este încărcată în variabila rezultat.

2.3. Instrucţiunea return

Ieşirea dintr-o funcţie (retransmiterea controlului către programul care a apelat-o) se face fie la întâlnirea instrucţiunii **return** fie (dacă aceasta lipseşte) la întâlnirea acoladei ce marchează sfârşitul funcţiei.

Sintaxa instrucţiunii **return**:

- prima formă:

return;

Această formă se foloseşte atunci când nu se transmite nici un rezultat către programul ce a apelat funcţia respectivă.

- a doua formă:

return rezultat;

Această formă se foloseşte atunci când se transmite un rezultat către programul ce a apelat funcţia respectivă. Se poate folosi şi varianta cu paranteze:

return(rezultat);

3. DESFĂŞURAREA LUCRĂRII

Se vor edita şi apoi executa programele descrise în continuare.

Programul nr. 1

Să re scrie o funcţie împreună cu un program de test, care decide dacă trei numere sunt în progresie aritmetică:

Sursa programului:

```
#include<conio.h>
#include<stdio.h>
int suntProgArit(int a, int b, int c);
 void main(void){
  clrscr();
  printf("%d",suntProgArit(1,2,4));
  getch();
```

```
}

int suntProgArit(int a, int b, int c)
{
  if(b-a==c-b)return 1;
  else return 0;
}
```

Programul nr. 2

Să se scrie o funcţie împreună cu un program de test, care stabileşte dacă două numere a şi b sunt prime între ele.

Sursa programului:

```
#include<conio.h>
#include<stdio.h>
int suntPrimeIntreEle(int a,int b);
  void main()
{
 clrscr();
 printf("%d",suntPrimeIntreEle(3,7));
 getch();
}

int suntPrimeIntreEle(int a,int b)
{
 int sunt;
 int i;
 int min=a;
 if(b<a)min=b;
 sunt=1;
 for(i=2;i<=min;i++)
   if((a%i==0)&&(b%i==0)){
     sunt=0;
     break;
   }
 return sunt;
}
```

Programul nr. 3

Să se scrie o funcţie care returnează primul număr prim mai mare strict decât un număr întreg N dat. Se va scrie şi programul principal ce testează această funcţie.

Sursa programului:

```
#include <conio.h>
#include <stdio.h>
#include <math.h> //pentru functia sqrt()
int primulNrPrim(int N);
int estePrim(int nr);
void main(void)
{
  int N;
  clrscr();
  printf("N=");
  scanf("%d",&N);
  printf("Primul nr. prim > %d este %d .", N, primulNrPrim(N));
  getch();
}
//definitia functiei primulNrPrim():
int primulNrPrim(int N)
{
  int nrCrt;
  nrCrt=N+1;
  for(;;){
    if(estePrim(nrCrt))return nrCrt;
    nrCrt++;
  }
}
//definitia functiei estePrim():
int estePrim(int nr)
{
  int i;
  for(i=2;i<=sqrt(nr);i++)
    if( nr % i == 0 ) return 0; //nu este numar prim
  return 1; //este numar prim
}
```

Programul nr. 4

Să se scrie o funcţie, împreună cu un program de test, care returnează minimul şi maximul dintre trei numere.

Sursa programului:

```
#include<conio.h>
#include<stdio.h>
void max_min(int a,int b,int c,int&max,int& min);
 void main(void){
 int min,max;
 clrscr();
 max_min(30,6,0,max,min);
 printf("%d %d",min,max);
 getch();
}

void max_min(int a,int b,int c,int&max,int& min)
{
 max=a;
 if(b>max)max=b;
 if(c>max)max=c;
 min=a;
 if(b<min)min=b;
 if(c<min)min=c;
}
```

Programul nr. 5

Să se scrie o funcţie care are ca parametru de intrare un string şi ca parametru de ieşire, stringul inversat.

Exemplu:

Dacă stringul de intrare este: "12abc" atunci stringul de ieşire va fi: "cba21" .

Sursa programului:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void  inversareStr(char * in, char* out);
void main(void)
{
 char s1[80]="12abc";
 char s1Inv[80];
```

```
 inversareStr(s1,s1Inv);
 clrscr();
 printf("Sirul inversat este:\n%s",s1Inv);
 getch();
}

void  inversareStr(char*  in, char* out)
{
 int i;
 int L; //lungime sir in
 L=strlen(in);
 for(i=L-1;i>=0;i--)
   out[L-1-i]=in[i];
 //terminatorul de string:
 out[L]='\0';
}
```

Programul nr. 6

Să se scrie o funcţie, împreună cu un program de test, ce are ca parametru de intrare un vector A şi dimensiunea lui şi scoate ca rezultat vectorul ce conţine toate numerele prime din A, precum şi dimensiunea lui.

Sursa programului:

```
#include<conio.h>
#include<stdio.h>
#include<math.h>
int estePrim(int nr);
void vectorPrime(int A[], int dimA, int B[], int& dimB);
 void main(void){
 int A[5]={7,9,8,19,31};
int B[5];
int dimB;
 int i;
 clrscr();
 vectorPrime(A,5,B,dimB);
 for(i=0;i<dimB;i++)printf("%d  ",B[i]);
 printf("\nDimensiunea vectorului de numere prime este %d",dimB);
 getch();
}
```

```
int estePrim(int nr)
{
 int i;
 int este=1;
 for(i=2;i<=sqrt(nr);i++)
   if(nr%i==0){
      este=0;
      break;
   }
 return este;
}
void vectorPrime(int A[], int dimA, int B[], int& dimB)
{
 int i;
 int j;
 j=0;
 for(i=0;i<dimA;i++)
   if(estePrim(A[i])){
     B[j]=A[i];
     j++;}
 dimB=j;
}
```

Programul nr. 7

Să se scrie o funcţie, împreună cu un program de test, care afişează dacă elementele unui vector sunt sau nu diferite.

Sursa programului:

```
#include<conio.h>
#include<stdio.h>
#include<math.h>
int suntDiferite(int A[],int dimA);
 void main(void){
 int A[6]={138,9,19,32,31,18};
 clrscr();
 printf("\n%d",suntDiferite(A,6));
 getch();
}

int suntDiferite(int A[],int dimA)
```

```
{
 int i,j;
 for(i=0;i<dimA;i++)
   for(j=i+1;j<dimA;j++)
   if(A[j]==A[i])return 0;
 return 1;
}
```

Programul nr. 8

Să se scrie o funcţie ce face rotirea spre stânga a unui vector, precum şi un program de test.

Sursa programului:

```
#include<conio.h>
#include<stdio.h>
void rotireSt(int A[], int dimA);
 void main(void){
 int i,A[6]={138,9,19,32,31,30};
 clrscr();
 rotireSt(A,6);
 for(i=0;i<6;i++)
 printf("%d ",A[i]);
 getch();
}

void rotireSt(int A[], int dimA)
{
 int primul;
 int i;
 primul=A[0];
 for(i=0;i<dimA-1;i++)
   A[i]=A[i+1];
 //ultimul:
 A[dimA-1]=primul;
}
```

Programul nr. 9

Să se scrie o funcţie, împreună cu programul de test care afişează dacă un vector este sau nu sortat crescător.

Sursa programului:

```
#include<conio.h>
```

```
#include<stdio.h>
int esteSortatCresc(int A[], int dimA);
 void main(void){
 int i,A[6]={1,9,19,30,31,36};
 clrscr();
 printf("%d",esteSortatCresc(A,6));
 getch();
}

int esteSortatCresc(int A[], int dimA)
{
 int i;
 int este=1;
 for(i=0;i<dimA-1;i++)
   if(A[i+1]<A[i]){
     este=0;
     break;
   }
 return este;
}
```

Programul nr. 10

Să se scrie o funcţie, împreună cu programul de test, care returnează minimul şi maximul dintr-un vector, precum şi poziţiile lor.

Sursa programului:

```
#include<conio.h>
#include<stdio.h>
void max_minV(int A[], int dimA, int& max, int& pozMax, int& min,
int & pozMin);
 void main(void){
 int min,max,pozMin,pozMax,A[6]={11,9,19,30,310,36};
 clrscr();
 max_minV(A,6,max,pozMax,min,pozMin);
 printf("Maximul este %d si are indexul %d\n",max,pozMax);
 printf("Minimul este %d si are indexul %d",min,pozMin);
 getch();
}

void max_minV(int A[], int dimA, int& max, int& pozMax, int& min,
int & pozMin)
```

```
{
 int i;
 max=A[0];
 pozMax=0;
 for(i=1;i<dimA;i++)
  if(A[i]>max){max=A[i];
               pozMax=i;}
 min=A[0];
 pozMin=0;
 for(i=1;i<dimA;i++)
  if(A[i]<min){min=A[i];
               pozMin=i;}
}
```

4. PROBLEME PROPUSE

1. Să se scrie o funcţie împreună cu un program de test care să afişeze toate perechile de numere prime între ele din intervalul [N1, N2].

2. Să se scrie o funcţie împreună cu un program de test care să afişeze toate numerele prime dintr-un interval [N1, N2].

3. Să se scrie o funcţie ce are ca rezultat intersecţia a două mulţimi, şi are antetul:

void intersectie(int A[], int nA, int B[], int nB, int C[], int &nC)

4. Să se scrie o funcţie ce face rotirea spre dreapta a unui vector.

5. Să se scrie o funcţie şi programul de test, care are ca intrare un număr de tip long int, şi scoate ca rezultat un vector ce conţine toate cifrele numărului.

6. Să se scrie o funcţie care returnează cel mai mare număr prim de patru cifre.

7. Să se scrie o funcţie care scoate drept rezultat soluţiile ecuaţiei de gradul al doilea, şi are antetul:

int ec2(double a, double b, double c, double &x1, double &x2)