FUNCŢII

1. Conţinutul lucrării

În lucrare se prezintă structura unei funcţii, apelul unei funcţii prin valoare şi prin referinţă, prototipul unei funcţii.

2. Consideraţii teoretice

Un program conţine una sau mai multe funcţii, dintre care una este funcţia principală având numele **main**. Celelalte au un nume dat de programator.

2.1. Structura unei funcţii

O funcţie are următoarea structură:

```
tip nume (lista_parametrilor_formali)
{
  declaraţii
  instrucţiuni
}
```

Primul rând din definiţia funcţiei se numeşte **antet**, iar restul se numeşte **corpul funcţiei**.

În limbajul C/C++ există două categorii de funcţii:

- funcţii care returnează în punctul de apel o valoare prin instrucţiunea **return expresie;** valoarea având tipul specificat în antet prin "tip";
- funcţii care nu returnează nici o valoare în punctul de apel, tip fiind înlocuit prin cuvântul cheie "**void**".

Lista parametrilor formali poate conţine:

- zero parametri, caz în care antetul funcţiei se reduce la:
  tip nume () sau **tip nume (void)**
- unul sau mai mulţi parametri formali, separaţi între ei prin virgulă; un parametru formal se indică prin: **tip nume**.

Exemplu:

int rezolv_sistem (int n, double a[10] [10], double b[10], double x[10])

Prototipul unei funcţii se obţine scriind punct şi virgulă după o construcţie identică cu antetul funcţiei respective sau obţinută prin eliminarea numelui parametrilor formali.

Exemplu:

```
int factorial (int n);
int factorial (int);
```

Funcţiile standard de bibliotecă au prototipurile în diferite fişiere cu extensia **.h,** cum ar fi **stdio.h, conio.h, math.h** etc. Funcţiile standard de bibliotecă se găsesc în format obiect (extensia **.o**), şi se adaugă în programe în faza de editare de legături. Prototipurile funcţiilor standard se includ în program înainte de apelul lor prin construcţia **#include.**

2.2. Apelul unei funcţii

O funcţie care nu returnează nici o valoare se apelează astfel:

nume (lista_parametrilor_efectivi);

Corespondenţa între parametrii formali şi cei efectivi este poziţională.

O funcţie care returnează o valoare poate fi apelată:
- fie printr-o instrucţiune de apel ca mai sus, caz în care valoarea returnată se pierde;
- fie ca un operand al unei expresii, valoarea returnată folosindu-se la evaluarea expresiei respective.

Tipul parametrilor formali şi cei actuali se recomandă să fie acelaşi. În caz contrar, în limbajul C tipul parametrului efectiv este convertit automat la tipul parametrului formal. În limbajul C++ se utilizează o verificare mai complexă pentru apelul funcţiilor. De aceea se recomandă utilizarea operatorului de conversie explicită **(tip)**, adică expresiile cast.

Exemplu:

f((double)n)

Revenirea dintr-o funcţie se face fie după execuţia ultimei instrucţiuni din corpul funcţiei, fie la întâlnirea instrucţiunii **return**.

Instrucţiunea **return** are formatele:

return;

sau

return expresie;

Observaţii:
- Dacă tipul expresiei din instrucţiune este cel din antetul funcţiei, se face conversia automată spre cel al funcţiei.
- Primul format al instrucţiunii **return** se foloseşte în funcţiile care nu returnează nici o valoare.

Transmiterea parametrilor efectivi (actuali) se poate face:
- prin valoare (*call by value*);
- prin referinţă (*call by reference*).

În cazul apelului prin valoare, unui parametru formal i se transferă valoarea parametrului efectiv. În acest caz, funcţia apelată nu poate modifica valoarea parametrul efectiv din funcţia care a făcut apelul. Programul L5Ex1 ilustrează acest lucru:

```
 /*Programul L5Ex1*/

 #include <stdio.h>
/*APEL PRIN VALOARE*/
 /*Procedura de interschimbare intre a si b*/
void interschimbare(int a, int b)
 {
   int aux;
   printf("\nIn functie la intrare a=%d b=%d\n",a,b);
   aux=a;a=b;b=aux;
   printf("\nIn functie la iesire a=%d b=%d\n",a,b);
 }
int main()
 {
   int a,b;
   a=2;b=3;
   printf("\nIn main inaintea apelului functiei interschimbare a=%d b=%d\n",a,b);
   interschimbare(a,b);
   printf("\nIn main la revenirea din functia interschimbare a=%d b=%d\n",a,b);
   return 0;
}
```

Se va constata că a şi b îşi păstrează vechile valori. Pentru ca interschimbarea să se producă, este necesară folosirea pointerilor. Programul L5Ex2 ilustrează acest lucru:

```
 /*Programul L5Ex2*/

 #include <stdio.h>
/*APEL PRIN VALOARE FOLOSIND POINTERI*/
 /*Procedura de interschimbare intre a si b*/
 void interschimbare(int *a, int *b)
 {
  int aux;
  printf("\nIn functie la intrare a=%d b=%d\n",*a,*b);
  aux=*a;*a=*b;*b=aux;
  printf("\nIn functie la iesire a=%d b=%d\n",*a,*b);
 }
 int main()
 {
  int a,b;
  a=2;b=3;
  printf("\nIn main inaintea apelului functiei interschimbare a=%d b=%d\n",a,b);
  interschimbare(&a,&b);
  printf("\nIn main la revenirea din functia interschimbare a=%d b=%d\n",a,b);
  return 0;
}
```

Se va constata că valorile a şi b au fost interschimbate între ele. Acest mod de transmitere a parametrilor este tot prin valoare, adică unui pointer i s-a transmis o adresă.

În cazul apelului prin referinţă, se transmit adresele parametrilor, nu valoarea lor. Acest mod de transmitere este valabil numai în limbajul C++. În acest caz, parametrii formali sunt definiţi ca fiind de tip referinţă. Programul L5Ex3 ilustrează acest lucru:

```
/*Programul L5Ex3 */

#include <stdio.h>
/* APEL PRIN REFERINTA */
/*Procedura de interschimbare intre a si b */
void interschimbare(int& a, int& b)
{
   int aux;
   printf("\nIn functie la intrare a=%d b=%d\n",a,b);
   aux=a;a=b;b=aux;
   printf("\nIn functie la iesire a=%d b=%d\n",a,b);
}
int main()
{
   int a,b;
   a=2;b=3;
   printf("\nIn main inaintea apelului functiei interschimbare a=%d b=%d\n",a,b);
   interschimbare(a,b);
   printf("\nIn main la revenirea din functia interschimbare a=%d b=%d\n",a,b);
   return 0;
}
```

Se va constata că valorile lui a şi b au fost interschimbate între ele.

Observaţie importantă: în cazul în care un parametru efectiv este numele unui tablou, atunci acesta are ca valoare adresa primului element, deci se transmite adresa ca valoare pentru parametrul formal corespunzător. În acest caz, deşi apelul s-a făcut prin valoare, funcţia respectivă poate modifica elementele tabloului al cărui nume s-a folosit ca parametru efectiv.

Drept exemplu de folosire a funcţiilor, în continuare se prezintă un program (L5Ex4) care realizează câteva operaţii asupra a două polinoame.

```
/*Programul L5Ex4*/

/* Operatii asupra polinoamelor de forma P(x)=p[0]+p[1]*x+ p[2]*x^2 +...p[n]* x^n  */
#include <stdio.h>
#include <conio.h>
# define GRADMAX 20

void produs(int n,float a[], int m,float b[], int *p,float c[])
{
   int i,j;
  *p=n+m;
   for(i=0;i<=n+m;i++) c[i]=0;
   for(i=0;i<=n;i++)
      for(j=0;j<=m;j++)
         c[i+j]+=a[i]*b[j];
}
```

```
void impartire(int n, float a[],int m,float b[], int *grad_cat,float cat[], int *grad_rest,
                    float rest[])
{
  int i,j,k;
  if (n<m) {
                *grad_cat=0;cat[0]=0;
                *grad_rest=n;
                for(i=0; i<=n; i++)
                  rest[i]=a[i];
              }
  else {
        *grad_cat=n-m;*grad_rest=m-1;
        for(i=n-m,j=n;i>=0;i--,j--) {
             cat[i]=a[j]/b[m];
             for (k=m;k>=0;k--)
                a[i+k]=a[i+k]-cat[i]*b[k];
             a[j]=0;
           }
         for(i=0;i<=m-1;i++)
           rest[i]=a[i];
       }
}

void citire_polinom(int *n,float a[])
{
  int i;
  printf("\nIntroduceti gradul polinomului  ");
  scanf("%d",n);
  for(i=0;i<=*n;i++) {
       printf("\na[%d]=",i);
       scanf("%f",&a[i]);
    }
}

float val_polinom(float x,int n,float a[])
{
 int i;
 float v=0;
 for(i=n;i>=0;i--) v=v*x+a[i];
 return v;
}

void afis_polinom(int n,float a[],char c)
{
 int i;
 printf("\n%c[x]=%g",c,a[0]);
 for(i=1;i<=n;i++)
   printf("+%g*x^%d",a[i],i);
 printf("\n");
}
```

```
int main()
{
  int n,m,grad_r,grad_cat,grad_rest;
  float x, v,p[GRADMAX+1],q[GRADMAX+1],r[GRADMAX+1],
        cat[GRADMAX+1],rest[GRADMAX+1];
  citire_polinom(&n,p);afis_polinom(n,p,'P');
  citire_polinom(&m,q);afis_polinom(m,q,'Q');
  printf("\nIntroduceti x=");scanf("%f",&x);
  v=val_polinom(x,n,p);
  printf("Valoarea polinomului p pentru x=%f este %f",x, v);
  getch();
  produs(n,p,m,q,&grad_r,r);
  printf("\nR[x]=P[x]*Q[x]\n");
  afis_polinom(grad_r,r,'R');
  getch();
  impartire(n,p,m,q,&grad_cat,cat,&grad_rest,rest);
  printf("\nRezultatul impartirii P[x]/Q[x]=>catul C[x] si restul R[x]\n");
  afis_polinom(grad_cat,cat,'C');
  afis_polinom(grad_rest,rest,'R');
  getch();
  printf("\nATENTIE! Polinomul p este modificat\n");
  afis_polinom(n,p,'P');
  return 0;
}
```

3. Mersul lucrării

3.1. Se va analiza modul de transmitere a parametrilor efectivi în programele din lucrare date ca exemplu și în materialul de curs.

În continuare se vor scrie programe pentru rezolvarea următoarelor probleme, folosind funcţii şi diverse moduri de transmitere a parametrilor. Se va evita folosirea variabilelor globale.

3.2. Se citeşte gradul unui polinom şi coeficienţii săi, care sunt numere întregi:

$$P(x) = p_0 + p_1 x^1 + ... + p_n x^n$$

în care p_0 este nenul. Ştiind că polinomul admite numai rădăcini întregi simple, să se găsească rădăcinile polinomului.

3.3. Se citeşte n şi perechile de numere întregi (x_i, y_i), $i = \overline{1,n}$ reprezentând o relaţie binară R peste mulţimea M. Admiţând că fiecare element din mulţimea M apare în cel puţin o pereche dintre cele citite, să se determine mulţimea M. Să se verifice dacă relaţia R este o relaţie de echivalenţă (reflexivă, simetrică şi tranzitivă).

3.4. Se dau două şiruri de caractere care reprezintă numere întregi zecimale foarte mari. Să se scrie un program de efectuare a operaţiilor aritmetice asupra lor.

3.5. Să se scrie funcţiile pentru adunarea, scăderea şi înmulţirea a două matrice şi apoi să se calculeze A=B*C–2*(B+C), unde B şi C sunt două matrice pătratice de ordinul n.

3.6. Să se scrie funcţia care realizează operaţiile aritmetice asupra a două matrice rare (matricea rară este o matrice de dimensiune mare, care are majoritatea elementelor nule).

3.7. Fiind date anul, luna, ziua, să se scrie o funcţie care să returneze a câtea zi din an este ziua respectivă şi câte zile au mai rămas din anul respectiv.

3.8. Să se scrie o funcţie care primind ca parametru un număr roman sub forma unui şir de caractere, returnează numărul respectiv ca număr arab în baza 10.

3.9. Să se scrie o funcţie care primind ca parametru un număr arab în baza 10, calculează şirul de caractere ce reprezintă numărul respectiv sub formă romană.

3.10. Se citeşte un număr întreg, reprezentând o sumă de bani. Să se găsească numărul minim de bancnote româneşti necesare pentru plata sumei respective.

3.11. Să se scrie o funcție pentru a verifica dacă un șir de caractere este subșir al altui șir de caractere. În caz afirmativ funcția trebuie să returneze poziția la care începe subșirul, iar în caz negativ trebuie să returneze valoarea -1.

3.12. Să se scrie o funcție pentru a verifica dacă parametrul ei (un întreg pozitiv) este pătrat perfect. Utilizați funcția pe un șir de întregi pozitivi cu scopul extragerii tuturor pătratelor perfecte într-un alt șir.

3.13. Să se scrie o funcție care verifică dacă parametrul ei (un întreg pozitiv) este număr perfect. Numărul perfect este un număr întreg egal cu suma divizorilor săi, din care se exclude numărul însuși.