Laborator 11-12. Funcţii. Exemple de programe. Aplicaţii în inginerie

11.1. Definirea unei funcţii

Sintaxa definirii unei funcţii:

```
tip_returnat nume_functie(lista_parametri_formali)
{
      secvenţa_de_declaraţii
      secvenţa_de_instrucţiuni
}
```

`tip_returnat` reprezintă tipul de date returnate de funcţie. O funcţie poate returna orice tip de date în afară de tablouri.
`lista_parametri_formali` este o listă de parametri separaţi prin virgule, conţinând tipul şi numele fiecărui parametru.
Prima linie din definirea unei funcţii poartă numele de *antet* al funcţiei.

11.2. Prototipuri de funcţii

Limbajul C impune fie declararea, fie definirea unei funcţii înainte de folosirea ei.
Declararea unei funcţii, numită **prototip** are aceeaşi formă ca şi antetul funcţiei urmată în plus de ";".

```
tip_returnat nume_functie(lista_parametri_formali);
```

11.3 Apelul funcţiilor

Apelul este realizat printr-o instrucţiune de apel, de forma:

```
nume_functie(lista_parametri_efectivi);
```

sau sub formă de operand al unei expresii, dacă funcţia returnează valoare.
`lista_parametri_efectivi` este o expresie sau mai multe expresii, separate prin virgulă sau vidă dacă funcţia nu are parametri.

11.4 Revenirea dintr-o funcţie

Revenirea dintr-o funcţie se poate realiza în două moduri:
1. după ce ultima instrucţiune din funcţie a fost executată şi a fost întâlnită acolada de închidere;
2. după ce a fost întâlnită instrucţiunea return.
Dacă funcţia nu returnează o valoare se va folosi instrucţiunea

```
return;
```

fără a include nici o expresie după cuvântul cheie return.
Dacă funcţia returnează o valoare se va folosi instrucţiunea

```
return expresie;
```

11.5. Exemple de programe

Ex1. Să se calculeze valoarea combinărilor $C_n^k = \frac{n!}{k!(n-k)!}$ utilizând o funcţie pentru calculul factorialului.

Programul va asigura citirea repetată a datelor de intrare n şi k.

```
/*calculul combinarilor folosind functia fact() pentru calculul factorialului*/
#include<stdio.h>
#include<conio.h>
double fact(int);
```

```
void main(void)
{
int n,k;
double combin;
char var;
do{
       printf("\nIntroduceti numarul n:");
       scanf("%d", &n);
       while(n<0)
       {
              printf("Dimensiune eronata:\n");
              printf("Introduceti alt n:");
              scanf("%d", &n);
       }
       printf("Introduceti numarul k<n:");
       scanf("%d", &k);
       while(n<k)
       {
              printf("Dimensiune eronata:\n");
              printf("Introduceti alt k<n:");
              scanf("%d", &k);
       }
       combin=fact(n)/fact(k)/fact(n-k);
       printf("Combinari de %d luate cate %d = %.0lf\n", n,k, combin);
       printf("Doriti sa continuati introducerea datelor? (D/N)");
}
while((var=getch())=='D' ||var=='d');
printf("\nProgramul s-a incheiat\n");
getch();
}

double fact(int n) /*calculeaza pe n!*/
{
double f;
int i;
 if(n==0)
        return 1.0;
 else
       for(i=2,f=1.0;i<=n;i++)
              f*=i;
 return f;
}
```

Ex2. Să se scrie un program care calculează produsul elementelor unui şir de numere întregi. Se va utiliza o funcţie pentru calculul produsului.

```
/*Transferul parametrilor ca tablouri*/
/*Produsul elementelor unui sir*/
#include <stdio.h>
#include <conio.h>

double produs(int sir[],int);

void main(void)
{
int n,i;
int a[20];
printf("Valoarea lui n=");
scanf("%d",&n);
while(n<=0||n>20){
   printf("Dimensiune eronata:%d\n",n);
   printf("Introduceti alt n:");
   scanf("%d",&n);
                          }
for(i=0;i<n;i++){
       printf("Elementul a[%d]=",i);
       scanf("%d",&a[i]);
                 }
printf("Valoarea produsului: %.0lf\n",produs(a,n));
printf("Apasati orice tasta pentru continuare\n");
getch();
}
```

```
double produs(int sir[], int n) /*functia produsul elementelor unui sir*/
{
double p;
for(p=1;n>0;n--)
      p*=sir[n-1];
return p;
}
```

Ex3. Să se scrie un program care citeşte gradul şi coeficienţii a două polinoame P(x) şi Q(x), precum şi argumentul x, apoi calculează produsul R(x) al celor două polinoame. În final, să calculeze valoarea polinoamelor P, Q şi R în punctul x.

```
/* Operatii asupra polinoamelor
   Un polinom are forma P(x)=p[0]+p[1]*x+ p[2]*x^2 +...p[n]* x^n  */

#include <conio.h>
#include <stdio.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]=0.0;
        for(i=0;i<=n;i++)
          for(j=0;j<=m;j++)
              c[i+j]+=a[i]*b[j];
   }

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

float val_polinom(float x,int n,float a[])
   {
        int i;
        float v;
        v=0.0;
        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("%c(x)=%g",c,a[0]);
        for(i=1;i<=n;i++)
          printf("+%g*x^%d",a[i],i);
        printf("\n");
   }

void main(void)
   {
        int n,m,grad_r;
        float x,v,p[GRADMAX+1],q[GRADMAX+1],r[GRADMAX+1];
        clrscr;
        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("\nValoarea polinomului P pentru x=%f este %f",x, v);
        v=val_polinom(x,m,q);
        printf("\nValoarea polinomului Q pentru x=%f este %f",x, v);
        getch();
        produs(n,p,m,q,&grad_r,r);
        printf("\n\nR(x)=P(x)*Q(x)\n");
        afis_polinom(grad_r,r,'R');
        v=val_polinom(x,m,r);
        printf("\nValoarea polinomului R pentru x=%f este %f",x, v);
        getch();
    }
```

Ex4. Studiul cinematic al mecanismului bielă-manivelă

```
/* Studiul cinematic al mecanismului biela-manivela */
#include <stdio.h>
#include <math.h>
#include <conio.h>

double r, l, rpl, omega;

double deg2rad(double x)
{
return x*M_PI/180;
}

double rad2deg(double x)
{
return x*180/M_PI;
}

double s(double fir)
{
return r*(1-cos(fir)+0.5*rpl*pow(sin(fir),2));
}

double v(double fir)
{
return omega*r*(sin(fir)+0.5*rpl*sin(2*fir));
}

double a(double fir)
{
return omega*omega*r*(cos(fir)+rpl*cos(2*fir));
}

void main (void)
{
double fid, fir, fi0r, fi[13], n, T;
unsigned g, i, index;
char welcome[]="Studiul cinematic al mecanismului biela-manivela";
printf("%s\n\nDate de intrare\n\nNumarul grupei:", welcome);
scanf("%u",&g);
g%=10;      //determinarea ultimei cifre din numarul grupei
printf("Numarul de ordine din grupa:");
scanf("%u",&i);
printf("\nDate de iesire\n\n");
r    =100+5*i;
rpl  =1./(3+0.1*i);
n    =200+2*g*i;
fi[0]=25+5*i;
omega=M_PI*n/30;
T    =2*M_PI/omega;
printf("Raza manivelei:    r     = %.2lf [mm]\nRaportul r/l:      r/l   = %.2lf\n\
Turatia:           n     = %.2lf [rot/min]\nUnghiul initial:   fi[0] = %.2lf  [deg]\n\
Viteza unghiulara: omega = %.2lf  [1/s]\nPerioada:          T     = %.2lf   [s]\
", r, rpl, n, fi[0], omega, T);
printf("\n\n\nApasati orice tasta...");
getch();
clrscr();
printf("        Tabelul de variatie a functiilor\n\n");
```

```
printf("+-----+------------+------------+--------------+\n");
printf("| fi  |    s(fi)   |    v(fi)   |     a(fi)    |\n");
printf("|-----+------------|------------|--------------|\n");
for(fid=0; fid<=360; fid+=30)
{
  fir=deg2rad(fid);
  printf("|%4.0lf | %10.4lf | %10.4lf | %12.4lf |\n", fid, s(fir), v(fir), a(fir));
}
printf("+-----+------------+------------+--------------+\n");
printf(" [deg]     [mm]        [mm/s]       [mm/s^2]    ");
printf("\n\n\nApasati orice tasta...");
getch();
clrscr();
printf("Valori caracteristice ale unghiului fi\n\n");
fi[1]=0;
fi[2]=2*M_PI;
fi[3]=0;
fi[4]=M_PI;
fi[5]=2*M_PI;
fi[6]=acos((-1+sqrt(1+rpl*rpl*8))/(4*rpl));
fi[7]=2*M_PI-fi[6];
fi[8]=0;
fi[9]=M_PI;
fi[10]=2*M_PI;
if(rpl>1./4)
 {
 fi[11]=acos(-1/(4*rpl));
 fi[12]=2*M_PI-fi[11];
 }
for(index=1;index<=10;index++)
  printf("fi[%2u]=%.4lf [rad] = %6.2lf [deg]\n", index, fi[index], rad2deg(fi[index]));
if(rpl>1./4)
  printf("fi[%2u]=%.4lf [rad] = %6.2lf [deg]\n\
fi[%2u]=%.4lf [rad] = %6.2lf [deg]\n", 11, fi[11], rad2deg(fi[11]), 12, fi[12], rad2deg(fi[12]));
fi0r=deg2rad(fi[0]);
printf("\nPentru unghiul fi[0] = %.4lf [rad] se obtin:\ns(%.2lf) = %.2lf\nv(%.2lf) \
= %.2lf\na(%.2lf) = %.2lf\n",fi0r,fi0r,s(fi0r),fi0r, v(fi0r), fi0r, a(fi0r));
}
```

Ex5. Studiul distribuţiei de viteze în mişcarea cardanică a unei bare drepte

```
/* Studiul distributiei de viteze in miscarea cardanica
                  a unei bare drepte                    */

#include <stdio.h>
#include <math.h>
#include <conio.h>

double deg2rad(double x)
{
return x*M_PI/180;
}

double rad2deg(double x)
{
return x*180/M_PI;
}

void main (void)
{
double fi[2], dta[5][2], dtb[5][2], dtc[5][2], dti[5][2];
double dta_med[2], dtb_med[2], dtc_med[2], dti_med[2];
double va[2], vb[2], vc[2], vi[2], fir;
double omega1[2], omega2[2], omega_med[2], tan_fi[2], tan_ex_fi[2];
double eps_tan_fi[2], eps_v[2], ds, l;
unsigned i, j, n;
char welcome[]="Studiul distributiei de viteze in miscarea cardanica a unei bare drepte";
printf("%s\n\nDate de intrare\nfi_1 [deg] = ", welcome);
scanf("%lf",&fi[0]);
printf("fi_2 [deg] = ");
```

```
scanf("%lf",&fi[1]);
ds=30;
l=400;
printf("\nParametrii instalatiei:\nds = %.2lf [mm]\nl  = %.2lf [mm]\n", ds, l);
printf("Introduceti numarul de masuratori in puncte: n = ");
scanf("%u",&n);
printf("\nIntroducerea datele experimentale");
for(j=0;j<=1;j++)
 {
 printf("\nPentru fi_%u = %.2lf [deg]\n", j+1, fi[j]);
 printf("\nIntroduceti intervalele de timp masurate [ms]:");

 printf("\nPunctul A:\n");
 dta_med[j]=0;
 for(i=0;i<n;i++)
   {
   printf("dta[%u] = ",i+1);
   scanf("%lf",&dta[i][j]);
   dta_med[j]+=dta[i][j];
   }
 dta_med[j]/=(n*1000);
 printf("Valoarea medie calculata: %lf [s]\n", dta_med[j]);
 printf("\nApasati orice tasta...");
 getch();
 clrscr();

 printf("\nPunctul B:\n");
 dtb_med[j]=0;
 for(i=0;i<n;i++)
   {
   printf("dtb[%u] = ",i+1);
   scanf("%lf",&dtb[i][j]);
   dtb_med[j]+=dtb[i][j];
   }
 dtb_med[j]/=(n*1000);
 printf("Valoarea medie calculata: %lf [s]\n", dtb_med[j]);

 printf("\nPunctul C:\n");
 dtc_med[j]=0;
 for(i=0;i<n;i++)
   {
   printf("dtc[%u] = ",i+1);
   scanf("%lf",&dtc[i][j]);
   dtc_med[j]+=dtc[i][j];
   }
 dtc_med[j]/=(n*1000);
 printf("Valoarea medie calculata: %lf [s]\n", dtc_med[j]);

 printf("\nPunctul I:\n");
 dti_med[j]=0;
 for(i=0;i<n;i++)
   {
   printf("dti[%u] = ",i+1);
   scanf("%lf",&dti[i][j]);
   dti_med[j]+=dti[i][j];
   }
 dti_med[j]/=(n*1000);
 printf("Valoarea medie calculata: %lf [s]\n", dti_med[j]);
 printf("\nApasati orice tasta...");
 getch();
 clrscr();
 }

printf("Date de iesire\n\nViteze liniare\n");
printf("+---------+----------+----------+----------+----------+\n");
printf("| fi[deg] | vA[mm/s] | vB[mm/s] | vC[mm/s] | vI[mm/s] |\n");
printf("+---------+----------+----------+----------+----------+\n");
for(j=0;j<=1;j++)
 {
 va[j]=ds/dta_med[j];
 vb[j]=ds/dtb_med[j];
 vc[j]=ds/dtc_med[j];
 vi[j]=ds/dti_med[j];
```

```
 printf("|   %3.0lf   | %8.2lf | %8.2lf | %8.2lf | %8.2lf |\n\
", fi[j], va[j], vb[j], vc[j], vi[j]);
 }
printf("+---------+---------+---------+---------+---------+\n");

printf("\nViteze unghiulare [rad/s]\n");
printf("+---------+---------+---------+----------+\n");
printf("| fi[deg] |  omega1  |  omega2  | omega_med |\n");
printf("+---------+---------+---------+----------+\n");
for(j=0;j<=1;j++)
 {
 fir=deg2rad(fi[j]);
 omega1[j]=va[j]/(l*cos(fir));
 omega2[j]=vb[j]/(l*sin(fir));
 omega_med[j]=(omega1[j]+omega2[j])/2;
 printf("|   %3.0lf   | %8.2lf | %8.2lf | %9.2lf |\n\
", fi[j], omega1[j], omega2[j], omega_med[j]);
 }
printf("+---------+---------+---------+----------+\n");

printf("\ntg(fi) si erori relative\n");
printf("+---------+-----------+----------+----------+----------+\n");
printf("| fi[deg] |tg(fi) aprox|tg(fi) real| eps_fi[%] | eps_v[%] |\n");
printf("+---------+-----------+----------+----------+----------+\n");
for(j=0;j<=1;j++)
 {
 tan_ex_fi[j]=tan(deg2rad(fi[j]));
 tan_fi[j]=dta_med[j]/dtb_med[j];
 eps_tan_fi[j]=fabs(tan_ex_fi[j]-tan_fi[j])/tan_fi[j]*100;
 eps_v[j]=fabs(vi[j]-vc[j])/vi[j]*100;
 printf("|   %3.0lf   | %10.2lf | %9.2lf | %9.2lf | %8.2lf |\n\
", fi[j], tan_fi[j], tan_ex_fi[j], eps_tan_fi[j], eps_v[j]);
 }
printf("+---------+-----------+----------+----------+----------+\n");

}
```