Funksionet ne gjuhen C

Zgjidhja e nje problemi kompleks, ndryshe nga problemet e thjeshta qe kemi trajtuar deri tani, kerkon zberthimin e tij ne nenprobleme. Secili nga nenproblemet mund te kerkoje nje zberthim ne nje varg nenproblemesh akoma me te thjeshta. Ky proces zberthimi vazhdon derisa arrihet ne probleme elementare te zgjithshme drejtpersedrejti.

Dekompozimi i nje problemi ne nje numer nenproblemesh me te thjeshte, perfaqeson te ashtuquajturen *analize zbritese* (*Top-Down*) per zgjidhjen e nje problemi kompleks. Ne kete rast algoritmi i pergjithshem do te perfshinte nenalgoritmet e shkruara per secilin nenproblem. Per çdo nenalgoritem mund te shkruhet nje program qe ne kete rast do te quhet *nenprogram* ose *modul*. Kombinimi i nenprogrameve ose moduleve do te jepte programin komplet qe zgjidh problemin fillestar.

Ne gjuhen C keto nenprograme quhen funksione, ekzekutimi i te cilave kontrollohet prej njesive te tjera te programit siç eshte programi kryesor ose nenprograme te tjere.

Stili modular i programimit lehteson lokalizimin e gabimeve dhe korrigjimin e tyre. Gjithashtu programet e ndertuar ne kete menyre jane zakonisht me te lehte per t'u kuptuar, pasi strukura e çdo njesie programi mund te studiohet pavaresisht nga njesite e tjera. Se fundi ata mund te modifikohen me lehtesi pasi module te veçante mund te shtohen, te fshihen ose te ndryshohen.

Funksioni në programim është një segment që grupon një numër instruksionesh të programit për të kryer nje detyrë të veçantë.

Një program ne C ka të paktën një funksion kryesor main (), pa te teknikisht s'eshte nje program C.

Llojet e funksioneve n gjuhen C

Në pergjithesi, ka dy lloje funksionesh në C, te përcaktuar nga përdoruesi ose jo.

1. Funksione nga librarite standarte
2. Funksione te përcaktuar nga perdoruesi

- **Funksione nga librarite standarte**

Funksionet nga librarite standarte jane funksione te gatshme ne gjuhes C. Per shembull:

main() - Ekzekutimi i cdo programi ne gjuhen c fillon me fuksionin main().
printf() - Perdoret per te afishuar te dhenat dalese ne C.
scanf() - Perdoret per te rregjistruar te dhenat hyrese ne C.

Libraria math.h permban funksionet e meposhtme:

acos() Mathematike $\cos^{-1}x$ = acos(x) ne gjuhen C

Llogarit arc kosinusin e argumentit (eshte inversi I kosinusit). Shembull perdorimi ne C:

```
#include <math.h>
#define PI 3.141592654
int main()
{
   float num;
   double rezultati;
   printf("Jepni vleren e variablit: ");
   scanf("%f",&num);
   result=acos(num);
   printf("Inversi i cos(%.2f)=%.2lf ne radian",num,rezultati);
   rezultati=(rezultati*180)/PI;
   printf("\nInversi I cos(%.2f)=%.2lf ne grade",num,rezultati);
   return 0;
}
```

Ekzekutimi per vleren 0.4:

Jepni vleren e argumentit: 0.4

Inversi i cos(0.40)=1.16 ne radian

Inversi i cos(0.40)=66.42 ne grade

acosh() Formula: $\cosh^{-1}x=\ln(x+\sqrt{(x^2-1)})$

Llogarit arc kosinusin hiperbolik te argumentit. Shembull perdorimi ne gjuhen C:

```
#include <stdio.h>
#include <math.h>
#define PI 3.141592654
int main()
{
   float num;
   double result;
   printf("Jepni vleren e argumentit: ");
   scanf("%f",&num);
   result=acosh(num);
   printf("Inversi I  cosh(%.2f)=%.2f ne radian",num,result);
```

```
   result=(result*180)/PI;
   printf("\nInversi i cosh(%.2f)=%.2f ne grade",num,result);
   return 0;
}
```

Ekzekutimi per vleren 1.2

Jepni vleren e argumentit: 1.2

Inversi i cosh(1.2)=0.62 ne radian

Inversi i cosh(1.2)=35.66 ne grade

asin() $\sin^{-1}x$

Llogarit arc sinusin e argumentit. Shembull perdorimi ne gjuhen C:

```
#include <stdio.h>
#include <math.h>
#define PI 3.141592654
int main()
{
float num;
double result;
printf("Jepni argumentin: ");
scanf("%f",&num);
result=asin(num);
printf("Inversi i sin(%.2f)=%.2f ne radian",num,result);
result=(result*180)/PI;
printf("\nInversi i sin(%.2f)=%.2f ne grade",num,result);
return 0;
}
```

Ekzekutimi per vleren 0.4:

Jepni vleren e argumentit: 0.4

Inversi i sin(0.40)=0.41 ne radian

Inversi i sin(0.40)=23.58 ne grade

asinh()

Llogarit arc sinusin hiperbolik te argumentit. Shembull perdorimi ne gjuhen C:

```
#include <stdio.h>
```

```
#include <math.h>
#define PI 3.141592654
int main(){
float num;
double result;
printf("Jepni vleren e argumentit: ");
scanf("%f",&num);
result=asinh(num);
printf("Inversi i sinh(%.2f)=%.2f in radians",num,result);
result=(result*180)/PI;
printf("\nInversi i sinh(%.2f)=%.2f in degrees",num,result);
return 0;
}
```

Ekzekutimi per vleren 8:

Jepni vleren e argumentit: 8

Inversi i sinh(8.00)=2.78 ne radian

Inversi i sinh(8.00)=159.08 ne grade

atan() Llogarit arc tangentin e argumentit.

atanh() Llogarit arc tangentin hiperbolik te argumentit

cos() Llogarit cosinusin e argumentit.

cosh() Llogarit cosinusin hiperbolit te argumentit

exp() Llogarit e ne fuqi te nje vlere te dhene per argumentin. Shembull perdorimi ne gjuhen C:

```
#include <stdio.h>
#include <math.h>
int main()
{
 double x, result;
 printf("Jepni vlere e x per te gjetur e^x: ");
 scanf("%lf",&x);
 result=exp(x);
 printf("Exp %.2lf=%.2lf",x,result);
 return 0;
}
```

Jepni vlere e x per te gjetur e^x: 12

Exp 12.00=162754.79

Funksionet e Percaktuara nga Perdoruesi

Gjuha C ju ofron programuesve të përcaktojne funksionin e tyre sipas kërkesave të njohura si funksione te percaktuara nga perdoruesi. Supozojme, një programues qee do të gjeje faktorialin e një numri te dhene si dhe të kontrolloje nëse është numer prim ose jo në të njëjtin program. Më pas, ai /ajo mund të krijojë dy funksione të veçanta user-defined në këtë program: një për gjetjen e faktorialit dhe tjetrin për të kontrolluar nëse është numer prim.

Si punojne ose funksionojne ne C keto funksione?

```
#include <stdio.h>
void function_name()
{
.....
.....
}
int main()
{
.....
.....
function_name();
.....
.....
}
```

Siç e dime, çdo program ne C fillon nga main() dhe programi fillon ekzekutimin e kodeve tij. Kur kontrolli i programit arrin të function_name() brenda funksionit main() kapercen emrin e tij dhe ekzekuton kodet e shkruara ne te. Kur, të gjitha kodet brenda funksion user-defined janë ekzekutuar, kontrolli i programit kapercen instruksionet vetëm pasi thirret function_name().

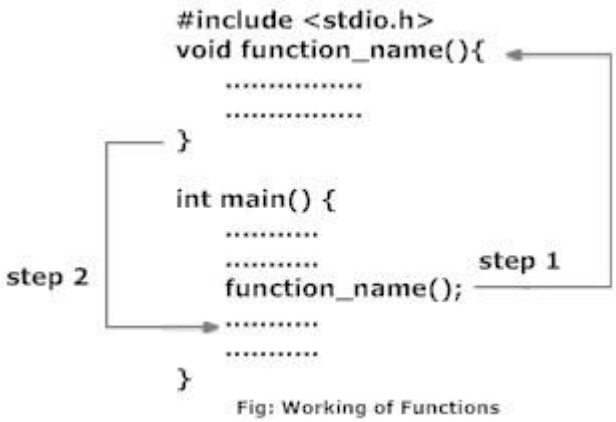

Fig: Working of Functions

Mos harroni, emri i funksionit është një identifikues dhe duhet të jetë unike. Avantazhet e tyre:

- Ndihmojne për të ndare programit kryesor në segmente të vogla që e bën programin te lehtë për tu kuptuar, ruajtur dhe kontrolluar
- Nëse kodi përsëritet në një program. Funksioni mund të përdoret për të përfshirë ato kodet dhe ekzekutohet kur është e nevojshme duke e therritur atë ne programin kryesor. 3. Programet punojne në projekte të medha dhe mund të ndajnë ngarkesën e punës duke bërë funksione të ndryshme.

Shembull 1 Shkruani nje program qe mbledh dy numra te plote. Krijoni nje funksion qe mbledh dy numra te plote dhe llogarit shumen ne programin kryesor.

```
#include <stdio.h>
int add(int a, int b);
int main(){
   int num1,num2,sum;
   printf("Jepni dy numra qe do te mblidhni\n");
   scanf("%d %d",&num1,&num2);
   sum=add(num1,num2);       //function call
   printf("sum=%d",sum);
   return 0;
}
int add(int a,int b)          //function declaratory
{
   int add;
   add=a+b;
   return add;               //return statement of function
}
```

Deklarimi i Funksionit

Çdo funksion në gjuhen C duhet të deklarohet para se të përdoren. Këto lloj deklarimesh janë quajtur edhe funksion prototip. Funksioni prototip jep informacion në lidhje me emrin përpilues te funksionit, llojit të argumenteve që duhet kaluar dhe llojin e kthimit.

Sintaksa e funksionit prototip

```
return_type function_name(type(1) argument(1),....,type(n) argument(n));
```

Ne shembullin e mesiperm ,int add(int a, int b); eshte funksion prototip i cili parashikon te jape kompilatorit informacionin e meposhtem:

1. Emrin e funksionit add()
2. Vlera e kthyer nga funksioni eshte int.
3. Te dy argumentet te cilet kalojne vlerat funksionit jane te tipit int.

Funksioni call

Kontrolli i programit mund të transferohen në funksion kur thirret nga programi kryesor:

Sintaksa e funksionit call

```
Emri I funksionit (argument(1),....argument(n));
```

Ne shembullin e mesiperm, function call eshte nga instruksioni add(num1,num2); pas main(). Kjo ka të bëjë me kontrollimin e kërcimit të programit nga ajo e deklaratë për të funksionuar përkufizim dhe ekzekuton kodet brenda atë funksion.

Instruksioni Return ne C

Return përfundon funksionin aktual dhe kthen kontrollin në pikën e thirrjes. Ajo gjithashtu mund të kthejë vlerën në atë pikë. Sintaksa e saj është si me poshte:

1) Kthimi i kontrollit nga funksioni që nuk kthen vlerën:
   return;
2) Kthimi i kontrollit nga funksioni që kthen vlerën:
   return <value>;
3) Vlera e kthimit mund të jetë çdo shprehje e vlefshme që kthen një vlerë:
   - *një konstante*
   - *një ndryshore*
   - *Një llogaritje, për shembull (a + b) * c*
   - *thirrni një funksion tjetër që kthen një vlerë*

Vlera duhet të jetë e llojit të njëjtë (ose të përputhshëm) me të cilin funksioni është përcaktuar. Për shembull, një funksion int nuk mund të kthejë një vlerë float.

Si funksionon? Kur thërrisni një funksion, ndodhin dy gjëra: 1. Ekzekutimi i funksionit aktual është ndaluar dhe 2. Funksioni që ju therrisni po ekzekutohet.

Kjo është ajo që ne e quajmë transferim të kontrollit. Kur thërrisni një funksion, kontrolli i programit transferohet nga funksioni i thirrjes në funksionin e thirrur. Instruksioni Return e kthen kontrollin tek rutina e mëparshme. Ky funksion do të vazhdojë ekzekutimin e tij nga pika ku u ndalua. Përkundrazi, një funksion që thërret instruksionin Return mbaron. Nuk është në gjendje të ndaloje dhe nuk mund të rifillojë.

Nëse një funksion përcaktohet si "void", nuk ka nevojë të kthejë një vlerë. Funksione të tilla kthejnë kontrollin automatikisht kur arrijnë në fund të trupit të tyre. Megjithatë, mund të përdorim "return;" për t'i dhënë fund ekzekutimit të tyre nga çdo pikë e trupit të tyre.

Kthimi i një vlere - Nëse një funksion është përcaktuar për kthimin e një vlere, ai duhet të përdorë instruksionin Return. Duhet gjithashtu të sigurohet që çdo rezultat i mundshëm të kthejë një vlerë. Kjo mund të bëhet e ndërlikuar nëse rutina juaj ka ndërtime të mbivendosur dhe nuk jeni të kujdesshëm.

Kthimi I vleres -1 nga instruksionin return tregon se ka nje problem ne kodim.

Llojet e funksioneve të përcaktuara nga përdoruesit në programimin C mund të kategorizohen si:

1. Funksioni pa argumente dhe pa vlerë kthimi

```
….
void prime();   //Deklarimi i funksionit pa argumenta
….
   prime();     //Therritja e funksionit ne programin kryesor
….
```

2. Funksioni pa argumente dhe me vlere e kthimit

```
….
int input()    //Deklarimi i funksionit pa argumenta dhe me vlere e kthimit
{
   int n;
   printf("jepni nje numer te plote:\n");
```

```
   scanf("%d",&n);
   return n;
}
…..
num=input();     //Therritja e funksionit ne programin kryesor
….
```

3. Funksioni me argumente por pa vlerë kthimi

```
….
void check_display(int n)  //Deklarimi i funksionit me  argumenta
….
check_display(num);   //Therritja e funksionit ne programin kryesor
….
```

4. Funksioni me argumente dhe me vlera e kthimit.

```
….
int check(int n) //Deklarimi i funksionit me  argumenta dhe me vlere kthimi
….
num_check=check(num); //Therritja e funksionit ne programin kryesor
….
```

Ushtrime me funksione

Ushtrimi 1 Ndertoni nje program qe gjen te gjithe numrat prim ne nje segment te dhene duke perdorur funsionet.

```
#include<stdio.h>
int check_prime(int num);
int main(){
  int n1,n2,i,flag;
  printf("Jepni dy numra te plote pozitive");
  scanf("%d %d",&n1, &n2);
  printf("numrat prim midis %d dhe %d jane: ", n1, n2);
  for(i=n1+1;i<n2;++i)
  {
    flag=check_prime(i);
    if(flag==0)
      printf("%d ",i);
```

```
  }
  return 0;
int check_prime(int num) /*funksioni qe gjeni numrat prim*/
{
  int j,flag=0;
  for(j=2;j<=num/2;++j)
     {
      if(num%j==0)
           {  flag=1;  break;
      }  }
  return flag;
}
```

Ushtrimi 2 Ndertoni nje progam qe gjen shumen nga 1 deri ne n numrave te plote positive, kur n jepet nga përdoruesi, duke perdorur thirrjen rekursive.

```
 #include<stdio.h>
int add(int n);
int main()
{
   int n;
   printf("Jepni nje numer te plote pozitiv: ");
   scanf("%d",&n);
   printf("Shuma = %d",add(n));
   return 0;
}
int add(int n)
{
   if(n!=0)
    return n+add(n-1);  /* recursive call */
}
```

Ushtrimi 3 Ndertoni nje progam qe gjen faktorialin e nje numri të dhene nga përdoruesi.

```
#include<stdio.h>
int factorial(int n);
```

```
int main()
{
   int n;
   printf("Jepni nje numer te plote pozitiv: ");
   scanf("%d",&n);
   printf("Faktoriali %d = %ld", n, factorial(n));
   return 0;
}
int factorial(int n)
{
   if(n!=1)
    return n*factorial(n-1);
}
```

Ushtrimi 4 Ndertoni nje progam qe kthimin mbrapsh te një fjalie të dhene nga përdoruesi pa përdorur funksionet e stringjeve.

```
#include <stdio.h>
void Reverse();
int main()
{
   printf("Jepni nje fjali: ");
   Reverse();
   return 0;
}
void Reverse()
{
   char c;
   scanf("%c",&c);
   if( c != '\n')
   {
      Reverse();
      printf("%c",c);
   }
}
```