## Funksionet matematikore të bibliotekës “cmath” në gjuhën programore C++

Përveç funksioneve të ndryshme që mund të krijojmë, C++ gjithashtu përfshin disa funksione të dobishme që mund ti përdorim. Këto funksione janë në dispozicion në bibliotekën standarde në C dhe C++ dhe quhen funksione të para ndërtuara (ang. **built-in** functions). Këto janë funksione që mund të përfshihen në programet që i krijojmë dhe pastaj ti përdorim.

C++ ka një seri të pasur të operacioneve matematikore të cilat mund të kryhen me numra të ndryshëm. Ja disa funksione të dobishme të para krijuara në bibliotekën cmath të C++-it

| Nr. | Funksioni & Qëllimi |
|---|---|
| 1 | **double cos(double);**<br>Ky funksion merr një kënd (si double) dhe kthen kosinusin e tij. |
| 2 | **double sin(double);**<br>Ky funksion merr një kënd (si double) dhe kthen sinusin e tij. |
| 3 | **double tan(double);**<br>Ky funksion merr një kënd (si double) dhe kthen tangjentin e tij. |
| 4 | **double log(double);**<br>Ky funksion merr një numër dhe kthen logaritmin natyral të atij numri. |
| 5 | **double pow(double, double)**<br>Numri i parë është baza kurse numri i dytë fuqia e bazës |
| 6 | **double hypot(double, double);**<br>Nësë këtij funksioni i japim gjatësitë e dy anëve të trekëndëshit kënddrejt, ai to dë kthejë gjatësinë e hypotenuzës. |
| 7 | **double sqrt(double);**<br>Funksion i cili kthen rrënjën katrore të numrit. |
| 8 | **int abs(int);**<br>Ky funksion kthen vlerën absolute të numrit të plotë. |
| 9 | **double fabs(double);**<br>Ky funksion kthen vlerën absolute të numrit decimal. |
| 10 | **double floor(double);**<br>Ky funksion kthen vlerën e plotë të numrit decimal edhe atë duke rrumbullaksuar kah më e vogla. |

Tipi i të dhënave që ceket në kllapa double është tip i numrave real me saktësi të dyfishtë(15 shifra të rëndësishme), kurse float po ashtu është tip i numrave real po me saktësi të thjeshtë (7 shifra të rëndësishme).

Për të shfrytëzuar këto funksione, duhet të përfshihet në krye të programit biblioteka <cmath>.

Ose duke e shtuar këtë rresht menjëherë para komandës para procesorike si në vijim:

```
#include<iostream>
#include<cmath>
using namespace std;
```

Ja një shembull me disa funksione matematikore:

```
#include <iostream>
#include <cmath>
using namespace std;

int main ()
{
    // deklarimi dhe inicializimi i variablave:
    short     s = 10;
    int       i = -1000;
    long      l = 100000; float
              f = 230.47; double
    d = 200.374;

    // operacione matematikore;
    cout << "sin(d) :" << sin(d) << endl;
    cout << "abs(i)          :" << abs(i) << endl; cout <<
    "floor(d) :" << floor(d) << endl; cout << "sqrt(f) :" << sqrt(f)
    << endl;
    cout << "pow( d, 2) :" << pow(d, 2) << endl;

    return 0;
}
```

Pasi të përkthejmë (kompajlojm), ndërtojmë dhe ekzekutojmë programin në dalje do të fitojmë këtë rezultat.

```
sin(d) :-0.634939 abs(i)
          :1000 floor(d)
:200 sqrt(f) :15.1812
pow( d, 2 ) :40149.7
```

1. Të shkruhet funksioni për llogaritjen e vlerës absolute për një vlerë të caktuar.

```
#include <iostream>
#include <cmath>
using namespace std;

int main ()
{
    cout<< "Vlera absolute e 3.1416 sht "
        << abs (3.1416)
        << endl;
    cout<< "Vlera absolute e -2.89 sht "
        << abs (-2.89) << endl;
    return 0;
}
```

2. Të shkruhet funksioni për llogaritjen e gjatësisë së hipotenuzës për trekëndëshin kënddrejtë, nëse dihen gjatësia e brinjës a dhe brinjës b.

```
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
        double a, b;
        cout<<"Llogaritja e hipotenuzës për trekëndshin këndrejt\n";
        cout << "Shëno gjatësinë e brinjës a:";
        cin >> a ;
        cout << "Shëno gjatësiën e brinjës b:";
        cin >>b;
        cout << "Gjatësia e hipotenuzës:                    " << hypot(a, b) << '\n';

        system("pause");
        return 0;
}
```

3. Të shkruhet programi për llogaritjen e sipërfaqes së rrethit duke përdorur funksionin **pow**.

```
#include<iostream >
using namespace std;

int main()
{
      float r, s;
      const float pi=3.14159;
      cout<< "Vlera e rrezes r = ";
      cin >> r;

      s = pi*pow(r,2);              //pow(r,2) e ngrit ne katror rrezen r cout<< "\nSiperfaqja
      e rrethit: s = "
                  << s << endl;
      return 0;
}
```

4. Të shkruhet programi për llogaritjen e rrënjës katrore të një numri të dhënë, duke e përdorur funksionin **sqrt**.

```
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
   double x = 25;
   cout<<"Rrenja katrore e "<<x
        <<" eshte " <<sqrt(x);
   system("pause");
   return 0;
}
```

5. Të shkruhet programi për llogaritjen e sinusit të një këndi të caktuar duke përdorur funksionin **sin**.

```
#include <iostream>
#include <cmath>
using namespace std;

#define PI 3.14159265

int main ()
{
	double kendi;
	kendi = 30.0;
	cout<<"Sinusi i " << kendi
		<< " shkalleve eshte "
		<< sin (kendi*PI/180)
		<<endl;
	system("pause");
	return 0;
}
```

6. Të shkruhet programi për llogaritjen e kosinusit të një këndi të caktuar duke përdorur funksionin **cos**.

```
#include <iostream>
#include <cmath>
using namespace std;

#define PI 3.14159265

int main ()
{
	double kendi;
	kendi = 45.0;
	cout<<"Kosinusi i " << kendi
		<< " shkalleve është
		<< cos (kendi*PI/180)
		<<endl;
	system("pause");
	return 0;
}
```

7. Të shkruhet programi për llogaritjen e tangjentit për një vlerë të caktuar, duke përdorur funksionin **tan**.

```
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
   double x = 0.3333;
   cout<<"Tangjenti i " <<x
         <<" është " << tan(x);

   return 0;
}
```

8. Të shkruhet programi për llogaritjen e shprehjes $\mathbf{y=e^x}$ duke përdorur funksionin **exp**.

```
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
        double x = 2.7;
    cout<<"'e' e ngritur ne fuqinë 2.7 është "
            <<exp(x)
            <<endl;
    system ("pause");
    return 0;
}
```

9. Të shkruhet programi për llogaritjen e ln(x) për një vlerë të caktuar duke e përdorur funksionin **log**.

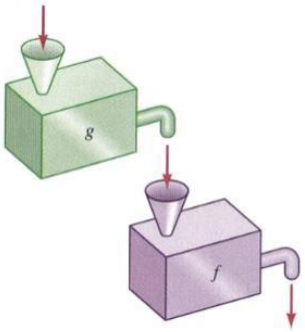
```
#include <iostream>
#include <cmath>
using namespace std;

int main ()
{
    double x=5.8;
    cout<<"ln("<<x<<") = "
        <<log (x);
    system("pause");
    return 0;
}
```

## Funksionet e caktuara nga përdoruesi

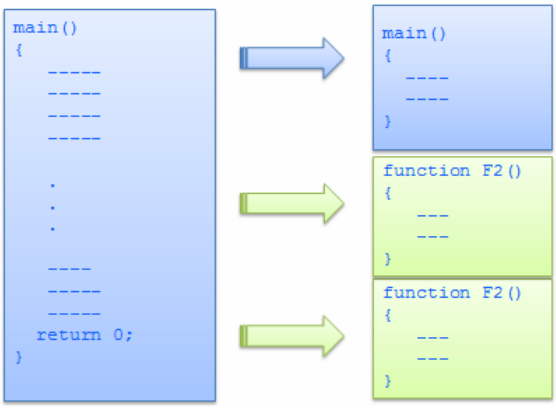
Gjatë shkruarjes së programeve të ndryshme përdoren biblioteka me funksione të gatshme, të përcaktuara nga përpiluesi i kompajlerit të gjuhës C++. Por, edhe vetë shfrytëzuesi mund të përcaktojë funksione të ndryshme dhe t'i shfrytëzojë ato sipas nevojës.
Një grumbull i komandave të caktuara brenda programit mund të përsëritet më shumë herë. Me qëllim të thjeshtimit të shkruarjes së programit, grumbulli i tillë deklarohet si pjesë e veçantë, e cila quhet nënprogram. Pastaj, nënprogrami thirret përmes emrit të tij, sa herë që nevojitet.
Në gjuhën C++ shfrytëzohen vetëm tipe të nënprogrameve që njihen si funksione (ang. function). Prandaj, zakonisht, në këtë gjuhë, kur flitet për nënprograme, mendohet në përcaktimin dhe në shfrytëzimin e funksioneve të ndryshme.
Në fillim është përmendur funksioni main(), i cili paraqitet në çdo program. Gjithashtu, në pjesët paraprake janë përdorur disa funksione matematikore të cilat gjenden në kuadër të modulit cmath.

## Funksionet

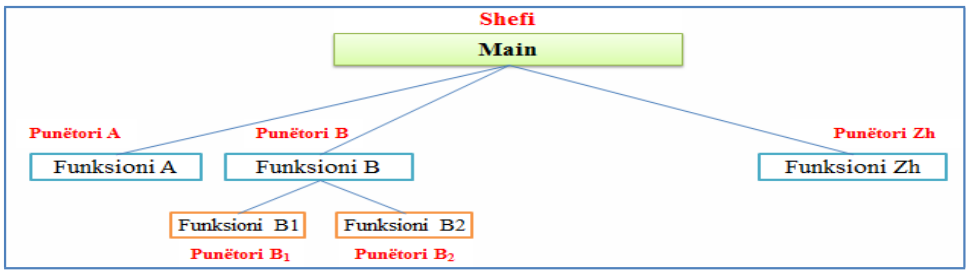
Funksioni është një grup i shprehjeve dhe komandave të ndryshme që kur ekzekutohet kryen një punë të caktuar. Funksioni pas kryerjes së punës mund të kthejë rezultat për punën e kryer apo edhe mund të mos kthejë rezultat (void). Forma abstrakte e funksionit është dhënë në figurën 1. Rezultati/dalja e një funksioni mund të jetë hyrje e një funksioni tjetër.




Përvojat tregojnë që mënyra më e mirë për të zhvilluar dhe mirëmbajtur programe të mëdha është konstruktimi i tyre nga pjesë të vogla (module/nënprograme). Figura 2 paraqet se si një program i shkruar i tëri në funksionin **main** është ndarë në pjesë të vogla (funksione).




Përdorimi i funksioneve lehtëson dizajnin, ndërtimin, debugimin, zgjerimin, modifikimin, ripërdorim dhe organizmin më të mirë të programit. Mënyra se si funksioni **main** i thirr funksionet e tjera ka një analogji sikurse shefi me punëtorët. Shefi (funksioni thirrës) i kërkon punëtorit (funksioni i thirrur) për të kryer një punë dhe për të kthyer përgjigje kur të kryhet puna.




Një nga arsyet kryesore për përdorimin e funksioneve është ndarja e programit në punë të vogla dhe nëse diçka nuk shkon mirë, atëherë është më e lehtë të identifikohet se ku është problemi.

Funksionet e caktuara nga përdoruesi, në C++ klasifikohen në dy kategori:

- Funksione që kthejnë vlerë – këto funksione kanë një tip kthyes dhe duke përdorur deklaratën return e kthejnë rezultatin e llojit të caktuar të të dhënave.
- Funksione boshe (që nuk kthejnë vlerë) – këto funksione nuk kanë tip kthyes të të dhënave. Këto funksione nuk e përdorin deklaratën return për të kthyer rezultat.

## Funksionet që kthejnë vlerë

Sintaksa e deklarimit të një funksioni të caktuar nga përdoruesi dhe i cili kthen rezultat është si më poshtë:

**tipi emri_funksionit(tipi1 par1, tipi2 par3 ...)**

**{**

**deklarimet dhe shprehjet tjera_**

**}**

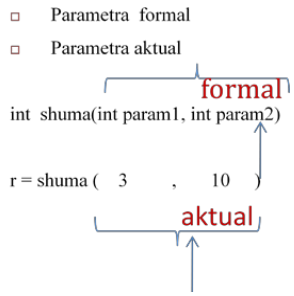
- emri_funksionit – secili identifikator valid
- tipi – tipi i të dhënave për funksionin/rezultatin
- tipi1, tipi2 – tipi i të dhënave për parametrat
- par1, par2 – emrat e parametrave

Shembull: Programi përmes së cilit llogaritet shuma e dy numrave përmes funksionit shuma

```
#include <iostream>
using namespace std;
//Nenprogrami shuma
int shuma (int a, int b);  // deklarimi i funksionit
int main ()
{
int z;
z = shuma (3,10);  // thirrja e nenprogramit, percjellja e ti me vlerat aktuale 3 dhe 10 cout <<
"Rezultati: " << z << "\n\n";
return 0;
}
```

```
int shuma (int a, int b)
{
int r;                // variable e brendshme r=a+b;
return (r);        // kthimi i rezultatit te funksionit
```

Parametrat e shënuar brenda kllapave në titullin e funksionit gjatë definimit të tij, quhen parametra formalë, sepse përmes tyre tregohet forma e llogaritjeve që kryhen brenda funksionit. Kurse, parametrat me të cilët zëvendësohen ato formalë gjatë thirrjes së nënprogramit, siç u përmend edhe më sipër, quhen parametra aktualë.




Parametrat formalë dhe ato aktualë duhet të përputhen mes vete për nga:

- numri - sa ka parametra formalë aq duhet të ketë edhe parametra aktualë;
- tipi - tipin e njëjtë duhet ta kenë parametrat formalë dhe parametrat aktualë përkatës;
- radha e shkruarjes - parametrit formalë në pozitë të caktuar i përgjigjet parametër aktual me pozitë të njëjtë.

Sh2. Programi përmes së cilit llogaritet syprina e katërkëndëshit me brinjët a dhe b, duke e shfrytëzuar funksionin Syprina

```
#include <iostream>
using namespace std;
double Syprina(float x,float y);
int main()
{
float a,b;
double s;
cout << "\nBrinja a: ";
cin >> a;
cout << "\nBrinja b: ";
cin >> b;
```

```
  s=Syprina(a,b);
  cout << "\nsyprina s="<< s << "\n\n";
  return 0;}
  // Nënprogrami Syprina
  double Syprina(float x,float y)
  {  return x*y; }
```

Detyra 1. Program përmes së cilit llogaritet ndryshimi i dy numrave me anë të funksionit zbritja

```
// shembull parametrat e funksioneve, zbritja
#include <iostream>
using namespace std;

int zbritja (int a, int b)
{
int r; r=a-b; return r;
}
int main ()
{
int x=5, y=3, z;
z = zbritja (7,2);
cout << "Rezultati i pare eshte: " << z << '\n';
cout << "Rezultati i dyte eshte: " << zbritja (7,2) << '\n';
cout << "Rezultati i trete eshte: " << zbritja (x,y) << '\n'; z= 4 + zbritja (x,y);
cout << "Rezultati i katert eshte: " << z << '\n';
return 0;
}
```

Detyra 2. Të shkruhet programi i cili përmes funksionit max e kthen si rezultat numrin më të madh në mes numrave x dhe y.

```
#include <iostream>

using namespace std;

double max(double x, double y)

{

double max;

if (x >= y)

max = x;

else

max = y;
```

```
return max;

}

int main()

{

double a,b;

a=42;

b=567;

cout<< "Numri më i madh është: "

<<max(a,b)

<<endl;

return 0;

}
```

Detyra 3. Krijo funksionin hipotenuza i cili llogarit gjatësinë e hipotenuzës së trekëndëshit kur dihet dihen gjatësitë e dy krahëve tjerë. Funksioni duhet të ketë dy parametra hyrës të tipit double dhe të kthejë hipotenuzën të tipit double.

```
#include <iostream>

#include <iomanip>

using namespace std;

double hypotenuza( double, double );

int main()

{

double brinja1, brinja2;

cout << setiosflags( ios::fixed | ios::showpoint );

cout << "Jep gjatsinë e brinjës b1= ";

cin >> brinja1;

cout << "Jep gjatsinë e brinjës b2= ";

cin >> brinja2;

cout << "Hypotenuza: " << setprecision(2)

<< hypotenuza( brinja1, brinja2 )

<< endl<<endl;
```

return 0;

}

double hypotenuza( double b1, double b2 )

{

return sqrt(b1 * b1 + b2 * b2 );

}

## Funksionet që nuk kthejnë vlerë (void)

Funksionet boshe (që nuk kthejnë vlerë) dhe funksionet që kthejnë vlerë e kanë strukturën e njëjtë. Të dy llojet e kanë edhe kokën (heading) dhe trupin (body). Funksionet boshe nuk kthejnë rezultat, por vetëm e kryejnë një punë të caktuar dhe nuk e përdorin deklaratën return. Megjithatë edhe tek këto funksione mund të përdoret return pa ndonjë vlerë kthyese. Për të deklaruar një funksion bosh përdoret fjala void para emrit të funksionit, për tipin kthyes të të dhënave. Sintaksa e deklarimit është si në vijim:

void emri_funksionit(tipi1 par1, tipi2 par3 ...)

{

deklarimet dhe shprehjet tjera_

}

Edhe funksionet që kthejnë vlerë edhe funksionet boshe, mund të kenë apo mos të kenë parametra formalë. Figura  paraqet deklarimin dhe implementimin e funksionit printoNumrin i cili shtyp vlerën e parametrit.

```
void printoNumrin(int numri)
{
        cout<< "Numri është "
            << numri << endl;
}


int main()
{
        printoNumrin(4);
        return 0;
}
```

Detyra 1. Të shkruhet funksioni void shtyp_numrin  i cili shtyp  vlerën e parametrit të tij.

#include <iostream>

using namespace std;

```
void shtyp_numrin(int numri)
{
cout << numri << '\n';
}
int main()
{
cout<<"Thirja e funksionit void\n";
shtyp_numrin(10);
return 0;
}
```

2. Të shkruhet funksioni void max i cili gjen dhe shtyp numrin më të madh nga dy parametrat formal të tipit double.

```
#include <iostream>
using namespace std;
void max(double x, double y)
{
double m;
if (x >= y)
m = x;
else
m = y;
cout<<"Numri më  madh është"
<<m;
}
int main()
{
double a=42,b=55;
max(a,b);        //thirrja e funksionit void
```

```
return 0;

}
```

3. Të shkruhet funksioni void shtypja i cili shtyp numrat e plotë nga 1 deri në n. Funksioni le ta ketë një parametër formal të tipit int.

```
#include <iostream>

using namespace std;

void shtypja(int);        //prototipi i funksionit

int main ()

{

shtypja(5);       //thirrja e funksionit

return 0;}

void shtypja (int n)

{

cout << "Shtypa e numrave nga 1 deri n <<n<< endl;

for (int i=1;i<=n;i++)

cout<<i<<endl;

}
```

- ✓ **Për funksionet që afishojnë mesazhe normalisht ska nevoje për tip kthimi.**
  Për shembull: Funksioni pa tip kthimi **hello** i cili shtyp mesazhin hello

```
#include
void hello (void)
{
  cout << "hello world!";
}

int main ()
{
  hello ();
  return 0;}
```

***shembulli 2. Funksioni pa tip kthimi hello i cili shtyp mesazhin*** **'**Une jam nje funksion'

```
#include <iostream>
using namespace std;

void shtypmesazhin ()
{
```

```
cout << "Une jam nje funksion";
}

int main ()
{
shtypmesazhin ();
return 0;
}
```

## Kalimi i argumentave

Deri tani, parametrat **i kalohen fuksionit me vlere**. Kjo dmth se kur thërasim një funksion me parametra i kemi kaluar funksionit vetëm vlerat e variablave dhe jo vetë variablat. Për shembull:

```
int x=5, y=3, z;
z = shuma ( x , y );
```

Ne këtë rast thiret funksioni duke i kaluar vlerat x dhe y, dmth 5 dhe 3 dhe jo vete variablat.

Në këtë mënyr kur funksioni shuma thirret vlerat e variablave të tij a dhe b bëhen 5 dhe 3 por çdo modifikim i a apo b brenda funksionit shuma nuk do të kenë efekt të x dhe y jashte tij, sepse variablat x dhe y kaluan thjesht vlerat e tyre përkatëse

Por mund te lind nevoja te manipulojme brenda funksionit vlera të variablave external(jashtëm). Për këto raste përdoren argumentet që kalojne me reference, si me poshte:

```
// kalimi me reference

#include

void dubliko (int& a, int& b, int& c)

{

  a*=2;

  b*=2;

  c*=2;

}

int main ()

{

  int x=1, y=3, z=7;

  dubliko (x, y, z);

  cout << "x=" << x << ", y=" << y << ", z=" << z;

  return 0;

}
```

Rezulti(output) do të jetë: x=2, y=6, z=14

Nëse vërejme deklarimin e funksionit dubliko tipi i çdo argumenti ndiqet nga një ampersand(&), që tregon se variabli do të kalohet me reference dhe jo me vlerë si zakonisht.

Kur i kalon parametrat me reference, po kalojmë vetë variablin dhe çdo modifikim që bejmë brenda funksionit për parametrat, do të kenë efekt jashtë tij.

Nëse e deklaronim funksionin si:

void dubliko (int a, int b, int c)

do të kalonin vetëm vlerat dhe variablat nuk do të ishin të modifikuar në main

Ky tip deklarimi "me reference" duke perdorur (&) përdoret vetëm te C++. Ne C perdoreshin pointerat per te bere dicka ekuvalente

shembull:

```
// me shume se nje vlere kthimi

#include

void prevnext (int x, int& prev, int& next)

{

  prev = x-1;

  next = x+1;

}

int main ()

{

  int x=100, y, z;

  prevnext (x, y, z);

  cout << "Previous=" << y << ", Next=" << z;

  return 0;

}
```

Output:Previous=99, Next=101

Vlerat default e argumentave (paracaktuar)

Kur deklarojme një funksion mund tu japim vlera default argumentave të tij. Kjo vlerë do të përdoret nëse parameteri është bosh në të majtë kur thiret funksioni. Për të bërë këtë thjesht i jep një vlerë argumentave te deklarimi i funksioneve. Nëse vlera për një parameter nuk kalohet kur funksioni thiret përdoret vlera default, por nëse vlera specifikohet atehere vlera default nuk meret parasysh:

```
// vlera default
```

```
#include <iostream>

using namespace std;

int divide (int a, int b=2)

{

  int r;

  r=a/b;

  return (r);

}


int main ()

{

  cout << divide (12);

  cout << endl;

  cout << divide (20,4);

  return 0;

}
```

6
5

siç shihet ka dy thirje të funksionit divide. E para:

divide (12)

kemi specifikur vetëm një argument , por funksioni kërkon dy. Kështu divide mer si vlerë 2 për parametrin e dyte . Rezultati i kesaj thirrje të funksionit është 6 (12/2).

Ne thirrjen e dyre:

divide (20,4)

dhe rezultati eshte  5 (20/4).

## Recursiviteti

Recursiviti eshte procesi, kur nje funksion therret vetveten . Per shembull: llogaritja e faktorielit të numrit n! = n * (n-1) * (n-2) * (n-3) ... * 1

n! = n * (n-1) * (n-2) * (n-3) ... * 1

```
// llogaritësi i faktorielit

#include

long factorial (long a)   //deklarimi i funskionit factorial

{

  if (a > 1)

   return (a * factorial (a-1));

  else

   return (1);

}

int main ()

{

  long l;

  cout << "Type a number: ";

  cin >> l;

  cout << "!" << l << " = " << factorial (l);

  return 0;

}
```

```
Type a number: 9
!9 = 362880
```

Pra te funksioni  factorial perfshijme nje thirrje te vetes.

## Prototipi i funksioneve

Deri tani, keni përcaktuar funksionet përpara shfaqjes së parë të thirjes për to, që zakonisht ishte në main, duke lëne funksionin main në fund. Nëse përsërisim shembujt duke vendosur funksionin main para funksioneve të tjera, do të shihnim një gabim. Arsyeja është se për te qenë e mundur thirrja e një funksioni, duhet të deklarohet paraprakisht ky si funksion

Por ka një alternative për të shmangur shkrimin e të gjithë funksioneve para main. Kjo realizohet me ane te prototipit. Që dmth të bësh një deklarim paraprak për funksionin. Kompilatori duhet te dijë argumentat dhe tipin e kthimit

Sintaksa është:

tipi emri_funksionit(tipi1 par1, tipi2 par3 ...);

Është identik me rreshtin e parë të deklarimit të funksionit perveç :

Nuk permban {}

Nuk ka ne fund (;).

Te argumentat është e mjaftueshme të shtypen tipet e tyre. Për shembull:

```
// prototipet
#include
void odd (int a);
void even (int a);
int main ()
{
  int i;
  do {
    cout << "Type a number: (0 to exit)";
    cin >> i;
    odd (i);
  } while (i!=0);
  return 0;
}
void odd (int a)
{
```

```
  if ((a%2)!=0) cout << "Number is odd.\n";
  else even (a);
}
void even (int a)
{
  if ((a%2)==0) cout << "Number is even.\n";
  else odd (a);
}
```

Output: Type a number (0 to exit): 9
Number is odd.
Type a number (0 to exit): 6
Number is even.
Type a number (0 to exit): 1030
Number is even.
Type a number (0 to exit): 0
Number is even.

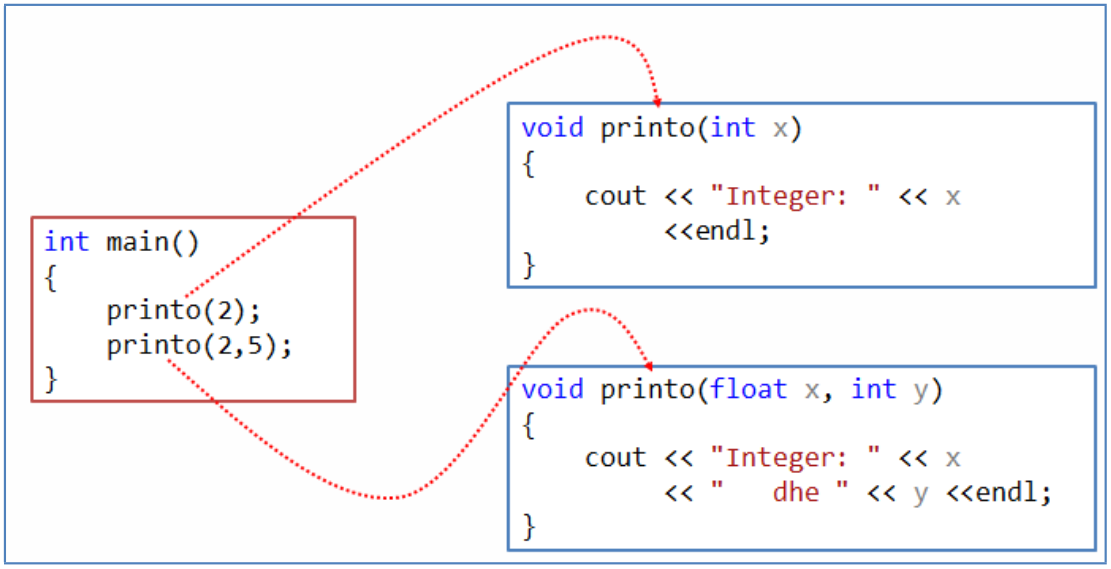
Ky shembull është ndërtuar në mënyrën më të gjatë për të shpjeguar prototipet.

Prototipet e funksioneve jane:
**void odd (int a);**
**void even (int a);**
që mundëson që funksionet të përdoren para se të deklarohen plotësisht,
për shembull në main.Shume programues rekomandojne që për çdo funksion duhet të shkruajme prototipin. Egzistenca e tyre në të njëjtin vend lehtëson punë gjatë programimit dhe gjithashtu lehtëson procesin e krijimit te 'header file'.

**Funksionet e mbingarkuara**

Funksionet e mbingarkuara janë dy ose më shumë funksione me të njëjtin emër, por me parametra të ndryshëm për nga numri i parametrave ose tipi i të dhënave të parametrave. C++ lejon disa funksione me emër të njëjtë deri sa e kanë nënshkrimin e ndryshëm. Kompajleri i C++ zgjedh funksionin e duhur për ta thirrur, duke u bazuar në numrin, tipin dhe rendin e parametrave. Zakonisht funksionet e mbingarkuara përdoren për të kryer punë të ngjashme por me tipa të ndryshëm të të dhënave.

Një shembull i funksionit të mbingarkuar shihet në figurën e mëposhtme.




//shembulli i funksionit të mbingarkuar(overloaded) divide

#include

int pjesto(int a, int b)

{

return (a/b);

}

float pjesto(float a, float b)

{

return (a/b);

}

```
int main ()
{
  int x=5,y=2;
  float n=5.0,m=2.0;
  cout << pjesto(x,y);
  cout << "\n";
  cout << pjesto(n,m);
  cout << "\n";
  return 0;
}
```

```
2
2.5
```

Në këtë rast ne kemi përcaktuar dy funksione me të njëjtin emër, por njeri pret argumenta të tipit int dhe tjetri të tipit float. Kompilatori e di cilin të therras në çdo rast duke ekzaminuar tipet, kur funksioni thirret. Nëse thirret me dy int-s si argument thërret funksionin që ka dy int ne prototip , nëse thirret me dy float thërret funksionin tjeter.

Për thjeshtesi, kemi shkruar të njejtin kod në trupin e funksioneve. Mund të ndërtojme dy funksione me të njejtin emër por me sjellje komplet të ndryshme.

**Detyra 1.** Të shkruhet programi i cili llogarit sipërfaqen e rrethit dhe drejtkëndëshit përmes funksionit të mbingarkuar siperfaqja().

```
#include<iostream>
using namespace std;
#define pi 3.14159265359
void siperfaqja(int r)
{
cout<<"Sipërfaqja e rrethit:"<<pi*r*r;
```

```
}
void siperfaqja(int a,int b)
{
cout<<"Sipërfaqja e drejtkëndëshit:"<<a*b;
}
int main()
{
int a,b,r;
cout<<"Rrezja r= "; cin>>r; siperfaqja(r);
cout<<"\nKrahet        a=?b=?"; cin>>a>>b; siperfaqja(a,b); cout<<endl; system("pause");
return 0;}
```