Функциски прототипови и принуда на аргументи

Во C++, е задолжително да се декларираат сите функции пред истите да може да се повикуваат. На пример, програма која со функција пресметува најголем од три дадени броја:

```
#include <iostream>
using namespace std;
int najgolem(int a, int b)
{
   int n;
   if (a>b) n=a;
   else n=b;
   return n;
}
int main()
{
   int a,b,c,m,k;
   cout<<"Vnesi tri veli broja"<<endl;
   cin>>a>>b>>c;
   m=najgolem(a,b);
   k=najgolem(m,c);
   cout << "Najgolem e" <<k;
   return 0;
}
```

во програмата, функцијата najgolem(int a, int b) мора, во програмскиот код, да е дефинирана пред функцијата main(). Во спротивно, компајлерот ќе пријави дека не знае што е тоа najgolem (int a, int b) и нема да успее да ја претвори програмата во извршна датотека.

Понекогаш, сакаме да го направиме обратното - прво да го запишеме поважниот дел од програмата, па потоа да ги наведеме подзадачите (функциите) кои служат како помошно средство при решавање на главниот проблем. Или пак, можеби е потребно да креираме две функции F1() и F2() кои се повикуваат една со друга - F1() ја повикува F2() и F2() ја повикува F1(). Во тој случај, не постои начин на кој F1() би ја запишале пред F2() и обратно - F2() би ја запишале пред F1(). Во вакви ситуации, C++ овозможува креирање на, т.н., **прототипови на функции**. Прототип претставува декларација на функција без пишување на нејзиниот код.

Синтаксата за креирање на прототипови е идентична со синтаксата што ја користевме за креирање на функции - само без наведување на содржината на функцијата (наредбите кои таа ги содржи). Наместо содржина го запишуваме знакот ';' (кој е задолжителен!). Притоа, бидејќи за креирање на прототипот не се потребни вистинските имиња на параметрите (туку само нивните типови) - нив можеме да ги прескокнеме.

```
#include <iostream>
using namespace std;
int najgolem(int a, int b);        //prototip so iminja na parametri
int najmal(int, int);              //prototip bez iminja na parametri
int main()
{
   int a,b,c,m,k,l,m1;
   cout<<"Vnesi tri celi broja"<<endl;
   cin>>a>>b>>c;
```

```
    m=najgolem(a,b);
    k=najgolem(m,c);
   cout << "Najgolem e  " <<k<<endl;
   m1=najmal(a,b);
   l=najmal(m1,c);
   cout<<"Najmal e  "<<l;
   return 0;
}

//definicija na najgolem(int a, int b)
int najgolem(int a, int b)
{
   int n;
   if (a>b) n=a;
   else n=b;
   return n;
}

//definicija na najmal(int a, int b)
int najmal(int a, int b)
{
   int n;
   if (a<b) n=a;
   else n=b;
   return n;
}
```

Пр. Програма што ќе пресмета и печати вкупна плоштина на три триаголници со внесени страни, при што да се користи функција за плоштина на еден триаголник

```
#include <iostream>
# include <cmath>
float plostina(float a, float b, float c);

using namespace std;
  int main()
{
   float i,s=0,a,b,c;
   for (i=1;i<=3;i++)
   {
      cout<<"vnesi gi stranite na triagolnikot"<<endl;
      cin>>a>>b>>c;
      s+=plostina(a,b,c);
   }
   cout << "Vkupnata plostina e "<<s;
   return 0;
}
float plostina(float a, float b, float c)
  {
     float p,s;
```

```
    s=(a+b+c)/2;
    p=sqrt(s*(s-a)*(s-b)*(s-c));
        return p;
  }
```

Пр. Да се напише програма која ќе ги отпечати сите четирицифрени природни броеви кои се деливи со збирот на двата броја составен од првите две цифри и од последните две цифри на четирицифрениот број, и на крајот ќе отпечати колку вакви броеви се пронајдени.

На пример: 3417 е делив со 34+17, 5265, 6578, ...

```
#include<iostream>
using namespace std;
int zbcif(int n);
int main()
{
   int br=0,i;
   for (i=1000; i<=9999; i++)
{
    if (i%zbcif(i)==0)
    {
      cout<<"Brojot "<<i<<" go zadovoluvauslovot\n";
      br++;
    }
   }
   cout<<endl;
cout<<"Pronajdeni se "<<br<<" broevi koi go zadovoluvaatuslovot\n";

return 0;
}

int zbcif(int n)
{
 int zbir;
 zbir=(n%100)+(n/100);
 return zbir;
}
```