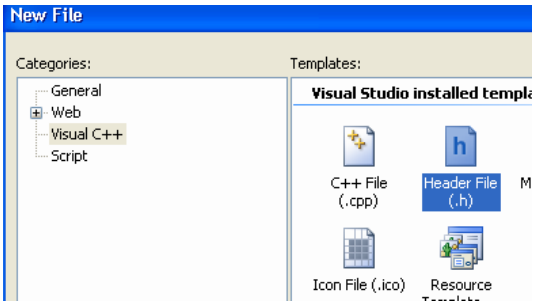
Zadaci – 10b. dio, *Funkcije (treći dio)*

Zadatak 109:

Napravite sljedeće funkcije koje nisu `void` (napravite funkcije sa povratnom vrijednošću):

- **`min`**: čiji će izlaz biti manji broj od dva ulazna broja
- **`max`**: čiji će izlaz biti veći broj od dva ulazna broja
- **`fakt`**: čiji će izlaz biti izračunati faktorijel
- **`prost`**: koja će vraćati vrijednost 1 (`true`) ako je njezin parametar prost broj, ako nije prost onda vraća vrijednost `0` (`false`)
- **`sumakvadrata`**: koja vraća sumu kvadrata od `u1` do `u2`

Zadatak riješite tako što ćete svaku funkciju pojedinačno implementirati, a zatim je testirati.
Testiranu funkciju snimite u jedan fajl (prototip i definiciju funkcije). Najjednostavnije vam je da u *Visual Studio*-u otvorite novi fajl (ne projekat) na sljedeći način:



File –> New File i odaberite *Header File*, i u njega prebacite (*cut - paste*) gotovu funkciju.

Premještanje implementacija funkcija možete činiti, također, pomoću nekog *editora*. (*Notepad* i sl.)

Svaku gotovu funkciju testirajte pojedinačno - jednim jednostavnim pozivom kao npr.:

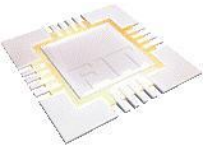
```
1:   void main()
2:   {
3:         int a, b;
4:         a = prost(17);
5:         b = prost(15);
6:         cout << a << endl;
7:         cout << b << endl;
8:   }
```

Ako ste ispravno implementirali funkciju `prost`, u liniji 4 bi se trebala ispisati vrijednost 1 (jer je 17 prost broj) a u liniji 5 vrijednost 0 (jer 15 nije prost broj).

Primjer za testiranje funkcije **`max`** i **`min`**:

```
1:   void main()
2:   {
3:         float x = max(2.5, 2.6);
4:         cout << x << endl;    // ispis: 2.6
5:   }
```

```
1:   void main()
2:   {
3:         float x = min(2.5, 2.6);
4:         cout << x << endl;    // ispis: 2.5
5:   }
```



Primjer za testiranje funkcije **`fakt`**:

```
1:   void main()
2:   {
3:           int x;
4:           x = fakt(5);
5:           cout << x << endl;
```

Ovaj program bi trebao ispisati vrijednost 120 - znamo da je:

$$5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 6 \cdot 20 = 120$$

Pomoć za funkciju `fakt`*:*

Pomoć 1:
Ovaj program je dosta sličan programu koji računa sumu brojeva od `1` do `n`

Pomoć 2:
U *for*-petlji za računanje sume morali ste ponavljati naredbu `s=s+i`, dok u petlji za računanje faktorijela morate ponavljati naredbu `f=f*i`.
Razmislite o slijedećem važnom pitanju za ovaj zadatka: Koja je početna vrijednost varijable `f`?

Pomoć 3:
Početna vrijednost varijable `f` nije nula, jer proizvod broja 0 i bilo kog drugog broja predstavlja opet nulu.

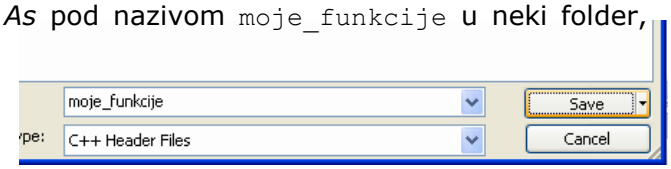
Primjer za testiranje funkcije **`sumakvadrata`**:

```
1:   void main()
2:   {
3:           int x;
4:           x = sumakvadrata(2,6);
5:           cout << x << endl;
6:   }
```

Ovaj program bi trebao ispisati vrijednost 90 - znamo da je:

$$2^2 + 3^2 + 4^2 + 5^2 + 6^2 = 4 + 9 + 16 + 25 + 36 = 90$$

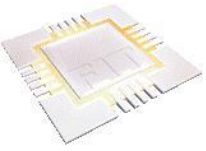
Nakon što ste iskopirali ispravno testirane definicije i prototipove funkcija u novi dokument, snimite fajl sa opcijom *Save As* pod nazivom `moje_funkcije` u neki folder, kao npr: `c:\temp`



Nakon što ste snimili fajl `moje_funkcije.h` zaboravite kako funkcioniraju funkcije. Nemojte se time više zamarati! Od sada vam nije više važno kako funkcije rade, nego šta funkcije rade. Morate da znate imena funkcija i koliko parametara prosljeđujete.
Rješenje ovog zadatka, tj. sadržaj fajla `moje_funkcije.h` (prototipovi i definicije funkcija) se nalazi na stranici 150.

Od sada ćemo koristiti funkcije, koje smo prethodno definisali, kao gotove funkcije.

Počnite novi program. Dodajte novu naredbu za uključivanje biblioteke `moje_funkcije.h`:



```
1:   #include <iostream>
2:   #include <math.h>
3:   using namespace std;
4:   #include <c:\temp\moje_funkcije.h>
5:
6:   void main()
7:   {
8:         int x;
9:         x = sumakvadrata(2,6);
10:        cout << x << endl;
11:  }
```

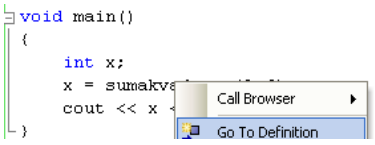
Naredba za uključivanje biblioteke `moje_funkcije.h` bi se trebala nalaziti ispod naredbe `using namespace std`, inače ne bismo mogli koristiti naredbe koje nudi biblioteka `iostream` (kao npr. `cin`, `cout`, `system`, …), jer u fajlu `moje_funkcije.h` nismo uključili `iostream` i `using namespace std`. Mada je to nebitno u ovoj lekciji, a ipak je bolje da biblioteku `moje_funkcije.h` uključimo kao zadnju.

U slučaju da smo u biblioteci `moje_funkcije.h` koristili funkciju `sqrt`, naredba za uključivanje biblioteke `math.h` bi se, također, trebala nalaziti ispred naredbe za uključivanje `moje_funkcije.h`. Prilikom pokretanja ovog programa linija br. 4 će se zamijeniti sa sadržajem fajla `moje_funkcije.h`.

Ako ne želite svaki put unositi čitavu putanju (*path*) do fajla `moje_funkcije.h`, onda snimite fajl `moje_funkcije.h` u folder u kojem se nalaze biblioteke (header fajlovi) vašeg C++ kompajlera. Taj folder bi treba biti:

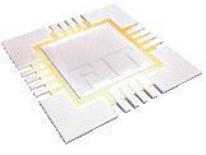
- za Visual Studio 6 :
  `'C:\Program Files\Microsoft Visual Studio\VC98\`**`Include`**`'`
- za Visual Studio 2003 .NET :
  `'C:\Program Files\Microsoft Visual Studio .NET 2003\VC7\`**`include`**`'`
- za Visual Studio 2005 :
  `'C:\Program Files\Microsoft Visual Studio 8\VC\`**`include`**`'`
- za Visual Studio 2008 :
  `'C:\Program Files\Microsoft Visual Studio 9\VC\`**`include`**`'`
- za Borland C++ 4.5 (napomena: ime fajla može biti maksimalne dužine do 8 karaktera):
  `'C:\BC45\INCLUDE'`

Header fajl možete otvoriti sa desnim klikom na ime fajla (u naredbi `#include`) i odabirom opcije *Open dokument* ili sa desnim klikom na ime neke funkcije i odabirom opcije *Go To Definition* ili *Go To Declaration*.



Ako ste snimili header fajl u folder `Include` onda uključivanje tog header fajla možete učiniti i bez navođenja pune putanje:

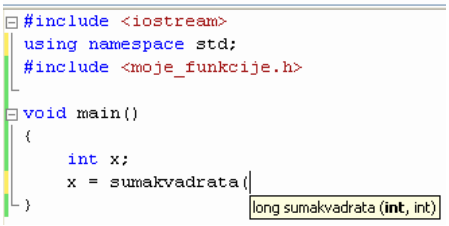
```
#include <moje_funkcije.h>
```



Naredbu `#include <math.h>` možete prebaciti u fajl `moje_funkcije.h`, tako da možete koristit matematičke funkcije (u fajlu `moje_funkcije.h` i u glavnom programu) bez dodavanja te naredbe u glavnom programu.

Isprobajte sljedeće:

Upišite ime neke funkcije i otvorite zagradu. Prikazat će vam se prototip funkcije, tako da možete vidjeti koliko i koje vrste parametara od vas očekuje funkcija. Pogledajte sliku.



Zadatak 110:

Napravite program koji će u funkciji `main` zahtijevati unos dvije vrijednosti `m` i `n` od korisnika. Funkcija `main` treba da proslijedi parametre `m` i `n` novoj funkciji `nad` koja treba izračunati vrijednost $\binom{m}{n}$ (čita se: *m* nad *n*) i vratiti tu vrijednost. U funkciji `main` pridružite izlaznu vrijednost ispisu `cout`. Koristite svoju gotovu funkciju `fakt`.

Formula koju biste trebali znati: $\binom{m}{n} = \frac{m!}{n!(m-n)!}$.

Riješen C++ kôd se nalaze na stranici 151.

Nakon testiranja funkcije `nad`, dodajte prototip i definiciju u *header* fajl `moje_funkcije.h`.

Pomoću ove funkcije možete, također, računati broj kombinacija **n**-te klase od **m** elemenata (ili obrnuto).

Zadatak 111:

Program iz prethodnog zadatka br. 131 je malo nepouzdan. Da bi program ispravno radio, korisnik mora da unese drugi broj koji je veći od prvog, inače bi program neispravno radio. Ovo bi mogli poboljšati kada bi vršili provjeru prilikom poziva funkcije `nad`. U slučaju da je prvi broj manji od drugog pozvali bi funkciju `nad` sa parametrima `n` i `m`, umjesto `m` i `n`.

Umjesto *if-else*-iskaza za provjeru, možemo koristiti naše gotove funkcije `min` i `max`:

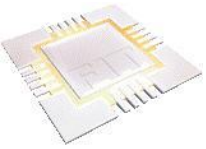
```
17:  ...
18:          int veci, manji;
19:          veci = max(m,n);
20:          manji = min(m,n);
21:
22:          cout << nad(veci, manji) << endl;
23:  }
```

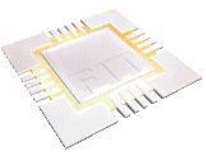
Da ne bi svaki put kad želimo koristiti funkciju `nad` morali dodavati funkcije `max` i `min`, bolje bi bilo da te funkcije dodamo u definiciju funkcije `nad`:

…Smjestite sljedeće naredbe (ako su ispravne) negdje u funkciji `nad` i isprobajte funkciju.

```
m = max(m, n);
n = min(m, n);
```

Nakon što se dodali naredbe, pročitajte rješenje na stranici na kraju dokumenta.





Zadatak 112:

Napravite program u kojem ćete u funkciji `main` zahtijevati od korisnika unos dvije vrijednosti, `m` za početak niza i `n` za kraj niza. Funkcija `main` treba proslijediti `m` i `n` novoj funkciji `svi_prosti` koja će ispisati sve proste brojeve koji se nalaze u tom nizu. Koristite gotovu funkciju `prost`.

Rješenje se nalaze na kraju dokumenta.

Pomoć:
- u funkciji `svi_prosti` dodajte *for*-petlju sa varijablom `i` koja će ponavljati naredbe:

  - vrijednost (izlaz) funkcije `prost` za parametar `i` pridružite varijabli `x` (koja je tipa *int* ili *bool*)
  - ako je vrijednost `x` različita od nule (tj. ako je broj `i` prost) ispišite broj `i` na ekran

Pokušajte riješiti ovaj program i bez dodatne varijable `x`.

Zadatak 113:

Napravite funkciju `brojac_prostih` sa dva ulazna parametra, čija će povratna vrijednost biti vrijednost koja predstavlja broj prostih brojeva od `u1` do `u2`, tj. izbrojane proste brojeve u rasponu od `u1` do `u2`.
Napravite test program za funkciju.

Rješenje se nalaze na kraju dokumenta.

Pomoć:
- prototip funkcije glasi: `int brojac_prostih(int, int);`

Zadatak 114:

a) Napravite funkciju `suma_faktorijela` sa dva ulazna parametra, čija će povratna vrijednost biti suma faktorijela za brojeve iz raspona od `u1` do `u2`.
Napravite test program za funkciju.

Rješenje se nalaze na kraju dokumenta.

b) Nakon što ste to odradili, prepravite funkciju `sumafakt` tako da ta funkcija može računati sumu ako korisnik unese prvi broj veći od drugog, a u slučaju da korisnik unese, kao početak ili kraj niza, negativan broj funkcija `sumafakt` treba vratiti vrijednost **0**. Koristite vaše gotove funkcije `min` i `max`.

Rješenje se nalaze na kraju dokumenta.

Zadatak 115:

Napravite program koji će za pet unesenih brojeva ispisati najmanji, koristeći samo gotovu funkciju `min` bez upotrebe *if-(else)*-iskaza.

Rješenje se nalaze na kraju dokumenta.

Funkcije: `call by reference`

Naredni zadatak može poslužiti kao uvod u ovu lekciju.

Zadatak 116:

Napravite dijagram toka, i na osnovu njega napišite C++ kôd za sljedeći zadatak:

Napravite funkciju `void ispisi_rjesenja(float, float, float)` koja na osnovu tri ulazna parametra `a`, `b`, `c` treba ispisati na ekran rješenja (x_1 i x_2) kvadratne funkcije ($ax^2+bx+c=0$). U funkciji `main` zahtijevati od korisnika unos članova kvadratne jednačine (`a`, `b`, `c`).

U slučaju da kvadratna jednačina postane linearna (za $a=0$), nemojte ispisati rješenja.

Riješen C++ kôd (bez dijagrama toka) se nalaze na kraju dokumenta

Pomoć:

- funkciji izračunajte vrijednost diskriminante $D = b^2 - 4ac$.
- na osnovu vrijednosti D i vrijednosti a možete zaključiti da li kvadratna jednačina ima rješenja:

<table>
<tr><th colspan="2">$D \geq 0$</th><th>$D < 0$</th></tr>
<tr><th>$a \neq 0$</th><th>$a = 0$</th><td rowspan="2">Nema realnih rješenja</td></tr>
<tr><td>Rješenja kvadratne jednčine:
$x_1 = \frac{-b+\sqrt{D}}{2a}$
$x_2 = \frac{-b-\sqrt{D}}{2a}$</td><td>Postaje linearna jednačina:
$bx + c = 0 \Rightarrow x = -\frac{c}{b}$
ali u ovom slučaju ćemo ispisati da nema rješenja</td></tr>
</table>

- uslov za računanje rješenja `x1` i `x2` glasi: `(D>=0 && a!=0)`
- testirajte program, unesite barem jednu negativnu vrijednost:
- za unesene vrijednosti **1**,**3**,**2** program treba ispisati:

```
x1 = 2
x2 = 1
```

- za unesene vrijednosti 0,8,2 i 1,2,3 program treba ispisati:

```
Greska: Nije moguce izracuanti rjesenja
```

Zadatak 117:

Kad bi ste željeli napraviti funkciju sa povratnom vrijednošću za računanje rješenja kvadratne jednačine (samo računanje a ne ispisivanje) možda bi ste imali jedan problem o implementacije. Problem je u tome što kvadratna jednačina ima dva rješenja. Funkcije u C++ mogu imati samo jednu povratnu vrijednost.

U tom slučaju mogli bi ste napraviti dvije slične funkcije, jednu koja računa rješenje kvadratne jednačine ***x1*** a drugu za ***x2***.

Napravite program za računanje rješenja kvadratne jednačine, tako da koristite funkcije `fx1` i `fx2` (funkcija `fx1` računa ***x1***, a funkcija `fx2` računa ***x2***).

Funkcije pozovite na sljedeći način:

```
1:   void main()
2:   {
3:          float am, bm, cm;
4:
5:          cout << "Unesite clanove a, b, c " << endl;
6:          cin >> am;
7:          cin >> bm;
8:          cin >> cm;
9:
10:         float x1m, x2m;
11:         x1m = fx1(am, bm, cm);
12:         cout << "Rjesenje kvadratne jednacine x1: " << x1m << endl;
13:
14:         x2m = fx2(am, bm, cm);
15:         cout << "Rjesenje kvadratne jednacine x2: " << x2m << endl;
16:  }
```

Ove funkcije ne bi trebale ispisivati rješenja na ekran, nego rješenja trebaju biti izlaz iz funkcije, a funkcija `main` bi trebala ispisivati rješenja kvadratne jednačine ***x1*** i ***x2***, tj. varijable `x1m` i `x2m`. Ovdje nam se javlja još jedan matematički problem: Šta ako jednačina nema rješenja? (npr. za vrijednosti 0,8,2 **i** 1,2,3)

Kako će nam funkcija koja ne ispisuje ništa, koja samo vraća gotovu vrijednost (broj), javiti da ne postoji rješenje. Taj problem ne možemo lahko riješiti pa ćemo se dogovoriti, u slučaju da ne postoje rješenja, da funkcija ispiše poruku o grešci i vrati vrijednost 0.

Rješenje se nalaze na kraju dokumenta.

U ovom primjeru smo koristili dvije funkcije sa po jednom izlaznom (povratnom) vrijednošću. Dvije izlazne vrijednosti iz jedne funkcije bi mogli postići pomoću globalnih varijabli. U našem slučaju bi koristili dvije ***globalne*** varijable `x1` i `x2` i jednu funkciju tipa ***void*** koja bi pomoću primljenih parametara `a`, `b`, `c` odnosno **a**`m`, **b**`m`, **c**`m` izračunala (izmijenila) globalne varijable `x1` i `x2`. To možete uraditi samostalno za vježbu.

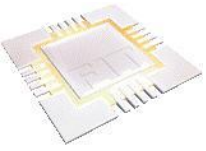
Kako smo već dosta puta rekli upotreba globalnih varijabli nije preporučljiva, stoga ćemo ovaj problem riješiti pomoću varijabli koje će biti ***referenca***. Deklarisat ćemo dodatne dvije lokalne varijable `x1m` i `x2m`, koje možemo proslijediti kao aktuelne parametre funkciji `fx` zajedno sa varijablama **a**`m`, **b**`m`, **c**`m`.

Funkcija `fx` imati pet ulaznih parametara. Prva tri parametra, koja predstavljaju članove kvadratne jednačine, će biti tipa *float*, dok ćemo parametre `x1m` i `x2m` (iz funkcije `main`) primati u funkciji `fx` pod imenom `x1` i `x2`. Ta dva parametra će biti tipa ***referenca*** na *float*.

Šta, inače, predstavlja referenca? Do sada smo parametre koristili za prenos vrijednosti u funkcije (*pass-by-value*). To znači, da se pri pozivu neke funkcije, vrijednosti aktuelnih parametara (**a**`m`, **b**`m`, **c**`m`) **kopiraju** u formalne parametre (varijable `a`, `b`, `c`). U ovom načinu poziva funkcije (*pass-by-value*) bilo koja promjena varijabli `a`, `b`, `c` u pozvanoj funkciji `fx` neće uticati na varijable **a**`m`, **b**`m`, **c**`m` funkcije `main`.

U prenosu parametara po referenci (*pass-by-reference*), vrijednosti aktuelnih parametara (**a**`m`, **b**`m`, **c**`m`) se **ne kopiraju** u formalne parametre (`a`, `b`, `c`), već se formalnim parametrima (varijable `a`, `b`, `c`) dodjeljuje zajednički memorijski prostor u RAM-u kao aktuelnim parametrima (varijablama **a**`m`, **b**`m`, **c**`m`). Pošto varijable `x1` i `x1m`, odnosno `x2` i `x2m` dijele zajednički memorijski prostor u RAM-u za 'pamćenje' njihovih vrijednosti, onda bilo koja promjena vrijednosti varijable `x1` u funkciji `fx` će uticati na promjenu vrijednost varijable `x1m` u funkciji `main`. Isto vrijedi i za par `x2` i `x2m`.

Aktuelni parametri, koji se prosljeđuju kao referenca, moraju biti varijable, ne smiju konstante, niti brojevi (npr. `7...`).



Kod poziva funkcije po referenci, aktuelni parametri i formalni parametri moraju biti varijable istog tipa (npr. ako je `x1m` tipa *float*, onda i `x1` mora biti *float*, a ne smije biti *long int*, *int* niti *double*).

Deklaracija formalnih parametara kao referenca

Formalne parametre (`x1` i `x2`) ćemo deklarirati kao referencu dodavanjem znaka ampersand '**&**' iza tipa varijable. Slijedi primjer deklaracije reference u funkciji `fx`.

```
     //       1-ulaz   2-ulaz   3-ulaz   4-izlaz    5-izlaz
1:   void fx(float a, float b, float c, float &x1, float &x2)
2:   {
3:   ...
```

referentni parametri se nazivaju i **izlazni**

Zadatak 118:

Prepravite prethodni program koristeći funkciju `fx`:

```
void fx(float, float, float, float &, float &);  // ← ovo je samo prototip
```

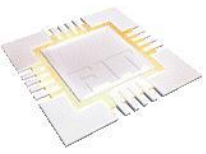
Rješenje se nalaze na kraju dokumenta.

Zadatak 119:

Napravite prototip i definiciju funkcije `uvecaj_r` i testirajte je u sljedećem programu:

```
1:   void main()
2:   {
3:         int x;
4:         x = 5;
5:
6:         uvecaj_r(x);        // vrijednost varijable x će se povećati sa 5 na 6
7:         uvecaj_r(x);        // vrijednost varijable x će se povećati sa 6 na 7
8:
9:         cout << x << endl;  // ispisat će se vrijednost 7
10:  }
```

Rješenje se nalaze na kraju dokumenta.



Zadatak 120:

Napravite program koji će u funkciji `main` zahtijevati od korisnika unos dva broja. Napravite funkciju `racunaj` koja će primiti 5 parametara. Prva dva parametra će biti ulazne vrijednost (pass-by-value), a ostala 3 će biti reference tipa *float* (pass-by-reference) – izlazna vrijednost. Funkcija `racunaj` treba izvršiti matematičke operacije (sabiranje, oduzimanje, množenje) i rezultat tih operacija treba smjestiti u zadnja tri parametra funkcije (*pass-by-reference*).

Rješenje se nalaze na kraju dokumenta.

Zadatak 121:

Napravite funkcije **`ff`** (za ***f****loat*) i **`fi`** (za ***i****nteger*), koje će ulaznom parametru ukloniti predznak.

Napravite test program za funkcije.

Rješenje se nalaze na kraju dokumenta.

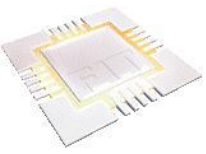
Pomoć:
Cilj ovih funkcija je da aktualni parametar nakon poziva uvijek bude pozitivan broj.

```
1:   void main()
2:   {
3:           int x;
4:           cout << "Unesi broj: ";
5:           cin >> x;
6:
7:           fi(x);        // uklanja predznak varijable x
8:
9:           cout << x << endl;  // vrijednost x ce uvijek biti pozitivna
10:  }
```

Rješenja

Rješenje zadatka br. **109**:

```
1:   //moje_funkcije.h
2:   double min(double, double);
3:   double max(double, double);
4:   long int fakt(int);
5:   int prost(int);
6:   long int sumakvadrata(int, int);
7:
8:   double min(double a, double b)
9:   {
10:         if (a<b)
11:               return a;
12:         else
13:               return b;
14:  }
15:
16:  double max(double a, double b)
17:  {
18:         if (a>b)
19:               return a;
20:         else
21:               return b;
22:  }
23:
24:  long int fakt(int n)     // u Visual Studio-u je nebitno da li koristite long int
25:  {                         // ili int, jer je njihova velicina u oba slucaja po 4 bajta
26:         long int f;
27:         f=1;              // pocetna vrijednost je 1 (zbog mnozenja)
28:
29:         for (int i=1; i<=n; i++)
30:               f=f*i;
31:
32:         return f;
33:  }
34:
35:  bool prost(int n)    // U Borland C++u nepostoji tip podatka bool, pa cete morati
36:  {                    // koristiti povratni tip int
37:         int brojac;
38:         brojac=0;
39:
40:         for (int i=2; i<=n/2; i++)
41:         {
42:               if (n % i == 0)
43:                     brojac++;
44:         }
45:
46:         if (brojac==0)
47:               return 1;  // ili return true;
48:         else
49:               return 0;  // ili return false;
50:  }
51:
52:  long int sumakvadrata(int m, int n)
53:  {
54:         long int suma;
55:         suma=0;
56:
57:         for (int i=m; i<=n; i++)
58:               suma = suma + i*i;
59:         return suma;
60   }
```



Rješenje zadatka br. **110**:

```
1:   #include <iostream>
2:   using namespace std;
3:   #include <moje_funkcije.h>
4:
5:   int nad(int m, int n)
6:   {
7:          int izlaz;
8:          izlaz = fakt(m) / ( fakt(n)*fakt(m-n) );
9:          return izlaz;
10:  }
11:
12:  void main()
13:  {
14:         int m, n;
15:         cout << "Unesi dva broja (drugi ne smije biti veci od prvog) \n";
16:         cin >> m >> n;
17:
18:         cout << nad(m,n) << endl;
19:  }
```

Iz ovog se vidi, da funkcija **`nad`** poziva funkciju **`fakt`** tri puta za tri različita parametra. Ovdje smo, također, koristili iste nazive varijabli (**`m`**, **`n`**) u funkciji **`main`** i u funkciju **`nad`** (možda i u funkciji **`fakt`**). Nazivi varijabli se smiju ponavljati u različitim funkcijama.

Testirajte program sa vrijednostima *6* i *2*: $\binom{6}{2}=\frac{6!}{2!(6-2)!}=\frac{(1\cdot 2\cdot 3\cdot 4)\cdot 5\cdot 6}{2!\cdot(1\cdot 2\cdot 3\cdot 4)}=\frac{5\cdot 6}{2!}=15$

Rješenje zadatka br. **111**:

Ako ste dodati naredbe …

```
1:   int nad(int m, int n)
2:   {
3:          m = max(m, n);
4:          n = min(m, n);
5:   ...
```

...onda vam funkcija nad neće raditi ispravno. Kad bi vrijedilo da je *n>m*, u liniji br. 3 bi se varijabli `m` dodijelila vrijednost `n`, jer je *n>m*. Obje varijable bi imale istu vrijednost (*m=n*). Tako bi i nakon linije br. 4 varijabla `n` imala istu vrijednost kao `m`.

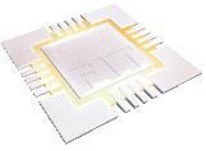
Slijedi ispravan primjer:

```
1:   int nad(int u1, int u2)
2:   {
3:          int m,n;
4:          m = max(u1, u2);
5:          n = min(u1, u2);
6:   ...
```

Rješenje zadatka br. **112** *(bez dodatne varijable `x`)*:

```
1:   void svi_prosti(int u1, int u2)
2:   {
3:          for (int i=u1; i<=u2; i++)
4:          {
5:                 if (prost(i))
6:                        cout << i << endl;
7:          }
8:   }
```

```
void svi_prosti(int u1, int u2)
{
       for (int i=u1; i<=u2; i++)
       {
              if (prost(i) != 0)
                     cout << i << endl;
       }
}
```

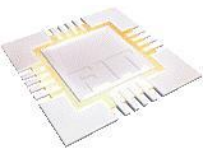


```
9:
10:  void main()
11:  {
12:         int m, n;
13:
14:         cout << "Unesi pocetak i kraj niza: \n";
15:         cin >> m >> n;
16:
17:         svi_prosti(m,n);
18:  }
```

Rješenje zadatka br. **113**:

```
1:   #include <iostream>
2:   using namespace std;
3:   #include <moje_funkcije.h>
4:
5:   int brojac_prostih(int u1, int u2)
6:   {
7:         int brojac;
8:         brojac=0;
9:
10:        for (int i=u1; i<=u2; i++)
11:        {
12:              if (prost(i))
13:              {                                  // ispis je moguc i u funkcijama koje
14:                    //cout << i << endl;    // imaju povratnu vrijednost
15:
16:                    brojac++;
17:              }
18:        }
19:        return brojac;
20:  }
21:
22:  void main()
23:  {
24:        int m, n;
25:        cout << "Unesi pocetak i kraj niza: \n";
26:        cin >> m >> n;
27:
28:        int ukupno;
29:        ukupno = brojac_prostih(m, n); // funkcija ima povratnu vrijednost
30:                                       // povratna vrijednost se pridruzuje varijabli ukupno
31:        cout << "Ukupno: "<< ukupno << endl;
32:  }
```

Rješenje zadatka br. **114** *(a)*:



```
1:   #include <iostream>
2:   using namespace std;
3:   #include <moje_funkcije.h>
4:
5:   int suma_faktorijela (int u1, int u2)  // u Borlandu obavezno koristite long int
6:   {                                      // kao povratnu vrijednost jer se ovdje
7:         int s = 0;                       // radi o velikim vrijednostima
8:
9:         for (int i=u1; i<=u2; i++)
10:        {
11:              s = s + fakt(i);
12:        }
13:        return s;
14:  }
15:
16:  void main()
17:  {
18:        int m, n;
19:        cout << "Unesi pocetak i kraj niza: \n";
20:        cin >> m >> n;
21:
22:        int suma;
23:        suma = suma_faktorijela(m, n);
24:        cout << "Suma: "<< suma << endl;
25:  }
```

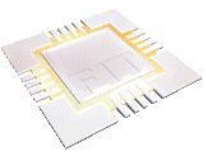
Rješenje zadatka br. **114** *(b)*:

```
1:   int suma_faktorijela (int u1, int u2)
2:   {
3:         if (u1 == 0 || u2 == 0)
4:               return 0;  // izlaz iz funkcije, ostatak naredbi se nece izvrsiti
5:
6:         int s = 0, m, n;
7:         m = min(u1, u2);
8:         n = max(u1, u2);
9:
10:        for (int i=m; i<=n; i++)
11:        {
12:              s = s + fakt(i);
13:        }
14:        return s;
15:  }
```

Samo da znate, i ovo je moguće:

```
1:   int suma_faktorijela (int u1, int u2)
2:   {
3:         if (u1 == 0 || u2 == 0)
4:               return 0;
5:
6:         int s = 0;
7:
8:         for (int i=min(u1, u2); i<=max(u1, u2); i++)
9:         {
10:              s = s + fakt(i);
11:        }
12:        return s;
13:  }
```

Rješenje zadatka br. **115**:



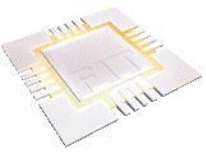
```
1:    #include <iostream>
2:    using namespace std;
3:    #include <moje_funkcije.h>
4:
5:    void main()
6:    {
7:          int a1, a2, a3, a4, a5;
8:          cout << "Unesi 5 brojeva \n";
9:          cin >> a1 >> a2 >> a3 >> a4 >> a5;
10:
11:         cout << "====\n";
12:
13:         cout << min( min( min(a1,a2), min(a3,a4)), a5) << endl;     // ili
14:
15:         cout << min(min(min(min(a4, a5), a3), a2), a1) << endl;     // ili
16:
17:         cout << min(a1, min(a2, min(a3, min(a4, a5)))) << endl;
18:   }
```

Rješenje zadatka br. **116**:

```
1:   #include <iostream>
2:   #include <math.h>
3:   using namespace std;
4:
5:   void ispisi_rjesenja(float, float, float); // prototip
6:
7:   void ispisi_rjesenja(float a, float b, float c)
8:   {
9:          float D = b*b - 4*a*c;
10:
11:         if (D>=0 && a!=0)
12:         {
13:                float x1, x2;
14:
15:                x1 = (b+sqrt(D)) / (2*a); // nemojte zaboraviti dodatne zagrade zbog nazivnika
16:                      x2 = (b-sqrt(D)) / (2*a);
17:
18:                cout << " x1 = " << x1 << endl;
19:                cout << " x2 = " << x2 << endl;
20:         }
21:         else
22:                cout << "Greska: Nije moguce izracuanti rjesenja \n";
23:  }
24:
25:  void main()
26:  {
27:         float am, bm, cm;  // ovdje su koristeni drugaciji nazivi varijabli nego u fx
28:                            // zbog lakseg objasnjenja koji slijedi poslije
29:         cout << "Unesite clanove a, b, c " << endl;
30:         cin >> am;                        // sufix m može asocirati na funkciju main
31:         cin >> bm;
32:         cin >> cm;
33:
34:         ispisi_rjesenja(am, bm, cm);
35:  }
```

Rješenje zadatka br. **117**:

```
1:   float fx1(float a, float b, float c)
2:   {
3:          float D = b*b - 4*a*c;
4:          float izlaz;
5:
6:          if (D>=0 && a!=0)
7:          {
8:                 izlaz = (b+sqrt(D)) / (2*a);
9:          }
10:         else
11:         {
12:                cout << "Greska: Nije moguce izracuanti rjesenja \n";
13:                izlaz = 0;
14:         }
15:
16:         return izlaz;
17:  }
18:
19:  float fx2(float a, float b, float c)
20:  {
21:  ...
```



Rješenje zadatka br. **118**:

```
1:   #include <iostream>
2:   #include <math.h>
3:   using namespace std;
4:
5:   void fx(float a, float b, float c, float &x1, float &x2)
6:   {
7:          float D = b*b - 4*a*c;
8:
9:          if (D>=0 && a!=0)
10:         {
11:                x1 = (b+sqrt(D)) / (2*a);
12:                x2 = (b-sqrt(D)) / (2*a);
13:         }
14:         else
15:         {
16:                cout << "Greska: Nije moguce izracuanti rjesenja \n";
17:                x1 = 0;
18:                x2 = 0;
19:         }
20:  }
21:
22:  void main()
23:  {
24:         float am, bm, cm;
25:         cout << "Unesite clanove a, b, c " << endl;
26:         cin >> am;
27:         cin >> bm;
28:         cin >> cm;
29:
30:         float x1m, x2m;
31:         fx(am, bm, cm, x1m, x2m); //funkcija je tipa void
32:
33:         cout << "Rjesenje kvadratne jednacine x1: " << x1m << endl;
34:         cout << "Rjesenje kvadratne jednacine x2: " << x2m << endl;
35:  }
```

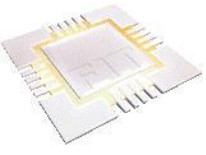
Rješenje zadatka br. **119**:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   void uvecaj_r(int &a)
5:   {
6:          a = a + 1;
7:   }
8:
9:   void main()
10:  {
11:         int x;
12:         x = 5;
13:
14:         uvecaj_r(x);
15:         uvecaj_r(x);
16:
17:         cout << x << endl;
18:  }
```

Pazite funkcija `uvecaj_r` je tipa *void* pa se ne smije pridružiti ispisu naredbi `cout`:

```
cout << uvecaj_r(x); //GREŠKA
```

...



Rješenje zadatka br. **120**:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   void racunaj(float a, float b, float &r1, float &r2, float &r3)
5:   {
6:          r1 = a+b;
7:          r2 = a-b;
8:          r3 = a*b;
9:   }
10:
11:  void main()
12:  {
13:         float x, y, mm1, mm2, mm3;
14:
15:         cout << "Unesi dva broja \n";
16:         cin >> x >> y;
17:
18:         racunaj(x,y, mm1, mm2, mm3);
19:
20:         cout << "============================== \n";
21:         cout << "Rezultat sabiranja:  " << mm1 << endl;
22:         cout << "Rezultat oduzimanja: " << mm2 << endl;
23:         cout << "Rezultat mnozenja:   " << mm3 << endl;
24:  }
```

Rješenje zadatka br. **121**:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   void fi(int &r)
5:   {
6:          if (r<0)    // ako je r negativno
7:                r=-r;    // neka promjeni predznak
8:   }
9:
10:  void fd(float &r)
11:  {
12:         if (r<0)
13:               r=-r;
14:  }
15:
16:  void main()
17:  {
18:         float d;
19:         int i;
20:
21:         cout << "Unesite cijeli broj: ";
22:         cin >> i;
23:         fi(i);
24:         cout << i << endl << endl;
25:
26:         cout << "Unesite decimalni broj: ";
27:         cin >> d;
28:         fd(d);
29:         cout << d << endl;
30:  }
```