Prototip funkcije

Većina programera definiciju funkcije piše nakon funkcije main, a prije funkcije main navodi prototip funkcije. **Prototip** funkcije je naredba koja sadrži iste elemente kao i zaglavlje funkcije, s tim što se, umjesto tijela funkcije, iza zaglavlja navodi znak „**;**“ umjesto bloka naredbi koje čine tijelo funkcije.
Ukoliko se koristi ovaj stil programiranja, prototip funkcije mora se obavezno navesti prije funkcije main. Ako se to ne uradi, kompajler će prijaviti grešku. U prototipu funkcije nije obavezno navođenje imena argumenata. Lista argumenata prototipa funkcije može sadržavati samo tipove podataka argumenata.
Prototip funkcije naziva se još i **deklaracija**.
Definicija funkcije naziva se i **implementacija funkcije**.
Odvajanje prototipa i implementacije nije obavezno, već je to izbor programera. Ukoliko programer ne želi koristiti ovaj stil pisanja funkcija, onda mora navesti definiciju funkcije prije funkcije main.

Prototip funkcije	
tipPovratneVrijednosti nazivFunkcije(**tip1,tip2,** t**ip2**,..... **tipN**); ili **tipPovratneVrijednosti** nazivFunkcije(**tip1** argument1, **tip2** argument2,..... **tipN** argumentN);	
Elementi prototipa funkcije	
tipPovratneVrijednosti	Određuje kakav podatak će funkcija vratiti
nazivFunkcije	Bilo koje valjan identifikator u C++
(tip1, tip2,..... tipN)	Lista fiktivnih argumenata funkcije. Sadrži tip argumenta. Može biti prazna, tj.možemo definisati funkciju bez argumenata.
(tip1 argument1, tip2 argument2,….tipN argumentN)	Opciono, lista fiktivnih argumenata funkcije sadrži tip i naziv argumenta.

Primjer 1	
Prototip funkcije	**Definicija**

int slucajanBroj();	int slucajanBroj()
	{
	int slucajno = 0;
	slucajno = 1 + rand() % (10 - 1 + 1); return slucajno;
	}

Primjer 2	
Prototip funkcije	**Definicija**

bool paranBroj (int);
ILI
bool paranBroj (int x);
	bool paranBroj (int x)
	{
	if (x%2 == 0) return true; else
	return false;
	}

Primjer 3	
Prototip funkcije	**Definicija**

int sumaN(int);
ILI
int sumaN(int n);
	int sumaN(int n)
	{
	int suma=0; for (int i=1;i<=n; i++)
	suma += i;
	return suma;
	}

Primjer 4	
Prototip funkcije	**Definicija**

int nNaM(int, int);
ILI
int nNaM(int n, int m);
	int nNaM(int n, int m)
	{
	int stepen=1; for (int i=1;i<=m; i++)
	stepen *= n;
	return stepen;
	}

Primjer 5	
Prototip funkcije	**Definicija**
double obim(double, double); ILI double obim(double a, double b);	double obim(double a, double b) { return (2*a+2*b); }

1. Napisati funkciju koja računa površinu pravouglog trougla. U glavnom programu učitati stranice pravouglog trougla a i b, a zatim pozvati funkciju i ispisati površinu trougla.Zadatak riješiti tako što ćete odvojiti deklaraciju(prototip) i definiciju

```
#include <iostream>

float Povrs(float x, float y);          Prototip(deklaracija funkcije)

using namespace std;
int main(int argc, char** argv) {
    float a,b,p;
    cout<<"Unesite dužinu stranice a: ";
    cin>>a;
    cout<<"Unesite dužinu stranice b: ";
    cin>>b;
    p=Povrs(a,b);          Poziv funkcije
    cout<<"Površina trougla je: "<<p<<endl;
    return 0;
}

float Povrs(float x, float y){          Definicija(implementacija) funkcije
    float P;
    P=(x*y)/2;
    return P;
}
```

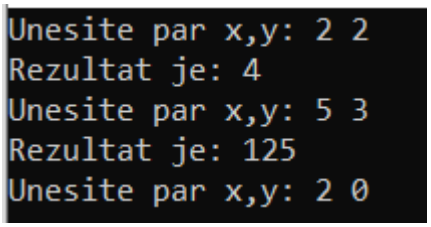
2. Napisati dvije funkcije. Prva funkcija KVADRAT računa kvadrat zadatog broja. Druga funkcija KUB računa kub zadatog broja. U glavnom programu ispisati sve kvadrate i kubove brojeva od 1 do zadatog N, sa korakom 0,1

```
Unesite n: 1
Kvadrati brojeva su:
0  0.01  0.04  0.09  0.16  0.25  0.36  0.49  0.64  0.81
Kubovi brojeva su:
0  0.001  0.008  0.027  0.064  0.125  0.216  0.343  0.512  0.729
```

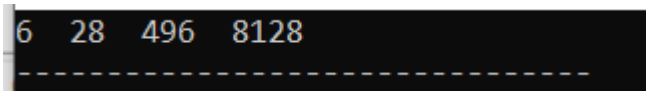
3. U glavnom programu učitati N. Za svaki broj od 1 do zadatog N, ispisati stepen broja 2 Pozivati funkciju Stepen2 za računanje stepena broja 2.(i na 2)

```
Upisite n: 5
0  0
1  1
2  4
3  9
4  16
5  25
```

4. Korisnik učitava parove cijelih brojava x i y tako dugo dok ne upiše obje nule. Nakon svakog učitanog para ispisati “x na y”. Pozivati funkciju Stepen, koja računa “x na y”.



5. Napisati funkciju SavrsenBroj koja provjerava da li je zadati cijeli broj savršen. Broj je savršen ako je zbir njegovih djelitelja (osim samog broja) jednak tom broju.
   Npr. 6 je savršen broj jer je 6 = 1+2+3.
   U glavnom programu ispisati sve savrsene brojeve od 1 do 19999.



6. Napisati funkciju koja ispisuje parne prirodne brojeve od 1 do zadanog N. Testirati funkciju.