Функције

Приликом решавања сложенијих задатака често се дешава да се одређени низови наредби понављају на више места. Да се не би трошило време на стално копирање и измене у случају грешке, такви низови наредби се групишу у самосталне програмске јединице које зовемо функцијама. Оне морају имати јединствено име и могу се позивати више пута ради извршавања одређених акција.

Функције морају имати повратну вредност. Ту вредност враћају приликом позива и она може користити у изразима. Уколико функција не враћа вредност, њен повратни тип је **void.** Свака функција мора бити унутар неке класе. Функције су заправо методи које ћемо креирати на већ постојећим класама које смо и раније помињали.

Ми ћемо у постојеће програме убацивати нове функције које ће наш код учинити разумљивијим и читљивијим. Функције пишемо у следећој форми:

```
povratni_tip ime_funkcije(definicija_parametara)
{
lokalne_promenljive
naredbe
}
```

Дефиниција функције обухвата повратну вредност, име функције, аргументе које функција користи (променљиве), тело методе функције унутар које се налазе наредбе и локалне променљиве. Заглавље функције чине повратна вредност, име функције и аргументи које функција користи. Декларација (прототип) функције садржи само заглавље функције. То је заправо дефиниција функције без тела функције.

```
//Primer deklaracije funkcije
int funkcija (int x, int y, int z);
```

Име функције мора почети словом, а остали карактери могу бити слова, цифре или карактер подвлака. Остали карактери нису дозвољени. Уколико желимо да функција има повратну вредност, морамо нагласити о којој се повратној вредности ради у дефиницији функције и морамо навести наредбу **return** уз вредност или израз који желимо вратити. На следећим примерима објаснићемо делове методе.

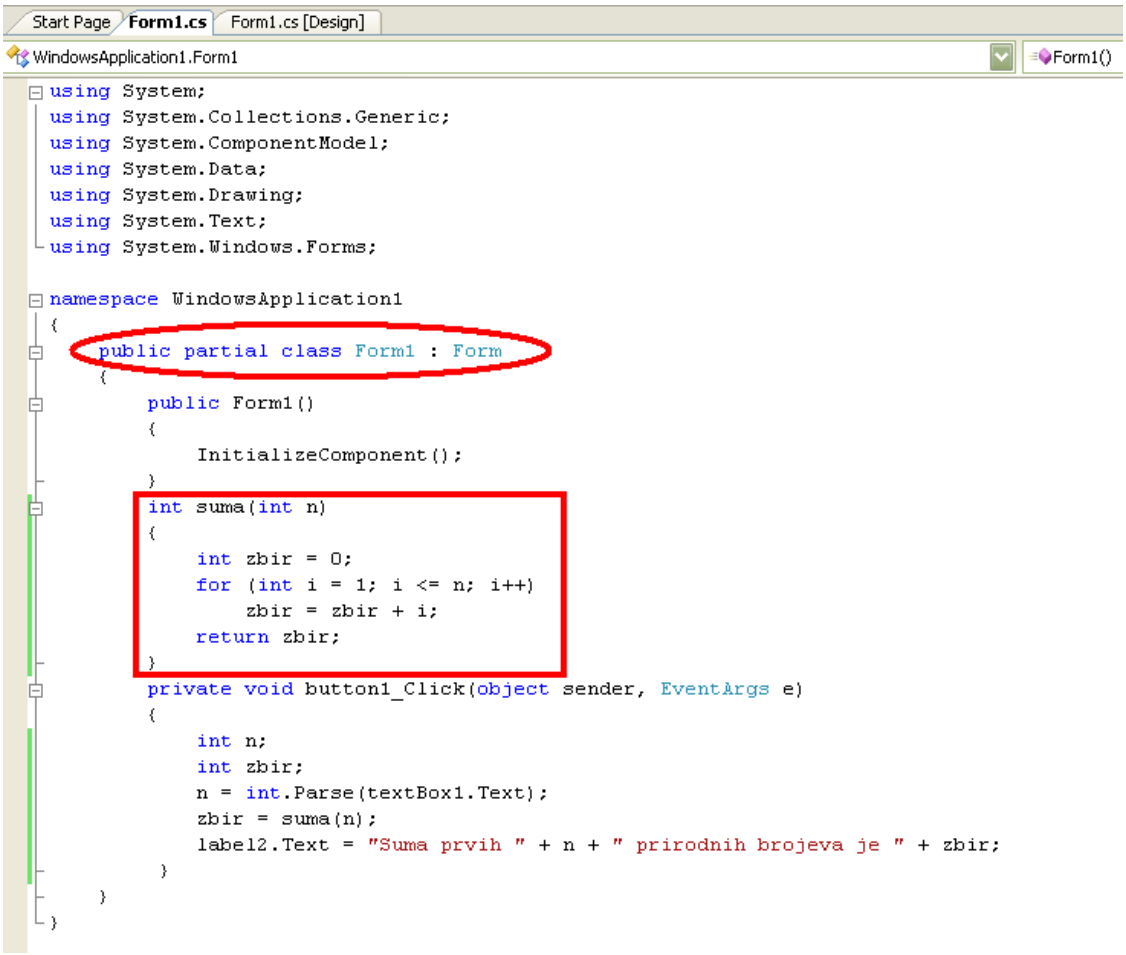
Пример 1. Написати функцију која израчунава суму првих n природних бројева.

```
int suma(int n)
{
    //Dajemo početnu vrednost sumi
    int zbir = 0;
    //Prolazimo kroz for petlju da bismo obuhvatili sve prirodne brojeve od 1 do n
    for (int i = 1; i <= n; i++)
       //Na postojeći zbir dodajemo vrednost narednog broja
        zbir = zbir + i;
    //Naredba return vraća vrednost funkcije
    return zbir;
}
```

Повратна вредност наведене функције је **int**, односно, цео број. Повратна вредност се пише испред имена функције. Затим следи **име функције**, које је у овом случају ***suma***. **Аргументи** које функција узима пишу се у заградама иза имена функције. Те променљиве зову се **класне променљиве**. У претходном примеру, то је **int n**. При томе, **n** је променљива, док је **int** тип те

променљиве. Потом се унутар витичастих заграда пише **код функције**. Све променљиве које се нађу унутар витичастих заграда зовемо **локалним променљивим.**

Код функције која ће се касније позивати у програму пишемо било где унутар класе Form1, али независно од других функција које се налазе у Form1. Form1 је класа која дефинише нашу форму и на којој описујемо акције везане за форму. На овај начин везујемо функцију за већ постојећу класу. На следећој слици је приказано где пишемо код наше функције.



Слика 9.5. Код функције пишемо унутар класе Form1

Пример 2. Написати функцију која израчунава суму квадрата природних бројева између бројева m и n.

```
int suma_kvadrata(int m, int n)
{
    //Dajemo početnu vrednost sumi
    int s = 0;
    //Prolazimo kroz for petlju da bismo obuhvatili sve brojeve između m i n
    for (int i = m; i <= n; i++)
        //Postojećoj sumi dodajemo kvadrat narednog broja
        s = s + i * i;
    //Naredba return vraća vrednost funkcije
    return s;
}
```

Повратна вредност наведене функције је ***int***, односно, цео број. Повратна вредност се пише испред имена функције. Затим следи **име функције**, које је у овом случају ***suma_kvadrata.*** **Аргументи** које функција узима пишу се у заградама иза имена функције. Те променљиве зову се **класне променљиве**. У претходном примеру, то су ***int m, int n***. Потом се унутар витичастих заграда

пише **код функције**. Све промењиве које се нађу унутар витичастих заграда зовемо **локалним променљивим.**

Пример 3. Написати функцију која исписује поруку.

Функција која ће исписивати поруку неће имати повратну вредност, дакле, повратни тип биће ***void***.

```
void ispisi(int x)
{
    //Funkcija prikazuje prozor sa porukom
    MessageBox.Show("Uneli ste broj " + x);
}
```

Формални параметри и тело методе функције

У претходом поглављу смо се упознали са деловима методе функције. Тада смо поменули аргументе које функција користи. Ти аргументи називају се ***формални параметри.*** Они се наводе у заглављу, иза имена функције, унутар заграда. ***Тело методе*** се састоји од сложених наредби које се налазе унутар витичастих заграда. Унутар тела методе се налазе локалне променљиве и наредбе.

Слика 9.6. Формални параметри и тело методе функције

Листа формалних параметара може садржати један или више параметара који се састоје од назива типова промењљивих и имена промењљивих, при чему се два или више аргумената раздвајају зарезима. Такође, може се десити да формални параметри не постоје и тада су обавезне празне заграде приликом дефинисања метода. Формални параметри су по својој природи локални и нису видљиви изван функције.

Проучићемо три различита примера. У првом примеру, функција ће имати само један формални параметар, у другом примеру ће имати више формалних параметара, а у трећем примеру функција неће имати формалних параметара.

Пример 1. Написати функцију која рачуна факторијел броја.

```
//Imamo jedan formalni parametar
int faktorijel(int n)
{
    //Početna vrednost za fakt je 1
    int fakt = 1;
    //Ulazimo u for petlju, moramo proći sve brojeve od 1 do n
    for(int i = 1; i <= n; i++)
        //Množimo prethodni broj sa sledećim
```

```
        fakt = fakt*i;
    //Vraćamo fakt kao vrednost faktorijela broja n
    return fakt;
}
```

Пример 2. Написати функцију која рачуна аритметичку средину четири задата реална броја.

```
//Imamo četiri formalna parametra
float najveci(float a, float b, float c, float d)
{
    //Pomoćna promenljiva
    float s;
    //Pomoćnoj promenljivoj dodeljujemo traženu vrednost
    s = (a + b + c + d)/4;
    //Vraćamo rešenje
    return s;
}
```

Пример 3. Написати функцију која исписује поруку.

```
 string obavestenje()
{
     return  "Pokrenuli ste funkciju obavestenje.";
}
```

Уколико функција има повратну вредност, онда унутар тела методе мора да се нађе наредба return. Тип повратне вредности који се нађе иза наредбе return мора одговарати типу података који функција приликом позива враћа. Одмах после наиласка на наредбу return и враћања вредности, метода се завршава.

Тело методе функције може бити и празно. У том случају говоримо само о декларацији функције - знамо да она постоји, коју повратну вредност има и које аргументе, али не знамо како она заправо изгледа и шта ради. Не постоји минимална дужина тела методе, као што не постоји ни максимална. Ипак, треба водити рачуна о томе да функција буде разумљива и прегледна.

Приметимо: У првом примеру локална променљива је ***fakt***, док је у другом примеру локална променљива ***s***. Све функције до сада приказане у примерима имале су тело функције.

Пример 4. Написати функцију која рачуна максимум два броја.

```
//Imamo dva formalna parametra
int maksimum(int a, int b)
{
    //Početna vrednost za max je a
    int max = a;
    //Ako je max manje od b
    if(max < b)
        //Tada je max jednako b
        max = b;
    //U protivnom, ako max nije manje od b
    else
```

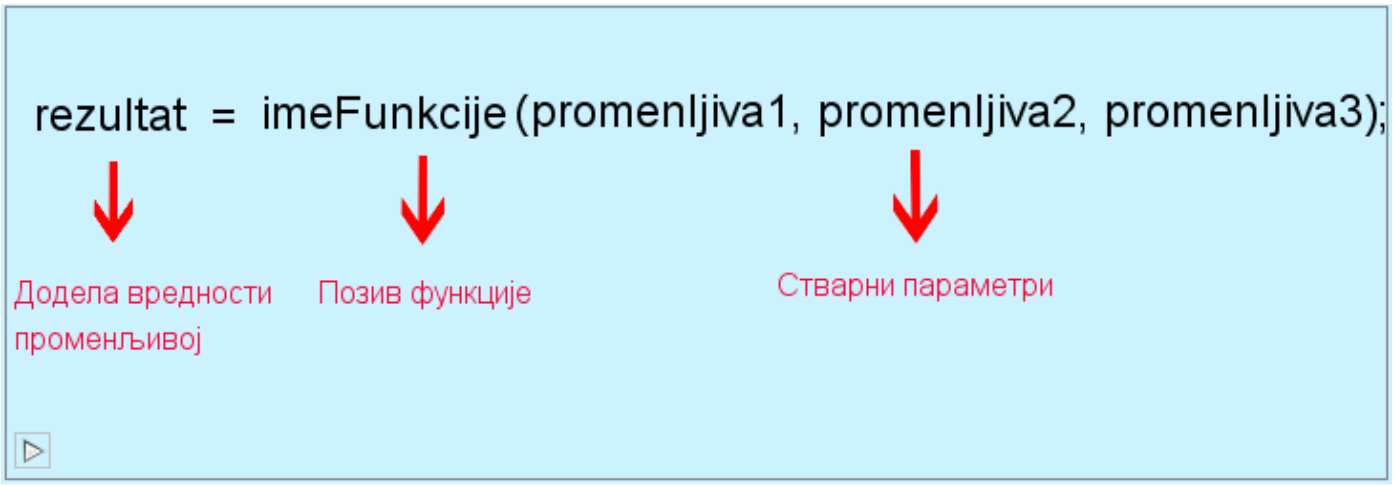
```
        //Tada je max jednako a
        max = a;
    //Vraćamo max kao maksimum dva zadata broja
    return max;
}
```

Стварни параметри и синтакса позива методе

Као што је познато, формални параметри су аргументи које пишемо у заградама иза имена функције. Међутим, формални параметри немају конкретну вредност, они нам само указују на то ког типа треба да буду вредности које ће се налазити унутар заграда приликом позива функције, као и колико ће аргумената функција имати. Позивање методе пишемо у следећој форми:

```
ime metoda (lista argumenata);
```

Позив метода се врши навођењем имена метода и навођењем евентуалне листе параметара унутар заграда иза имена метода. У заградама се наводе само аргументи, без њиховог типа и међусобно се одвајају зарезима. Уколико повратна вредност постоји, можемо је доделити одређеној промењивој.



Аргументи који се наводе у заградама уместо формалних параметара приликом позива метода јесу **стварни параметри.** Они представљају конкретне вредности које ће функција коју позивамо обрадити на одређен начин. Стварни параметри морају одговарати формалним параметрима по броју, редоследу и типу параметара.

Аргументи се методу могу пренети:

1. Преко референце
2. Преко вредности.

У C# се аргументи преносе по вредности уколико се не наведе другачије. Све промењиве морају бити иницијализоване пре него што се могу пренети методу, без обзира на то да ли се преноси преко вредности или референце.

Пренос преко вредности: Вредносни параметар се наводи тако што се наведе име типа иза кога следи име промењиве. Када вршимо пренос по вредности метода прави копију референце, тада две референце (оригинал и копија) референцирају на исти објекат. Оригинална референца се не може променити. Унутар методе може се променити вредност параметара, а да то не утиче на промењиву изван методе.

Пренос преко референце: Референтни параметар представља референцу на меморијску локацију. Када вршимо пренос преко референце метода не прави копију референце, већ се ради са оригиналном референцом. Оригинална референца се не може променити. Референтни параметри се декларишу навођењем кључне речи ***ref*** испред типа. Кључна реч ***ref*** односи се само на параметар испред кога је наведена, а не на целу листу параметара. Приликом позива методе такође се мора навести кључна реч ***ref***.

Код преноса преко вредности подаци се могу пренети методу, али се не могу пренети из ње, а код преноса преко референце подаци се могу пренети у методу и из методе.

Уколико метод нема параметре, обавезне су празне заграде приликом позива метода, као и приликом њеног дефинисања.

Пример 1. Написати програм који рачуна број комбинација од n елемената k-те класе користећи функцију факторијел.

Решење: Број комбинација од n елемената k-те класе једнак је:

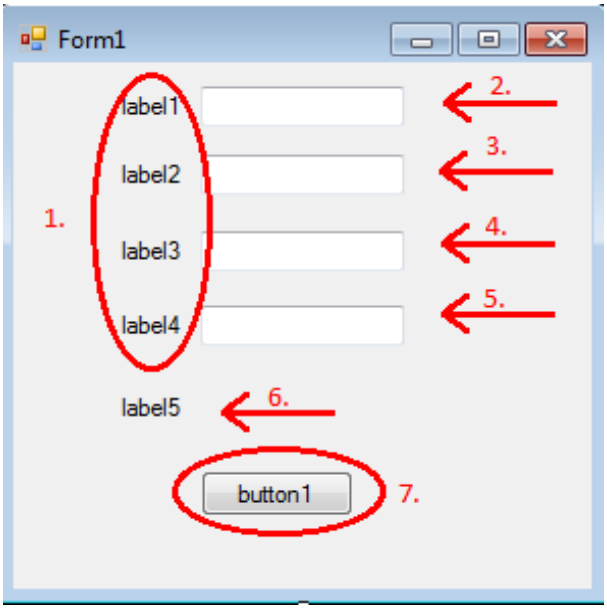
$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!}$$

За израчунавање броја комбинација потребно нам је да израчунамо факторијеле бројева који се појављују у наведеној формули. Да то не бисмо одвојено радили за сваки број користићемо функцију факторијел.

ПРИМЕРИ

Пример 1. Написати програм који рачуна максимум четири задата броја.

Правимо форму облика:



Слика 9.40. Изглед форме пре преименовања у примеру 1

label1 - преименоваћемо у a, label2 - преименоваћемо у b, label3 - преименоваћемо у c, label4 - преименоваћемо у d

textBox1 - у ово поље корисник уноси реалан број

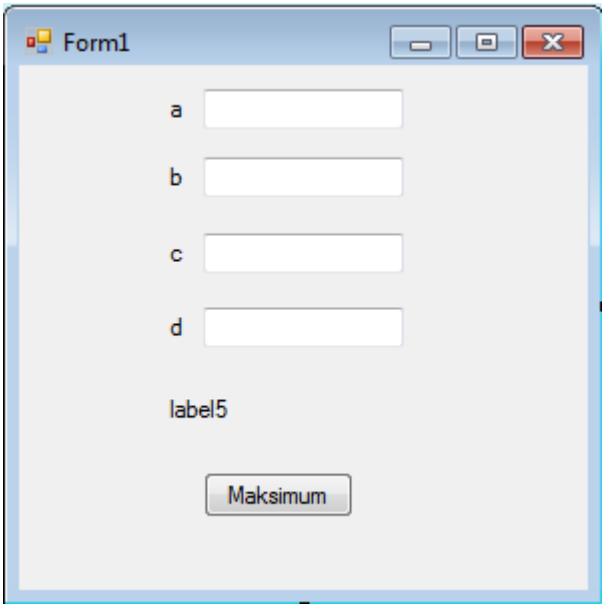
textBox2 - у ово поље корисник уноси реалан број

textBox3 - у ово поље корисник уноси реалан број

textBox4 - у ово поље корисник уноси реалан број

label5 - овде ћемо исписати резултат рада програма

button1 - преименујемо дугме у Maksimum



Двокликом на дугме Maksimum отвара нам се прозор у коме пишемо код за наш програм. Унутар класе Form1 пишемо код за функцију која рачуна максимум, а унутар button1_Click пишемо код програма у коме ћемо више пута позивати функцију. Да бисмо израчунали максимум четири броја позваћемо функцију три пута. Прво ћемо израчунати максимум прва два броја, потом максимум друга два броја и на крају ћемо утврдити који од нађена два максимума је већи.

Слика 9.41. Изглед форме након преименовања у примеру 1

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }
    //Definišemo funkciju maksimum koja računa maksimum dva broja
    double maksimum(double a, double b)
    {
        //Inicijalizujemo vrednost promenljive max na a
        double max = a;
        //Ako je max manje od b
        if (max < b)
          //Tada je max jednako b
            max = b;
        //U protivnom, ako je max veće od b
        else
          //Tada je max jednako a
            max = a;
        //Vraćamo max kao maksimalnu vrednost
        return max;

    }

    //Klikom na dugme računamo maksimum četiri broja
    private void button1_Click(object sender, EventArgs e)
    {
        //Ako nijedan textBox nije prazan računamo maksimum četiri broja
        if (textBox1.Text != "" && textBox2.Text != "" && textBox3.Text != "" &&
textBox4.Text != "")
        {
          //Deklaracija promenljivih
            double a, b, c, d;
          //Deklaracija promenljive koja označava maksimum četiri broja
            double max;
          //Učitavamo vrednosti iz textBox-ova
           //Prebacujemo ih iz formata string u format double - realan broj
            a = double.Parse(textBox1.Text);
            b = double.Parse(textBox2.Text);
            c = double.Parse(textBox3.Text);
            d = double.Parse(textBox4.Text);
           //Promenljivoj max dodeljujemo vrednost - pozivamo funkciju maksimum tri puta
            max = maksimum(maksimum(a, b), maksimum(c, d));
```

```
            //Ispisujemo rezultat u labeli
                label5.Text = "Maksimum uneta cetiri broja je " + max;

        }
        //Ako je neki od textBox-ova prazan
        else
        {
            //Ispisujemo poruku u labeli
                label5.Text = "Unesite vrednosti za a, b, c i d!";
        }
    }
}
```