РЕКУРЗИВНИ МЕТОД

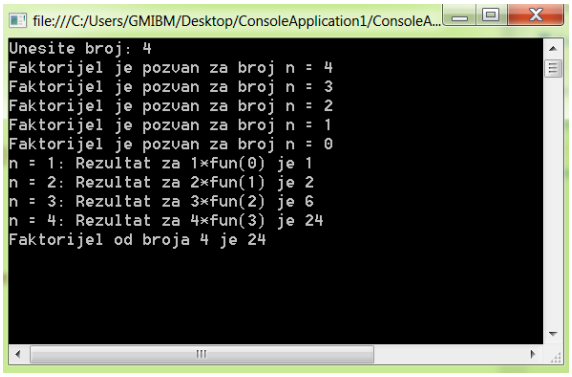
Рекурзивне методе су методе које позивају саме себе и једино су по томе посебне. Ништа их не издваја у односу на друге методе осим што себе позивају, при чему треба осигурати напуштање методе како не би отишла у бесконачно позивање саме себе.

Задатак 1: Написати програм за израчунавање факторијела.

```
using System;

class Program
{
    static void Main(string[] args)
    {
        //Program za izracunavanje faktorijela
        Console.Write("Unesite broj");
        int n = int.Parse(Console.ReadLine());
        Console.WriteLine("Faktorijel od broja {0} je {1}", n, fun(n));
        Console.ReadKey();
    }
    public static int fun(int n)
    {
        Console.WriteLine("Faktorijel je pozvan za broj n = {0}", n);
        if (n == 0) return 1;    //bazni slucaj
        else
        {
            int rezultat =  n * fun(n - 1);
            Console.WriteLine("n = {0}: Rezultat za {0}*fun({1}) je {2}", n, n-1, rezultat);
            return rezultat;
        }

    }
}
```



Ако унесемо број 4. Позива се метода **fun** са аргументом 4 тј. fun(4).

На слици видимо да се ова метода позива 5 пута за n=4,3,2,1,0.

Први позив: 4 * fun(3)
Други позив: 4 * 3 * fun(2)
Трећи позив: 4 * 3 * 2 * fun(1)
Четврти позив:4 * 3 * 2 * 1 * fun(0)
Пети позив: 4 * 3 * 2 * 1 * 1 [прва повратна вредност из петог позива 1*1]
= 4 * 3 * 2 * 1 [друга повратна вредност из четвртог позива (2 * 1)]
= 4 * 3 * 2 [трећа повратна вредност из трећег позива (3 * 2)]
= 4 * 6 [четврта повратна вредност из другог позива (4 * 6)]
= 24 [пета (последња) повратна вредност из првог позива]

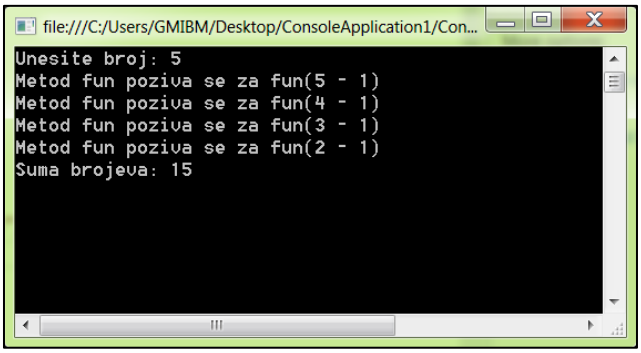
Приликом сваког позива методе у стек сегменту меморије ствара се нови стек оквир. У овим стек оквирима локална променљива n имаће редом вредности: 4,3,2,1,0. Стек оквир инстанце методе fun "памти" докле је та истанца методе стигла са извршавањем кода. Када се заврши повратак из методе у позивајућу рутину, параметри позива се скидају са стека.

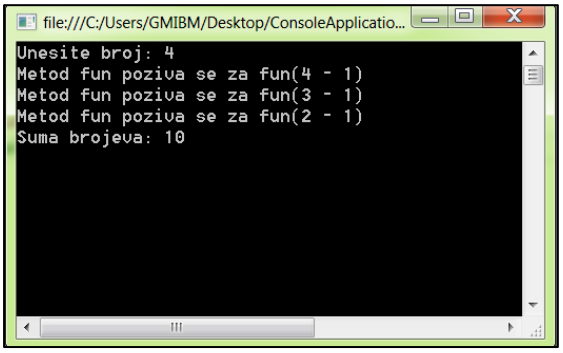
2. Задатак: Анализирати програмски код и одредити колико пута се метод fun() позива за fun(5) и fun(4).

```
using System;

class Program
{
    static void Main(string[] args)
    {
        Console.Write("Unesite broj: ");
        int n = int.Parse(Console.ReadLine());
        Console.WriteLine("Suma brojeva: {0}", fun(n));
        Console.ReadKey();
    }
    public static int fun(int n)
    {
        if (n == 1) return 1;           //bazni slucaj
        else {
            Console.WriteLine("Metod fun poziva se za fun({0} - 1)", n);
            return  n + fun(n - 1);    //rekurzivni korak
        }
    }
}
```

Ако тестирамо програм можемо да приметимо да се за n = 5 метод fun() позива још 4 пута за fun(4), fun(3), fun(2) и fun(1). Односно, за n = 4, метод fun() позива још 3 пута за fun(3), fun(2) и fun(1).





3. Задатак : Анализирати код и одредити резултат. Унети вредност преко тастатуре нпр. 3 и објаснити шта ће програм приказати на екрану: 1,2,6 или нешто друго.

```
using System;

class Program
{
        static void Main(string[] args) {

            Console.Write("Unesite ceo broj:");
            int i = int.Parse(Console.ReadLine());
            Console.Write(fun(i));
            Console.ReadKey();
        }
        static long fun(int n) {
            return n * fun(n - 1);
        }
}
```

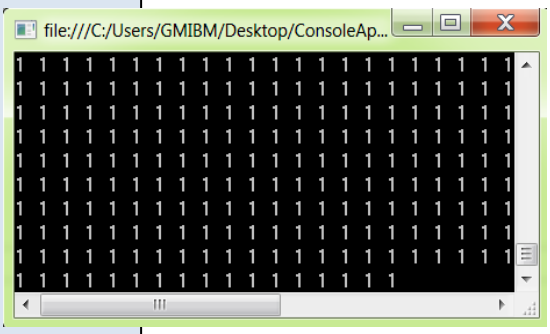
Када унесемо цео број (3) преко тастатуре (i = 3), позива се метода fun(i). Метод fun() је рекурзивна метода, зато што се у оквиру те методе налази метода за истим називом тј. fun(n-1). Имајући у виду да у оквиру рекурзивне методе не постоји механизам за напуштање методе она ће бесконачно да позива саму себе и програм ће пријављивати грешку.

4. Задатак : Анализирати код и одредити резултат који ће се приказати на екрану.

```
using System;

class Program
{
        static void Main(string[] args) {

            fun(2);
        }
        public static void fun(int n) {
            while (n > 1) {
                Console.Write((n - 1) + " ");
                fun(n - 1);
            }
        }
}
```

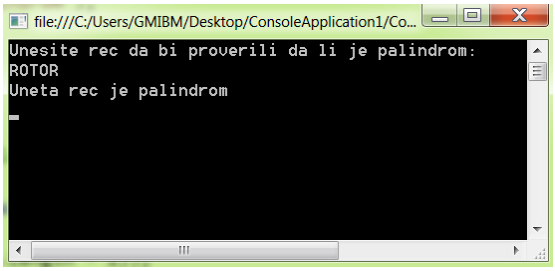


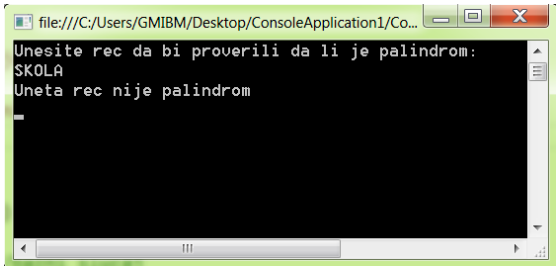
5. Задатак : Написати програм који одређује да ли је неки стринг палиндром.

```
using System;

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Unesite rec da bi proverili da li je palindrom: ");
        string s = Console.ReadLine();
        bool provera = palindrom(s);
        if (provera == true)
            Console.WriteLine("Uneta rec je palindrom");
        else
            Console.WriteLine("Uneta rec nije palindrom");
        Console.ReadKey();

    }
    public static bool palindrom(string s)
    {
        if (s.Length <= 1) return true; //bazni slucaj
        else
            if (s[0] != s[s.Length - 1]) return false;  //provera krajeva stringa
            else
                return palindrom(s.Substring(1, s.Length - 2)); //skrati krajeve
    }
}
```





НАПОМЕНА: Метода **Substring()** , садржи два аргумента: почетак и дужина стринга. Дакле, ако је:

s = “SKOLA”

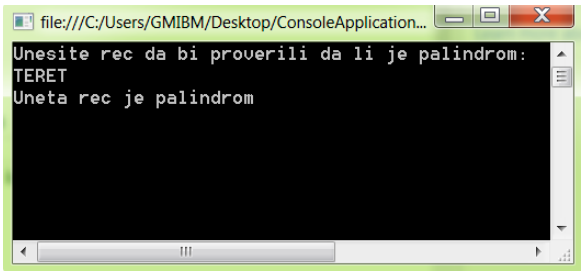
proba = s.Substring(1, s.Length – 2); вредност промењиве proba = “KOL”;

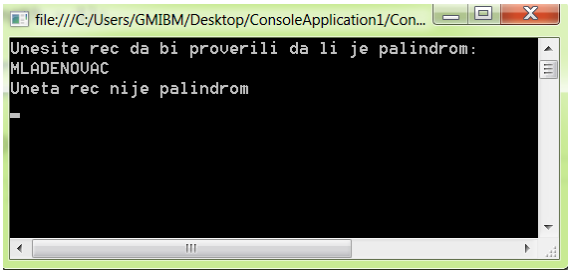
6. Задатак : Написати програм који одређује да ли је неки стринг палиндром.

```
using System;

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Unesite rec da bi proverili da li je palindrom: ");
        string s = Console.ReadLine();
        bool provera = Palindrom(s);
        if (provera == true)
            Console.WriteLine("Uneta rec je palindrom");
        else
            Console.WriteLine("Uneta rec nije palindrom");
        Console.ReadKey();

    }
    public static bool Palindrom(string s)
    {
        return Palindrom(s, 0, s.Length - 1);
    }
    public static bool Palindrom(string s, int levi, int desni) {
        if (desni <= levi) return true;
        else
            if (s[levi] != s[desni]) return false; //provera krajeva stringa
            else
                return Palindrom(s, levi + 1, desni - 1); // skrati krajeve
    }
}
```





Принцип за одређивање да ли је реч палиндром је слична као у претходном примеру. Могли смо да изоставимо методу Palindrom(string s).

7. Задатак : Написати програм за сортирање низа рационалних бројева у опадајућем редоследу.

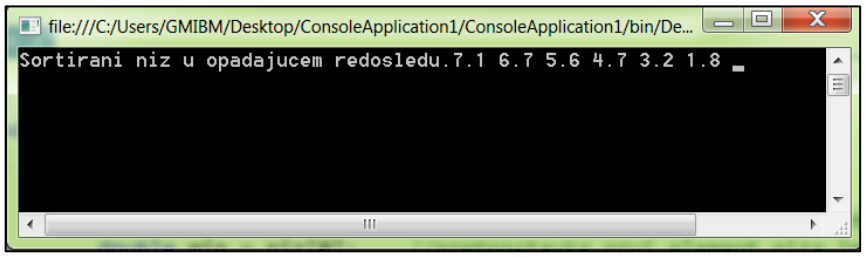
```
using System;

class Program
{
    static void Main(string[] args)
    {
        double[] niz = { 3.2, 5.6 , 1.8 , 7.1, 4.7, 6.7};
        Sortiranje(niz);
        Console.Write("Sortirani niz u opadajucem redosledu.");
        foreach (double x in niz)
            Console.Write(x + " ");
        Console.ReadKey();

    }
```

```
    public static void Sortiranje(double [] niz)
    {
        Sortiranje(niz, niz.Length - 1);
    }
    public static void Sortiranje(double [] niz, int kraj)
    {
        if (kraj > 0) {
            int imin = 0;           //index od elementa niza koji ima najmanju vrednost
            double min = niz[0];    //pretpostavka prvi element niza ima najmanju vrednost
            for (int i = 1; i <= kraj; i++)
                if (niz[i] < min)
                {
                    min = niz[i];
                    imin = i;
                }
            //zamenjujemo mesta posledenjem elementu u nizu
            //i elementu koji ima manju vrednost
            niz[imin] = niz[kraj];
            niz[kraj] = min;

            Sortiranje(niz, kraj - 1);
        }
    }
}
```



АНАЛИЗИРАТИ ПРИМЕРЕ И ЗАОКРУЖИТИ ТАЧАН ОДГОВОР

189. Дат је код у програмском језику C#, који дефинише рекурзивни метод. Анализирати код и одредити резултат извршавања задатог метода:

```
publicstaticint fun(int n) {
if (n == 1) return 1;
elsereturn n + fun(n - 1);
}
```

1. Позивом fun (4) се исти метод fun () позива још 2 пута.
2. Позивом fun (5) се исти метод fun () позива још 4 пута.
3. Позивом fun (4) се исти метод fun () позива још 4 пута.
4. Позивом fun (5) се исти метод fun () позива још 6 пута.

2

187. Дат је код у програмском језику C#, који дефинише рекурзивни метод. Анализирати код и одредити резултат извршавања задатог метода:

```
publiclong fun(int n){
return n * fun(n - 1);
}
```

1. Резултат позива fun(3) је 1.
2. Резултат позива fun(3) је 2.
3. Резултат позива fun(3) је 6.
4. Позив fun(3) изазива грешку јер производи бесконачан ланац позива истог метода fun(…).

2

188. Дат је код у програмском језику C#, који дефинише рекурзивни метод. Анализирати код и одредити резултат који ће се приказати на екрану:

```
namespace TestPrimer {
classProgram{
staticvoid Main(string[] args){
            fun(2);
        }
publicstaticvoid fun(int n) {
while (n > 1){
Console.Write((n - 1) + " ");
                fun(n - 1);
            }
        }
    }
}
```

1. Програм на екрану не приказује ништа
2. Програм на екрану приказује 1 2 3
3. Програм на екрану приказује 3 2 1.
4. Програм на екрану бесконачно приказује 1 1 1 1 1 ….
5. Програм на екрану бесконачно приказује 2 2 2 2 2 ….

2

190. Дат је код у програмском језику C#, рекурзивни метод, који проверава да ли је неки стринг палиндром. Да би код био комплетиран потребно је допунити трећи ред условом **if** наредбе.

```
1.publicstaticbool palindrom(String s)
2. {
3. if (s.Length <= 1) returntrue;  //bazni slučaj
4.elseif (____________________) returnfalse;
5.elsereturn palindrom(s.Substring(1, s.Length - 2));
6. }
```

Заокружити број испред траженог одговора:

1. `s[0] != s[s.Length - 1]`
2. `s[0] != s[s.Length]`
3. `s[1] != s[s.Length - 1]`
4. `s[1] != s[s.Length]`

2

191. Дат је код у програмском језику C#, рекурзивни метод, који проверава да ли је неки string палиндром. Да би код био комплетиран потребно је допунити седми ред.

```
1.publicstaticbool Palindrom(String s){
2. return Palindrom(s, 0, s.Length - 1);
3. }
4.publicstaticbool Palindrom(String s, int levi, int desni){
5.if (desni <= levi) returntrue; // bazni slucaj
6.elseif (s[levi] != s[desni]) returnfalse;
7.elsereturn_____________________;
8. }
```

Заокружити број испред траженог одговора:

1. `Palindrom(s)`
2. `Palindrom(s, levi, desni)`
3. `Palindrom(s, levi + 1, desni - 1)`
4. `Palindrom(s, levi + 1, desni)`
5. `Palindrom(s, levi, desni - 1)`

2

192. Дат је код у програмском језику C#, рекурзивни метод за сортирање низа рационалних бројева у опадајућем редоследу. Да би код био комплетиран потребно је допунити други ред помоћу једног од понуђених одговора.

```
1.  publicstaticvoid Sortiranje(double[] niz){
2.      _____________________________;
3.  }
4.  publicstaticvoid Sortiranje(double[] niz, int kraj){
5.  if (kraj > 0){
6.  int imin = 0;
7.  double min = niz[0];
8.  for (int i = 1; i <= kraj; i++)
9.  if (niz[i] < min){
10.                 min = niz[i];
11.                 imin = i;
12.             }
13.          niz[imin] = niz[kraj];
14.          niz[kraj] = min;
15.          Sortiranje(niz, kraj - 1);
16.      }
17. }
```

2

Заокружити број испред траженог одговора:

1. `Sortiranje(niz)`
2. `Sortiranje(niz, niz.length)`
3. `Sortiranje(niz, niz.length+1)`
4. `Sortiranje(niz, niz.length-1)`