بسم اللّه الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

بازگشت

مدرس : مهندس افشین رفوآ

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 - واحد 7
88146323 - 88446780 - 88146330

بازگشت

مقدمه

تصور کنید می خواهید اعداد فرد مثبت را از بیشنه مشخصی تا کمینه ی معینی بشمارید. به عنوان مثال، برای شمردن اعداد فرد از 1 تا 9 به این ترتیب عمل می کنیم.

1، 3، 5، 7،9

توجه داشته باشید که برای انجام این عملیات، بالاترین مقدار را در نظر می گیریم، سپس 2 را از آن کم می کنیم تا مقدار قبلی را به دست آوریم. در برنامه نویسی برای حل این مشکل، ابتدا تابع را می نویسیم سپس کاری می کنیم که تابع خود را فرابخواند. این پایه ی بازگشت در برنامه نویسی است.

معرفی بازگشت

1. برای راه اندازی برنامه ی کاربردی جدید، در فهرست گزینه ی اصلی، File -> New Project... را کلیک کنید.
2. Empty Project را از لیست میانی انتخاب کنید.
3. اسم پروژه را به Recursions تغییر دهید، کلید Enter را بزنید.
4. در پنجره ی Solution Explorer، راست کلیک کرده سپس : Recursions -> Add -> New Item
5. در لیست میانی روی Code File کلیک کنید.
6. اسم فایل را Calculator انتخاب کنید و کلید Enter را فشار دهید.

ایجاد متد بازگشتی

یکی از فرمول های موجود برای ایجاد متد بازگشتی به صورت زیر می باشد.

```
ReturnValue Function(Arguments، if any)
        {
            Optional Action...
            Function();
```

```
        Optionan Action...
    }
```

متد بازگشتی با یک مقدار بازگشتی شروع می شود. در صورت برنگرداندن یک مقدار، می توان آن را با کلید واژه ی void تعریف کرد. پس از اسمش، متد قادر است یک یا چند آرگومان بگیرد. اغلب مواقع، متد بازگشتی حداقل یک آرگومان می گیرد، سپس آن را اصلاح می کند. در بدنه ی متد، می توان هر کاری که لازم است انجام داد. دو قاعده ی اصلی وجود دارد که حین پیاده سازی متد بازگشتی باید رعایت کرد.

متد باید خود را در بدنه ی خود فراخوانی کند.

پیش یا پس از فراخوانی خود، متد باید شرطی را بررسی کند که به متد اجازه ی توقف بدهد. در غیر این صورت، متد تا ابد ادامه پیدا می کند.

برای مثال بالا (منظور مثال بازگشت می باشد)، می توان ابتدا متدی ایجاد کرد که عدد صحیح (integer) به عنوان آرگومان می گیرد. به منظور درک بهتر این مطلب، فقط اعداد مثبت را در نظر می گیریم. در بدنه ی متد، مقدار جاری آرگومان را نمایش می دهیم، سپس 2 واحد از آن کم می کنیم و در مرحله ی پایانی خود متد را فراخوانی می کنیم.

```
using System;
public class Exercise
{
        static void OddNumbers(int a)
        {
            if (a >= 1)
            {
                Console.Write("{0}، ", a);
                a -= 2;
                OddNumbers(a);
            }
        }
        public static int Main()
        {
            const int number = 9;
            Console.WriteLine("Odd Numbers");
            OddNumbers(number);
            Console.WriteLine();
            return 0;
        }
    }
```

همان طور که مشاهده می کنید متد خود را در بدنه اش فرا می خواند. نتیجه ی زیر به دست می آید.

```
Odd Numbers
        9، 7، 5، 3، 1،
        Press any key to continue...
```

ایجاد متد بازگشتی

1. برای ایجاد تابع بازگشتی، ابتدا فایل را به صورت زیر اصلاح کنید.

```
using System;
public class Calculator
{
            private long Factorial(long number)
            {
                if (number <= 1)
                    return 1;
                return number * Factorial(number - 1);
            }
            public static int Main()
            {
                long factor = 0;
                Calculator exo = new Calculator();
                Console.Write("To calculate a factorial، enter a
(small positive) number: ");
                factor = long.Parse(Console.ReadLine());
                Console.WriteLine("The factorial of {0} = {1}"،
                    factor، exo.Factorial(factor));
                System.Console.ReadKey();
                return 0;
            }
        }
```

2. برنامه را تست کنید.

3. عدد 8 را وارد کرده و Enter را بزنید.

```
To calculate a factorial، enter a (small positive) number: 8
        The factorial of 8 = 40320
        Press any key to continue...
```

4. با زدن کلید Enter از پنجره ی DOS خارج شوید.

استفاده از متدهای بازگشتی

متد های بازگشتی مکانیزم های ارزشمندی برای ساختن سری ها و لیست ها عرضه می کنند که در حقیقت مقادیر افزایشی یا کاهشی هستند که از یک الگو پیروی می کنند. تصور کنید به جای تنها نمایش دادن اعداد فرد (همان طور که بالا نشان دادیم)، می خواهیم آن ها را به صورت تصاعدی / افزایشی اضافه کنیم. در صورت داشتن 1، 1 تولید می کند. اگر 5 دارید و 1 را به 3 اضافه می کنید و بعد نتیجه ی آن را به 5 (اضافه کنید) و غیره.... این فرایند را می توان به ترتیب زیر نمایش داد.

1 = 1
1 + 3 = 4
1 + 3 + 5 = 9
1 + 3 + 5 + 7 = 16
1 + 3 + 5 + 7 + 9 = 25

برای اجرای این عملیات، ابتدا 1 را در نظر می گیریم. چنانچه عدد مساوی 1 یا کوچک تر از آن بود، متد 1 باز می گرداند. در غیر این صورت، 2 را به 1 اضافه می کنیم، سپس 2 را به نتیجه ی حاصله (2+1) اضافه می کنیم. این پروسه را ادامه دهید تا به مقدار آرگومان برسید. متد به ترتیب زیر ایجاد یا پیاده سازی می شود.

```
using System;
public class Exercise
{
            static int AdditionalOdd(int a)
            {
                if (a <= 1)
                    return 1;
                return a + AdditionalOdd(a - 2);
            }
            static void OddNumbers(int a)
            {
                if (a >= 1)
                {
                    Console.Write("{0}، ", a);
                    a -= 2;
                    OddNumbers(a);
                }
            }
            public static int Main()
```

```
            {
                const int Number = 9;
                Console.WriteLine("Odd Numbers");
                Console.WriteLine();
                Console.WriteLine("Sum of Odds: {0}\n",
AdditionalOdd(Number));
                return 0;
            }
        }
```

نتیجه ی زیر حاصل می گردد.

```
Odd Numbers
        9، 7، 5، 3، 1،
        Sum of Odds: 25
        Press any key to continue...
```

به کاربردن متدهای بازگشتی

1. فایل را به ترتیب زیر اصلاح کنید.

```
using System;
public class Calculator
{
            private long Factorial(long number)
            {
                if (number <= 1)
                    return 1;
                return number * Factorial(number - 1);
            }
            private long Permutation(long n، long r)
            {
                if (r == 0)
                    return 0;
                if (n == 0)
                    return 0;
                if ((r >= 0) && (r <= n))
                    return Factorial(n) / Factorial(n - r);
                else
                    return 0;
            }
            private long Combinatorial(long a، long b)
            {
                if (a <= 1)
                    return 1;
                return Factorial(a) / ((Factorial(b) *
Factorial(a - b)));
```

```
        }
        public static int Main()
        {
            long factor = 0;
            long second = 0;
            Calculator exo = new Calculator();
            Console.Write("To calculate a factorial، enter a
(small positive) number: ");
            factor = long.Parse(Console.ReadLine());
            Console.Write("To calculate a permutation and
the combination، enter a second (small positive) number: ");
            second = long.Parse(Console.ReadLine());
            Console.WriteLine("Factorial:   F({0}) = {1}"،
                factor، exo.Factorial(factor));
            Console.WriteLine("Permutation: P({0}، {1}) =
{2}"،
                factor، second، exo.Permutation(factor،
second));
            Console.WriteLine("Combination: C({0}، {1}) =
{2}"،
            factor، second، exo.Combinatorial(factor،
second));
            System.Console.ReadKey();
            return 0;
        }
```

1. برنامه را تست کنید.
2. به عنوان اولین عدد درخواستی، 20 را وارد کنید و بعد Enter را بزنید.
3. به عنوان عدد دوم، 5 را وارد کرده و کلید Enter را فشار دهید.

```
To calculate a factorial، enter a (small positive) number: 20
        To calculate a permutation and the combination، enter a
second (small positive)
         number: 5
        Factorial:   F(20) = 2432902008176640000
        Permutation: P(20، 5) = 1860480
        Combination: C(20، 5) = 15504
```