آموزش سی شارپ (C#) قسمت ۱۲ : حلقه ها (دستور for) (نسخه چاپی)

با سـلام مجدد به دوسـتان عزیزم در انجمن تخصصی فناوری اطلاعات ایران، در صورتی که از دنبال کنندگان سری آموزشی زبان سی شارپ باشید، می دانید که در دو بخش قبلی به بررسی دستوران کنترلی پرداختیم. در بخش سوم از قسمت هشتم سری آموزشی، به بررسی حلقه ها و دستور for خواهیم پرداخت. اما حلقه ها چه کاربردهایی دارنـد؟ خیلی از اوقات، نیاز داریم قطعه ای از کد چندین بار تکرار شود. برای مثال، فرض کنید میخواهیم اعداد ۱ تا ۱۰۰ را چاپ کنیم. راه حل چیست؟ صد بار پشت سر هم دستور Console.WriteLine را بنویسیم؟؟؟!!! صد در صد اینطور نیست. در اینجور مواقع باید از حلقه ها اسـتفاده کنیم. حلقه ها به ما این قابلیت را می دهند که دستوراتی را به تعداد دلخواه تکرار کنیم. حلقه ها را در دو بخش مورد بررسـی قرار خواهیم داد. در بخش اول به بررسی دستور for پرداخته و در بخش بعدی به بررسی دستورات while و do-while می پردازیم. در ابتدا، ساختار کلی دستور for را با هم بررسی می کنیم. بوسیله دستور for می توان یک قطعه از کد را تعداد دلخواهی که مد نظر داریم اجرا کنیم. ساختار کلی این دستور به شکل زیر است:

```
for({init};{condition};{increment})
{
   // loop body
}
```

خوب، بهتره که به کاربرد و مفهوم هر یک از بخش های دستور بالا بپردازیم:

۱. init: در این بخش بایـد متغیری که شـمارنده بر اساس آن عمل خواهد کرد را تعریف کنیم. هر حلقه for نیاز به یک متغیر شمارنده دارد که در هر تکرار، مقـدار آن یک واحـد اضافه می شود. این متغیر در بخش init تعریف می شود. این متغیر بایـد مقدار اولیه نیز داشته باشد.
۲. condition: بعد از تعریف متغیر شـمارنده، باید شـرطی برای اتمام حلقه مشـخص کنیم، برای مثال، می خواهیم شرط ما ۱۰ بار تکرار شود. در این قسمت تعیین می کنیم که حلقه ما در چه حالتی به اتمام می رسد. این قسمت باید مقدار بازگشتی از نوع bool داشته باشد.
۳. increment: در این بخش مقداری که با هر بار تکرار به شمارنده ما اضافه می شود یا کم می شود را مشخص می کنیم.
۴. loop body: بدنه حلقه for یا دستوراتی که می خواهیم با هر بار تکرار حلقه for اجرا شوند را در این قسمت می نویسیم. دقت کنید که بدنه حلقه for با Brace باز و بسـته مشخص می شود. در اینجا مانند دستور if، در صورتی که تعداد دستورات حلقه for تنها یک دستور بود می توان از نوشتن Brace ها خودداری کرد.

در ادامه با یک مثال ساده، با کاربرد واقعی دسـتور for آشـنا می شویم. فرض کنید تصمیم داریم اعداد یک تا ۱۰ را در خروجی چاپ کنیم. برای اینکار کافیست یک حلقه for تعریف کرده و داخل آن مقدار مورد نظر را در خروجی چاپ کنیم:

```
for(int counter = 1; counter <= 10; counter++)
   Console.WriteLine(counter);
```

همانطور که مشاهده می کنید، داخل دستور if سه بخش داریم که با علامت ; از هم جدا شده اند. در قسـمت اول متغیری به نام counter تعریف کرده و مقـدار پیش فرض ۱ را به آن اختصاص دادیم. در قسـمت دوم شـرط for را نوشتیم که حلقه تـا زمانی اجرا شود که مقـدار counter کوچکتر یا مساوی عدد ۱۰ می باشـد. در انتها هم گفتیم که با هر بار تکرار، یک واحد به متغیر counter اضافه شود. در داخل بدنه for نیز با دستور WriteLine مقدار متغیر counter را که با هر بار تکرار یک واحد به آن اضافه می شود را در خروجی چاپ می کنیم.

در مثال دوم تصمیم داریم اعداد ۱۰ تا ۱ را به صورت معکوس چاپ کنیم:

```
for(int counter = 10; counter >= 1; counter--)
   Console.WriteLine(counter);
```

در اینجا، تغیراتی که دادیم، ابتدا مقدار اولیه متغیر counter را برابر ۱۰ قرار دادیم و شرط را نیز جوری تغییر دادیم که دستورات for تا زمانی اجرا شوند که مقدار counter بزرگتر یـا مساوی یک می باشـد. در انتها نیز گفتیم با هر بار اجرای حلقه for، از مقدار counter یک واحد کم کن. با این کار اعداد ۱۰ تا ۱ به صورت معکوس در خروجی چاپ خواهند شد.

در مثال بعدی، می خواهیم اعداد زوج بین ۱ تا ۱۰۰ را در خروجی چـاپ کنیم، یک راه آن به این صورت است که یک حلقه با تکرار ۱۰۰ بار ایجاد کنیم و در داخل بدنه چک کنیم که باقی مانده تقسـیم متغیر شـمارنده بر ۲ مساوی صـفر هست یا نه. اگر بود مقدار را در خروجی چاپ کن:

```
for(int counter = 1; counter <= 100; counter++)
{
    if(counter % 2 == 0)
        Console.WriteLine(counter);
}
```

در قسمت های قبلی در مورد عملگر % گفتیم که باقی مانده تقسیم دو عدد را برای ما بر میگرداند.

در روش بعدی، می توانیم حلقه را جوری بنویسیم که مقدار counter در هر بار تکرار ۲ واحد اضافه شود:

```
for(int counter = 2; counter <= 100; counter = counter + 2)
    Console.WriteLine(counter);
```

کد بالا نیز کلیه اعداد زوج بین ۱ تا ۱۰۰ را برای ما در خروجی نمایش خواهد داد.

در هنگام استفاده از حلقه for ما می توانیم عملیات تعریف متغیر شمارنده را خارج از حلقه و قبل از شروع آن تعریف کنیم:

```
var counter = 1;
for(; counter <= 10; counter++)
    Console.WriteLine(counter);
```

دقت کنید که نوشتن علامت ; اولی برای حلقه for الزامی است، در غیر اینصورت کامپایلر از کد نوشته شده خطا خواهد گرفت.

حلقه های تو در تو (Nested Loops)

زمانی که ما چند حلقه for را داخل هم بنویسـیم، به این حلقه ها، حلقه های تو در تو گفته می شود. برای مثال، فرض کنید که قصد داریم جدول ضرب ۹ در ۹ را در خروجی چاپ کنیم. برای اینکار ما نیاز به حلقه های تو در تو خواهیم داشت. نمونه کد زیر یک جدول ضرب ۹ در ۹ را برای ما در خروجی چاپ می کند:

```
for (int i = 1; i <= 9; i++)
{
    for (int j = 1; j <= 9; j++)
    {
        Console.Write((i*j) + "\t");
    }
    Console.WriteLine();
}
```

```
}
```

حلقه اول، ۹ بار، و با هر بار اجرای حلقه اول، حلقه دوم ۹ بار اجرا خواهد شد و خروجی ضرب متغیرهای i و j در خروجی چاپ خواهد شد.

دستور break

بوسیله دستور break می توانیم در شـرایطی که مد نظر داریم از حلقه خارج شویم. برای مثال، در حلقه زیر، زمانی که شـمارنده به عدد ۵۰ برسد، حلقه for متوقف شده و از حلقه خارج می شویم:

```
for(int counter = 2; counter <= 100; counter++)
{
    if(counter == 50)
        break;
    Console.WriteLine(counter);
}
Console.WriteLine("End of program!");
```

دستور continue

به کمک این دسـتور می توانیم رونـد اجرای حلقه را متوقف کرده و به ابتدای حلقه برگردیم. فقط به این نکته توجه داشـته باشـید که این دستور باعث reset شـدن حلقه نمی شود و حلقه، تکرار بعدی را اجرا می کند. برای مثال، اگر ما در تکرار ۵۰ امین تکرار، دستور continue را اجرا کنیم، ۵۱ امین تکرار اجرا خواهـد شد و روند اجرا به ابتدای حلقه باز خواهد گشت. برای مثال، برنامه ای که اعداد زوج بین ۱ تا ۱۰۰ را در خروجی چاپ می کرد را به صورت زیر نیز می توانیم بنویسیم:

```
for(int counter = 1; counter <= 100; counter++)
{
    if(counter % 2 != 0)
        continue;
    Console.WriteLine(counter);
}
```

در کـد بالا گفتیم که اگر باقیمانـده تقسـیم متغیر شـمارنده بر عدد ۲، مخـالف صـفر بود، تکرار بعـدی اجرا شود، در غیر اینصورت دسـتور WriteLine اجرا خواهد شد.

چند مثال دیگر از دستور for

۱. دریافت دو عدد از کاربر و محاسبه عدد اول به توان عدد دوم و چاپ آن در خروجی:

```
Console.Write("Enter first number: ");
var firstNumber = int.Parse(Console.ReadLine());
Console.Write("Enter second number: ");
var secondNumber = int.Parse(Console.ReadLine());

var pow = 1;
for (var counter = 1; counter <= secondNumber; counter++)
{
```

```
    pow = pow*firstNumber;
}

Console.WriteLine(firstNumber + " ^ " + secondNumber + " = " + pow);
Console.ReadKey();
```

در کـد بالا، ابتـدا دو عـدد را از کـاربر دریـافت کرده و عـدد هـای وارد شـده را بـه عـدد تبـدیل می کنیـم و در دو متغیر firstNumber و secondNumber ذخیره می کنیم. سپس برای خروجی متغیری با نام pow تعریف می کنیم و مقدار اولیه آن را برابر ۱ قرار می دهیم.

دلیل اینکه مقدار آن را یک قرار دادیم، دستور داخل حلقه for می باشد، در صورتی که مقدار اولیه را صفر قرار می دادیم، نتیجه ضرب عدد اول در صـفر، صـفر می شد و نهایتاً خروجی ما عددی جز عدد صـفر نخواهد بود. سپس حلقه ای تشکیل داده که به تعداد عدد دوم تکرار می شود و در داخل بـدنه عـدد اول را در pow ضـرب کرده و نتیجه را در pow ذخیره می کنیم. پس از تمـام حلقه، در خروجی بـا پیغـام مناسب مقدار pow را چاپ می کنیم.

۲. محاسبه میانگین جمع اعداد ۱ تا ۱۰۰:

```
var sum = 0;
for (int counter = 1; counter <= 100; counter++)
{
    sum = sum + counter;
}

Console.WriteLine("Average: " + (sum/100));
```

در این بخش با دسـتور if و کاربرد آن آشـنا شـدیم. در قسـمت بعدی به بررسـی حلقه های while و do-while خواهیم پرداخت تـا مبحث دستورات کنترلی و حلقه ها به پایان برسد. **ITPRO باشید**

نویسنده : حسین احمدی

منبع : جزیره برنامه نویسی وب سایت توسینسو

هرگونه نشر و کپی برداری بدون ذکر منبع و نام نویسنده دارای اشکال اخلاقی است

#آموزش_زبان_سی_شارپ #آموزش_سی_شارپ #دوره_آموزشی_سی_شارپ #آموزش_گام_به_گام_سی_شارپ #دستور_for_در_زبان_سی_شارپ #آموزش_برنامه_نویسی_شئ_گرا #آموزش_مقدماتی_سی_شارپ #آموزش_برنامه_نویسی #دستور_for #حلقه_ها_در_زبان_c#

محمد محرابی

چند مثال از حلقه for

مثال ۱. استفاده از اعداد منفی

```
int x;
for(x = 100; x > -100; x -= 5)
  Console.WriteLine(x);
```

خروجی

```
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
-5
-10
-15
-20
-25
-30
-35
-40
-45
-50
-55
-60
-65
-70
-75
-80
-85
-90
-95
```

مثال ۲. در صورتی که شمارنده مثبت شد با دستور break از حلقه خارج میشود.

```
// use break to exit this loop
for(int i=-10; i <= 10; i++) {
  if(i > 0) break; // terminate loop when i is positive
  Console.Write(i + " ");
}
Console.WriteLine("Done");
```

خروجی

```
-10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 Done
```

مثال ۳. تعریف شمارنده (init) خارج از حلقه و استفاده از افزایشگر شمارنده (increment) در داخل بدنه حلقه.

```
i = 0; // move initialization out of loop
for(; i < 10; ) {
  Console.WriteLine("Pass #" + i);
  i++; // increment loop control var
}
```

خروجی

```
Pass #0
Pass #1
Pass #2
Pass #3
Pass #4
Pass #5
Pass #6
Pass #7
Pass #8
Pass #9
```

مثال۴. استفاده از ویرگول در تعریف حلقه

```
int i, j;

for(i=0, j=10; i < j; i++, j--) {
  Console.WriteLine("i and j: " + i + " " + j);
}
```

خروجی

```
i and j: 0 10
```

```
i and j: 1 9
i and j: 2 8
i and j: 3 7
i and j: 4 6
```

zahra .m

```
for (int i = 1; i <= 9; i++)
{
    for (int j = 1; j <= 9; j++)
    {
        Console.Write((i*j) + "\t");
    }
    Console.WriteLine();
}
```

استاد تو این قطعه کد از "\t"استفاده کردید. که به اندازه یه tab بین خروجی ها فاصـله ایجاد کرده. میشه توضیح بدید که گزینه های دیگه ای هم مثل این داریم ؟اسم خاصی دارن؟

حسین احمدی

سلام، به این کاراکتر ها اصطلاحاً Escape Character گفته میشه، مثلاً \b معادل Backspace هست یا \n خط جدید ایجاد می کنه، تو گوگل جستجو کنید C# Escape Characters براتون لیست کاملش رو میاره.

zahra .m

متشکرم استاد.

معمولا چه زمانی از این کاراکتر ها استفاده میشه؟

مطلب اصلی