برنامه نویسی مبتنی بر وب با استفاده از ASP.Net

مدرس:سمیه عادلی

مقدمه

منظور ازبرنامه نویسی مبتنی بر وب، نشانه گذاری و کدنویسی مربوط به طراحی صفحات وب است که شامل محتوای وب، برنامه نویسی سمت کاربر و سمت سرور و امنیت شبکه می شود.

برنامه نویسی مبتنی بر وب به دو بخش تقسیم می شود:

- برنامه نویسی سمت کاربر
- برنامه نویسی سمت سرور

برنامه نویسی سمت کاربر

- در سمت کاربر بیشتر نیاز است داده هایی که کاربر وارد می کند گردآوری شود.
- همچنین اطمینان حاصل شود که افزونه های لازم برای ارائه یک واسط کاربری گرافیکی وجود دارد.
- برای کدنویسی سمت کاربر از JavaScript، VBScript استفاده می شود. HTMLو CSS وظایف دیگر سمت کاربر مانند طراحی و قالب بندی را انجام می دهند.

برنامه نویسی سمت سرور

- برنامه‌نویسی سمت سرور عملکرد سیستم‌های زیرین را پوشش می‌دهد.
  - مانند دسترسی و بازیابی داده از پایگاه داده، امنیت و کارائی.
- بعضی ابزارهایی که برای این منظور استفاده می شوند عبارتند از: ASP، PHP، Java و MySQL.

تفاوت کدنویسی سمت کاربر و سمت سرور

- کدهای سمت کاربر بر روی کامپیوتر کاربر در یک مرورگر وب ذخیره و اجرا می شود.
- کدهای سمت سرور برای کاربر قابل دسترس نیستند. بلکه بر روی وب سرور اجرا شده و سپس صفحه تولید شده، برای کامپیوتر کاربر ارسال می شود.

اجزای اصلی برنامه های تحت وب

- سرور وب
- مرورگر وب
- HTML
- VBScript
- jQuery
- JavaScript
- CSS
- ASP
- php

ASP

- ASP: Active Server Page

- ASP یک پلت فرم وقالب کاری برای ایجاد برنامه های مبتنی بر وب است.
- ASP.net روی قالب کاری .net ساخته شده، بنابراین تمام ویژگی های آن روی برنامه های کاربردی ASP.net قابل دسترس می باشد.

ASP

- برنامه های کاربردی تولید شده در ASP.net می توانند با هر زبان برنامه نویسی قابل ترجمه در net. نوشته شوند، مانند:
- C#
- Visual Basic

- برای توسعه ی برنامه های کاربردی با ASP.net می توان از visual studio استفاده کرد.

فضاهای نام

- فضاهای نام (NameSpaces)

✓فضای نام روشی برای مدیریت کدنویسی است.

✓فضای نام از ایجاد تداخل بین نام های توابع در برنامه جلوگیری می کند.

فضاهای نام

برای ایجاد فضای نام به صورت زیر عمل می شود:

```
namespace anyName
{
.....
    Class anyClassName
{
  .....
}

.....
}
```

فضاهای نام

- یکی از فضاهای نام پایه ای در net.، فضای نام System است. برای استفاده از آن می توان از کد زیر استفاده کرد:

```
Using System;
```

- معمولا تمامی فضاهای نام به صورت public می باشند و در خارج از کد قابل دسترسی هستند.

فضاهای نام

۲ روش برای استفاده از کدهای فضای نام وجود دارد:

- روش اول

ProjectName.NameSpace.ClassName.MemberName

- روش دوم

Using NameSpace;

ClassName. MemberName;

متغیرها

متغیر مکانی از حافظه است که مقداری در آن ذخیره می شود و بعدها بر اساس آن مقدار تصمیم گیری هایی انجام می شود یا عملیاتی (مثلا ریاضیاتی و محاسباتی) بر روی آن انجام می شود.

تعریف متغیر و مقداردهی آن

- برای تعریف متغیر باید ابتدا نوع داده ای آن را مشخص کرد و سپس نامی را به آن اختصاص داد، تا متغیر در طول برنامه قابل دسترسی باشد.
- برای تعریف یک متغیر، به صورت زیر عمل می شود:
  - نوع متغیر نام متغیر، به عنوان مثال: int x,y
- نکته: مقداردهی اولیه یک متغیر، از بسیاری از خطاهای زمان اجرا مانند انجام عملیات ریاضی روی دو متغیر بدون مقدار جلوگیری خواهد کرد.

انواع داده ای(Data Types)

نوع داده ای، نوع اطلاعاتی که قرار است در یک متغیر ذخیره شود، را مشخص می کند.

نوع داده ای یک متغیر، تاثیر زیادی بر چگونگی رفتار کامپیوتر با برنامه ی شما دارد.

انواع داده ای(Data Types)

انواع داده ای در C# به صورت زیر دسته بندی می شوند:

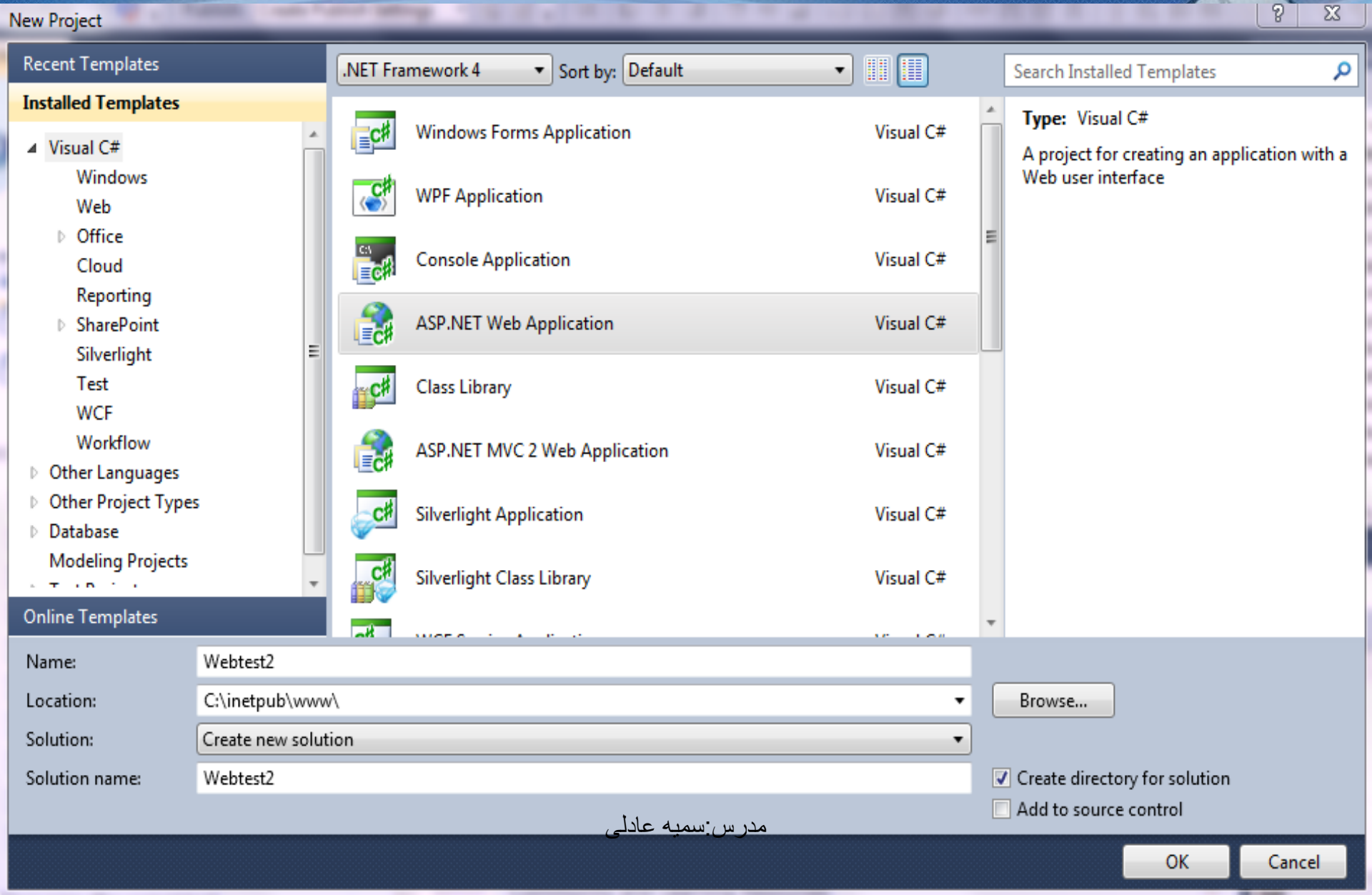
- اعداد صحیح
- اعداد اعشاری
- رشته ها و کاراکترها
- تاریخ
- بولی

C#	C#	C#	C#	C#
long	decimal	string	DateTime	bool
int	double			
short				
byte				

مثال

- ویژوال استودیو را اجرا کنید.
- در صفحه ی باز شده، روی دکمه new project کلیک نمایید.
- از پنل project گزینه VisualC# Project را انتخاب نمایید.
- از پنل سمت چپ، گزینه ASP.net Web Application را انتخاب نمایید.
- در قسمت location، نامی را برای پروژه ی خود در دایرکتوری Home انتخاب نمایید.

مثال



مدرس:سمیه عادلی

مثال

- از ToolBox سمت چپ صفحه یک Label و یک Button را روی فرم قرار دهید.
- روی دکمه ۲بار کلیک کنید، تا بتوانید کد مربوط به رویدادی که هنگام کلیک کردن روی دکمه می خواهید انجام شود، را بنویسید.
- نکته: به صفحه ی باز شده بدین صورت، Code Behind می گویند.

مثال

فضاهای نام مفید و کاربردی در این نوع برنامه ها به صورت پیش فرض در صفحه ی Code Behind تعریف شده اند.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

مثال

هدف مثال این است که :

هر بار کاربر روی دکمه کلیک کرد به او جمله ی "سلام این اولین برنامه ی من است" را نشان دهد.

برای این منظور باید مراحل زیر را انجام دهید:

- ✓ یک متغیر از نوع string و با نام strtext تعریف کنید.

```
String strtext;
```

- ✓ متغیر strtext را با جمله ی مذکور در هدف مثال مقداردهی کنید.

```
Strtext="سلام این اولین برنامه ی من است";
```

مثال

- مقدارمتغیر strtext باید به لیبل نسبت داده شود تا روی فرم نمایش داده شود. برای این منظور باید به صورت زیر عمل کرد:

```
Label1.Text=strtext;
```

اکنون می توانید برنامه را اجرا کنید.

عملگرها

- عملگرها، علامت هایی هستند که بر روی یک یا چند عبارت کار خاصی را انجام می دهند.
- تعدادی از انواع عملگرها عبارتند از:
  - عملگرهای محاسباتی
  - عملگرهای رابطه ای
  - عملگرهای منطقی
  - عملگرهای بیتی
  - عملگرهای انتساب

عملگرهای محاسباتی

- این عملگرها برای اعمال محاسبات ریاضی مانند جمع، تفریق و ... استفاده می شوند.
- عملگر + : برای جمع دو عدد استفاده می شود.

```
int n1 = 10;
int n2 = 15;
int result = n1 + n2;
```

عملگرهای محاسباتی

- عملگر –: عملیات تفریق را انجام می دهد.
- مثال:

```
int result = 34 - 15;
```

- عملگر *: عملیات ضرب را انجام می دهد.
- مثال:

```
int result = 20* 5;
```

عملگرهای محاسباتی

- عملگر /: عملیات تقسیم را انجام می دهد.
- مثال:

```
int result = 20/5;
```

- عملگر %: باقیمانده تقسیم دو عدد را بر می گرداند.
- مثال:

```
int result = 13% 2;
```

عملگرهای محاسباتی

- عملگر ++: این عملگر به مقدار موجود یک عدد اضافه می کند.
- مثال

```
int num = 1;
num++;
```

- عملگر --: این عملگر از مقدار موجود یک عدد کم می کند.
- مثال

```
int num = 10;
num--;
```

عملگرهای رابطه ای

- این عملگرها، برای مقایسه بین دو عبارت یا مقدار استفاده می شوند و نتیجه ای که بر می گردانند از نوع Boolean است.
  - عملگر ==: این عملگر بررسی می کند که دو عبارت با هم برابر هستند یا خیر. در صورت برابر بودن مقدار true و در غیر اینصورت مقدار false بر می گرداند.
  - عملگر =!: این عملگر بررسی می کند که دو عبارت مخالف یکدیگر هستند یا خیر.

عملگرهای رابطه ای

- عملگر <: این عملگر در صورتی true بر میگرداند که عملوند سمت چپ کوچکتر از عملوند سمت راست باشد در غیر اینصورت مقدار false بر می گرداند.
- مثال

```
bool result = num1 < num2;
```

- عملگر =<: این عملگر در صورتی true بر میگرداند که عملوند سمت چپ کوچکتر یا مساوی عملوند سمت راست باشد در غیر اینصورت مقدار false بر می گرداند.

عملگرهای رابطه ای

- عملگر <: این عملگر در صورتی true بر میگرداند که عملوند سمت چپ بزرگتر از عملوند سمت چپ باشد.
- عملگر =<: این عملگر در صورتی true بر میگرداند که عملوند سمت چپ بزرگتر یا مساوی از عملوند سمت راست باشد.
- مثال:

```
bool result1 = num1 >= num2;
```

عملگرهای منطقی

این نوع عملگرها بر روی نوع های Boolean اعمال می شوند. یعنی عملوندهای مورد استفاده باید از نوعBoolean بوده یا مقدار Boolean بر گردانند.

- عملگر &&: یا عملگر AND در صورتی مقدار True بر می گرداند که هر دو عملوند مقدار True بر گردانند.
- حالت های مختلف && به شرح زیر می باشد:

- true && true => true
- true && false => false
- false && true => false
- false && false => false

عملگرهای منطقی

- عملگر || : یا عملگر OR در صورتی مقدار True بر می گرداند که یکی از عملوندها نتیجه True داشته باشد.
- حالت های مختلف || به شرح زیر می باشد:

- true || true => true
- true || false => true
- false || true => true
- false || false => false

عملگرهای منطقی

- عملگر !: یا عملگر NOT، نتیجه یک عبارت Boolean را برعکس می کند، یعنی اگر یک عبارت نتیجه true برگرداند، این عملگر نتیجه را به false تبدیل می کند.

عملگرهای بیتی

- این عملگرها، بر روی بیت ها اعمال می شوند.
- عملیات مربوط به عملگر به صورت بیت به بیت انجام می شود.
- نکته: در سیستم باینری، عبارات به صورت اعداد ۰ و ۱ نمایش داده می شوند. به هر یک از این اعداد یک بیت گفته می شود.

عملگرهای بیتی

- عملگر &: در این عملگر، در صورتی که هر دو بیت ۱ باشند، خروجی ۱ خواهد بود. در غیر اینصورت خروجی ۰ خواهد شد. نام این عملگر AND می باشد.
- مثال:

- Num1 :0 1 0
- Num2: 1 1 0
- Result: 0 1 0

عملگرهای بیتی

- عملگر |: نام دیگر این عملگر OR می باشد.
- خروجی آن، در صورتی که یکی از بیت ها معادل ۱ باشد، ۱ خواهد بود، در غیر اینصورت بیت نتیجه ۰ خواهد شد.
- مثال:

- Num1 :0 1 0
- Num2: 1 1 0
- Result: 1 1 0

عملگرهای بیتی

- عملگر ^: نام دیگر این عملگر XOR می باشد
- بیت خروجی در صورتی ۱ خواهد بود که بیت های معادل با یکدیگر تفاوت داشته باشند.
- مثال:

- Num1 :0 1 0
- Num2: 1 1 0
- Result: 1 0 0

عملگرهای انتساب

عملگرهای انتساب جهت ریختن مقدار داخل یک متغیر و یا تغییر مقدار آن استفاده می شوند.

- عملگر =: عملیات انتساب ساده را انجام می دهد. یعنی مقدار سمت راست را داخل عملوند سمت چپ قرار می دهد.
- عملگر =+: این عملگر، مقدار سمت راست را به مقدار موجود عملوند سمت چپ اضافه کرده و نتیجه را داخل عملوند سمت چپ میریزد.

- int num = 1;
- num += 1;
- Result:num=2;

عملگرهای انتساب

عملگر =-: این عملگر، مقدار سمت راست را از مقدار موجود عملوند سمت چپ کم کرده و نتیجه را داخل عملوند سمت چپ میریزد.

- int num = 2;
- num -= 1;
- Result:num=1;

- عملگر =*: این عملگر، مقدار سمت راست را در مقدار موجود عملوند سمت چپ ضرب کرده و نتیجه را داخل عملوند سمت چپ میریزد.

عملگرهای انتساب

- عملگر =/: این عملگر، مقدار موجود در عملوند سمت چپ را بر مقدار موجود عملوند سمت راست تقسیم کرده و نتیجه را داخل عملوند سمت چپ میریزد.

- int num = 20;
- num /= 4;
- Result:num=5;

- عملگر =%: این عملگر، باقیمانده تقسیم عملوند سمت چپ بر مقدار سمت راست را در عملوند سمت چپ میریزد.

حق تقدم عملگرها

- در عبارتی که چندین عملگر در آن استفاده شده باشد، انجام عملیات بر اساس حق تقدم عملگر ها انجام می شود.
- در زیر حق تقدم ها را مشاهده می کنید. حق تقدم ها از درجه زیاد به کم نوشته شده اند. همچنین در هر سطر حق تقدم ها از چپ به راست می باشند:

- **() , []**
- **! , ++ , --**
- *** , / , %**
- **+ , -**
- **<< , >>**
- **< , <= , > , >=**
- **== , !=**
- **& , ^ , | , && , ||**

مثال

- برنامه ای با نام Mathtest طراحی کنید، که عدد اعشاری n را از ورودی خوانده و عملیات زیر را بر روی آن انجام دهد:
- n=n+56
- n=n-33
- n=n*45
- n=n/2
- n=n%5

مثال

- ویژوال استودیو را اجرا کنید.
- در صفحه ی باز شده، روی دکمه new project کلیک نمایید.
- از پنل project گزینه VisualC# Project را انتخاب نمایید.
- از پنل سمت چپ، گزینه ASP.net Web Application را انتخاب نمایید.
- در قسمت location، نام Mathtest را برای پروژه ی خود در دایرکتوری Home انتخاب نمایید.

مثال

- با استفاده از جعبه ابزار ۶ کنترل Label با خواص زیر، به فرم اضافه کنید:
  - خاصیت ID برابر با lbln و خاصیت text برابر با "عدد اعشاری را وارد کنید".
  - خاصیت ID برابر با lblAdd
  - خاصیت ID برابر با lblSub
  - خاصیت ID برابر با lblMul
  - خاصیت ID برابر با lblDiv
  - خاصیت ID برابر با lblRem

مثال

- ✓ با استفاده از جعبه ابزار یک کنترل TextBox با خاصیت ID برابر با txtn به فرم اضافه کنید.
- ✓ با استفاده از جعبه ابزار یک کنترل Button با خاصیت ID برابر با btnmath و خاصیت text برابر با Math Test به فرم اضافه کنید.
- ✓ روی کنترل ۲ بار کلیک کنید و کد مربوط به برنامه ی مدنظر را بنویسید.

دستورات کنترلی

- بعضی از مواقع، در برنامه نویسی شرایطی ایجاد می شود که نیاز است بر اساس شرایط خاص و یا ورودی های کاربر، روند اجرای برنامه تغییر کند.
- برای مثال گفته می شود:
- اگر x به این حالت بود، A را انجام بده در غیر اینصورت B را انجام بده.
- برای این منظور، باید از شرط ها و دستورات کنترلی استفاده شود

دستور if

- راحت ترین راه برای تصمیم گیری در برنامه ها استفاده از دستور if است.
- دستورif برنامه نویس را قادر خواهد کرد بر اساس شرط های مختلف کدهای مورد نظر خود را اجرا کند.
- ساختار دستور if به صورت زیر است:

```
if(condition)
{
   // statements
}
```

دستور if

- در دستور if:
- در قسمت condition شرط مورد نظر نوشته می شود.
- در قسمت statements هم دستوراتی که در صورت درست بودن شرط اجرا خواهند شد، نوشته می شوند.

دستور if else

- فرض کنید شرطی را مشخص کردید، می خواهید در صورت برقرار نبودن شرط، قطعه کد دیگری اجرا شود. در این حالت دستور else به کار برده می شود.
- دستور else کدی که در صورت برقرار نبودن شرط باید اجرا شود را مشخص می کند.

دستور if else

ساختار کلی دستور else به صورت زیر است:

```
if(condition)
{
   // if body
}
else
{
   // else body
}
```

- قسمت اول همان بخش if است که در قسمت قبلی مطرح شد.
- کدی که در قسمت else نوشته می شود در صورت برقرار نبودن شرط اجرا خواهد شد.

دستور if-elseif

- به وسیله ی این دستور، می توان چندین شرط متوالی را در برنامه تعیین کرد. ساختار کلی این دستور به صورت زیر است:

```
if(condition1)
{ // if body
}
else if(condition2)
{ // if body
}
else if(conditionN)
{ // if body
}
else
{ // if body
}
```

- اگر شرط ۱ برقرار بود:دستورات اول
- در غیر اینصورت اگر شرط ۲ برقرار بود : دستورات دوم
- در غیر اینصورت اگر شرط n برقرار بود دستورات nام:
- در غیر این صورت : سایر دستورات

دستور switch

- دستور switch بر اساس یک مقدار حالت های مختلف را بررسی کرده و دستورات قسمت مربوطه را اجرا خواهد کرد. ساختار کلی دستور switch به صورت رو به رو است:

```
switch({variable})
{
    case {value1}:
        statements
        break;
    case {value2}:
        statements
        break
        .
        .
        .
    case {value-n}:
        statements
        break;
    default:
        statements
        break;
}
```

دستور switch

- ساختار switch با دستور switch شروع شده و مقابل آن در داخل پرانتز نام متغیری که باید مقدار آن بررسی شود، قرار می گیرد. یعنی به جای variable نام متغیر مورد نظر نوشته می شود.
- بوسیله دستور case به ترتیب شرایط مختلف مشخص می شود. به جای value در مقابل دستور case مقدار مورد نظر قرار می گیرد.

دستور switch

- بعد از نوشتن دستور مربوط به هر case باید انتهای caseبا دستور break مشخص شود. در غیر اینصورت خطا ایجاد می شود.
- در بخش default، دستوراتی که در صورت برقرار نبودن هیچ کدام از شرط های case باید اجرا شود، تعیین می شود.
- نوشتن بخش default دلخواه است و می توان از نوشتن آن صرفنظر کرد.

دستور switch

```
switch(name)
{
    case "Value1":
    {
        int number = 1;
        break;
    }
    case "Value2":
    {
        int number = 2;
        break;
    }
}
```

- **نکته:** زمانی که قصد تعریف یک متغیر داخل قسمت case را دارید و همان متغیر در case دیگری، در همان دستور switch نیز تعریف می شود، باید بدنه case را با علامت های brace باز و بسته مشخص کنید.

تمرین

- ✓برنامه ای با نام Simpletest طراحی کنید، که ۲ عدد و یک عملگر را از ورودی خوانده و بر اساس عملگر وارد شده، محاسبه را انجام داده و نتیجه را به کاربر نشان دهد.
- ✓عملگرهای موردنظر به صورت زیر هستند:

✓ +

✓ -

✓ *

✓ /

✓ %

- ✓ برنامه را به ۲ صورت زیر طراحی کنید:
  - با استفاده از دستور if-elseif
  - با استفاده از دستور switch

حلقه ها

- زمانی که نیاز باشد قطعه ای از کد چندین بار تکرار شود از حلقه استفاده می شود.
- انواع دستورات موجود برای ایجاد حلقه به صورت زیر است:
- دستورfor
- دستورwhile
- دستورdo-while

دستور for

بوسیله دستور for می توان یک قطعه از کد را به تعداد دفعات موردنظر اجرا کرد. ساختار کلی این دستور به شکل زیر است:

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

دستور for

- int i: در این بخش متغیری که شمارنده بر اساس آن عمل خواهد کرد تعریف می شود. هر حلقه for نیاز به یک متغیر شمارنده دارد که در هر تکرار، مقدار آن یک واحد اضافه می شود. این متغیر در بخش int i تعریف می شود. این متغیر باید مقدار اولیه نیز داشته باشد.
- Condition: بعد از تعریف متغیر شمارنده، باید شرطی برای اتمام حلقه مشخص شود.

دستور for

- increment: در این بخش مقداری که با هر بار تکرار به شمارنده اضافه می شود یا کم می شود، تعیین می شود.
- loop body:در این قسمت، بدنه حلقه for یا دستوراتی که باید با هر بار تکرار حلقه for اجرا شوند تعیین می شود.
- نکته: بدنه حلقه for با Brace باز و بسته مشخص می شود. در اینجا مانند دستور if، در صورتی که تعداد دستورات حلقه for تنها یک دستور بود می توان از نوشتن Braceها خودداری کرد.

حلقه های تو در تو

- زمانی که چند حلقه for داخل هم نوشته شوند، به این حلقه ها، حلقه های تو در تو گفته می شود.
- برای مثال، فرض کنید که هدف ایجاد جدول ضرب ۹ در ۹ است. برای این منظور، به حلقه های تو در تو نیاز است.
- نمونه کد رو به رو یک جدول ضرب ۹ در ۹ را ایجاد می کند:

```
for (int i = 1; i <= 9; i++)
{
    for (int j = 1; j <= 9; j++)
    {
    p=i*j;
    }
    }
```

دستور while

- دستور while همانند دستورfor برای تکرار اجرای یک یا چند دستور مورد استفاده قرار می گیرد.
- تفاوت این دستور با دستور for در این است که در ساختار while عملیات تعریف متغیر انجام نمی شود و تنها یک شرط برای خروج از حلقه مشخص می شود. ساختار کلی دستورwhile به صورت رو به رو است:

```
while({condition})
{
   // loop syntaxes
}
```

دستور while

- برای دستور while در بخش condition باید شرطی که بر اساس آن حلقه به اتمام خواهد رسید مشخص شود.
- دستورات داخل بدنه while تا زمانی اجرا خواهند شد که شرط while برابر true باشد.

دستور while

به کد مقابل دقت کنید، ابتدا متغیری خارج از حلقه while تعریف شده است.

سپس حلقه while نوشته شده است. بر اساس شرط حلقه while، کد بدنه حلقه تا زمانی اجرا خواهد شد که مقدار counter کوچکتر یا مساوی ۱۰ باشد.

بوسیله دستور آخر با هر بار اجرای حلقه یک واحد به متغیر counter اضافه می شود. در صورتی که این خط نوشته نشود، حلقه به صورت بی نهایت اجر خواهد شد زیرا مقدار counter اضافه نشده و شرط حلقه while برای همیشه مقدار true بر می گرداند.

```
int counter = 1;
int a=1;
while (counter <= 10)
{
    a=a+2;
    counter++;
}
```

دستور do-while

ساختار حلقه do-while به صورت زیر است:

```
do
{
  // syntaxes
} while({condition});
```

تفاوت حلقه while با حلقه do-while در این است که حلقه do-while حداقل یکبار اجرا خواهد شد، به این خاطر که شرط اتمام حلقه در انتهای حلقه قرار دارد، اما حلقه while در صورت برقرار نبودن شرط، اصلاً وارد حلقه نمی شود.

آرایه ها

- برای تعریف مجموعه ای از اطلاعات هم نوع از آرایه ها استفاده می شود.
- آرایه ها هم همانند متغیرها باید تعریف و مقداردهی اولیه شوند.
- علاوه براین، باید نوع و تعداد اعضای آرایه ها نیز تعیین گردند.
- نکته: حد پایین آرایه صفر است.

آرایه ها

برای تعریف و مقداردهی آرایه ها چندین روش وجود دارد.

- تعریف آرایه

datatype[] arrayname= new datatype[arraylength]

مثال: int[] intdata=new int[4];

مثال فوق، آرایه ای از نوع عدد صحیح و به طول ۴ تعریف می کند.

- تعریف آرایه و هم زمان مقداردهی آن

datatype[] arrayname={“6”,”7”,”8”,”9”}

آرایه ها

- روشی دیگر برای مقداردهی آرایه بعد از تعریف آن

```
datatype[] arrayname= new datatype[arraylength];
arrayname[0]=“a”;
arrayname[1]=“b”;
   ⋮
arrayname[n]=“z”;
```

پیمایش در آرایه

برای حرکت بین اعضای یک آرایه با طول زیاد، باید از حلقه ی for یا foreach استفاده کرد.

به عنوان نمونه:

n: طول آرایه

```
for(int i=0;i<n;i++)
{
    .....
}
```

```
foreach({type} {variable-name} in {collection})
{
    .....
 }
```

خواص(properties)

- property استانداردی در سی شارپ برای دسترسی به ویژگی های یک شی از یک کلاس است.
- بسیاری از اشیای ذاتی Net framework. خواص مفید زیادی را همراه خود دارند مانند طول رشته، اندازه فونت، عنوان فرم و....
- برای استفاده از یک خاصیت لازم است تا کلاس تعریف کننده شی در برنامه فراهم شده باشد.(منظور استفاده از فضای نام مربوطه می باشد)

خواص(properties)

- به عنوان نمونه:
- برای به دست آوردن زمان و تاریخ جاری سیستم می توان از خاصیت Now مربوط به شی DateTime استفاده کرد.
- برای این منظور، به صورت زیر عمل می شود:
- System.DateTime.Now;

خاصیت Length

- فرض کنید، طول آرایه ای که تعریف شده است را نمی دانید!
- در چنین موقعیتی می توان با خاصیت Length طول آرایه را بدست آورد.
- مثال:

```
string[] str={“a”,”b”,”c”}
int lnth=str.Length;
Result: lnth= 3
```

مثال

برنامه ای بنویسید که تمامی اعداد فرد کوچکتر از ۱۰۰ را در یک آرایه ذخیره کرده و چاپ نماید. برای این منظور باید به صورت زیر عمل کنید:

- یک حلقه ایجاد کنید که اعداد یک تا ۱۰۰ را پیمایش کند.
- داخل حلقه باید بررسی کنید که باقیمانده ی تقسیم عدد بر ۲ برابر با یک باشد(مطابق تعریف اعداد فرد).
  - اگر باقیمانده ی تقسیم بر ۲ برابر با یک بود باید آن را در داخل عنصری از آرایه ذخیره کرده و در خروجی چاپ نماید.

مثال

- ویژوال استودیو را اجرا کنید.
- در صفحه ی باز شده، روی دکمه new project کلیک نمایید.
- از پنل project گزینه VisualC# Project را انتخاب نمایید.
- از پنل سمت چپ، گزینه ASP.net Web Application را انتخاب نمایید.
- در قسمت location، نام Arraytest را برای پروژه ی خود در دایرکتوری Home انتخاب نمایید.

مثال

- ✓ با استفاده از جعبه ابزار یک کنترل Label با خاصیت ID برابر با lblarr و یک کنترل Button با خاصیت ID برابر با btnarr و خاصیت text برابر با "نمایش" به فرم اضافه کنید.
- ✓ روی کنترل Button ۲بار کلیک کرده و کد مربوطه را بنویسید.

کلاس ها و متدها

- متدها قطعه کدهایی هستند که هر یک، وظیفه ای مشخص را در برنامه انجام می دهند.
- متدها در واقع همان رفتارهای کلاس هستند.
- بسیاری از کلاس های Net framework. متدها و توابع مفید و کاربردی آماده ای را دارا می باشند.

کلاس ها و متدها

- برای تعریف یک کلاس به صورت زیر عمل می شود:

```
Class classname
{
}
```

- برای تعریف متد به صورت زیر عمل می شود:

```
{modifier} {return-type} {name}({parameters})
{
   // method body
}
```

کلاس ها و متدها

- {modifier} نمایانگر سطح دسترسی به متد می باشد. مانند public، private و...
- {return-type} نوع بازگشتی متد را مشخص می کند.
  - متدها می توانند نوع بازگشتی داشته باشند یا نداشته باشند.
  - برای مثال زمانی که متدی با نوع بازگشتی string نوشته می شود، جای {return-type} کلمه کلیدی string نوشته می شود.
  - مقدار مورد نظر داخل بدنه متد با کلمه کلیدی return برگردانده می شود.
  - اگر متد نوع بازگشتی نداشته باشد از کلمه void استفاده می شود.

مدرس:سمیه عادلی