C# - CODE

Reading and writing to a console……………………………….……..……..

C# Built -in data Type…………………………………………..……..……..

C# String data Type……………………………………………..……..……..

Operators………………………………………………………………..……..

Datatype ………………………………………………………….……..……..

Conversions……………………………………………………………..……..

Array …………………………………………………………….……..……..

If Statement……………………………………………………….……..……..

Switch …………………………………………………………….……..……..

While Loop………………………………………………………………..……..

Methods…………………………………………………………….……..……..

Methods & Methods Parameter…………………………………………..……..

Namespaces……………………………………………………………………..

Class-introduction …………………………………………………………..……..

Static & instance members……………………………………………………

Inheritance……………………………………………………………………

Method hiding ……………………………………………………………………

Polymorphism……………………………………………………………………

Method overloading Vs hiding……………………………………………………

Method overloading………………………………………………………………..

Why properties …………………………………………………………………..

Properties…………………………………………………………………………..

Structs…………………………………………………………………….……..

Classes Vs Structs …………………………………………………………..……..

Interfaces……………………………………………………………………….

Explicit interface implementation……………………………………………………

Abstract classes ………………………………………………………………..

Abstract classes Vs interface……………………………………………………

Reading and writing to a console

Console.WriteLine()

ينزل سطر جديد

Console.Write ()

بدون ما ينزل سطر جديد

ملاحظة \*

بعد الانتهاء من كتابة console.write لازم كتابة console.write

1. using System;
3. namespace Sample
4. {
5. class Test
6. {
7. public static void Main(string[] args)
8. {
9. Console.WriteLine("C# is cool");
10. Console.ReadLine();
11. }
12. }
13. }

Console.WriteLine("C# is cool")

Output 🡪 c# is cool

1. int value = 10;
2. // Variable
3. Console.WriteLine(value);
4. // Literal
5. Console.WriteLine(50.05);

Console.ReadLine();

Console.WriteLine(value);

Console.WriteLine(50.05);

Output 🡪 10

50.05

1. int val = 55;
2. Console.WriteLine("Hello " + "World");
3. Console.WriteLine("Value = " + val);

Console.ReadLine();

Console.WriteLine("Hello " + "World");

Console.WriteLine("Value = " + val);

Output 🡪 Hello World

Value 55

1. public static void Main(string[] args)
2. {
3. int firstNumber = 5, secondNumber = 10, result;
4. result = firstNumber + secondNumber;
5. Console.WriteLine("{0} + {1} = {2}", firstNumber, secondNumber, result);
6. Console.ReadLine();
7. }

Output 🡪 5 + 10 = 15

ملاحظة \*

كتابة Console.ReadLine معناه كتابة القيمة من الكيبورد ( انت تدخل القيمة بعد تشغيل البرنامج)

1. string testString;
2. Console.Write("Enter a string - ");
3. testString = Console.ReadLine();
4. Console.WriteLine("You entered '{0}'", testString);
5. Console.ReadLine();

Output 🡪 Enter a string – الان اثناء تشغيل البرنامج يطلب منك ادخال قيمه

You entered ‘ القيمة المدخله‘

1. string userInput;
2. int intVal;
3. double doubleVal;
4. Console.Write("Enter integer value: ");
5. userInput = Console.ReadLine();
6. /\* Converts to integer type \*/
7. intVal = Convert.ToInt32(userInput);
8. Console.WriteLine("You entered {0}",intVal);
9. Console.Write("Enter double value: ");
10. userInput = Console.ReadLine();
11. /\* Converts to double type \*/
12. doubleVal = Convert.ToDouble(userInput);
13. Console.WriteLine("You entered {0}",doubleVal);
14. Console.ReadLine();

ملاحظة \*

تتم القراءة من الكيبورد بقيمة نصيه اذا يجب عليك تحويل القيمة الى القيمه المراده بستخدام امر

Convert.typeValue

C# Built -in data Type

Example : Variables of Deafferent Data Type

امثله على تعريف البيانات ( نوع البيانات + القيمة )

string stringVar = "Hello World!!";

int intVar = 100;

float floatVar = 10.2f;

char charVar = 'A';

bool boolVar = true;

Example : Variables of Deafferent Data Type )Empty Value )

امثله على تعريف البيانات ( نوع البيانات + القيمة فارغه )

string stringVar;

int intVar;

float floatVar;

char charVar;

bool boolVar;

Example: You can use (var) data type like

var stringVar = “ABCD”;

var integerVar = 10000;

ملاحظة \*

الفرق بين var وكتابة النوع مثل int , string ,….

اذا كان عندك عدة كلاسات وفيه كلاس واحد تبي تدعي اسم متغير وتبي بسرعه تدعيه بدون من تدوره بالكلاسات وتعرف نوعه تحط var وهو يستدعي المتغير حسب النوع المعطى سابقاً

C# String data Type

// Declare without initializing.

// تعريف متغير نوع نص بدون قيمه

string message1;

// Initialize to null.

// تعريف متغير نص قيمه فارغه

string message2 = null;

//Initialize with a regular string literal.

// تعريف متغير مع قيمة

string oldPath = "c:\\Program Files\\Microsoft Visual Studio 8.0";

Example 1

string s1 = "Hello ";

string s2 = s1;

s1 += "World";

System.Console.WriteLine(s2);

Output 🡪 Hello

ملاحظة\*

طبع Hello فقط ،، لانه يقراء سطر سطر

Example 2

string s1 = "Hello ";

s1 += "World";

string s2 = s1;

Console.WriteLine(s2);

Output 🡪 Hello World

ملاحظة\*

طبع Hello World

الملخص في هذا المثال : s1 فيه قيمه Hello وبعده اضفنا قيمة World الان صبحة قيمة s1 ( Hello World )

ومن ثم اضفنا متغير جديد واعطيناه قيمة s1

الان طباعة s2

الناتج ( Hello World (

Operators

ملاحظة \*

تقدر تستخدم var

كذالك تقدر تستخدم المتغيرات int …

var a = 2 + 2 \* 2;

Console.WriteLine(a);

Output 🡪 6

var a = (2 + 2) \* 2;

Console.WriteLine(a)

Output 🡪 8

int a = 13 / 5 / 2;

int b = 13 / (5 / 2);

Console.WriteLine($"a = {a}, b = {b}");

Output 🡪 a = 1, b = 6

// n طريقة جمع عددين

double x, y, r;

Console.Write("this is Calcution \n X: ");

// n البانات القادمه من نوع سترنق الان لازم احولها الى نوع دبل

x = Convert.ToDouble(Console.ReadLine());

Console.Write(" Y: ");

y = Convert.ToDouble(Console.ReadLine());

r = x + y;

Console.WriteLine(Convert.ToSingle(r));

Datatype Conversions

عندنا في هذا المثال قيمه نوع نص ،، اضفنا قيمه نوع ارقام والمطلوب منه تحويل القيمه النصيه الى قيمه رقميه

string sAge = "26";

int age;

age = Convert.ToInt32(sAge);

Console.WriteLine( sAge);

Output 🡪 26

ملاحظة \*

دائماً القيمه المستقبله من الكيبورد قيمه نصيه يجب عليك تحويلها الى أي قيمه مراده

Array in C#

تعريف مصفوفه بدون قيم

// defining array with size 5.

// But not assigns values

int[] intArray1 = new int[5];

تعريف مصفوفه مع إضافة قيم

// defining array with size 5 and assigning

// values at the same time

int[] intArray2 = new int[5] {1, 2, 3, 4, 5};

We can assign values as follows:

myNumber [0] = 10;

myNumber [1] = 22;

myNumber [2] = 14;

myNumber [3] = 66;

myNumber [4] = 22;

Example

// تعريف مصفوفه بدون قيم

string[] names = new string[5];

// تعريف مصفوفه مع إضافة القيم

int[] number = { 1,2,3,4,5,6,7};

names[0] = "A";

names[1] = "B";

names[2] = "C";

names[3] = "D";

names[4] = "E";

Console.WriteLine(names[2]);

Console.WriteLine(number[4]);

Output 🡪 C

5

//The array can be defined as follows:

//تستطيع تعريف المصفوفة كما يلي

int[] myNumber = new int[5];

//We can assign values as follows:

//تستطيع إسناد قيم كما يلي

myNumber[0] = 10;

myNumber[1] = 22;

myNumber[2] = 14;

myNumber[3] = 66;

myNumber[4] = 22;

//The cell in brackets [] is called index.

//Now, what if I wanted to do a call to values?

//الآن، ماذا لو أردنا إستدعاء القيم؟

Console.WriteLine(myNumber[2]);

Output:🡪

14

//Note:

//There is another way to define the array and assign the values directly.

//يوجد طريقة أخرى لتعريف المصفوفة وإسناد قيم لها مباشرة

int[] myNumbes = new int[] { 1, 5, 3, 4, 66, 55 };

//We can use any other type!

//نستطيع إستخدام نوع آخر

string[] myStrings = new string[] { "Ahmad", "Mohammad", "Shady", "Safaa" };

Console.WriteLine(myStrings[2]);

Output:🡪

Shady

ترتيب العداد في المصفوفه

// **sort int array**

int[] intArray = new int[5] { 8, 10, 2, 6, 3 };

**Array.Sort**(intArray);

// write array

foreach (int i in intArray)

Console.Write(i + " ");

Output 🡪 2 3 6 8 10

// **sort string array**

string[]stringArray= new string[5]{ "X", "B", "Z", "Y", "A" };

**Array.Sort**(stringArray);

// write array

foreach (string str in stringArray)

Console.Write(str + " ");

Output 🡪 A B X Y Z

If Statement

Let's take an example directly:

public static void Main()

{

Console.WriteLine("\*\*Ahmed's Big Givaway\*\*");

Console.Write("Choose 1,2 or 3 : ");

string userValue = Console.ReadLine();

string message;

if (userValue == "1")

message = "Wow!! You won a new car";

else if (userValue == "2")

message = "Wow!! You won a new boat";

else if (userValue == "3")

message = "Wow!! You won a soup";

else

{

message = "wrong choice";

message = message + ", You Lose!!";

}

Console.WriteLine(message);

Console.ReadLine();

}

Note:

//We need to set all the possibilities that the user will enter.

Switch Statement

سويتش تعتبر شبيه في if ولكن بشكل ارتب

// تعريف متغير

int i = 1;

// ادخال المتغير في دالة switch

switch (i)

{

case 1:

Console.WriteLine("One");

break;

case 2:

Console.WriteLine("Two");

break;

default:

Console.WriteLine("Other");

break;

}

Output 🡪 One

// واحد او اثنين جميعها تنفذ نفس الشرط

int i = 1;

switch (i)

{

case 1:

case 2:

Console.WriteLine("One or Two");

break;

default:

Console.WriteLine("Other");

break;

}

Output 🡪 One or Tow

المطلوب من هذاالمثال ادخال رقم الشهر من قبل الكيبورد ويطبع لك اسم الشهر

Console.WriteLine("Write number of Months");

// كما تعلمنا في السابق الكيبورد يستقبل قيمه نصيه لذا يجب عليك تحويل القيمه الى قيمه رقميه

int month = Convert.ToInt32(Console.ReadLine());

switch (month)

{

case 1: Console.WriteLine("jan");

break;

case 2: Console.WriteLine("Feb");

break;

case 3: Console.WriteLine("March");

break;

case 4:Console.WriteLine("April");

break;

case 5: Console.WriteLine("May");

break;

case 6: Console.WriteLine("June");

break;

case 7: Console.WriteLine("July");

break;

case 8:Console.WriteLine("Aug");

break;

case 9: Console.WriteLine("Sep");

break;

case 10: Console.WriteLine("Oct");

break;

case 11: Console.WriteLine("Nov");

break;

case 12:Console.WriteLine("Dec");

break;

// اذا ما تنفذ الشرط ينفذ هذه الداله

default:

Console.WriteLine("Sorry");

break;

}

While loop

The structure of While loop:

data type x = value;

while (test-expression)

{

// body of while

}

If the test-expression is evaluated to true, statements inside the while loop are executed.

after execution, the test-expression is evaluated again.

Until the execution is evaluated to false, the while loop terminates.

//Example:

public static void Main(string[] args)

{

int i = 1;

while (i <= 5)

{

Console.WriteLine("C# For Loop: Iteration {0}", i);

i++;

}

}

Output:

C# For Loop: Iteration 1

C# For Loop: Iteration 2

C# For Loop: Iteration 3

C# For Loop: Iteration 4

C# For Loop: Iteration 5

For & foreache loop

For loop can be defined as follows:

For ( int i = 0; i < 10 ; i++ )

//Example:

public static void Main()

{

for (int i = 0; i < 10; i++)

{

Console.WriteLine(i);

}

}

Output:

0

1

2

3

4

5

6

7

8

9

Printing array using foreache loop

public static void Main(string[] args)

{

char[] myArray = { 'H', 'e', 'l', 'l', 'o' };

foreach (char ch in myArray)

{

Console.WriteLine(ch);

}

}

Output:

H

e

l

l

o

Methods

ماھي الدالة : الدالة عباره عن مجموعة أوامر بداخلھا لھا وظیفھ محدده ولھا اسم ،، یمكن استدعاءھا من خلال مناداتھا باسمھا دون الحاجھ لكتابة الاكواد الموجوده داخلھا مره أخرى

Access\_Specifier: It is used to define an access level either public or private, etc. to allow other classes to access the method. If we didn’t mention any access modifier, then by default it is private.

Return\_Type: It is used to specify the type of value the method can return. In case, if method is not returning any value, then we need to mention void as return type.

Parameters: The method parameters are used to send or receive a data from method and these method parameters are enclosed within parentheses and are separated by commas. In case, if no parameters are required for a method then, we need to define a method with empty parentheses.

public static void Main()

{

PrintMessage();

Console.ReadLine();

}

private static void PrintMessage()

{

Console.WriteLine("Hello World !");

}

Output:

Hello World !

Methods & Methods parameter

NameSpace

فوائدها :

- تساعد على ترتيب الكود

- عند انشاء كلاسات تحتوي على نفس اسماء المتغيرات نستخدم NameSpace للفصل بينهم فلا يحدث تعارض

Namespace also solves the problem of **naming conflict**. Two or more classes when put into different namespaces can have same name.

We can define a namespace in C# using the *namespace*keyword as:

namespace Project

{

//Body of namespace

}

For example:

1. namespace Project
2. {
3. class MyClass
4. {
5. public void Message))
6. {
7. System.Console.WriteLine("Creating my namespace");
8. }
9. }
10. }

In the above example, a namespace Project is created. It consists of a class MyClass as its member. Message is a method of class MyClass.

**Accessing Members of Namespace in C#**

The members of a namespace can be accessed using the dot(.) operator. The syntax for accessing the member of namespace is,

طريقة الوصول للنيم سبيس والكلاس وليس اسمائها الحقيقة

Namespace\_Name.Member\_Name // شرح توضيحي لطريقة الوصول

Project.Myclass // طريقة المناداة, اسم النيم سبيس.اسم الكلاس

Example :

Porject.MyClass C = new Project.MyClass();

**Example 1: Introducing Namespace in C# Program**

1. using System;
2. namespace Project
3. {
4. public class Example
5. {
6. public static void Message()
7. {
8. Console.WriteLine("Creating my namespace");
9. }
10. }
11. }
13. namespace MyProgram
14. {
15. public class MyClass
16. {
17. public static void Main()
18. {
19. Project.Example.Message();
20. // لطباعة المسج نقوم بالوصول لها عن طريق الدخول لproject ثم الى
21. example ثم Message ويطبع مابداخلها
22. }
23. }}

// When we run the program the output will be : Creating my namespace

## Using a Namespace in C# : (using) Keyword

A namespace can be included in a program using the (using) keyword. The syntax is,

using Namespace\_Name; // اسم توضيحي لطريقة كتابة النيم سبيس

بدلا من كتابة system في كل مرة نستخدم فيها console مثل هذا المثال :

System.Console.WriteLine("Hello World!");

نضع كلمة system بداية الملف اختصارا للوقت

using System;

وفي كل مرة نستخدمها نكتبها هكذا بدون system

Console.WriteLine("Hello World!");

## Nested Namespace in C#

A namespace can contain another namespace. It is called nested namespace. The nested namespace and its members can also be accessed using the dot (.) operator.

Nested namespace بالامكان وجود اكثر من نيم سبيس وتسمى

وطريقة الوصول اليها بنفس الطريقة السابقة

The syntax for creating nested namespace is as follows:

namespace Project

{

namespace NestedNamespace

{

// Body of nested namespace}

}

Example : Nested Namespace in C#

1. using System;
3. // Nested Namespace
4. namespace Project
5. {
6. namespace Nested
7. {
8. public class Example
9. {
10. public static void Message()
11. {
12. Console.WriteLine("Nested Namespace Example");
13. }
14. }
15. }
16. }
18. namespace MyProgram
19. {
20. public class MyClass
21. {
22. public static void Main()
23. {
24. MyNamespace.Nested.SampleClass.myMethod();
25. }
26. }
27. }

When we run the program, the output will be:

Nested Namespace Example

Class

الكلاس عبارة عن القالب الأساسي للكتابة ويكون بداخله تعريف المتغيرات والميثود

لتعريف الكلاس لابد من تسميته , ولا ننسى الاقواس

Example:

public class users {

// Properties, Methods, Events, etc.

}

Method overloading

هي عندما تكون لدينا عمليات لديها نفس الاسم وفي كل مرة نريد استخدامها بطرق مختلفة

له 3 أنواع :

النوع الأول ( عدد البراميتر ) : -

using System;

class number of parameter {

    // adding two integer values.

    public int Add(int a, int b)

لدينا 2 ميثود لديها نفس الاسم  
**Add**

وفي المثال الأول عدد البراميتر 2

وفي المثال الثاني عدد البراميتر 3

    {

        int sum = a + b;

        return sum;

    }

    // adding three integer values.

    public int Add(int a, int b, int c)

    {

        int sum = a + b + c;

        return sum;

    }

النوع الثاني ( نوع البراميتر ) : -

using System;

class different\_data\_type {

    // adding three integer values.

    public int Add(int a, int b, int c)

لدينا 2 ميثود لديها نفس الاسم  
**Add**

ولكن نوع البيانات مختلفة

integerفي المثال الأول:

doubleفي المثال الثاني:

    {

        int sum = a + b + c;

        return sum;

    }

    // adding three double values.

    public double Add(double a,

                      double b, double c)

    {

        double sum = a + b + c;

        return sum;

    }

النوع الثالث ( ترتيب البراميتر ) : -

using System;

class GFG {

    // Method

لدينا 2 ميثود لديها نفس الاسم  
**Identity**

Id ثم nameفي المثال الأول جاءت

name ثم idفي المثال الأول جاءت

    public void Identity(String name, int id)

    {

        Console.WriteLine("Name : " + name + ", "

                          + "Id : " + id);

    }

    // Method

    public void Identity(int id, String name)

    {

        Console.WriteLine("Name : " + name + ", "

                          + "Id : " + id);

    }

C# Static class

**ملاحظة مهمة :**

**Object اذا المتغير ستتك ستكون قيمته ثابته مع ال**

**للوصول لأي متغير او ميثود من نوع ستتك أسم الكلاس (.)اسم المتغير او الميثود**

public static class MyStaticClass

{

public static int myStaticVariable = 0;

public static void MyStaticMethod()

{

Console.WriteLine("This is a static method.");

}

public static int MyStaticProperty {

get; set; }

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine(MyStaticClass.myStaticVariable);

// قمنا باستدعائه عن طريق اسم الكلاس(.)اسم المتغير , هذه الحالة فقط اذا كان نوعه ستتك

MyStaticClass.MyStaticMethod();

// قمنا باستدعاءه عن طريق اسم الكلاس(.)اسم المثود لانه نوعه ستتك

MyStaticClass.MyStaticProperty = 100;

// يمكن اسناد قيمه للمتغير بنفس الطريقة

Console.WriteLine(MyStaticClass.MyStaticProperty);

// ومن ثم استدعائه للطباعه بنفس الطريقة السابقة

}

}

The output is :

0

This is a static method.

100

**ملاحظه مهمه :**

All the static members can be access using className without creating an object of a class e.g. MyStaticClass.MyStaticMethod().

It is also possible to have static members in non-static classes just like a normal class. You can instantiate non static classes using the new keyword as usual. However, the instance variable can only access the non-static methods and variables, it cannot access the static methods and variables.

ملاحظه مهمة :

اذا الممبرز (المتغير او الميثود) والكلاس ستتكك , لانحتاج لانشاء أوبجكت ويتم استدعائها بطريقة السابقة.

اذا الممبرز ستتك (المتغير او الميثود) والكلاس ليس ستتك , نحتاج لانشاء أوبجكت

Static Constructor تكون موجود ف حالتين :

1. In none static class
2. In static class

A static or non-static class can have a static constructor without any access modifiers like public, private, protected, etc.

A static constructor in a non-static class runs only once when the class is instantiated for the first time.

A static constructor in a static class runs only once when any of its static members accessed for the first time.

Static Constructor in Static Class

public static class MyStaticClass

{

static MyStaticClass()

//هذا الكونستركتر نوعه ستتك

{

Console.WriteLine("Inside static constructor.");

}

public static int myStaticVariable = 0;

public static void myStaticMethod()

{

Console.WriteLine("This is static method.");

}

public static int MyStaticProperty { get; set; }

}

class Program

{

static void Main(string[] args)

{

MyStaticClass.myStaticVariable = 100;

MyStaticClass.MyStaticProperty = 200;

MyStaticClass.myStaticVariable = 300;

MyStaticClass.MyStaticProperty = 400;

}

}

The output is :

Inside static constructor.

Static constructor in a non-static class

public class MyNonStaticClass

{

static MyNonStaticClass()

{

Console.WriteLine("Inside static constructor.");

}

public void myNonStaticMethod()

{

Console.WriteLine("Non-static method");

}

}

class Program

{

static void Main(string[] args)

{

MyNonStaticClass mnsObj1 = new MyNonStaticClass();

MyNonStaticClass mnsObj2 = new MyNonStaticClass();

MyNonStaticClass mnsObj3 = new MyNonStaticClass();

}

}

The output is :

Inside static constructor

Inheritance

فائدة الوراثة :

Now you can reuse the members of your parent class. So, there is no need to define the

member again. So less code is required in the class.

Example 1:

public class X {

// كلاس السوبر

    public void ADD()

    {

        // Method implementation

    }

}

public class Y : X

{

    // كلاس السب يرث من السوبر .. super = X , sub = y

}

class Program

{

    static void Main(string[] args)

    {

        Y Obj = new Y();

        Obj.ADD(); // هنا راح يسوي امبلمنتيشين للمثود الموجود بالسوبر كلاس

    }

}

Example 2 :

using System;

namespace ConsoleApplication1 {

// Base class

class Proj {

   // data members

    public string name;

    public string subject;

    // public method of base class

    public void readers(string name, string subject)

    {

        this.name = name;

        this.subject = subject;

        Console.WriteLine("Myself: " + name);

        Console.WriteLine("My Favorite Subject is: " + subject);

    }

}

class B : Proj {

   // inheriting the Proj class using ( : ) اشارة النقطتين الرأسيتين تدل على ان الكلاس بي يرث كلاس السوبر

    public B()

    {

        Console.WriteLine("Hello PEOPLE");

    }

}

// Main class

class Test {

    // Main Method

    static void Main(string[] args)

    {

        // creating object of derived class

        B g = new B(); // راح ينادي الكونستركتر الموجود في السب كلاس اللي ماله براميتير

        // calling the method of base class

        // using the derived class object

        g.readers("Kirti", "C#"); // راح يطبع مثود الريدرز واسند لها قيم

    }

}

}

The output is :

Hello PEOPLE

Myself: Kirti

My Favorite Subject is: C#