C#

Base Code Reference

Base Code Reference

- **Base C# Page**
- **Data Type**
- **Variable**
  - Var
  - Const
- **= Operator**
- **Convert**
  - Cast
  - Convert
  - Parse
- **Console**
- **Operators**
  - Mathematical
  - Logical
  - Bit
- **Selection**
  - if
  - switch
- **Try-Catch**

- **Array**
  - Array
  - 2D Array
- **Collection**
  - List
  - Queue
  - Stack
- **Loops**
  - for
  - while
  - do-while
  - foreach
- **Text File**
- **Function**
- **Jumping**
  - break
  - continue
  - return
  - goto

- **String**
  - Length
  - Replace
  - Substring
  - Contains
  - Remove
  - IndexOf ,LastIndexOf
  - Trim
  - Split
  - ToCharArray
- **Class (Object)**
- **Database**
  - SQL
  - DatabaseContext
    - Model
    - DbContext
- **Database : CRUD (EF)**
  - **C**reate (Insert)
  - **R**ead (Select)
  - **U**pdate
  - **D**elete

Base C# page

```
using ...[library]...;

namespace ...[project name]...
{
    class Program
    {
        static void Main(string[] args)
        {
            ...[main code]...
        }
    }
}
```

Data type

- **int** (integer) ➔ *[8 bit(byte), 16 bit(short), 32 bit, 64 bit(long)]*
- **float** ➔ *[32 bit(7-8digit)]*
- **double** ➔ *[64 bit(15-16 digit)]*
- **decimal** ➔ *[128 bit(28-29 digit)]*
- **char** (character) ➔ *16 bit]*
- **string** ➔ *[length*sizeOfChar]*
- **bool** (boolean) ➔ *[true, false]*
- **object** ➔ *[parent!]*

*// **u** ➔ unsigned integer **[positive integer] => ushort, uint, ulong***

*// **s** ➔ signed integer **[negative & positive] => sbyte***

Data type

Type	Size (bit)	Range
byte	8	0 to 255
sbyte	8	-128 to 127
short	16	-32768 to 32767
ushort	16	0 to 65535
int (integer)	32	-2147483648 to 2147483647
unit	32	0 to 4294967295
long	64	-9223372036854775808 to 9223372036854775807
ulong	64	0 to 18446744073709551615
*char (character)	16	0 to 65535

Type	Size (bit)	Precision
float	32	7 digits
double	64	15-16 digits
decimal	128	28-29 digits

Type	Value
string	"hello"
*char (character)	'a'
bool (boolean)	true , false

Variable

[dataType] [variableName] = [defaultValue];

- int i = 5; *// 5+2 => 7*
- float f = 3.5f; *//wrong : 3.5*
- double db = 2.7;
- decimal dc = 0.346;
- char ch = 'a'; *//wrong : 'ali' , 'm'+'y' => digit!!!(ASCII Code)*
- string str = "my name"; *// "ali"+"reza" => "alireza" , "5"+"2" => "52"*
- bool b = false;

*// **[defaultValue]** optional : int number;*

*// **[variableName]** wrongs : int **2a**; int **string**; int **my num**;*

Variable

- **var** *[variableName]* **=** *[defaultValue]*;

```
var a = 5;
var b="ali";
var c = 3.5;
```

- **const** *[dataType] [variableName]* **=** *[defaultValue]*;

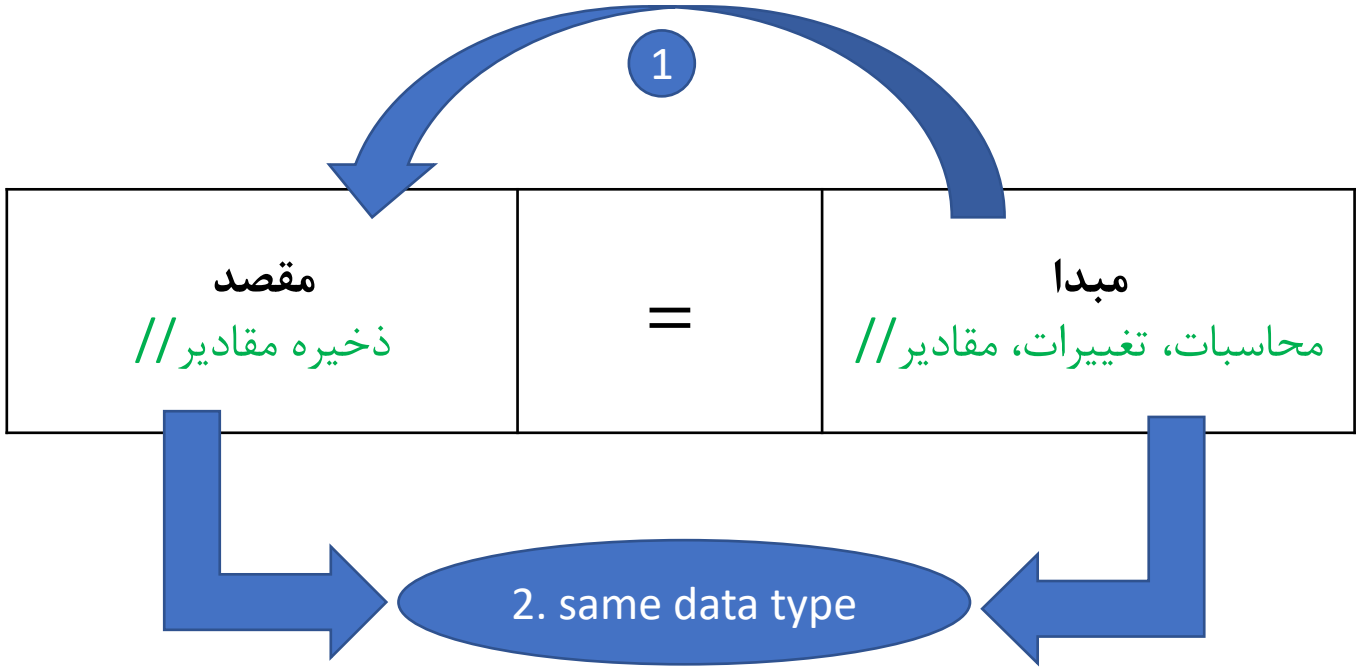
```
const  double  pi = 3.14;
```

*// **[defaultValue]** not optional ➔ wrong : var a;*

*//**const** wrong : const double pi = 3.14;*

pi = pi + 2;

= operator



- **قانون اول**

مسیر حرکت محاسبات از راست به چپ

- **قانون دوم**

دوسر عملگر = باید هم نوع باشند

// در غیر اینصورت مقدار مبدا به نوع مقصد **تبدیل** خواهد شد

Convert

- **(***[dataType]***)** *[value]//just numerical data type & value*

```
int number = (int) 3.5;
```

- *[dataType]***.Parse(***[string value]***)**

```
int number = int.Parse(3.5);//خطا
int number = int.Parse("12");//12
```

- **Convert.To***[dataType]***(***[value]***)**

```
string str = Convert.ToString(12);//"12"
```

- *[value]***.ToString()**

```
int number = 5;
string str = number.ToString();//"5"
```

Console

- using **System**;*//library*
- Console.**ReadLine()** *//string value*

```
string name = Console.ReadLine();
int number = int.parse(Console.ReadLine());
```

- Console.**WriteLine(***[value]***)** | Console.**Write(***[value]***)**

```
string result = "ali";  |  int result = 5+7;
Console.WriteLine(result);  |  Console.Write(result);
```

- Console.**ReadKey**()

```
Console.ReadKey();//wait for key
```

Console

```
//Example
int num = 5;
string str = "5";
Console.WriteLine(num + str);

//Example
ConsoleKeyInfo ch = Console.ReadKey();
Console.WriteLine(ch.Key);
Console.WriteLine(ch.KeyChar);
```

Console

- ConsoleColor.***[colorName]***

ConsoleColor.Red

- Console.**BackgroundColor** = ConsoleColor***.[colorName]***;*//back color*
- Console.**ForegroundColor** = ConsoleColor***.[colorName]***;*//font color*

```
Console.BackgroundColor = ConsoleColor.Blue;
Console.ForegroundColor = ConsoleColor.Yellow;
```

```
Console.WriteLine("text");
```

- Console.**Clear()**;

Operators

- Mathematical Operators

*	/	%	+	-	++	--

- Logical operators

>	>=	<	<=	==	!=	!

- Bit operators

~	<<	>>	&	|	^

Selection

- **if - else**

```
if ([???])
{
    //output 1
}
else
{
    //output 2
}
```

string password ← *[get value]*
string userPassword ← *[input value]*

```
if (password == userPassword)
{
    //login
}
else
{
    //error
}
```

Selection

- **[condition] ? [true] : [false];**

int **input** ← *[input value]*

string **result = input%2==0 ? "even" : "odd"**

[output] → **result**

Selection

- **if - else if**

```
if ([?1?])
{
    //output 1
}
else if ([?2?])
{
    //output 2
}
else if ([?3?])
{
    //output 3
}
...
else
{
    //output n
}
```

Color lightColor ← *[input value]*

```
if (lightColor == red)
{
    //STOP
}
else if (lightColor == Green)
{
    //GO
}
else if (lightColor == yellow)
{
    //SLOW
}
else
{
    //driver
}
```

Selection

- **OR(||) , AND(&&)**

if (*[condition]* AND *[condition]*) ➔ if (*[condition]* **&&** *[condition]*)

if (*[condition]* OR *[condition]*) ➔ if (*[condition]* **||** *[condition]*)

OR		
Condition 1	*Condition 2*	**||**
T	T	**T**
T	F	**T**
F	T	**T**
F	F	**F**

AND		
Condition 1	*Condition 2*	**&&**
T	T	**T**
T	F	**F**
F	T	**F**
F	F	**F**

T ➔ True
F ➔ False

Selection

If (a > 0 && a < 10) = ?

If (a < 0 || a > 10) = ?

Selection

- **switch - case**

Switch (*[variable]***)**
{
case *[value]* **:** *//outpot 1*;**break;**
case *[value]* **:** *//outpot 2*;**break;**
case *[value]* **:** *//outpot 3*;**break;**
…
default : *//outpot 4*;**break;**
}
*//**note:** variable & values => same dataType*
*//**break** : jump out of switch & every statement*

char letter ← *[input value]*

```
switch (letter)
{
   case 'b' : //bahar; break;
   case 't' : //tabestan; break;
   case 'p' : //paeiz; break;
   case 'z' : //zemestan; break;
   default : //invalid input; break;
}
```

Try-Catch

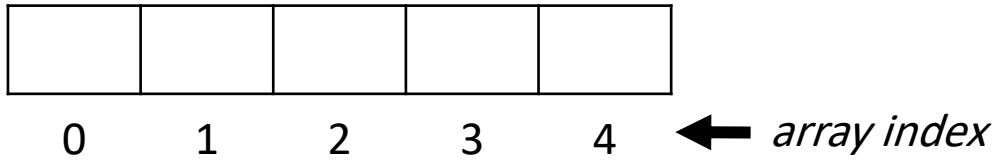
```
try
{
        //your main code
}
catch (Exception)
{
        throw;//error place and details
}
```

```
try
{
        //your main code
}
catch (Exception e)
{
        //e ➔ error details
}
finally
{
        //optional : after try or catch, final run
}
```

Array

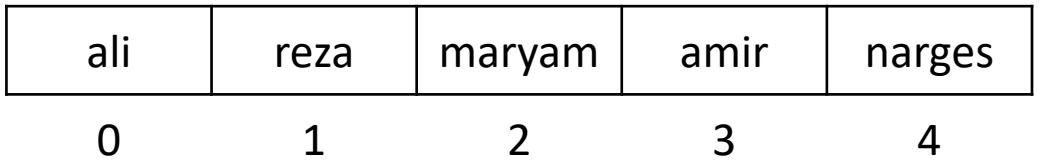
- dataType**[]** arrayName = **new** dataType**[*arraySize*]**;

  int[] arrNum = new int[5]; //size & endOfIndex



- dataType**[]** arrayName = **{***item1***,** *item2***,** *item3***, …,** *itemn***}**; //same type datatype

  string[] arrStud = {"ali", "reza", "maryam", "amir", "narges"};

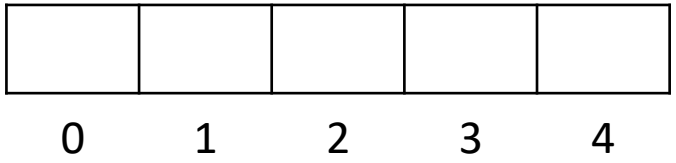


Array

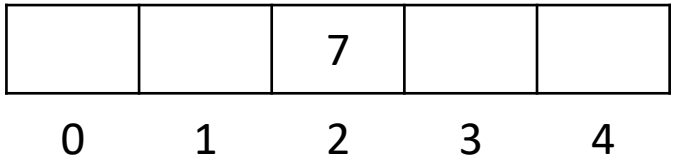
- set value

  arrName[***index***] = ***value***;

  int[] arrNum = new int[5];



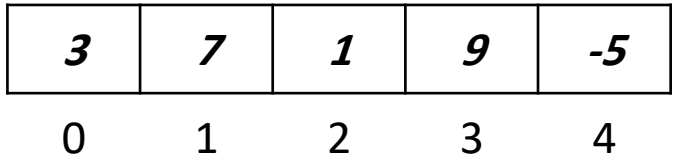
  arrNum[2] = 7;



- get value

  variable = arrName[***index***] ;

  arrNum



  int n = arrNum[3];

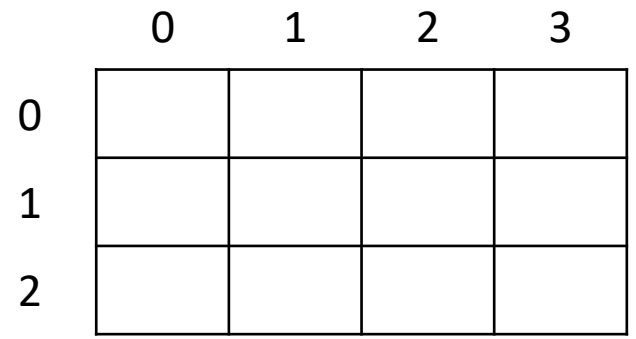
  *// wrong : int n = arrNum[**5**], arrNum[**5**] = 7*

  out of bound!!!

2D Array

- dataType**[,]** arrayName = **new** dataType**[*rowSize , columnSize*]**;

  int[] arrNum = new int[3,4];



- dataType**[]** arrayName = **{***{item1,…, itemn}***, … ,** *{item1,…, itemn}***}**;

  string[] arrStud = {{"ali", "reza"}, {"maryam", "amir"}};

	0	1
0	ali	reza
1	maryam	amir

2D Array

- set value

arrName[***rowIndex, columnIndex***] = ***value***;

```
int[] arrNum = new int[3,4];
arrNum[1,3] = 7;
```

	0	1	2	3
0				
1				7
2				

- get value

variable = arrName[***rowIndex, columnIndex***] ;

arrPerson

	0	1
0	ali	reza
1	maryam	amir

```
string name = arrPerson[0,1];
```

*//wrong : arrPerson[**2,2**],arrPerson[**1,2**],*
out of bound!!!

Collection

- **List**

1. List<***[dataType]***> ***[listName]*** = new List<***[dataType]***>();
2. List<***[dataType]***> ***[listName]*** = new List<***[dataType]***> {*item1*, *item2*, …, *itemn*};

[listName].**Add(**item**)**

[listName].**ElementAt(**index**)** يا *[listName]* **[**index**]**

[listName].**RemoveAt(**index**)**

Collection

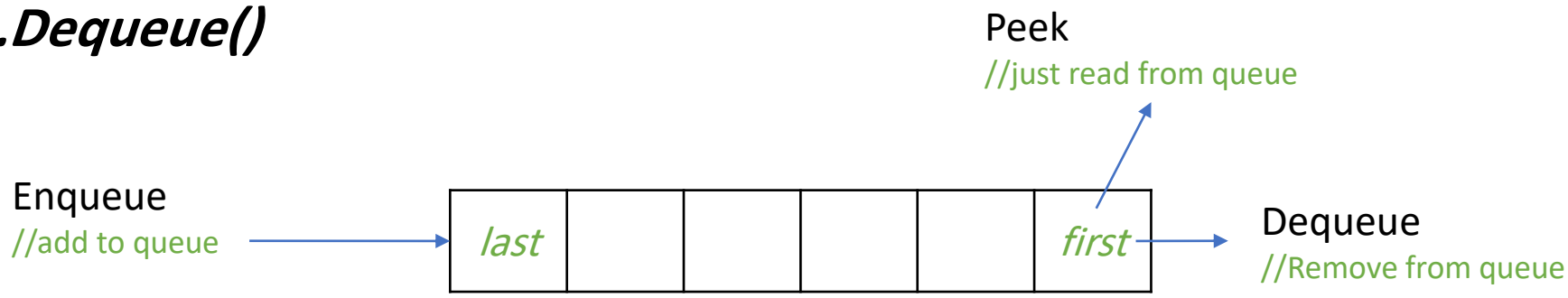
- **Queue**

Queue<***[dataType]***> ***[listName]*** = new Queue <***[dataType]***>();

[listName].**Enqueue***(item)*

[listName].***Peek()***

[listName].***Dequeue()***



//note : *first in first out ➔ FIFO*

Collection

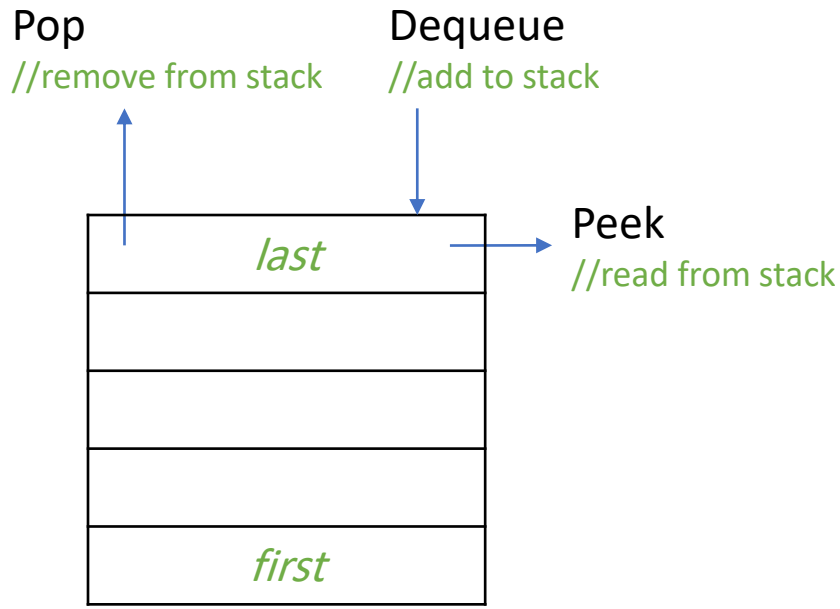
- **Stack**

Stack<***[dataType]***> ***[listName]*** = new Stack <***[dataType]***>();

[listName].***Push(****item****)***

[listName].***Peek()***

[listName].***Pop()***



*//**note :** first in last out ➔ FILO*

Loops

- **for**

```
for(int [start] ; [condition] ; [step])
{
        //code
}
```

```
int n=0;
for(int i=0 ; i<10 ; i++)
{
        n+=1;
}
```

[output] ➔ n;

//about step
// i--
//i+=2,3,4....

Loops

- **while**

```
while([condition])
{
        //code
        //something must change condition
        //warning : infinite loop
}
```

```
int n ← [input value]
while( n>0 )
{
        [output] → n%10;
        n=n/10;
}
```

Loops

- **do while**

```
do
{
        //code
        //must be run once
} while([condition])
```

```
int n ← [input value]
do
{
        [output] → n%10;
        n=n/10;
} while( n>0 )
```

Loops

- **foreach**

foreach(var *[name]* **in** *[array* ***or*** *list]* **)**

{

//code

}

```
List<string>  myList = new List<string> {"ali", "reza", "sadegh"};
String str;

foreach(var item in myList)
{
        str += item + ",";
}
```

Text File

- Using **System.IO;** //پیش نیاز استفاده از دستورات
- String **fileAddress =**

  @""+ Application.StartupPath +"*[filename].[filetype]*";

  @""+ Application.StartupPath +"\\myFile.txt"; //:مثال

- File.**Exist(**fileAddress**);** //بررسی وجود فایل در آدرس داده شده
- File.**Create(**fileAddress**).Dispos**(); //ایجاد فایل در آدرس داده شده
- File.**ReadAllText(**fileAddress**);** //خواندن تمام محتوای فایل
- File.**WriteAllText(**fileAddress**, *[stringValue]*);**

  //بررسی وجود فایل در آدرس داده شده و نوشتن متن داده شده در فایل
  *ایجاد فایل در صورت عدم وجود آن

- ....**Dispose**(); یا ....**Close**(); //رها کردن دستور(جریان) فعلی

Text File

- **StreamWriter** //نویسنده

```
//ساخت یک نمونه نویسنده متصل به آدرس فایل
StreamWriter writer = new StreamWriter(fileAddress);
writer.WriteLine([stringValue]);
writer.Dispose(); //رها کردن فایل جهت امکان اجرای سایر دستورات روی فایل
```

- **StreamReader** //خواننده

```
//ساخت یک نمونه خواننده متصل به آدرس فایل
StreamReader reader = new StreamReader(fileAddress);
var text = reader.ReadToEnd(); //خواندن تمام محتوای فایل
reader.Dispose(); //رها کردن فایل جهت امکان اجرای سایر دستورات روی فایل
```

Text File

```
if(!File.Exist(fileAddress)) //بررسی شرط موجود نبودن فایل در آدرس داده شده
{
File.Create(fileAddress); //ایجاد فایل در آدرس داده شده
StreamWriter writer = new StreamWriter(fileAddress);
writer.WriteLine("my text save to file");
writer.Dispose(); //رها کردن فایل برای عملیات بعدی
}
else
{
StreamReader reader = new StreamReader(fileAddress);
var text = reader.ReadToEnd();
reader.Dispose(); //رها کردن فایل برای عملیات بعدی
}
```

Function

- **[public/private]** *[returnType]* *[functionName]* **(***[functionParameter]***)**

[public/private] void *[functionName]* **(***[functionParameter]***)**
{
	//functionCode
}

public ← عمومی، قابل دسترسی در کل فضای namespace پروژه
private ← اختصاصی، قابل دسترسی فقط در بدنع کد فعلی

تعریف تابع	فراخوانی (اجرا) تابع

public **void** func1(**int a**, **int b**)
{
 c = a + b;
 Console.Write(c);
}
	func1(5 , 7);
	
	یا
	
	func1(n , m);

Function

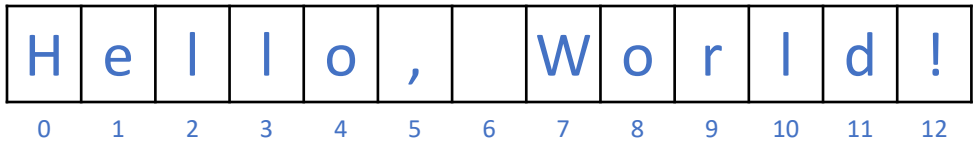
```
[public/private] [dataType] [functionName] ([functionParameter])
{
        //functionCode
        return [value]; //مقدار بازگشتی هم نوع با نوع تعریف شده در ابتدای تعریف تابع
}
```

تعریف تابع	فراخوانی (اجرا) تابع

public **int** func1(**int a**, **int b**)
{
c = a + b;
return c;
}
	int result = **func1(5 , 7)**;
	
	یا
	
	Console.Write(**func1(n , m)**);

String

string str = **"Hello, World!"**;



- **Length** //برای آرایه ها هم کاربرد دارد

str.Length ➔ خروجی ➔ 13

- **Replace(*[oldValue],[replaceValue]*)**

str.Replace("o","*") ➔ خروجی ➔ Hell*, W*rld!

- **Substring(*[startIndex],[subLength]*)** //subLength می تواند مقدار نداشته باشد

str.Substring(3 , 5) ➔ خروجی ➔ o, Wo

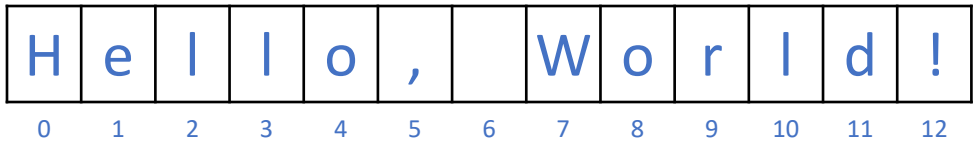
str.Substring(3) ➔ خروجی ➔ o, World!

- **Contains(*[Value]*)** //برای آرایه ها هم کاربرد دارد

str. Contains("or") ➔ خروجی ➔ true

String

string str = **"Hello, World!"**;



- **Remove(*[index],[removeLength]*)** //می تواند مقدار نداشته باشد removeLength

str.Remove(3 , 5) ➔ خروجی ➔ Hellrld!

str.Remove(3) ➔ خروجی ➔ Hell

- **IndexOf(*[value]*)** //اولین اندیس معادل

str.IndexOf("o") ➔ خروجی ➔ 4

- **LastIndexOf(*[value]*)** //آخرین اندیس معادل

str.LastIndexOf("o") ➔ خروجی ➔ 8

- **Trim()** //حذف فضای خالی ابتدا و انتهای متن

String

string str = **"Hello, World!"**;

- **Split(*[charValue]*)** //تبدیل رشته متن فعلی به آرایه از رشته های متنی

str.Split(",") ➔ خروجی ➔

"Hello"	" World!"
0	1

- **ToCharArray()** //تبدیل رشته متن فعلی به آرایه از کاراکترها

str.ToCharArray() ➔ خروجی ➔

'H'	'e'	'l'	'l'	'o'	','	' '	'W'	'o'	'r'	'l'	'd'	'!'
0	1	2	3	4	5	6	7	8	9	10	11	12

Class (object)

```
Class [className]
{
//property area
        public [propertyType] [propertyName] { get; set;}

//method area
        [public/private] [dataType] [methodName] ([methodParameter])
        {
        //methodCode
        return [value];
        }
}
//ساختار تعریف متد و تابع و کاملا مشابه است
```

Class (object)

- **دسترسی به کلاس تعریف شده در بدنه اصلی برنامه**

```
//یک نمونه (کپی) از کلاس اصلی
[className] [sampleName] = new [className]();

//مقداردهی به property کلاس (set)
[sampleName].[propertyName] = [value];

//دریافت مقدار از property کلاس (get)
var n = [sampleName].[propertyName] ;

//فراخوانی method کلاس
var result = [sampleName].[methodName](parameter);
```

Class (object)

```
Public class Circle
{
        const pi = 3.14;
        public int radius { get; set;}

        public int Enviroment()
        {
                int env = 2*radius*pi;
                return env;
        }
}
```

```
Circle circle = new Circle();
circle.radius = 7;

var result = circle.Enviroment();
```

Data Base

- **SQL**

//نرم افزار پایگاه داده

Microsoft SQL Server

SQL Server Downloads | Microsoft

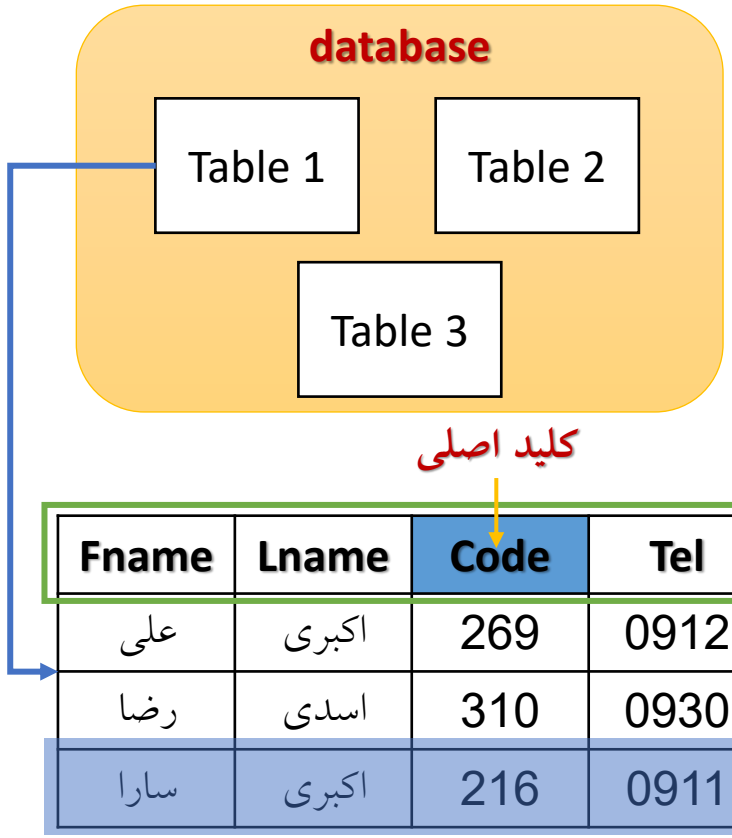
//نرم افزار مدیریت پایگاه داده

Microsoft SQL management studio

Download SQL Server Management Studio (SSMS) - SQL Server Management Studio (SSMS) | Microsoft Learn

Data Base

- **SQL**



پایگاه داده (Database)

– مجموعه یک یا چند جدول از اطلاعات

جدول (Table)

– مجموعه اطلاعات مرتبط با یک موضوع یا مسئله //مثال : اطلاعات فردی دانش آموزان

فیلد (Field)

– **عناوین** اطلاعات هر جدول شامل نام و نوع فیلد //متن، عدد، تاریخ و ...

رکورد (Record)

– یک **سطر** اطلاعات از هر جدول

کلید اصلی (Primary Key)

– فیلد منحصر به فرد (محتوای غیر تکراری) که **نباید** خالی باشد

کلید خارجی (Foreign Key)

– فیلد **ارتباط** از یک جدول به جدول دیگر

Data Base

- **DatabaseContext** // پیاده سازی ساختار دیتابیس
  - **Model** //مدل ها مجموعه ای از **ویژگی ها** که در نهایت تبدیل به **فیلدهای جدول** دیتابیس خواهند شد

```
public class [modelName]
{
        [property Attribute]//الزامی نیست
        public [propertyType] [propertyName] { get; set;}
                ...
}
```

//الزامی به تبدیل مدل به جدول دیتابیس نیست و می توان از آن صرفا برای انتقال اطلاعات مرتبط با یک موضوع استفاده کرد.

*در زیرشاخه عنوان پروژه در پنجره **Solution** یک **پوشه** به نام **Models** ایجاد کنید

Data Base

- **DatabaseContext ➔ Model**

*در زیر مجموعه پوشه Models یک Class جدید ایجاد کنید //مثال : Book.cs

```
using System.ComponentModel.DataAnnotations;

public class Book
 {
     [Key] //using ...DataAnnotations
     public int Id { get; set; }//primary key
     public string Name { get; set; }
     public string Author { get; set; }
     public string Publisher { get; set; }
     public string? Image { get; set; }
     public string Group { get; set; }
}
```

Data Base

- **DatabaseContext** // پیاده سازی ساختار دیتابیس
  - **DbContext** //فایل شامل **آدرس دیتابیس** و **لیست مدل هایی** که باید تبدیل به جداول دیتابیس شوند

```
using Microsoft.EntityFrameworkCore;

public class DatabaseContext:DbContext //using ...EntityFrameworkCore
{
     protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
     {
      optionsBuilder.UseSqlServer(@"Data Source=.; //local server
                                    Initial Catalog=[databaseName];
                                    Integrated Security=SSPI");
     }
        public DbSet<[modelName]> [tableName] { get; set; }
}
```

Data Base

- **DatabaseContext ➔ DbContext**

```
using Microsoft.EntityFrameworkCore;

public class DatabaseContext:DbContext
{
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer(@"Data Source=.;
                                          Initial Catalog=LibraryDb;
                                          Integrated Security=SSPI");
        }

        public DbSet<Book> Books { get; set; }
                                                  //مدل book تبدیل به جدول books در دیتابیس LibraryDb خواهد شد
}
```

Data Base : CRUD

Table 1: **Account**

Fname	Lname	Code	Tel
علی	اکبری	269	0912
رضا	اسدی	310	0930
سارا	اکبری	216	0911

SELECT نمایش اطلاعات

SELECT	**نام فیلد**	**From**	**نام جدول**
SELECT	Fname, Lname	**From**	Account
SELECT	Fname, Lname, Code, Tel یا *	**From**	Account

SELECT جستجو

SELECT	**نام فیلد**	**From**	**نام جدول**	**where**	**شرط**
SELECT	*	**From**	Account	**where**	Lname = 'اکبری'

DELETE حذف رکورد

DELETE	**From**	**نام جدول**	**where**	**شرط**
DELETE	**From**	Account	**where**	Code = 216

INSERT ثبت رکورد

INSERT	**Into**	**نام جدول (فیلدها)**	**values**	**مقادیر متناظر هر فیلد**
INSERT	**Into**	Account(Fname, Lname, Code, Tel)	**values**	(**@F**='مریم', **@L**='میرزاخانی', **@C**=190, **@T**='0915')

UPDATE ویرایش رکورد

UPDATE	**نام جدول**	**Set**	**مقادیر متناظر هر فیلد**	**where**	**شرط**
UPDATE	Account	**Set**	**Tel**='0915'	**where**	Code = 310

Data Base : CRUD

- **Create (Insert)**

```
DatabaseContext db = new DatabaseContext();//پیش نیاز دسترسی به دیتابیس

Book book = new Book()
            {
                Name = "آموزش سی شارپ",
                Author = "مهدی مرسلی",
                Publisher = "نشر کیان",
                Group = "برنامه نویسی",
              };

db.Books.Add(book);
db.SaveChanges();
```

Data Base : CRUD

- **Read (Select)**

```
db.Books.ToList();

db.Books.Select(b => new Book(){ Name = b.Name,Author = b.Author}).ToList();

db.Books.Where(b => b.Name.Contains(search)).ToList();

db.Books.Where(b => b.Name.Contains(search))
        .Select(b => new Book(){ Name = b.Name,Author = b.Author}).ToList();
```

Data Base : CRUD

- **Update**

```
var book = db.Books.Find(Id);

if (book != null)
{
        book.Name = "آموزش زبان برنامه نویسی سی شارپ";
        book.Publisher = "کیان کامپیوتر";

        db.SaveChanges();
}
```

Data Base : CRUD

- **Delete**

```
var book = db.Books.Find(Id);

if (book != null)
{
        db.Books.Remove(book);
        db.SaveChanges();
}
```