```
// With two slashes you can start a one-line comment.
/*
With slash-star you can start a multi-line comment and
end it with the star-slash combo.
*/
//Variables
int i = 10;//You can declare variables with this format;
           //<type> <name> = <value>;
           // = used for assignment.
string s = "Hello world";// Strings can be declared using double quotes.
double d = 42.05;//You can use float,decimal and double to represent
                 //floating point numbers.
char c = 'q';//You can use single-quotes to store characters.
string s2 = null;//Reference type values can be null , be aware !
bool isGood = true;//You can store boolean values using bool type.
var v = 100;//You can use the "var"keyword to omit the type.

//Clauses
if (i == 10)// == means equals
{
    i = 20;
}
else if (i != 15)//You ca use "else if" when your if condition fails.
// != means 'not' equals by the way.
else //And at last , the else clause when your every conditions fails.
 {
 }

while (i < 50)//This is what the while condition looks like.
{
    i++;// C# has"++" too.
}

for(int x = 0; x<50; x++)//The classic forloop.
                         //Nothing is out of the ordinary here.
{
    i += x;//It's like"i=i+x;"but shorter.You can do "-=","*=","/=" too
    //if you want to do your math lile that.
}

string s3 = i == 50 ? "It's 50" : "It's not 50";//We have ternary operator too...
//Arrays
string[] sArray = new string[10];//create an array with a lenght of 10 like this.
sArray[0] = s;//Indexes starts from zero,just like it should !
sArray[1] = s2;
sArray[2] = s3;
//Methods
//You can declare methods like this;<type> <name>
//|if any|<parameter type> <parameter name>
void SayHi()//you can use the "void" as the type ifyou don't want to return
  anythingh.
{
    Console.WriteLine("Hi");
}
int Add(int x, int y)
{
    return x + y;
}
//And you call them like this.
SayHi();
int i2 = Add(10, 20);

//Classes
//You can declare classes like this.
/*
class <class name>
{
|if any|<property type> <property name>;
}
*/
class Person
{
    string Name;
    string Surname;
    int Age;

    string FullName()//You can declare methods in your classes like this.
    {
        return Name + " " + Surname;//string concatenation
    }
}
//You can create an instance of your class like this;
Person p = new Person();
p.Name = "Petean";//you can reach the properties and methods inside your class
                  //using dot notation
p.Surname = "Ionel";
p.Age = 54;
string fd = p.FullName();
//Lists and key value pairs
using System.Collections.Generic;//If you want to store your objects as a list
//but want more flexible "thing" from arrays,you can use the generic List class
// from the "System.Collections.Generic" namespace.You can reference the namespace
//with the helpof "using" keyword.
List<Person> pList = = new List<Person>();//Because this classis generic ,it must
//include the type in the declaration.You can give it any type you want.It can be
//a class you declared too. Of course not every type is acceptable and you can filter
//the types you want when you are declaring a generic class,but let's keep it simple
//shall we ?
pList.Add(p);//you can add an object to a list like this.
pList.Remove(p);//and you can remove one like this.
```

```

Dictionary<int, string>();//You can use Dictionary class to keep your key value pairs.
//This class is a generic class too an utilizes two separate types to represent the
//key and value types.In this case,the key is integer and the value is a string type.






```