```
// 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.
```

```
53
54  sArray[0] = s;//Indexes starts from zero,just like it should !
55
56  sArray[1] = s2;
57
58  sArray[2] = s3;
59
60  //Methods
61  //You can declare methods like this;<type> <name>
62  //|if any|<parameter type> <parameter name>
63
64  void SayHi()//you can use the "void" as the type ifyou don't want to return
      anythingh.
65  {
66      Console.WriteLine("Hi");
67  }
68  int Add(int x, int y)
69  {
70      return x + y;
71  }
72
73  //And you call them like this.
74
75  SayHi();
76  int i2 = Add(10, 20);
77
78  //Classes
79  //You can declare classes like this.
80  /*
81  class <class name>
82  {
83  |if any|<property type> <property name>;
84  }
85  */
86
87  class Person
88  {
89      string Name;
90      string Surname;
91      int Age;
92
93      string FullName()//You can declare methods in your classes like this.
94      {
95          return Name + " " + Surname;//string concatenation
96      }
97  }
98
99  //You can create an instance of your class like this;
100
101 Person p = new Person();
102 p.Name = "Petean";//you can reach the properties and methods inside your class
103                   //using dot notation
```

```
104  p.Surname = "Ionel";
105  p.Age = 54;
106  string fd = p.FullName();
107
108  //Lists and key value pairs
109
110  using System.Collections.Generic;//If you want to store your objects as a list
111  //but want more flexible "thing" from arrays,you can use the generic List class
112  // from the "System.Collections.Generic" namespace.You can reference the
       namespace
113  //with the helpof "using" keyword.
114
115  List<Person> pList = = new List<Person>();//Because this classis generic ,it must
116  //include the type in the declaration.You can give it any type you want.It can be

117  //a class you declared too. Of course not every type is acceptable and you can
       filter
118  //the types you want when you are declaring a generic class,but let's keep it
       simple
119  //shall we ?
120
121  pList.Add(p);//you can add an object to a list like this.
122
123  pList.Remove(p);//and you can remove one like this.
124
125  Dictionary<int, string>();//You can use Dictionary class to keep your key value
       pairs.
126  //This class is a generic class too an utilizes two separate types to represent
       the
127  //key and value types.In this case,the key is integer and the value is a string
       type.
128
129
130
131
132
133
```