```
1  // With two slashes you can start a one-line comment.
2  /*
3  With slash-star you can start a multi-line comment and
4  end it with the star-slash combo.
5  */
6  //Variables
7
8  int i = 10;//You can declare variables with this format;
9             //<type> <name> = <value>;
10            // = used for assignment.
11 string s = "Hello world";// Strings can be declared using double quotes.
12
13 double d = 42.05;//You can use float,decimal and double to represent
14                  //floating point numbers.
15
16 char c = 'q';//You can use single-quotes to store characters.
17
18 string s2 = null;//Reference type values can be null , be aware !
19
20 bool isGood = true;//You can store boolean values using bool type.
21
22 var v = 100;//You can use the "var"keyword to omit the type.
23
24
25 //Clauses
26
27 if (i == 10)// == means equals
28 {
29     i = 20;
30 }
31 else if (i != 15)//You ca use "else if" when your if condition fails.
32 // != means 'not' equals by the way.
33 else //And at last , the else clause when your every conditions fails.
34  {
35  }
36
37 while (i < 50)//This is what the while condition looks like.
38 {
39     i++;// C# has"++" too.
40 }
41
42 for(int x = 0; x<50; x++)//The classic forloop.
43                          //Nothing is out of the ordinary here.
44 {
45     i += x;//It's like"i=i+x;"but shorter.You can do "-=","*=","/=" too
46     //if you want to do your math lile that.
47 }
48
49
50 string s3 = i == 50 ? "It's 50" : "It's not 50";//We have ternary operator too...
51
52
```

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

```
104 }
105
106 //You can create an instance of your class like this;
107
108 Person p = new Person();
109 p.Name = "Petean";//you can reach the properties and methods inside your class
110                   //using dot notation
111 p.Surname = "Ionel";
112
113 p.Age = 54;
114
115 string fd = p.FullName();
116
117 //Lists and key value pairs
118
119 using System.Collections.Generic;//If you want to store your objects as a list
120 //but want more flexible "thing" from arrays,you can use the generic List class
121 // from the "System.Collections.Generic" namespace.You can reference the
      namespace
122 //with the helpof "using" keyword.
123
124 List<Person> pList = = new List<Person>();//Because this classis generic ,it must
125 //include the type in the declaration.You can give it any type you want.It can be

126 //a class you declared too. Of course not every type is acceptable and you can
      filter
127 //the types you want when you are declaring a generic class,but let's keep it
      simple
128 //shall we ?
129
130 pList.Add(p);//you can add an object to a list like this.
131
132 pList.Remove(p);//and you can remove one like this.
133
134 Dictionary<int, string>();//You can use Dictionary class to keep your key value
      pairs.
135 //This class is a generic class too an utilizes two separate types to represent
      the
136 //key and value types.In this case,the key is integer and the value is a string
      type.
137
138
139
140
141
142
```