# More Exercise: C# Intro and Basic Syntax

Problems for exercises and homework for the ["C# Fundamentals" course @ SoftUni](https://softuni.bg/trainings/3836/programming-fundamentals-with-csharp-september-2022)  
You can check your solutions in [Judge](https://judge.softuni.org/Contests/1453/Basic-Syntax-Conditional-Statements-and-Loops-More-Exercise)

## Sort Numbers

Create a program that receives three real numbers and sorts them in descending order. Print each number on a new line.

using System;

namespace \_01.\_Sort\_Numbers

{

class Program

{

static void Main(string[] args)

{

double a = double.Parse(Console.ReadLine());

double b= double.Parse(Console.ReadLine());

double c = double.Parse(Console.ReadLine());

if (a > b && a > c)

{

Console.WriteLine(a);

if (b > c)

{

Console.WriteLine(b);

Console.WriteLine(c);

}

else

{

Console.WriteLine(c);

Console.WriteLine(b);

}

}

else if (b > a && b > c)

{

Console.WriteLine(b);

if (a > c)

{

Console.WriteLine(a);

Console.WriteLine(c);

}

else

{

Console.WriteLine(c);

Console.WriteLine(a);

}

}

else if (c> a && c >b)

{

Console.WriteLine(c);

if (a > b)

{

Console.WriteLine(a);

Console.WriteLine(b);

}

else

{

Console.WriteLine(b);

Console.WriteLine(a);

}

}

}

}

}

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  1  3 | 3  2  1 |
| -2  1  3 | 3  1  -2 |
| 0  0  2 | 2  0  0 |

## English Name of the Last Digit

Create a **method** that returns the **English spelling** of the last digit of a given number. Write a program that reads an integer and prints the returned value from this method.

using System;

namespace \_02.\_English\_Name\_of\_the\_Last\_Digit

{

class Program

{

static void Main(string[] args)

{

int n = int.Parse(Console.ReadLine());

int lastDigit = n % 10;

if(lastDigit==1)

{ Console.WriteLine("one"); }

else if (lastDigit==2)

{

Console.WriteLine("two");

}

else if (lastDigit == 3)

{

Console.WriteLine("three");

}

else if (lastDigit == 4)

{

Console.WriteLine("four");

}

else if (lastDigit == 5)

{

Console.WriteLine("five");

}

else if (lastDigit == 6)

{

Console.WriteLine("six");

}

else if (lastDigit == 7)

{

Console.WriteLine("seven");

}

else if (lastDigit == 8)

{

Console.WriteLine("eight");

}

else if(lastDigit==9)

{ Console.WriteLine("nine"); }

else if(lastDigit==0)

{ Console.WriteLine("zero"); }

}

}

}

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 512 | two |
| 1 | one |
| 1643 | three |

## Gaming Store

Create a program, whichhelps you buy the games. The **valid games** are the following games in this table:

using System;

namespace \_03.\_Gaming\_Store

{

class Program

{

static void Main(string[] args)

{

double balanceFirstly = double.Parse(Console.ReadLine());

double balance = balanceFirstly;

string input = Console.ReadLine(); ;

bool flag = true;

while(input!= "Game Time")

{

if (input== "OutFall 4")

{

if(balance>= 39.99)

{

Console.WriteLine($"Bought OutFall 4");

balance -= 39.99;

}

else

{ Console.WriteLine("Too Expensive"); }

}

else if (input == "CS: OG")

{

if (balance >= 15.99)

{

Console.WriteLine($"Bought CS: OG");

balance -= 15.99;

}

else

{ Console.WriteLine("Too Expensive"); }

}

else if (input == "Zplinter Zell")

{

if (balance >= 19.99)

{

Console.WriteLine($"Bought Zplinter Zell");

balance -= 19.99;

}

else

{ Console.WriteLine("Too Expensive"); }

}

else if (input == "Honored 2")

{

if (balance >= 59.99)

{

Console.WriteLine($"Bought Honored 2");

balance -= 59.99;

}

else

{ Console.WriteLine("Too Expensive"); }

}

else if (input == "RoverWatch")

{

if (balance >= 29.99)

{

Console.WriteLine($"Bought RoverWatch");

balance -= 29.99;

}

else

{ Console.WriteLine("Too Expensive"); }

}

else if (input == "RoverWatch Origins Edition")

{

if (balance >= 39.99)

{

Console.WriteLine($"Bought RoverWatch Origins Edition");

balance -= 39.99;

}

else

{ Console.WriteLine("Too Expensive"); }

}

else { Console.WriteLine("Not Found"); }

if (balance == 0.00)

{

Console.WriteLine("Out of money!");

break;

flag = false;

}

input = Console.ReadLine();

}

if (flag)

{

Console.WriteLine($"Total spent: ${(balanceFirstly-balance):f2}. Remaining: ${balance:f2}");

}

}

}

}

|  |  |
| --- | --- |
| **Name** | **Price** |
| OutFall 4 | $39.99 |
| CS: OG | $15.99 |
| Zplinter Zell | $19.99 |
| Honored 2 | $59.99 |
| RoverWatch | $29.99 |
| RoverWatch Origins Edition | $39.99 |

On the first line, you will receive your **current balance** – a **floating-point** number in the range **[0.00…5 000.00]**.

Until you receive the command "Game Time", you have to keep **buying games**. When a **game** is **bought**, the user's **balance** decreases by the **price** of the game.

Additionally, the program should obey the following conditions:

* If a game the user is trying to buy is **not present** in the table above, print "Not Found" and **read the next line**.
* If at any point, the user has **$0** left, print "Out of money!" and **end the program**.
* Alternatively, if the user is trying to buy a game that they **can't afford**, print "Too Expensive" and **read the next line**.
* If the game exists and the player has the money for it, print "**Bought {nameOfGame}**"

When you receive "Game Time", **print** the user's **remaining money** and **total spent on games**, **rounded** to the **2nd decimal place**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 120  RoverWatch  Honored 2  Game Time | Bought RoverWatch  Bought Honored 2  Total spent: $89.98. Remaining: $30.02 |
| 19.99  Reimen origin  RoverWatch  Zplinter Zell  Game Time | Not Found  Too Expensive  Bought Zplinter Zell  Out of money! |
| 79.99  OutFall 4  RoverWatch Origins Edition  Game Time | Bought OutFall 4  Bought RoverWatch Origins Edition  Total spent: $79.98. Remaining: $0.01 |

## Reverse String

Create a program which reverses a string and prints it on the console.

using System;

namespace \_04.\_Reverse\_String

{

class Program

{

static void Main(string[] args)

{

string input = Console.ReadLine();

for (int i = input.Length-1; i >=0; i--)

{

Console.Write(input[i]);

}

}

}

}

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Hello | olleH |
| SoftUni | inUtfoS |
| 1234 | 4321 |

## Messages

Create a program, which emulates **typing an SMS**, following this guide:

|  |  |  |
| --- | --- | --- |
| **1** | **2**  abc | **3**  def |
| **4**  ghi | **5**  jkl | **6**  mno |
| **7**  pqrs | **8**  tuv | **9**  wxyz |
|  | **0**  space |  |

Following the guide, **2** becomes “**a**”, **22** becomes “**b**” and so on.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 5  44  33  555  555  666 | hello | 9  44  33  999  0  8  44  33  777  33 | hey there | 7  6  33  33  8  0  6  33 | meet me |

### Hints

* A naive approach would be to just put all the possible combinations of digits in a giant switch statement.
* A cleverer approach would be to come up with a **mathematical formula**, which **converts** a **number** to its **alphabet** representation:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Digit** | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| * **Index** | * 0 1 2 | * 3 4 5 | * 6 7 8 | * 9 11 12 | * 13 14 15 | * 16 17 18 19 | * 20 21 22 | * 23 24 25 26 |
| * **Letter** | * a b c | * d e f | * g h i | * j k l | * m n o | * p q r s | * t u v | * w x y z |

* Let's take the number **222** (**c**) for example. Our algorithm would look like this:
  + Find the **number of digits** the number has, e.g. **222** 🡺 **3 digits**
  + Find the **main digit** of the number, e.g. **222** 🡺 **2**
  + Find the **offset** of the number. To do that, you can use the formula: (main digit - 2) \* 3
  + If the main digit is **8 or 9**, you need to **add 1** to the **offset**, since the digits **7** and **9** have **4 letters each**
  + Finally, find the **letter index** (a 🡺 0, c 🡺 2, etc.). To do that, you can use the following formula: (offset + digit length - 1).
  + After you've found the **letter index**, you can just add that to **the ASCII code** of the lowercase letter "**a**" (97)

using System;

using System.Diagnostics;

namespace \_05.\_Messages

{

class Program

{

static void Main(string[] args)

{

int n = int.Parse(Console.ReadLine());

string output = string.Empty;

Stopwatch stopWatch = new Stopwatch();

stopWatch.Start();

for (int i = 1; i <=n; i++)

{

int N = int.Parse(Console.ReadLine());

if(N==0)

{ output+=" "; }

else if(N==2)

{ output +="a"; }

else if (N == 22)

{ output += "b"; }

else if (N == 222)

{ output += ("c"); }

else if (N == 3)

{ output += ("d"); }

else if (N == 33)

{ output += ("e"); }

else if (N == 333)

{ output += ("f"); }

else if (N == 4)

{ output += ("g"); }

else if (N == 44)

{ output += ("h"); }

else if (N == 444)

{ output += ("i"); }

else if (N == 5)

{ output += ("j"); }

else if (N == 55)

{ output += ("k"); }

else if (N == 555)

{ output += ("l"); }

else if (N == 6)

{ output += ("m"); }

else if (N == 66)

{ output += ("n"); }

else if (N == 666)

{ output += ("o"); }

else if (N == 7)

{ output += ("p"); }

else if (N == 77)

{ output += ("q"); }

else if (N == 777)

{ output += ("r"); }

else if (N == 7777)

{ output += ("s"); }

else if (N == 8)

{ output += ("t"); }

else if (N == 88)

{ output += ("u"); }

else if (N == 888)

{ output += ("v"); }

else if (N == 9)

{ output += ("w"); }

else if (N == 99)

{ output += ("x"); }

else if (N == 999)

{ output += ("y"); }

else if (N == 9999)

{ output += ("z"); }

}

Console.WriteLine(output);

stopWatch.Stop();

// Get the elapsed time as a TimeSpan value.

Console.WriteLine(stopWatch.ElapsedMilliseconds);

}

}

}