# Lab: Methods

Problems for in-class lab 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/1208/Methods-Lab)

# Declaring and Invoking Methods

## Sign of Integer Numbers

A single integer is given, create a method that checks if the given number is **positive**, **negative,** or **zero.** As a result print:

* For positive number: **"The number {number} is positive. "**
* For negative number: **"The number {number} is negative. "**
* For zero number: **"The number {number} is zero. "**

using System;

namespace \_01.\_Sign\_of\_Integer\_Numbers

{

class Program

{

static void Main(string[] args)

{

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

checkNumber(n);

}

static void checkNumber(int number)

{

if(number>0)

{ Console.WriteLine($"The number {number} is positive. "); }

else if(number<0)

{ Console.WriteLine($"The number {number} is negative. "); }

else

{ Console.WriteLine($"The number {number} is zero. "); }

}

}

}

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 | The number 2 is positive. |
| -9 | The number -9 is negative. |

## Grades

Create a method **that receives** a grade between **2.00** and **6.00** and **prints the corresponding grade definition**:

* 2.00 – 2.99 - "**Fail**"
* 3.00 – 3.49 - "**Poor**"
* 3.50 – 4.49 - "**Good**"
* 4.50 – 5.49 - "**Very good**"
* 5.50 – 6.00 - "**Excellent**"

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3.33 | Poor |
| 4.50 | Very good |
| 2.99 | Fail |

### Hints

1. Read the grade from the console and pass it to a method



1. Then create the method and make the if statements for each case



using System;

namespace \_02.\_Grades

{

class Program

{

static void Main(string[] args)

{

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

grades(grade);

}

static void grades(double grade)

{

if (grade>=2.00&&grade <= 2.99)

{ Console.WriteLine("Fail"); }

else if(grade >= 3.00 && grade <= 3.49)

{ Console.WriteLine("Poor"); }

else if (grade >= 3.50 && grade <= 4.49)

{ Console.WriteLine("Good"); }

else if (grade >= 4.50 && grade <= 5.49)

{ Console.WriteLine("Very good"); }

else if (grade >= 5.50 && grade <= 6.00)

{ Console.WriteLine("Excellent"); }

}

}

}

## Calculations

Create a program that receives three lines of input:

* On the first line – a **string** – "**add**", "**multiply**", "**subtract**", "**divide**".
* On the second line – a number.
* On the third line – another number.

You should create **four** **methods** (for each calculation) and invoke the corresponding method depending on the command. The method should also print the result (needs to be void).

using System;

namespace \_03.\_Calculations

{

class Program

{

static void Main(string[] args)

{

string command = Console.ReadLine();

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

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

if (command == "add")

{

Console.WriteLine( adding(fn, sn));

}

else if(command=="multiply")

{

Console.WriteLine(multiply(fn, sn));

}

else if (command == "subtract")

{

Console.WriteLine(subtract(fn, sn));

}

else if(command=="divide")

{

Console.WriteLine(divide(fn, sn));

}

}

static int adding(int a, int b)

{

int sum = a + b;

return sum;

}

static int multiply(int a, int b)

{

int sum = a \* b;

return sum;

}

static int subtract(int a, int b)

{

int sum = a - b;

return sum;

}

static int divide(int a, int b)

{

int sum = a / b;

return sum;

}

}

}

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| subtract  5  4 | 1 |
| divide  8  4 | 2 |

### Hints

1. Read the command on the first line, and the two numbers, and then make an if/switch statement for each type of calculation



1. Then create the four methods and print the result



## Printing Triangle

Create a method for printing triangles as shown below:

using System;

namespace \_03.\_Calculations

{

class Program

{

static void Main(string[] args)

{

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

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

{

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

{

Console.Write(j + " ");

}

Console.WriteLine("");

}

for (int i = number - 1; i >= 1; i--)

{

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

{

Console.Write(j + " ");

}

Console.WriteLine("");

}

}

}

}

### Examples

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

### Hints

1. After you read the input
2. Start by creating a method **for printing a single line** from a **given start** to a **given end**. Choose a **meaningful name** for it, describing its purpose:



1. Create another method for printing the whole triangle. Again choose a **meaningful name** for it, describing its purpose.
2. Think how you can use the **PrintLine()** method to solve the problem.
3. After you spent some time thinking, you should have concluded that you need two loops.
4. In the first loop you can print the first half of the triangle:



1. In the second loop you can print the second half of the triangle:



## Orders

Create a program that calculates and prints the total price of an order. The method should receive two parameters:

* A **string, representing a product** - "**coffee**", "**water**", "**coke**", "**snacks**"
* An integer, representing the **quantity** of the product

The prices for a single item of each product are:

* coffee – 1.50
* water – 1.00
* coke – 1.40
* snacks – 2.00

Print the result, rounded to the second decimal place.

using System;

namespace \_05.\_Orders

{

class Program

{

static void Main(string[] args)

{

string product = Console.ReadLine();

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

if (product == "coffee")

{

Console.WriteLine($"{coffee(quantity):F2}");

}

else if (product == "water")

{

Console.WriteLine($"{water(quantity):F2}");

}

else if (product == "coke")

{

Console.WriteLine($"{coke(quantity):F2}");

}

else if (product == "snacks")

{

Console.WriteLine($"{snacks(quantity):F2}");

}

}

static double coffee(int a)

{

double sum = a\*1.5;

return sum;

}

static double water(int a)

{

double sum = a \* 1.0;

return sum;

}

static double coke(int a)

{

double sum = a\*1.4;

return sum;

}

static double snacks(int a)

{

double sum = a \*2;

return sum;

}

}

}

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| water  5 | 5.00 |
| coffee  2 | 3.00 |

### Hints

1. Read the first two lines
2. Create a method to pass the two variables in
3. Print the result in the method

# Returning Values and Overloading

## Calculate Rectangle Area

Create a method that calculates and **returns** the [area](http://www.mathopenref.com/trianglearea.html) of a rectangle.

using System;

namespace \_06.\_Calculate\_Rectangle\_Area

{

class Program

{

static void Main(string[] args)

{

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

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

Console.WriteLine(rectangleArea(width,height));

}

static double rectangleArea(double a,double b)

{

double area = a \* b;

return area;

}

}

}

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  4 | 12 |
| 6  2 | 12 |

### Hints

1. Read the input.
2. Create a method, but this time **instead** of typing "**static void**" before its name, type "**static double**" as this will make it **return a value of type double**:



1. **Invoke** the method in the main and **save the return value in a new variable**:



## Repeat String

### Create a method that receives two parameters:

### A string

### A number n (integer) represents how many times the string will be repeated

### The method should return a new string, containing the initial one, repeated n times without space.

using System;

namespace \_07.\_Repeat\_String

{

class Program

{

static void Main(string[] args)

{

string input = Console.ReadLine();

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

string result=reverseString(input, n);

Console.WriteLine(result);

}

private static string reverseString(string input, int n)

{

string result="";

for (int i = 0; i < n; i++)

{

result += input;

}

return result;

}

}

}

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| abc  3 | abcabcabc |
| String  2 | StringString |

### Hints

1. First, read the **string** and the repeat count **n**
2. Then create the method and pass the variables to it



1. In the main method, print the result

## Math Power

Create a method, which receives two numbers as parameters:

* The first number – the **base**
* The second number – the **power**

The method should return the **base** raised to the given **power**.

using System;

namespace \_08.\_Math\_Power

{

class Program

{

static void Main(string[] args)

{

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

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

Console.WriteLine(mathPow(num,power));

}

static double mathPow(double a,double b)

{

double num = 1;

if (b == 0)

{ return 1; }

else

{

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

{

num \*= a;

}

}

return num;

}

}

}

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  8 | 256 |
| 3  4 | 81 |

### Hints

1. As usual, read the input.
2. Create a method that will have two parameters - the number and the power, and will return a result of type double:



1. Print the result.

## Greater of Two Values

You are given an input of two values of the same type. The values can be of type **int**, **char** or **string**. Create methods called **GetMax(), which can compare int, char or string** and returns the **biggest of the two values**.

using System;

namespace \_08.\_Math\_Power

{

class Program

{

static void Main(string[] args)

{

string type = Console.ReadLine();

string firstValue = Console.ReadLine();

string secondValue = Console.ReadLine();

if (type == "int")

{

int firstInt = int.Parse(firstValue);

int secondInt = int.Parse(secondValue);

getMax(firstInt, secondInt);

}

else if (type == "char")

{

char firstChar = char.Parse(firstValue);

char secondChar = char.Parse(secondValue);

getMax(firstChar, secondChar);

}

else if (type == "string")

{

getMax(firstValue, secondValue);

}

}

static void getMax(int a, int b)

{

if (a > b)

{ Console.WriteLine(a); }

else if(b>a)

{ Console.WriteLine(b); }

}

static void getMax(char a, char b)

{

if (a > b)

{ Console.WriteLine(a); }

else if (b > a)

{ Console.WriteLine(b); }

}

static void getMax(string a, string b)

{

int rersult=a.CompareTo(b);

if (rersult >0)

{ Console.WriteLine(a); }

else

{ Console.WriteLine(b); }

}

}

}

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| int  2  16 | 16 |
| char  a  z | z |
| string  aaa  bbb | bbb |

### Hints

Use method overloading.

## Multiply Evens by Odds

Create a program that **multiplies the sum** of **all even digits** of a number **by the sum of all odd digits** of the same number:

* Create a method called **GetMultipleOfEvenAndOdds()**
* Create a method **GetSumOfEvenDigits()**
* Create **GetSumOfOddDigits()**
* You may need to use **Math.Abs()** for negative numbers

using System;

namespace \_10.\_Multiply\_Evens\_by\_Odds

{

class Program

{

static void Main(string[] args)

{

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

Console.WriteLine(GetMultipleOfEvenAndOdds(GetSumOfEvenDigits(num), GetSumOfOddDigits(num)));

}

static double GetSumOfEvenDigits(int num)

{

double sum = 0;

int absNum = Math.Abs(num);

while(absNum!=0)

{

int lastDigit = absNum % 10;

if(lastDigit%2==0)

{ sum += lastDigit; }

absNum /= 10;

}

return sum;

}

static double GetSumOfOddDigits(int num)

{

double sum = 0;

int absNum = Math.Abs(num);

while (absNum != 0)

{

int lastDigit = absNum % 10;

if (lastDigit % 2 ==1)

{ sum += lastDigit; }

absNum /= 10;

}

return sum;

}

static double GetMultipleOfEvenAndOdds(double a,double b)

{

double multiply = a \* b;

return multiply;

}

}

}

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comment** |
| -12345 | 54 | Evens: 2 4  Odds: 1 3 5  Even sum: 6  Odd sum: 9  6 \* 9 = 54 |
| 3453466 | 220 |  |

1. **Math Operations**

Write a method that receives **two numbers** and an **operator**, calculates the result and returns it. You will be given **three lines of input**. The first will be the first **number**, the second one will be the **operator** and the last one will be the **second number**.

The possible operators are: **/**, **\***, **+** and **-**.

using System;

namespace \_11.\_Math\_operations

{

class Program

{

static void Main(string[] args)

{

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

string command = Console.ReadLine();

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

if (command == "+")

{

Console.WriteLine(adding(fn, sn));

}

else if (command == "\*")

{

Console.WriteLine(multiply(fn, sn));

}

else if (command == "-")

{

Console.WriteLine(subtract(fn, sn));

}

else if (command == "/")

{

Console.WriteLine(divide(fn, sn));

}

}

static double adding(double a, double b)

{

double sum = a + b;

return sum;

}

static double multiply(double a, double b)

{

double sum = a \* b;

return sum;

}

static double subtract(double a, double b)

{

double sum = a - b;

return sum;

}

static double divide(double a, double b)

{

double sum = a / b;

return sum;

}

}

}

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5  \*  5 | 25 |
| 4  +  8 | 12 |

### Hint

1. Read the inputs and create a method that returns a double (the result of the operation)

