# Homework: Primitive Data Types and Variables / Console Input / Output

Please submit as homework a single zip / rar / 7z archive holding the solutions (source code only) of all below described problems.

## Declare Variables

Declare five variables choosing for each of them the most appropriate of the types byte, sbyte, short, ushort, int, uint, long, ulong to represent the following values: 52130, -115, 4825932, 97, -10000. Choose a large enough type for each number to ensure it will fit in it. Try to compile the code. Submit the source code of your Visual Studio project as part of your homework submission.

## Float or Double?

Which of the following values can be assigned to a variable of type float and which to a variable of type double: 34.567839023, 12.345, 8923.1234857, 3456.091? Write a program to assign the numbers in variables and **print** them to ensure no precision is lost.

## Quotes in Strings

Declare two string variables and assign them with following value:

|  |
| --- |
| The "use" of quotations causes difficulties. |

Do the above in two different ways: with and without using **quoted strings**. Print the variables to ensure that their value was correctly defined.

|  |
| --- |
| **Expected Output** |
| The "use" of quotations causes difficulties.  The "use" of quotations causes difficulties. |

## Isosceles Triangle

Write a program that prints an isosceles triangle of 9 copyright symbols ©, something like this:

|  |
| --- |
| ©  © ©  © ©  © © © © |

Note that the © symbol may be displayed incorrectly at the console so you may need to change the console character encoding to UTF-8 and assign a Unicode-friendly font in the console. Note also, that under old versions of Windows the © symbol may still be displayed incorrectly, regardless of how much effort you put to fix it.

## Null Values Arithmetic

Create a program that assigns null values to an integer and to a double variable. Try to print these variables at the console. Try to add some number or the null literal to these variables and print the result.

## \* Comparing Floats

Write a program that **safely compares floating-point numbers** (double) with precision eps = 0.000001. Note that we cannot directly compare two floating-point numbers a and b by a==b because of the nature of the floating-point arithmetic. Therefore, we assume two numbers are equal if they are more closely to each other than a fixed constant eps. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **Number a** | **Number b** | **Equal (with precision eps=0.000001)** | **Explanation** |
| 5.3 | 6.01 | false | The difference of 0.71 is too big (> eps) |
| 5.00000001 | 5.00000003 | true | The difference 0.00000002 < eps |
| 5.00000005 | 5.00000001 | true | The difference 0.00000004 < eps |
| -0.0000007 | 0.00000007 | true | The difference 0.00000077 < eps |
| -4.999999 | -4.999998 | false | Border case. The difference 0.000001 == eps. We consider the numbers are different. |
| 4.999999 | 4.999998 | false | Border case. The difference 0.000001 == eps. We consider the numbers are different. |

## \* Print the ASCII Table

Find online more information about [**ASCII**](http://www.ascii-code.com/) (American Standard Code for Information Interchange) and write a program to prints the entire ASCII table of characters at the console (characters from 0 to 255). Note that some characters have a special purpose and will not be displayed as expected. You may skip them or display them differently. You may need to use for-loops (learn in Internet how).

## Circle Perimeter and Area

Write a program that reads the radius **r** of a circle and prints its perimeter and area formatted with 2 digits after the decimal point. Examples:

|  |  |  |
| --- | --- | --- |
| **r** | **perimeter** | **area** |
| 2 | 12.57 | 12.57 |
| 3.5 | 21.99 | 38.48 |

## Formatting Numbers

Write a program that reads 3 numbers: an integer a (0 ≤ a ≤ 500), a floating-point b and a floating-point c and **prints them in 4 virtual columns** on the console. Each column should have a width of 10 characters. The number a should be printed in **hexadecimal, left aligned**; then the number a should be printed in binary form, padded with zeroes, then the number b should be **printed with 2 digits after the decimal point, right aligned**; the number c should be **printed with 3 digits after the decimal point, left aligned**. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **a** | **b** | **c** | **result** |
| 254 | 11.6 | 0.5 | |FE |0011111110| 11.6|0.500 | |
| 499 | -0.5559 | 10000 | |1F3 |0111110011| -0.56|10000.000 | |
| 0 | 3 | -0.1234 | |0 |0000000000| 3|-0.123 | |

## \* Fibonacci Numbers

Write a program that reads a number n and prints on the console the first n members of the [**Fibonacci sequence**](http://en.wikipedia.org/wiki/Fibonacci_number) (at a single line, separated by spaces) : 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, …. Note that you may need to learn how to use loops. Examples:

|  |  |
| --- | --- |
| **n** | **comments** |
| 1 | 0 |
| 3 | 0 1 1 |
| 10 | 0 1 1 2 3 5 8 13 21 34 |

## \*\* Magic Wand

As we all know programmers often make mistakes in their code. They spend hours and hours trying to figure out where the problem is. Some are praying for the code to fix itself, others are searching for magical rainbow unicorns to help them with their problem. One day, the programmers Gesho and Posho discovered a way to build magic wands that solve their coding problems. Your task is to help Gesho and Posho to build a **magic wand**.

### Input

The input data should be read from the console.

On the only input line you have an integer number **N**. The **width** of the wand is **3\*N+2**.

The input data will always be valid and in the format described. There is no need to check it explicitly.

### Output

The output data should be printed on the console. You must print the magic wand on the console.

Each row contains characters "**.**" (dot) and "**\***" (asterisk).

### Constraints

* The number **N** will always be an **odd** integer number in the range [5…39].
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 5 | ........\*........  .......\*.\*.......  ......\*...\*......  .....\*.....\*.....  \*\*\*\*\*.......\*\*\*\*\*  .\*.............\*.  ..\*...........\*..  .\*..\*\*.....\*\*..\*.  \*..\*.\*.....\*.\*..\*  \*\*\*..\*.....\*..\*\*\*  .....\*.....\*.....  .....\*.....\*.....  .....\*.....\*.....  .....\*.....\*.....  .....\*.....\*.....  .....\*\*\*\*\*\*\*..... |  | 7 | ...........\*...........  ..........\*.\*..........  .........\*...\*.........  ........\*.....\*........  .......\*.......\*.......  \*\*\*\*\*\*\*.........\*\*\*\*\*\*\*  .\*...................\*.  ..\*.................\*..  ...\*...............\*...  ..\*...\*\*.......\*\*...\*..  .\*...\*.\*.......\*.\*...\*.  \*...\*..\*.......\*..\*...\*  \*\*\*\*...\*.......\*...\*\*\*\*  .......\*.......\*.......  .......\*.......\*.......  .......\*.......\*.......  .......\*.......\*.......  .......\*.......\*.......  .......\*.......\*.......  .......\*.......\*.......  .......\*\*\*\*\*\*\*\*\*....... |