Tasks for Introduction To Programming

Hello, world!

1. Print `Hello, world!` on the console.

Basic output

1. Print the words `one, 2, three` on the console, each of them on a new line.
2. Print `John said, "No"`.

Basic input

1. Prompt the user to input a number and print it on the console.
2. Prompt the user to input 3 characters and print them.

Variables

1. Create a variable with the value `12` and print it on the console.
2. Create two variables with values `3` and `7`. Print them on the console. Make a third variable holding their sum. Print the sum on the console.
3. Create 2 variables. Ask the user to input 2 numbers and save them in the variables. Print the product of the 2 numbers.
4. Create a variable holding the letter `A`. Add `1` to the variable. Print the variable.
5. Create 2 variables representing the 2 sides of a rectangle with values `3.4` and `10.0`. Print the area of the rectangle.

Expressions

1. Ask the user for two numbers - the side of a triangle and the height of the triangle. Print the area of the triangle.
2. Calculate the circumference of a circle with radius `0.159154943`, knowing that `Pi = 3.141592654`.
3. Calculate the face of a circle with radius `0.564189584`. Before printing the result include the library `<iomanip>` and write `cout << setprecision(10);`.
4. Prompt the user for their age and print how many seconds have passed since their birth. e.g if the user is 1 year old it would be `1 * 365 * 24 * 60 * 60`.
5. Calculate the roots of `x^2-130x-4056`.

Functions

1. Write a function which prints `Hello, world!`.
2. Write a function which has **1 int parameter** and prints that parameter to the console.
3. Write a function which calculates the **sum of 2 ints**.
4. Write a function which calculates the **product of 2 chars**.

5. Write a function which calculates the **real roots** of a **quadratic equation** of the form `a*x^2 + b*x + c = 0`. The function takes **3 parameters** - `a, b, c`.
6. Write a function which accepts **4 coordinates** in the form `(x1, y1, x2, y2)` and calculates the distance between the points with coordinates `(x1, y1)` and `(x2, y2)`.

Variables (Advanced)

1. Create 2 variables, assign them values. Swap the values of the variables. Print the swapped values.
2. Print the value of `5 == 5`.
3. Print the size of `int` (how many bytes each int has).
4. Find the maximum value of `unsigned int`.
5. Create a variable with value `325`. Print the variable as a char.
6. Create a variable holding the letter `E`. Print the variable as an int.
7. Write a function which **reverses a number**.
8. Write a function which **prints all the digits** of a number on new lines.

Conditionals

1. Prompt the user for **3 numbers** and print them in increasing order.
2. Write a function which takes **an int** representing a year as a parameter and returns whether it's a leap year. e.g `2004` returns `true`, `2003` returns `false`.
3. Write a function which takes **an int** representing a month (1 = Jan, 12 = Dec) and returns the number of days it has.
4. Prompt the user to enter **20 numbers** and print the largest of them.
5. Write a function which takes **a char** and prints on the console if it is a letter, number or other symbol.
6. Write a function which takes in **a char** and returns its hex value if it's a valid digit `{ 0-9, A-F }`.
7. Write a function which takes **an int** from 1 to 7 and prints the weekday as a string on the console. e.g `printDay(1);` outputs `Monday`.
8. Write a function which takes an enum `WeekDay` and prints the weekday as a string on the console. e.g `printDay(Monday);` outputs `Monday`.

Loops

1. Print the numbers from **1 to 1000** on new lines.
2. Write a function to compute **factorial**.
3. Prompt the user to enter **a number** `N` and print on the console all odd numbers between 0 and `N`.
4. Write a function which finds **how many digits** a number has.
5. Write a function which takes **an int** as a parameter and prints **all of its divisors**.
6. Write a function to calculate the `N`th **Fibonacci number** where `N` is the parameter of the function.
7. Write a function which given 2 numbers `N` and `K` computes the binomial coefficient $\binom{N}{K} = \frac{N!}{K!(N-K)!}$

8. Write a function which takes **an int** `N` and prints a pyramid of numbers from 1 to `N`, as the first row has 1 number, 2nd has 2 numbers and so on until we reach the number `N`. The last row may not obey the stated rule.

   e.g `printPyramid(12);`

```
1
2 3
4 5 6
7 8 9 10
11 12
```

Structs

1. Create a struct representing a **point with 2 integer coordinates**. Write a function which finds the **distance** between 2 points.
2. Create a struct representing **a point with 2 float coordinates**. Write a function which finds the **distance** between 2 points.
3. Create a struct representing a **linear equation** of the form `y = A * x + B`. Expand the struct to support the line `x = 2` for example. Write a method that takes a point and returns if the point lies on the line.
4. Create a struct representing a **vector**. A vector is defined with 2 points, a start and end point.

Recursion

1. Print the numbers from **1 to 100** on the console in **ascending order**.
2. Print the even numbers from **100 to 5** on the console in **descending order**.
3. Write a function to **compute factorial**.
4. Write a function which takes **1 int parameter**. The function calculates the **sum** of all numbers **between 0 and the parameter**.
5. Write a function which takes **an in** as a parameter and prints on the console **all its divisors**.
6. Write a function which takes **an int** as a parameter and finds **how many digits** it has.
7. Write a function to calculate the `N`th Fibonacci number where `N` is the parameter of the function.
8. Write a function which takes **2 int parameters** - Base and Power and calculates $Base^{Power}$. e.g `pow(2, 10)` returns `1024`.

Recursion (Double recursion)

1. Write a program which generates a random number between 1 and 100 and prompts the user to guess the number. The program tells the user if he guessed bigger, smaller or the exact number.
2. Print all possible numbers with up to 5 digits containing the numbers `1,2`.
3. Write a recursive function which takes an int array and a number `N`. The function prints all possible combinations of `N` elements of the given array.

Arrays (Fixed size)

1. Create an array with **3 ints** `3, 4, 7` and print them on the console.

2. Create an array with **100 ints** from 1 to 100. Divide each element of the array by 2 and print the array.
3. Write a function which accepts an **array of ints with 5 elements** and subtracts 1 from each element.
4. Write a function which accepts **2 arrays of integers with 3 elements** and adds the elements of the second array to the first. As a result `arrayOneProcessed[i] = arrayOne[i] + arrayTwo[i]`.

Pointers

1. Create a variable `D` with value `420`. Create another variable `X` of the same type. Create a pointer to the variable `X`. Try to find the address of `D` by moving the pointer around.
2. Determine if your machine is using little or big endianness.

Strings

1. Write a function which finds the **length of a string**.
2. Write a function which **compares two strings**.
3. Write a function which takes a **string** and returns a **copy of it**.
4. Write a function which takes a **string** and prints it **reversed**.
5. Create a string variable of length **at most 100 letters**. Ask the user to input a word and save it in that string variable. Print the length of the string.
6. Prompt the user how long their name is. Then prompt for their name and print on the console `Hello, <UserNameHere>!`.
7. Write a function which takes **a string** and determines if it is a **palindrome**. A palindrome is a string which is **read the same way forwards and backwards**.
8. Write a function which takes in a **string** and prints what characters we need to add to the end of the string for it to become a palindrome.
9. Write a function which converts a **binary string** (composed from 0s and 1s) to a **decimal number**.
10. Write a function which converts an **int** to a **binary number** as a string (reversed), e.g `convertToBinary(13)` returns `"1011"`.
11. Write a **recursive function** which prints the **binary representation** of a **number in reverse****.

Arrays (One dimensional with arbitary size)

1. Write a function which accepts **an array of integers** of arbitary size and makes all the integers negative. (Hint: the function must also accept the size of the array. Array and pointer here are interchangeable)
2. Write a function which takes **an array** (pointer), its size and element we are looking for and retuns the **index at which that element is found** in the array or -1 if the array doesn't have such element.
3. Write a function which accepts **an array of chars** with arbitary size and returns the **sum of all the elements**.
4. Write a function which takes **an array of ints** and outputs **all the even numbers** on the console.
5. Write a function which takes **an array of chars** and calculates the **average of all odd numbers**.
6. Write a function which takes **an array of ints** and **reverses** it in place.
7. Write a **recursive function** which finds the **max number in an array**.

Arrays (Two dimensional)

1. Prompt the user for a matrix size they'd like to have. Create the matrix and prompt the user to input its values. Print the matrix to the console.
2. Create a struct representing a 3x3 matrix. Write a method to add two matrices.
3. Create a struct representing 5x5 matrix. Write a method to transpose the matrix.
4. Create a struct representing a `NxN` matrix where N is specified by the user. Write a method which calculates the sum of each row and prints it to the console.