LECTURE #3

Readings (10th Edition):
Section 1.2: Programming and Problem Solving
Chapter 2 until 2.3: Data

SOME THINGS TO REMEMBER

ALGORITHM

- **Algorithm : A sequence of steps to solve a problem**
  - When it is and when it is not an algorithm
    - It is an algorithm when you can follow a set of steps to complete a certain task
    - It is not an algorithm when you randomly try things hoping that by pure luck things will magically work out and complete the task
  - The biggest secret of programming is …
    - think about the problem before you code!!!!
    - Understand the problem, if you don't understand the problem how can you find solutions for it
  - **Remember:** In order to be an algorithm you need:
    - A task that needs to completed
    - A set of rules and steps to accomplish it
    - A final state where you can consider yourself done

YOUR FIRST PROGRAM:

```
// CSC 125
// File Name: testFile.cpp
// Author: Alex Jerez

// IMPORTANT: this is our legal contract, every program
// you code has to have your name and date when you created your code.
// if your program doesn't have this information or your code matches
// somebody elses code disciplinary action will be taken against both of you.
```

```
// Description: First "hello world" program used as an example for class.

#include <iostream>

int main()
{
	// descriptions throughout your code are very important. Explain what you are doing
	// this program will display “hello world” in a console

	std::cout << "Hello World" << std::endl;

	getchar();

	return 0;
}
```

ANATOMY OF OUR PROGRAM

- **Include Sections:**
  - Adding standard libraries (or user created libraries to our program)
- **Statements**
  - Every statement in C++ should end with a ;
  - Remember this because it will be the source of many mistakes (including my own)
- **Blocks of code**
  - Defined by { } brackets, allow for breaking down the code into specific sections
  - Do not require a ; at the end
- **Main Function Definition**
  - This function has to be part of any program we code.

BASIC MATH OPERATIONS

- As simple as it sounds you have to remember your math. There is a huge difference between 2*5+7 and 2*(5+7).
- Remember your basic math operators
  - +
  - –
  - /
  - *
  - ( )
  - %

DATA TYPES

- Numerical notations
  - Decimal Notations
  - Binary Notation
  - Hexadecimal Notation
  - Nth Notation

- A computer doesn't understand numbers but it does understand voltage changes
  - High Voltage (~5V) → 1
  - Low Voltage (0) → 0
  - Therefore, a binary system is the perfect solution for a computer
  - BIT
    - Short for **BI***nary digi***T**
    - It can only have two possible values : 1 or 0

- Bit -> bytes -> Kilobytes -> Megabytes -> GigaTerabytes -> PetaBytes -> Exabytes
  - · 1 Bit = Binary Digit
    · 8 Bits = 1 Byte
    · 1024 Bytes = 1 Kilobyte (2^10)
    · 1024 Kilobytes = 1 Megabyte (2^20)
    · 1024 Megabytes = 1 Gigabyte (2^30)
    · 1024 Gigabytes = 1 Terabyte (2^40)
    · 1024 Terabytes = 1 Petabyte (2^50)
    · 1024 Petabytes = 1 Exabyte (2^60)

  - What is the meaning of all these numbers?
    - 1 Gigabyt ~= 7 minutes of HD-TV video
    - 1 Petabyte = 13.3 years of HD-TV
    - 20 Petabytes = amount of data google process every day
    - 1 Hexabyte = enough to write down every single piece of human knowledge and you will only use 5% of it.

  - Where do we see this?
    - Encryption
    - OS
    - Gaming Machines

NOW WE CAN TALK ABOUT **DATA TYPE**S AND **VARIABLES**

- A **variable** is a memory location that is used to store information
  - Computer Memory is linear (starts at 0 and increments to the amount installed – Remember this, it will be very important later)
  - **A variable is then a memory location** whose content is always changing
  - This is also something important to remember. Memory Location vs Memory content:
    - **Memory Location or Memory Address** is a place in memory that has an address that we can use to acces
    - **Memory Content** is what exists within that **Memory Location**
    - For example: Our program can execute the following code:

`int a;`
`int b;`
 	
`a = 10;`
`b = a;`
 	
`int c;`
	// create a variable A in memory
	
	* We start at the first address in memory: **1200** (normally a hexadecimal number, this is just an example)
		…: ??
		1208: ??
		1204: ??
		1200: **A = ?**

`int a;`
`int b;`
 	
`a = 10;`
`b = a;`
`int c;`
	// create a variable B in memory
	* We move to the next available memory address: **1204** (why 4?, read the next section and it will make sense)
		…: ??
		1208: ??
		1204: **B = ?**
		1200: **A = ?**

`int a;`
`int b;`
 	
`a = 10;`
`b = a;`
`int c;`
	// assign the value 10 to the variable **a** located at **1200**	…: ??
		1208: ??
		1204: **B = ?**
		1200: **A = 10**

`int a;`
`int b;`
 	
`a = 10;`
`b = a;`
`int c;`
	// assign the value stored in **a** to **b,** but just the value, so we make a copy of it.	…: ??
		1208: ??
		1204: **B = 10**
		1200: **A = 10**

`int a;`
`int b;`
 	
`a = 10;`
`b = a;`
`int c;`
	// We create a new variable c in the next available memory location.
	** Although this is allowed, you should never define variables in random locations, all your variables should be defined at the top of your code
		… ??
		 1208 **C = ?**
		 1204 **B = 10**
		 1200 **A = 10**

- A **data type** is an assigned format (or size) given to a variables
- Data types:
  - Char / strings
  - Short Int / Int / Long Int
  - Bool
  - Float /double / long double

Name	Description	Size*	Range*
`char`	Character or small integer.	1byte	signed: -128 to 127 unsigned: 0 to 255
`short int (short)`	Short Integer.	2bytes	signed: -32768 to 32767 unsigned: 0 to 65535
`int`	Integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
`long int (long)`	Long integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
`bool`	Boolean value. It can take one of two values: true or false.	1byte	`true` or `false`
`float`	Floating point number.	4bytes	+/- 3.4e +/- 38 (~7 digits)
`double`	Double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
`long double`	Long double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
`wchar_t`	Wide character.	2 *or* 4 bytes	1 wide character

DEFINING A VARIABLE

- Formal definition of a variable
  - [data type] [**variable identifie**r] {= *assignment*} ;
  - Data type:
    - char/int/float/etc
  - Variable Identifier (Left Hand Side):
    - Unique name given to a variable
    - It cannot be a keyword used by c++
    - Case sensitive
    - No spaces, no starting with numbers, no using special characters
    - Example:
      - int variable_a;

        - bool variable_b;
        - int variable_c = 3; // a value assigned at creation (Right Hand Side)
        - char variable_d = 'f';
        - float _variable_e = 1.4;
    - Just as we can create a variable and assign a value, once we have a variable we can assign new values to it as long as:
        - We have matching data types
        - Data types are super sets of other data types
        - We use casting

CASTING AND SIZE

- What does it mean?
    - We can convert from one data type to another (in some cases it won't make sense)
    - a = int(5.6) // now a = 5 (**it is not rounding**, it just drops the decimal part)
    - a = float(5) // now a = 5.0

- Messing with values
    - What happens when you cast an int to a character?
        - char a = int('a');
        - Create a single program and test the solution.
        - Hint, ASCII values!
    - sizeof()
        - int a = 9
        - cout << sizeof(a); // look back a few sections and you will find what this means

USER INPUT/ CONSOLE OUTPUT

- **How do we get information into our programs?**
    - Very simple, if **cout** is way to display information to a console (output) then **cin** is going to be the way we bring information from the keyboard.

    std::cin >> test1; // important, noticed the direction of the >>

- Getting information displayed to the console
    - std::cout << test1; // again the direction of the << is important

EXAMPLE: A PROGRAM TO READ A VARIABLE AND PRINT IT

```
// Define Section
#include <iostream>

// Using namespace ( prevent us from having to use std:: all the time
using namespace std;

// main function
int main()
{
	// variable used to get information from the user
	int num_variable;

	cout << "================= CSC123: Enter a number ================="<< endl;
	cout << "Enter an integer number: ";
	 cin >> num_variable;

	cout << "Your number is: " << num_variable << endl;
	cout << "Your number multiplied by 2 is: " << num_variable*2 << endl;

	getchar( );

	return 0;
}
```