CMP 1001

Introduction to Programming

Part 1: Introduction to programming and C++

by

Assist. Prof. M. Şükrü Kuran

WELCOME!

Introduction to Programming

+

OUTLINE

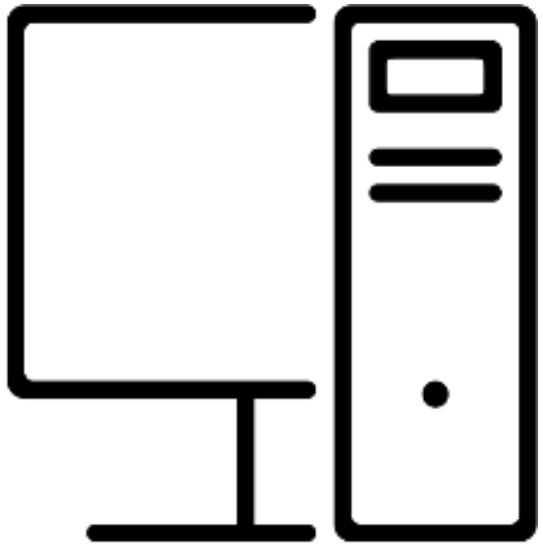
CMP 1001

Section 1: Introduction to computing & programming

INTRO TO COMPUTING & PROGRAMMING

Computer:

1. A device for counting and computing
2. A machine that can process data



A computer **CAN NOT** reason as we do.

To perform some task by using computers, we need to tell **ALL** necessary steps (i.e., instructions or statements) to the computer for it to perform the task.

The list of instructions is called a "**program**".

INTRO TO COMPUTING & PROGRAMMING

"The **purpose of programming** is to find a sequence of instructions that will **automate performing a specific task** or solving a given problem."

The instructions must be written as **programming languages**

1. Machine language (i.e., Assembler)
2. High-level programming languages (i.e., C++, C, Python, Java, etc…)

PROGRAMMING LANGUAGES

1. Machine language

a) Native tongue of a computer (kind of)

b) Each instruction is a binary string. (i.e., 01011011100110)

2. High level languages

a) Human-readable languages, (e.g., C++, C, Python, C#, GoLang).

b) Each language has its own standards and predefined instructions.

c) Each language has libraries that are coded by other people (i.e. **Application Programming Interface (API)**).

d) Translated in machine language by either a compiler or interpreter

CMP 1001

Section 2: Introduction to C++

C++ LANGUAGE

C++ is a general-purpose "**high level computer programming language**". It is a compiled language with performance, efficiency, and flexibility as its design highlights.

For the last decade, C++ is one of the most popular computer languages.

The language derives much of its syntax from C.

C++ VERSIONS

C++ is an ever-changing language by keeping its roots and general concepts but adapting to new needs.

C++98 C++03 C++11

C++14 C++17 C++20

CLion

How can I write programs in C++?

We use some specialized tools called

Integrated development environments (IDEs).

CLion is one of the most commonly used IDE for software development in the C++ language.

- It is a free IDE for academic purposes
- Used extensively both for personal and professional uses

CLion IDE for C/C++ download

https://www.jetbrains.com/clion/

MinGW

GNU Compiler Collection for MS Windows

- GCC is the standard C/C++ compiler system in Linux & MacOS
- MinGW and Cygwin are Windows ports of GCC (well… basically)
- To compile and run C/C++ programs in Windows MinGW or Cygwin are very good candidates
- **You have to install MinGW/Cygwin before your IDE**

MinGW for Windows download

http://www.mingw.org/

C++ PROJECTS

How can I write programs in C++?

Using CLion, C++ programs are written in "**classes**" which are inside "**C++ Projects**".

Example:

Open up a new empty C++ project in CLion. Select C++ Executable with language standard C++14

STATEMENTS

Programming Commands

We type the commands into .cpp files. Commands are named "**statements**" in C++.

```
<statement>;
```

1 Line = 1 Statement

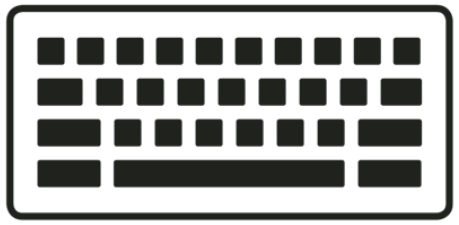
NOTE: Each statement ends with a ";" character.

NOTE: If you are making a mistake while writing a statement, Clion will warn you with an error icon "near" the erroneous code line.

INPUT/OUTPUT IN C++

C++ is a general programming language. Therefore we get the input from **ANY** input source we want (*i.e., keyboard, mouse, touchscreen, scanner, Internet, etc…*)

Similarly, we give out the output on **ANY** output source we want (*i.e.,* ***monitor/console****, printer, Internet, etc…*)



INPUT: Keyboard, File

OUTPUT: Monitor/Console, File

STD::COUT

Output to the Console

We use the "**std::cout**" function to write something to the console.

```
std::cout << "Greetings!";
```

```
std::cout << "Greetings!" << std::endl;
```

Example:
Write a program that writes "Hello CMP 1001!" on the console

STD::CIN

Input from the Keyboard

Getting an input from the keyboard is a bit trickier than writing something to the console.

1. Define where to put the input we get from the keyboard.
2. Use the "**std::cin function**" to read the input.

STD::CIN

Step 1: Define where to put the input we get from the keyboard

For this purpose, we declare a variable into the program which can store a value. The most basic variable type is called the "**integer**" which can hold an integer value.

```
int number;
```

You can declare an integer with any name you want. But there are some naming rules.

STD::CIN

Step 2: Using std::cin function to read the value from the keyboard

```
std::cin >> number;
```

Example:
Write a program that reads 2 numbers as integers, calculates the summation of them, and prints the sum.

Example:
Write a program that reads 2 numbers as integers, calculates and prints the summation, subtraction, multiplication, division, and mod of these numbers.

CMP 1001

Section 2: Variables

VARIABLES

Main Variable Types

In most common programming languages (i.e., imperative languages) we have something called "**variables**" to keep data values.

```
int number;
```

There are two main types of variables in C++:

1. **Fundamental Data Types**
2. **Classes**.

Fundamental Data Types are simple constructs that can hold a **SINGLE VALUE** of the given type.

INTEGER

Storing Integer Number Values

Integer (**int**) is used to store a single integer number.

Declaration: We use "**int**" for declaring integer variables.

```
int number;
```

Data: An "**int**" variable can take any positive, negative integer value or "0".

```
number = 6674;
```

```
std::cin >> number;
```

INTEGER

Integer Types

There are several integer variants in C++. Their sizes are dependent on the implementation and environment.

Primitive Name	Size (in bits)	Value Range
int	At least 16	$-2^{15}\ to\ 2^{15}-1$
short int	At least 16	$-2^{15}\ to\ 2^{15}-1$
long int	At least 32	$-2^{32}\ to\ 2^{32}-1$
long long int	At least 64	$-2^{64}\ to\ 2^{64}-1$

DOUBLE

Storing Real Number Values

Double (**double**) is used to store real numbers (i.e., 1.54, 6.783)

Declaration: We use "**double**" for declaring double variables.

```
double realNumber;
```

Data: A "**double**" variable can take any positive, negative real value or "0".

```
realNumber = 316.074;
```

```
std::cin >> realNumber;
```

DOUBLE

Double types

There are several integer variants in C++. Their sizes are dependent on the implementation and environment.

Primitive Name	Size (in bits)	Value Range
float	32	$1.175 \cdot e^{-38}$ *to* $3.4 \cdot e^{38}$
double	64	$2.25 \cdot e^{-308}$ *to* $1.79 \cdot e^{308}$
long double	96	$3.3621 \cdot e^{-4932}$ *to* $1.189 \cdot e^{4932}$

Example:

Write a program that reads 2 real numbers as doubles, calculates the summation of them, and prints the sum.

BOOLEAN

Storing a yes/no information

Boolean (**bool**) is used to store a true/false value.

Declaration: We use "**bool**" for declaring bool variables.

```
bool flag;
```

Data: A "**bool**" variable can take either "0" (i.e., false) or "1" (i.e., true)

```
flag = true;
```

```
flag = -5;
```

```
flag = 1;
```

```
std::cin >> flag;
```

NOTE: Any value other than "0" is considered true (i.e., "1")

CHAR

Storing Character Values

Character (**char**) is used to store a single character.

Declaration: We use "**char**" for declaring character variables.

```
char ch;
```

Data: A "**char**" variable can take any character value (letter, digit, …) including non-ASCII characters.

```
char ch = 'A';
```

```
std::cin >> ch;
```

CHAR

ASCII/UNICODE Value

Each character value has a corresponding "**ASCII/UNICODE value**". Imagine this as the numerical representation of that character for the computer.

(i.e., ASCII code of 'A' is 65. So the code of 'A' is 65 for the computer)

```
char ch;
int asciiValue;
std::cin >> ch;
asciiValue = ch;
```

Example:

Write a program that reads a character variable, prints the character and its ASCII/UNICODE value on the console.

STRING

Storing Multiple Character Values

String (**std::string**) is used to store multiple characters (i.e., words).

Declaration: We use "**std::string**" for declaring string variables.

```
std::string word;
```

Data: A "**std::string**" variable can take a word of any length. including non-ASCII characters and whitespace characters.

```
std::string word = "Example";
```

```
std::cin >> word;
```

STRING

Storing Multiple Character Values

NOTE: String is NOT a fundamental data type, it is a Class but it is a very CORE Class.

NOTE: std::cin reads characters until it finds a whitespace character. Space, end of line character (i.e., enter), and tab are whitespace characters.

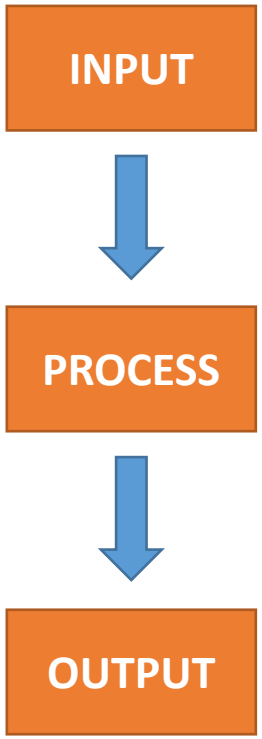
You can use the getline() function to read a line (including blanks and tabs) to a String.

```
getline (std::cin, word);
```

Example:

Write a program that reads a String variable as a line and prints it to the screen.

STEPS OF SOFTWARE DEVELOPMENT



1. **State the problem**

   Understand what the problem is? What do you want the program to do?

2. **Analyze the problem (i.e., identify inputs and outputs)**

   What will be given to the program as the input?

   What will be asked to the program as the output?

3. **Design an algorithm (i.e., specify the list of steps)**

   A program is written with a sequence of "**statements**".

4. **Implement the steps by using a programming language**

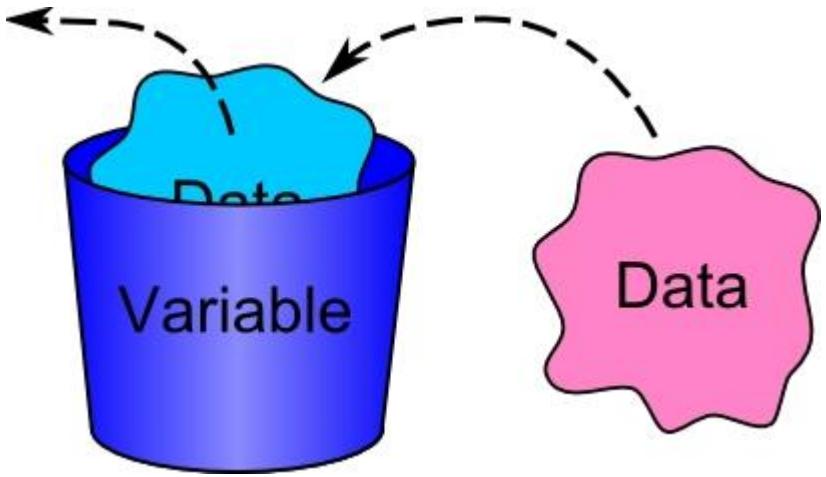
5. **Compile and execute the program.**

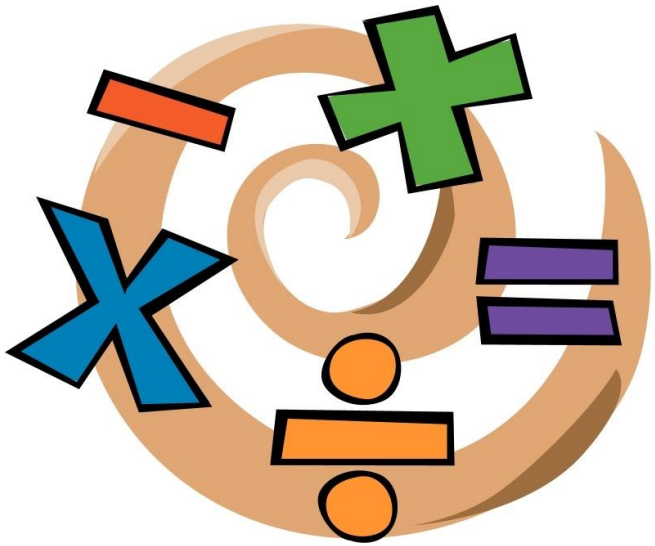
   After finishing writing your program, you have to "**compile**" it according to the C++ rules.

   If there are no errors (i.e., mistakes), execute (i.e., run) your program with some inputs.

COMING SOON...

Next week on CMP 1001





VARIABLES & OPERATORS