

# Lecture 1 Notes

## Introduction

CSE 201 is one of the first CSE courses that students at CSUSB enroll in. Many new students are unaware of what Computer Science demands from its potential scholars. It is first important to know that we are not studying the computer; the computer is simply a tool for us to apply our theories. What we are studying is: not the computer, but computation itself, or (in other words) algorithms. An algorithm is a series of instructions for performing a particular task.

Creating and applying algorithms involves significant application of logic and mathematics. Inside a computer, an algorithm can physically exist in the form of a computer program. The process of creating a computer program is called computer programming.

CSE 201 serves as an introduction to computer programming. We will use a programming language known as C++.

## Your first program

The first program that you usually create in any programming language is known as the “hello world” program. The only function of this program is to display a message. In C++ that program can be expressed in the following form:

Listing 1: hello.cpp

```
#include <iostream>
using namespace std;

int main()
{
    cout << "hello world" << endl;
    return 0;
}
```

This is what we call the source code of a computer program. Source code is the human readable version of a computer program’s algorithm. We must convert source code into a computer program before we can execute it (a process known as compiling), this is done by using a program known as a compiler.

Compilers as well as other programmer tools can be executed through the operating system’s command-line interface.

Assuming that you are using Linux, if you saved hello.cpp in your Documents folder, then you can compile it by typing these commands into your terminal emulator:

```
cd ~/Documents
c++ hello.cpp
```

A program named “a.out” will be produced. You may execute the a.out program with the following command:

```
./a.out
```

Take note that the compiler and the hello world program both execute within an environment known as the command-line interface. This environment has text-only graphics;

devoid of any buttons, scrollbars, or images. This is a programmer-friendly environment that is not only easier to create programs in, but also extends more power to the user.

## Data types

A computer program is usually created for processing data. Processing data requires being able to distinguish between different types of data. The C++ programming language supports many elementary data types, four of them that you should know are in the following table:

Data type	Size	Range
bool	1 byte	true or false
char	1 byte	-128 to 127
int	4 bytes	-2147483648 to 2147483647
double	8 bytes	$\pm 1.7 \times 10^{308}$

bool is short for boolean. Data of type bool can have one of two values: true or false. Some expressions in C++ often result in a boolean value, this is typically the case when you are mathematically comparing two values.

char is short for character. It represents almost any key that you can press from your keyboard. A character can have a value such as: 'a', '3', '%', '.', ' ', 'J'. All text that you see on your computer screen is stored in the computer's memory as sequences of characters. While C++ allows you to express characters as their symbol in single-quotes, each character actually represents an integer within the range of 1 byte.

int stands for integer. It represents a negative or positive whole number. Integers are useful for basic arithmetic where fractional values are not required.

double stands for double-precision floating-point. You do not need to know what that means exactly, just know that double is used for representing fractional numbers. If you are performing any scientific or financial calculations, you should use this data type as opposed to an integer.

There are many other basic data types in C++, some of them are variations of the int and double data types. In this course, we will stick to these four basic data types.

## Variables

In addition to distinguishing between different types of data, computer programs should have a way to store data. C++, as well as many other programming languages, allow you to do this by declaring variables.

In C++, all variables have a data type, a name, and a value. To declare a variable, you must specify at least the data type and the name of the variable. For instance:

Listing 2: A series of variable declarations in C++

```
int x;
double amount;
bool cointoss;
int weight = 150;
double balance = 5071.12;
char q = 'A';
bool verified = true;
```

As you can see, variables can optionally have a value assigned upon declaration. If no value is assigned, the default value is used:

Data type	Default value
bool	false
char	'\0'
int	0
double	0.0

After a variable has been declared, it's stored value can be fetched by simply referring to the name of the variable.

## Expressions and Statements

Formally, every C++ program is a sequence of statements. Each statement performs an action. There are two types of C++ statements, compiler statements and preprocessor statements.

Preprocessor statements begin with a '#' symbol at the start of a line. These type of statements are used for modifying or generating compiler statements. For example, the '#include' preprocessor statment is used for inserting source code from an external file into the current source code.

Compiler statements describe what the program will do. Compiler statements are delimited by a semicolon, and you may have multiple of these statements on a single line.

Some statements result in a value which can be assigned to a variable, in which case the statement is also an expression. All expressions are statements, but not all statements are expressions.

This is a statement:

```
cout << "hello.";
```

The action this statement performs is the displaying of the character sequence "hello.". This statement does not produce a value.

This is an expression:

```
a + 4;
```

This expression calculates the value of variable a added to the value 4. This produces a value which can be assigned to a variable, displayed to the screen, or used in another statement.

## Analysis of the Hello World program

Let us return to your first program, the hello world program. Here is hello.cpp again with line numbers on the left:

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     cout << "hello world" << endl;
6     return 0;
7 }
```

Line 1 is a preprocessor statement, `iostream` is the name of a file included in every C++ distribution. `iostream` contains statements that allows us to receive and display messages in our terminal window. One variable that `iostream` exports is the “cout” variable. `cout` stands for character out. All data that you write into `cout` gets displayed in your terminal window, or more generally what we call “standard output”.

Line 2 is related to a concept in C++ known as namespaces. In C++, when we declare variables or functions we can choose to place them inside a namespace. Namespaces can exist inside other namespaces too. The perceived benefit of namespaces is to prevent source code from different programmers from conflicting with each other if their code happens to use the same variable names. To access a namespaced variable, requires that you type the name of all the containing namespaces as well as the variable name. The built-in source code bundled with every C++ distribution exists under the namespace called “std”. All variables that do not exist inside a declared namespace are said to exist in the global namespace.

The “using namespace std” statement means that we wish to take all variables from inside the `std` namespace and move them into the global namespace. Without this statement, we would have to type “`std::cout`” every time we wish to use `cout`. We would also have to type “`std::`” before using any other variable from the C++ library. This command allows us to use `cout` without typing the preceding “`std::`”.

Line 3 is the declaration of a function called `main`. You will learn more about functions later on in this course, for now you should just know that the `main` function signifies the starting point of your program’s execution.

Lines 4 and 7 encloses the sequence of statements that gets executed inside the `main` function.

Line 5 is the statement that displays the character sequence “hello world”. Recall that a value of type `char` is enclosed in single-quotes. C++ allows us to create a sequence of characters by using double-quotes. This sequence of characters is what we call a C-string. C-strings have the type “`const char *`” which you will learn more about later. On line 5, we are taking the C-string value “hello world” and writing it into our variable “`cout`”. The `<<` operator indicates that we are writing data from the right-hand side of the operator to the variable on the left-hand side of the operator. The `<<` operator used in this context only works on variables that are of type “`ostream`”. `ostream` is a special data type included in C++. You may write multiple data into an `ostream` object in one statement by chaining the `<<` operator. In our example, we write both “hello world” and the value of variable `endl` into `cout`.

`endl` is a special built-in variable that inserts a line-break into any `ostream` object that it is written to.

Line 6 is the final statement that gets executed while our program is running. It instructs the execution of our `main` function to stop and give the value zero as the exit status code to our operating system. After running any program in our terminal, you may use the following command to check that program’s exit status:

```
echo $?
```

The result will be 0 if the program exited normally, any non-zero value indicates an error.

You may change the value in `hello.cpp` on line 6 to something like “`return 2;`”. After recompiling and running your program, use the “`echo $?`” command to see the exit status.