C++ Chapter 2: Getting Started

Contents

How Does Typing in a Code Tutorial Help with Learning?

Typing in a code tutorial can significantly enhance learning in several ways:

- **Active Engagement**: Typing the code yourself forces you to actively engage with the material, rather than passively reading or watching. This active participation helps reinforce the concepts being taught.
- **Muscle Memory**: Repeatedly typing code helps build muscle memory, making it easier to recall syntax and structure when you write code independently.

- **Error Handling**: When you type code, you're likely to make mistakes. Debugging these errors helps you understand common pitfalls and how to resolve them, which is a crucial skill for any programmer.
- **Understanding**: Typing out code allows you to see how different parts of the code interact. This deeper understanding can help you apply similar concepts to different problems.
- **Retention**: Studies have shown that actively doing something helps with retention. By typing out the code, you're more likely to remember the concepts and techniques.

Time required: 90 minutes

How Popular is C++?

https://pypl.github.io/PYPL.html

https://www.tiobe.com/tiobe-index/

Read: Think C++

- Chapter 2 Variables and Types

Do: Online Tutorials

- C++ Tutorial
- C++ Intro
- C++ Syntax
- C++ Output (Print Text)
- C++ New Lines
- C++ Comments
- C++ Variables
    - C++ Declare Multiple Variables
    - C++ Identifiers
    - C++ Constants
- C++ User Input

- C++ Data Types
    - C++ Numbers
    - C++ Booleans
    - C++ Characters
    - C++ Strings
- C++ Operators
    - C++ Assignment Operators
    - C++ Comparison Operators
    - C++ Logical Operators

Introduction to C++

C++ (pronounced see plus plus) is a general-purpose programming language that supports both procedural and object-oriented programming paradigms. It is compiled into a binary executable. It is regarded as an intermediate-level language, as it comprises both high-level and low-level language features. Key features include strong typing, manual memory management, and support for classes and objects.

Machine Independent but Platform Dependent: A C++ executable is not platform-independent (compiled programs on Linux won't run on Windows), however they are machine independent.

A C++ program can be compiled and executed on any operating system that has a C++ compiler.

C++ is one of the most popular programming languages and is implemented on a wide variety of hardware and operating system platforms. As an efficient performance driven programming language, it is used in systems software, application software, device drivers, embedded software, high-performance server and client applications, and entertainment software such as video games.

What is C++?

C++ is a cross-platform language that can be used to create high-performance applications.

Who developed C++?

Bjarne Stroustrup developed C++ (originally named "C with Classes") in 1983 at Bell Labs as an enhancement to the C programming language.

What Is a Program?

A program is a sequence of instructions that specifies how to perform a computation. The computation might be something mathematical, like solving a system of equations or finding the roots of a polynomial. It can also be a symbolic computation, like searching and replacing text in a document or (strangely enough) compiling a program.

The instructions (or commands, or statements) look different in different programming languages, but there are a few basic functions that appear in just about every language:

input: Get data from the keyboard, or a file, or some other device.

output: Display data on the screen or send data to a file or other device.

math: Perform basic mathematical operations like addition and multiplication.

testing: Check for certain conditions and execute the appropriate statements.

repetition: Perform some action repeatedly, usually with some variation.

Believe it or not, that's pretty much all there is to it. Every program you've ever used, no matter how complicated, is made up of functions that look like these. One way to describe programming is the process of breaking a large, complex task up into smaller and smaller subtasks until eventually the subtasks are simple enough to be performed with one of these simple functions.

Values and Variables

The number four (4) is an example of a numeric value. In mathematics, 4 is an integer value. Integers are whole numbers, which means they have no fractional parts, and an integer can be positive, negative, or zero. Examples of integers include 4, -19, 0, and -1005. In contrast, 4.5 is not an integer, since it is not a whole number.

Constants

C++ supports constants. PI is not included in C++. We have to create our own constant. Constants are declared like variables with the addition of the **const** keyword.

```
const double PI{3.14159265358979};
```

This is a double data type representation of PI in C++ as a constant.

Once declared and initialized, a constant can be used like a variable in all but one way—a constant may not be reassigned. It is illegal for a constant to appear on the left side of the assignment operator (=) outside its declaration statement. A subsequent statement like

```
PI = 2.5;
```

would cause the compiler to issue an error message.

Operators and Operands

Symbol	Operation	Example	Description
+	Addition	a + b	Adds two numbers.
–	Subtraction	a – b	Subtracts one number from another.
-	Negation	- a	Change sign of operand.
*****	Multiplication	a * b	Multiplies one number by another.
/	Division	a / b	Divides one number by another and gives the result as a floating-point number. If both values are int, the result is integer division.
%	Remainder or Modulus	a % b	Divides one number by another and gives the remainder.

Operators are special symbols that represent computations like addition and multiplication. The values the operator is applied to are called operands.

Variables in C++ must be defined before they can be used.

```
int number;
double a_double{24.0};
float b_double{60.0};
```

The following table lists some of the data types, sizes, and ranges in C++.

Type	Size (in bytes)	Range
short int	2	-32,768 to 32,767

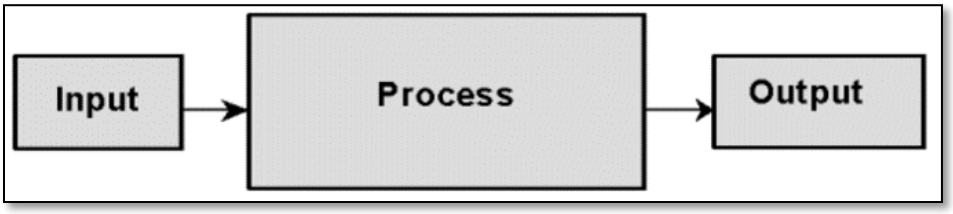
unsigned short int	2	0 to 65,535
unsigned int	4	0 to 4,294,967,295
int	4	-2,147,483,648 to 2,147,483,647
long int	4	-2,147,483,648 to 2,147,483,647
unsigned long int	4	0 to 4,294,967,295
signed char	1	-128 to 127
unsigned char	1	0 to 255
char	1	-128 to 127
float	4	7 decimal points
double	8	15 decimal points
long double	16	30
bool	1	

Expressions

- **statement**: a unit of code that does something – a basic building block of a program.
- **expression:** a statement that has a value – for instance, a number, a string, the sum of two numbers, etc. 4+2, x-1, and "Hello, world!\n" are all expressions.

Input - Process - Output

All programs have three basic components.



Input

User-entered values are passed to the input stream using the **cin** keyword which takes input from the keyboard using the operator **>>**

```
/**
 * Filename: temperature_converter.cpp
 * Written by:
 * Written on:
 * Description: Convert fahrenheit to celsius
 */

#include <iostream>
int main()
{
    double fahrenheit;
    double celsius;
    std::cout << "+-----------------------------------+" << std::endl;
    std::cout << "|-- Convert Fahrenheit to Celsius --|" << std::endl;
    std::cout << "+-----------------------------------+" << std::endl;
    std::cout << " Enter temperature in F: ";
    // Program pauses until user enters a value and presses Enter
    // User input is assigned to fahrenheit
    std::cin >> fahrenheit;
    // Convert fahrenheit to Celsius
    celsius = ((fahrenheit - 32) * 5.0) / 9.0;
    // Output results to the console
    std::cout << fahrenheit << " Fahrenheit " =
              << celsius << " Celsius" << std::endl;
    // Return success status to the OS
    return 0;
}
```

Example run:

```
Enter temperature in degrees F: -40
-40 Fahrenheit = -40 Celsius
```

Tutorial 1: Favorite Number

Create a C++ program named **favorite_number.cpp**

```cpp
/**
 * @name favorite_number.cpp
 * @date
 * @brief Get name and favorite number from user
 */
#include <iostream>

int main()
{
    // Declare variable
    int favorite_number;
    std::string name;
    // Prompt the user for input
    std::cout << "\nHello, what is your name?: ";
    // Get input from keyboard
    std::cin >> name;
    // Prompt the user for a number
    std::cout << "\nEnter your favorite number between 1 and 100: ";
    // Get input from keyboard
    std::cin >> favorite_number;
    // Output to console, split long line of code
    std::cout << "Amazing " << name << "! "
              << favorite_number << " is my favorite number too.\n";
    return 0;
}
```

Example run:

```
Hello, what is your name?: Bill

Enter your favorite number between 1 and 100: 42
Amazing Bill! 42 is my favorite number too.
```

Tutorial 2: Wages

```cpp
/**
 * Filename: wages.cpp
 * Written by:
 * Written on:
 * Description: Calculate wages per month
 */

#include <iostream>

int main()
{
    // Declare variables to store hourly wage, hours worked, monthly salary
    double hourlyWage;
    double hoursWorked;
    double monthlySalary;

    // Ask the user to input their hourly wage
    std::cout << "Enter your hourly wage: $";
    // Store the user's input in 'hourlyWage'
    std::cin >> hourlyWage;

    // Ask the user to input the number of hours worked
    std::cout << "Enter the number of hours worked in a month: ";
    // Store the user's input in 'hoursWorked'
    std::cin >> hoursWorked;

    // Calculate the monthly salary by multiplying wage and hours
    monthlySalary = hourlyWage * hoursWorked;

    // Display the result to the user
    std::cout << "Your monthly salary is: $" << monthlySalary << std::endl;

    // Return 0 to indicate successful program execution
    return 0;
}
```

Example run:

```
Enter your hourly wage: $75
Enter the number of hours worked in a month: 40
Your monthly salary is: $3000
```

Assignment 1: Miles to Kilometers

Copy this program, paste it into a file named **MilesToKilometers.cpp**

Convert the following program from Java to C++.

```java
/**
* Filename: MilesToKilometers.java
* Written by:
* Written on:
* Miles to Kilometers converter
*/
import java.util.Scanner;

class MilesToKilometers {
    public static void main(String[] args) {

        // TODO Declare conversion constant
        // 1 mile = 1.609344 kilometers
        final double MILES_IN_KILOMETERS = 1.609344;
        // Declare variables
        double miles;
        double km;

        // TODO: Print creative title
        System.out.println("************************************");
        System.out.println("*                                  *");
        System.out.println("*    Convert Miles to Kilometers   *");
        System.out.println("*                                  *");
        System.out.println("************************************");

        // TODO: Create Scanner object for user input
        Scanner input = new Scanner(System.in);

        // TODO: Get user input
        System.out.print("Enter miles >> ");
        miles = input.nextDouble();

        // TODO: Convert miles to kilometers
        km = miles * MILES_IN_KILOMETERS;

        // TODO: Print results
        System.out.println(miles + " miles: " + km + " kilometers");

        input.close();
    }
}
```

Example run:

```
+-----------------------------------+
|--  Convert Miles to Kilometers  --|
+-----------------------------------+
Enter the distance in miles: 10
10.0 miles = 16.09 kilometers
```

Assignment Submission

1. Use pseudocode or TODO.
2. Comment your code to show evidence of understanding.
3. Attach the program files.
4. Attach screenshots showing the successful operation of the program.
5. Submit in Blackboard.