Programming in C++

Fundamentals of C++ Programming

Dr. Zeyad Safaa Younus Saffawi

Introduction

- **What is C++?**
- A **general-purpose**, **case-sensitive**, **free-form programming language**.
- It supports **procedural**, **object-oriented**, and **generic programming**.
- It is **statically typed** and **compiled**.
- **Its Level:** It is considered a **middle-level language** because it combines both **high-level** and **low-level** language features.

Environment Setup

- **The Easiest Option (Recommended for Beginners):** Use an online compiler and executor.
  - **Advantage:** You don't need to install any software to start learning. You can compile and execute examples immediately.
  - **Example Site:** http://www.compileonline.com/
- **The Second Option (For Local Setup):** You need two main pieces of software:
  - **Text Editor:** To type your program (e.g., Windows Notepad, VS Code, etc).
  - **C++ Compiler:** The tool that converts your source code into the final executable program.
  - **Note:** C++ source files typically end with the extension .cpp, .cp, or .c.

Structure of a C++ Program

- A **C++ program** is made up of several main parts that must appear in a specific order. Here's the **general structure**:

#include <libraryname> **for example: #include <iostream> // For input/output**

```
using namespace std;
int main ()
{
   // Local Declarations
   // Constants and Local Variables
   // Statements
   return 0;
}
```

Line	Purpose / Function
#include <iostream>	Includes input/output library (must appear **before** the main() function)
using namespace std;	Allows using standard names directly (e.g., cout, cin)
int main()	Starting point of the program
{	Beginning of program body
// Declarations	Place to declare variables/constants
// Statements	Place to write instructions (input, process, output)
return 0;	Ends the program successfully
}	End of main function

Basic Program Structure of a Simple C++ Program

- **Code Example:**

```
# include <iostream>
using namespace std;
int main()
{
  cout << "Hello World";       // prints Hello World to the screen
  return 0;
}
```

- **Explanation of Key Parts:**
  - **# include <iostream>** : Includes a header file containing information useful for the program, which is needed for input/output operations like cout, cin.
  - **using namespace std;** : Allows using standard names directly (e.g., cout, cin), so we can write **cout** instead of **std::cout**.
  - **int main()** : The main function where program execution begins.
  - **cout << "Hello World";** : Prints the message "Hello World" to be displayed on the screen.
  - **return 0;** : Terminates the main() function and returns the value 0 to the calling process (signalling successful execution).

Fundamental Syntax Rules (C++ Writing Rules)

1. **Semicolons**:
   - The semicolon (**;**) is a **statement terminator** in C++.

Example:

```
x = y;
y = y + 1;
 add(x, y);
```

2. **Blocks**:
   - A set of logically connected statements surrounded by **opening and closing braces ({ }).** Used in functions, loops, conditions, etc

Example:

```
{
cout << "Hello World";          // prints Hello World
return 0;
}
```

Fundamental Syntax Rules

3. Identifiers (Names):

- In the **C++ language**, identifier names are used to represent **variables**, **functions**, **classes, objects**, and other program elements.
- An identifier **must begin with a letter (A–Z or a–z) or an underscore (_)** followed by zero or more letters**, digits (0–9), or underscores**.
- It **cannot contain spaces** or **special characters** such as **#, !, @, $, %,** etc.
- Also, **C++ is a case-sensitive** programming language, which means that uppercase and lowercase letters are treated as different

Example*:* **Manpower ≠ manpower** Sum ≠ sum.

Fundamental Syntax Rules

Valid Identifier	Invalid Identifier	Reason
total	2total	Cannot start with a digit
sum_1	total value	Space not allowed
_average	@result	@ is not allowed
NumberOfStudents	grade%	% not allowed
x_y_z	data-file	- not allowed
studentID	float	float is a C++ keyword
value123	123value	Cannot begin with a number
_tempVar	temp var	Space not allowed
MAX_VALUE	#define	# not allowed
countItems	while	while is a reserved keyword
user_name_1	user$name	$ not allowed in C++ identifiers
A1B2C3	C++Name	+ not allowed
Data_Set_2025	!error	! not allowed
isValid	do.	. not allowed
Result_Value	return	return is a reserved word
Speed_of_Light	speed-of-light	Hyphen - not allowed
userInput	int	int is a keyword
Matrix_A_B	Matrix A B	Space not allowed
__config	config@	@ not allowed
Name123_	name#1	# not allowed

Fundamental Syntax Rules

- **Comments** are **non-executable statements** used to explain and document your program.
  - All characters inside a comment are ignored by the C++ compiler.
    - Single-line comment: Starts with **//** and extends to the end of the line.
    - Multi-line / Block comment: Starts with **/*** and ends with ***/**.
    - **Comments** can appear **anywhere in the code**, except in the **middle of an instruction**.
- **Reserved Words:**
  - Words that have a fixed, special meaning in the C++ language.
  - They **can not** be used as variable, constant, or any other identifier names.

    Examples: int, float, return, for, class, if, while, bool.

Fundamental Syntax Rules (Variables and Data Types)

- **DATA TYPES**: While writing a program in any language, you need to use various variables to store various information.
- **Variables** are nothing but **reserved memory locations** to store values. This means that when you **create a variable,** you reserve some space in memory. Every variable has a specific **type** (determines **size** and **layout in memory**). variable definition tells the compiler where and how much storage to create for the variable. It holds data that can be changed during program execution.
  - The name of a variable can be **composed of letters, digits, and the underscore character**. It must begin with either a letter or an underscore. **Upper and lowercase letters are different because C++ is case-sensitive**.

Fundamental Syntax Rules

Basic Data Types

Type	Keyword	Description
boolean	**bool**	Stores logical values: true (1) or false (0).
character	**char**	Stores a single character (1 byte).
integer	**int**	Stores integers (whole numbers).
Floating point	**float**	Single-precision floating point numbers.
Double Floating point	**double**	Double-precision floating point numbers.
Valueless	**void**	Represents the absence of type.
Wide character	**wchar_t**	Represents the wide character

Fundamental Syntax Rules

❖ **Type Modifiers:**

- Can be used to modify the basic types.

*M*odifiers*:* signed, unsigned, short, long.

Fundamental Syntax Rules

Size and Range of Data Types (For Reference)

- **Note:** The size (in bytes) and range (min/max values) of variables can be different depending on the compiler and the computer you are using. For example:

Type	Typical Bit Width	Typical Range
char	1byte	-127 to 127 or 0 to 255
unsigned char	1byte	0 to 255
signed char	1byte	-127 to 127
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647

- How to Check the Size on Your Machine: Use the **sizeof( )** function.

Fundamental Syntax Rules

short int	2bytes	-32768 to 32767
unsigned short int	Range	0 to 65,535
signed short int	Range	-32768 to 32767
long int	4bytes	-2,147,483,647 to 2,147,483,647
signed long int	4bytes	same as long int
unsigned long int	4bytes	0 to 4,294,967,295
float	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	8bytes	+/- 1.7e +/- 308 (~15 digits)
wchar_t	2 or 4 bytes	1 wide character

Fundamental Syntax Rules

The example below, explain the correct size of various data types on your computer.

```
#include <iostream>
using namespace std;
int main()
{
cout << "Size of char : " << sizeof(char) << endl;
cout << "Size of int : " << sizeof(int) << endl;
cout << "Size of short int : " << sizeof(short int) << endl;
cout << "Size of long int : " << sizeof(long int) << endl;
cout << "Size of float : " << sizeof(float) << endl;
cout << "Size of double : " << sizeof(double) << endl;
cout << "Size of wchar_t : " << sizeof(wchar_t) << endl;
return 0;
}
```

This example uses **endl**, which inserts a new-line character after every line and << operator is being used to pass multiple values out to the screen. Also, using **sizeof( )** is used as function to get size of various data types.

Fundamental Syntax Rules

- When the code is compiled and executed, it produces the following result which can vary from machine to machine:

```
Size of char : 1
Size of int : 4
Size of short int : 2
Size of long int : 4
Size of float : 4
Size of double : 8
Size of wchar_t : 4
```

Variable Definition in C++

- A **variable definition** specifies a **data type**, and **contains a list** of **one or more variables** of that type as follows:

  Syntax: **type** **variable_name**;

- **type** must be a valid C++ data type containing char, int, float, bool, etc.
- **variable_name** consists of **one** or **more identifier names** separated by **commas** (**,**). Some valid declarations are shown below:

```
int       x, y, z;          // Integer variables
float     f , salary;       // Floating-point variable
char      c, ch;            / Character variable
double    d;                // double Floating-point variable
```

Variable Definition in C++

- **Variables** can be **initialized** (**assigned an initial value** to a variable when it is created) in their declaration. The initializer consists of an **equal sign followed by a constant** expression as follows:
- This helps the program know the starting value of the variable before using it in calculations or logical operations

Syntax: **type variable_name = value**;

Example:

```
float    P = 10.5;          // Declaration + Initialization
int      x = 6, y = 9;      // Declaration + Initialization
```

Variable Scope

- There are **three main scopes** in C++:
- **Local Variables:**
  - Declared **inside** a function or block.
  - Can only be used within that specific function.
- **Global Variables:**
  - Declared **outside** all functions (usually at the top).
  - Can be accessed by **any** function in the program.
- **Formal Parameters:**
  - Variables defined in **function parameters**.

Variable Scope

- **Local variables** are variables that are declared **inside a function or inside a block { }**.
- A **local variable** is like a "**secret**" variable that only one function knows about. Once the function finishes, the **variable disappears**. For example :

```
#include <iostream>
using namespace std;
int main ()
{
   // Local variable declaration:
      int x, y;
      int z;
  // actual initialization
      x = 15;
      y = 32;
      z = x + y;
      cout << z;
      return 0;
      }
```

Variable Scope

- **Global variable** are variables that are declared **outside all functions**, usually at the **top of the program**.
- They can be used by **any function** in the program.
- They stay alive for the **entire lifetime** of the program — they **do not disappear** after a function end
- A **global variable** is like a "**public**" variable that **every function can see and use**.
- If **one function changes it**, the **change will be visible to all other functions too**.

Variable Scope

- Example of global variable

```
#include <iostream>
using namespace std;
// Global variable declaration:
    int m;
int main ()
{
  // Local variable declaration:
    int x, y;
  // actual initialization
    x = 15;
    y = 32;
    m = x + y;
   cout << m;
   return 0;
 }
```

Variable Scope

- **Note:** If a **local variable** has the **same name** as a **global variable**, the **local** variable takes **priority inside its function**.

```
#include <iostream>
using namespace std;
// Global variable declaration:
int m = 32;
int main ( )
{
// Local variable declaration:
   int m = 15;
   cout << m;
   return 0;
}
```

Initializing Local and Global Variables

- **Local Variables**
  - When you create a **local variable** (inside a function), **the system does not give it an initial value**. **You must give it a value yourself.**
- **Global Variables**
  - **Global variables** (declared outside all functions) **are automatically initialized by the system.** The system gives them default values:

Type	Default Value
int	0
float	0.0
char	'\0'
pointers	NULL

CONSTANTS/LITERALS

- **What is a Constant?**
- **Fixed values** that the program **cannot change during execution**, also known as **Literals**.
- Constants can be of any of the **basic data types** and can be divided into Integer, Floating-Point, Characters, Strings, and Boolean Values.
- Once declared, a constant **cannot be modified**; trying to change its value will cause a compile-time error.
- **Two Ways to Define Constants:**
  - **Using # define Preprocessor**
  - **Using const Keyword**

CONSTANTS/LITERALS

1. The # define Preprocessor

Syntax: **# define identifier value**

Example :

```
#include <iostream>
using namespace std;
# define  L   9
# define  W   7
# define NEWLINE '\n'
int main()
{
    int A;
    A = L * W;
    cout << A;
    cout << NEWLINE;
    return 0;
}
```

- When the code is compiled and executed, the result is 63

CONSTANTS/LITERALS

2. Using const Keyword:

- Use **const** prefix to declare constants with a specific type.

Syntax: **const type variable = value**;

Example:

```
#include <iostream>
using namespace std;
int main()
{
const  int    L = 9;
const  int  W = 7;
const char NEWLINE = '\n';
int A;
A = L * W;
cout << A;
cout << NEWLINE;
return 0;
}
```

- When the code is compiled and executed, the result is 63

Input and Output Instructions

- **Output (cout):** Used to **display** values, text, or results on the screen.

  Syntax: cout << exp1 << exp2 << exp3 << ... << expN;

  Example: cout << "The average is: " << avg;

- **Input (cin):** Used to **read** input values from the user.

  Syntax: cin >> var1 >> var2 >> var3 >> ... >> varN;

  Example: int num1, num2;

  cin >> num1 >> num2;

- Input and output in C++ use the **stream operators**:
  - **<<** (Insertion operator for output)
  - **>>** (Extraction operator for input)

Example of Complete Program

write a C++ program to find the sum of two numbers

```
#include <iostream>
using namespace std;
int main()
{
    int num1, num2, sum;
    cout << "Enter two numbers: ";
    cin >> num1 >> num2;
    sum = num1 + num2;
    cout << "The sum is: " << sum << endl;
    return 0;
}
```

(Escape Characters) in C++ :

- **\n** (New line (moves to next line))
- **\t** (Horizontal tab (adds space like Tab key))
- **\0** (Null character (end of string))

ASCII Table (American Standard Code for Information Interchange)

- The **ASCII table** assigns a **numeric code (0–127)** to each **character** (letters, digits, symbols, etc.) so that computers can understand and process text. Each **character** has a **unique integer value**.
  - **A–Z (Uppercase letters)** → ASCII codes **65 to 90**
  - **a–z (Lowercase letters)** → ASCII codes **97 to 122**
  - **0–9 (Digits)** → ASCII codes **48 to 57**
  - **Symbols** → ASCII codes **32 to 47**, **58 to 64**, etc.

Character	ASCII Code	Character	ASCII Code
'a'	97	'A'	65
'b'	98	'B'	66
'z'	122	'Z'	90
'0'	48	'1'	49
'9'	57	'&'	38
'*'	42	'+'	43

References

- Gaddis, T. (2014). *Starting out with C++: From control structures through objects* (8th ed.). Pearson.
- Soulié, J. (2007, April 24). C++ language tutorial. cplusplus.com.
- Tutorials Point. (n.d.). *Learn C++ programming language*. Tutorials Point.