Introduction to C++

Learning Objectives

After the completion of this chapter, the student will be able to

- Understand the basic building blocks of C++ programming language
- Able to construct simple C++ programs
- Execute and debug C++programs

9.1 Introduction

C++ is one of the most popular programming language which supports both procedural and Object Oriented Programming paradigms. Thus, C++ is called as a **hybrid language.** C++ is a superset (extension) of its predecessor C language. Bjarne Stroustrup named his new language as "C with Classes". The name C++ was coined by Rick Mascitti where ++ is the C language increment operator.

Bjarne Stroustrup
Inventor of C++ Programming Language

Bjarne is a Danish Computer Scientist born on 30th December 1950. He has a Master degree in Mathematics and Computer Science in 1975 from Aarhus University, Denmark and Ph.D in Computer Science in 1979 from the University of Cambridge, England.

History of C++

C++ was developed by Bjarne Stroustrup at AT & T Bell Laboratory during 1979. C++ is originally derived from C language and influenced by many languages like Simula, BCPL, Ada, ML, CLU and ALGOL 68. Till 1983, it was referred "New C" and "C with Classes". In 1983, the name was changed as C++ by Rick Mascitti.

Benefits of learning C++

- C++ is a highly portable language and is often the language of choice for multi-device, multi-platform app development.
- C++ is an object-oriented programming language and includes **classes**, **inheritance**, **polymorphism**, **data abstraction** and **encapsulation**.
- C++ has a rich function library.
- C++ allows exception handling, inheritance and function overloading which are not possible in C.
- C++ is a powerful, efficient and fast language. It finds a wide range of applications – from GUI applications to 3D graphics for games to real-time mathematical simulations.

9.2 Character set

Character set is a set of characters which are used to write a C++ program. A character represents any alphabet, number or any other symbol (special characters) mostly available in the keyboard. C++ accepts the following characters.

Alphabets	A …. Z, a ….. z
Numeric	0 …. 9
Special Characters	+ - * / ~ ! @ # $ % ^& [ ] ( ) { } = >< _ \ | ? . , : ‘ “ ;
White space	Blank space, Horizontal tab (→), Carriage return (↲), Newline, Form feed
Other characters	C++ can process any of the 256 ASCII characters as data.

9.3 Lexical Units (Tokens):

C++ program statements are constructed by many different small elements such as commands, variables, constants and many more symbols called as operators and punctuators. These individual elements are collectively called as **Lexical units** or **Lexical elements** or **Tokens**. C++ has the following tokens:

- Keywords
- Literals
- Punctuators
- Identifiers
- Operators

TOKEN:

The smallest individual unit in a program is known as a Token or a Lexical unit

9.3.1 Keywords

Keywords are the reserved words which convey specific meaning to the C++ compiler. They are the essential elements to construct C++ programs. Most of the keywords are common to C, C++ and Java.

C++ is a case sensitive programming language so, all the keywords must be in lowercase.

Table 9.1 C++ Keywords

asm	auto	break	case	catch
char	class	const	continue	default
delete	do	double	else	enum
extern	float	for	friend	goto
if	inline	int	long	new
operator	private	protected	public	register
return	short	signed	sizeof	static
struct	switch	template	this	throw
try	typedef	union	unsigned	virtual
void	volatile	while		

- With revisions and additions, the recent list of keywords also includes:

using, namespace, bal, static_cast, const_cast, dynamic_cast, true, false

- Identifiers containing a double underscore are reserved for use by C++ implementations and standard libraries and should be avoided by users.

9.3.2 Identifiers

Identifiers are the user-defined names given to different parts of the C++ program viz. variables, functions, arrays, classes etc., These are the fundamental building blocks of a program. Every language has specific rules for naming the identifiers.

Rules for naming an identifier:

- The first character of an identifier must be an alphabet or an underscore (_).
- Only alphabets, digits and underscore are permitted. Other special characters are not allowed as part of an identifier.
- C++ is case sensitive as it treats upper and lower-case characters differently.
- Reserved words or keywords cannot be used as an identifier name.

As per ANSI standards, C++ places no limit on its length and therefore all the characters are significant.

Identifiers	Valid / Invalid	Reason for invalid
Num	Valid	
NUM	Valid	
_add	Valid	
total_sales	Valid	
tamilMark	Valid	
num-add	Invalid	Contains special character (-)
this	Invalid	This is one of the keyword. Keyword cannot be used as identifier names.
2myfile	Invalid	Name must start begins with an alphabet or an underscore

- You may use an underscore in variable names to separate different parts of the name (eg: ***total_sales*** is a valid identifier where as the variable called ***total sales*** is an invalid identifier).
- You may use capital style notation, such as ***tamilMark*** ie. capitalizing the first letter of the second word.

9.3.3 Literals (Constants)

Literals are data items whose values do not change during the execution of a program. Therefore Literals are called as Constants. C++ has several kinds of literals:

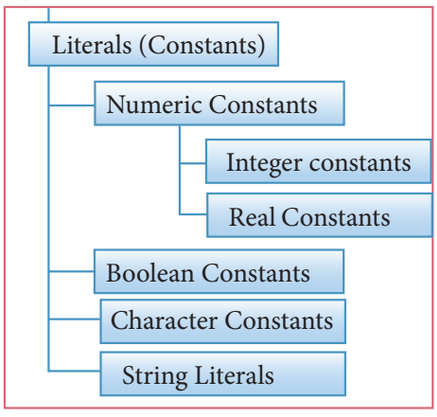


Figure 9.1 Types of Constants

Numeric Constants:

As the name indicates, the numeric constants are numeric values, which are used as constants. Numeric constants are further classified as:

1. Integer Constants (or) Fixed point constants.
2. Real constants (or) Floating point constants.

(1) Integer Constants (or) Fixed point constants

Integers are whole numbers without any fractions. An integer constant must have at least one digit without a decimal point. It may be signed or unsigned. Signed integers are considered as negative, commas and blank spaces are not allowed as part of it. In C++, there are three types of integer constants: (i) Decimal (ii) Octal (iii) Hexadecimal

(i) Decimal

Any sequence of one or more digits (0 …. 9)

Valid	Invalid
725	7,500 (Comma is not allowed)
-27	66 5(Blank space is not allowed)
4.56	9$ (Special Character not allowed)

If you assign **4.56** as an integer decimal constant, the compiler will accept only the integer portion of **4.56** ie. **4**. It will simply ignore **.56.**

Notes

If a Decimal constant declared with fractions, then the compiler will take only the integer part of the value and it will ignore its fractional part. This is called as "Implicit Conversion". It will be discussed later.

(ii) Octal

Any sequence of one or more octal values (0 …. 7) that begins with 0 is considered as an Octal constant.

Valid	Invalid
012	05,600(Commas is not allowed)
-027	04.56 (Decimal point is not allowed)**
+0231	0158 (8 is not a permissible digit in octal system)

Notes

** When you use a fractional number that begins with 0, C++ considers the number as an integer not an Octal.

(iii) Hexadecimal

Any sequence of one or more Hexadecimal values (0 …. 9, A …. F) that starts with 0x or 0Xis considered as an Hexadecimal constant.

Valid	Invalid
0x123	0x1,A5 (Commas is not allowed)
0X568	0x.14E (Decimal point is not allowed like this)

The suffix L or l and U or u added with any constant forces it to be represented as a long or unsigned constant respectively.

(2) Real Constants (or) Floating point constants

A real or floating point constant is a numeric constant having a fractional component. These constants may be written in fractional form or in exponent form.

Fractional form of a real constant is a signed or unsigned sequence of digits including a decimal point between the digits. It must have at least one digit before and after a decimal point. It may be prefixed with + or - sign. A real constant without any sign will be considered as positive.

Exponent form of real constants consists of two parts: **(1) Mantissa** and **(2) Exponent.** The mantissa must be either an integer or a real constant. The mantissa followed by a letter E or e and the exponent, should also be an integer.

For example, 58000000.00 may be written as 0.58×10^8 or 0.58E8.

Mantissa (Before E)	Exponent (After E)
0.58	8

Example:

5.864 E1 $10^1 \times 5.864$ ➔ 58.64

5864 E-2 $10^{-2} \times 5864$ ➔ 58.64

0.5864 E2 $10^2 \times 0.5864$ ➔ 58.64

Boolean Literals

Boolean literals are used to represent one of the Boolean values (True or false). Internally true has value 1 and false has value 0.

Character constant

A character constant in C++ is any valid single character enclosed within single quotes.

Valid character constants : 'A', '2', '$'

Invalid character constants : "A"

The value of a single character constant has an equivalent ASCII value. For example, the value of 'A' is 65.

Escape sequences (or) Non-graphic characters

C++ allows certain non-printable characters represented as character constants. Non-printable characters are also called as non-graphic characters. Non-printable characters are those characters that cannot be typed directly from a keyboard during the execution of a program in C++, for example: backspace, tabs etc. These non-printable characters can be represented by using escape sequences. An escape sequence is represented by a backslash followed by one or two characters.

Table 9.2 Escape Sequences

Escape sequence	Non-graphical character
\a	Audible or alert bell
\b	Backspace
\f	Form feed
\n	Newline or linefeed
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\\	Backslash
\'	Single quote
\"	Double quote
\?	Question Mark
\On	Octal number
\xHn	Hexadecimal number
\0	Null

Even though an escape sequence contains two characters, they should be enclosed within single quotes because, C++ consider escape sequences as character constants and allocates one byte in ASCII representation.

ASCII (American Standard Code for Information Interchange) was first developed and published in 1963 by the X3 committee, a part of the American Standards Association (ASA).

Evaluate Yourself

1. What is meant by literals? How many types of integer literals are available in C++?
2. What kind of constants are following?
   i) 26 ii) 015 iii) 0xF iv) 014.9
3. What is character constant in C++?
4. How are non graphic characters represented in C++?
5. Write the following real constants into exponent form:
   i) 32.179 ii) 8.124 iii) 0.00007
6. Write the following real constants in fractional form:
   i) 0.23E4 ii) 0.517E-3 iii) 0.5E-5
7. What is the significance of null (\0) character in a string?

String Literals

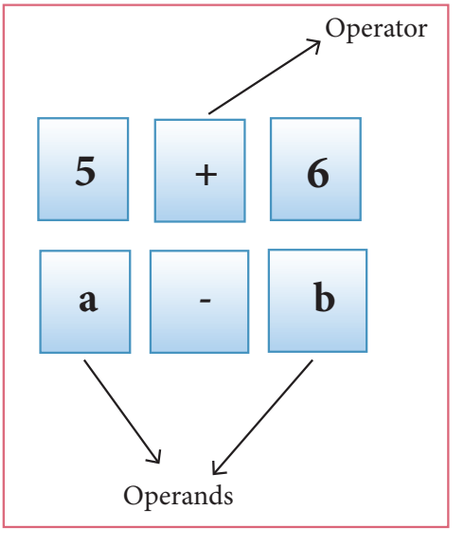
Sequence of characters enclosed within double quotes are called as String literals. By default, string literals are automatically added with a special character **'\0' (Null)** at the end. Therefore, the string "welcome" will actually be represented as "welcome\0" in memory and the size of this string is not 7 but 8 characters i.e., inclusive of the last character \0.

Valid string Literals : "A", "Welcome" "1234"

Invalid String Literals : 'Welcome', '1234'

9.3.4 Operators

The symbols which are used to do some mathematical or logical operations are called as **"Operators".** The data items or values that the oper ators act upon are called as **"Operands".**



In C++, The operators are classified on the basis of the number of operands.

(i) **Unary Operators** - Require only one operand

(ii) **Binary Operators** - Require two operands

(iii) **Ternary Operators** - Require three operands

C++ Binary Operators are classified as:

(1) Arithmetic Operators
(2) Relational Operators
(3) Logical Operators
(4) Assignment Operators
(5) Conditional Operator

(1) Arithmetic Operators

Arithmetic operators perform simple arithmetic operations like addition, subtraction, multiplication, division etc.,

Operator	Operation	Example
+	Addition	10 + 5 = 15
-	Subtraction	10 – 5 = 5
*	Multiplication	10 * 5 = 50
/	Division	10 / 5 = 2 (Quotient of the division)
%	Modulus (To find the remind-er of a division)	10 % 3 = 1(Remainder of the division)

- The above mentioned arithmetic operators are binary operators which requires minimum of two operands.

Increment and Decrement Operators (Unary Operators)

++ (Plus, Plus) Increment operator
-- (Minus, Minus) Decrement operator

An increment or decrement operator acts upon a single operand and returns a new value. Thus, these operators are unary operators. The increment operator adds 1 to its operand and the decrement operator subtracts 1 from its operand. For example,

- **x++ or ++ x** is the same as **x = x+1;**
  It adds 1 to the present value of x
- **x -- or -- x** is the same as to **x = x–1**;
  It subtracts 1 from the present value of x

The ++ or -- operators can be placed either as prefix (before) or as postfix (after) to a variable. With the prefix version, C++ performs the increment / decrement before using the operand.

(2) Relational Operators

Relational operators are used to determine the relationship between its operands. When the relational operators are applied on two operands, the result will be a Boolean value i.e 1 or 0 to represents True or False respectively. C++ provides six relational operators. They are,

Operator	Operation	Example
>	Greater than	a > b
<	Less than	a < b
>=	Greater than or equal to	a >= b
<=	Less than or equal to	a <= b
==	Equal to	a == b
!=	Not equal	a != b

- In the above examples, the operand *a* is compared with *b* and depending on the relation, the result will be either 1 or 0. i.e., 1 for true, 0 for false.
- All six relational operators are binary operators.

(3)Logical Operators

A logical operator is used to evaluate logical and relational expressions. The logical operators act upon the operands that are themselves called as logical expressions. C++ provides three logical operators.

Table 9.3 Logical Operators

Operator	Operation	Description
&&	AND	The logical AND combines two different relational expressions in to one. It returns 1 (True), if both expression are true, otherwise it returns 0 (false).

||	**OR**	The logical OR combines two different relational expressions in to one. It returns 1 (True), if either one of the expression is true. It returns 0 (false), if both the expressions are false.
!	**NOT**	NOT works on a single expression / operand. It simply negates or inverts the truth value. i.e., if an operand / expression is 1 (true) then this operator returns 0 (false) and vice versa

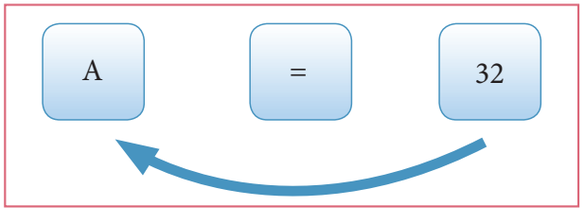
- AND, OR both are binary operators where as NOT is an unary operator.

Example: a = 5, b = 6, c = 7;

Expression	Result
(a<b) && (b<c)	1 (True)
(a>b) && (b<c)	0 (False)
(a<b) || (b>c)	1 (True)
!(a>b)	1 (True)

(4)Assignment Operator:

Assignment operator is used to assign a value to a variable which is on the left hand side of an assignment statement. = (equal to) is commonly used as the assignment operator in all computer programming languages. This operator copies the value at the right side of the operator to the left side variable. It is also a binary operator.



C++ uses different types of assignment operators. They are called as Shorthand assignment operators.

Operator	Name of Operator	Example
+=	Addition Assignment	a = 10; c = a += 5; (ie, a = a + 5) c = 15
-=	Subtraction Assignment	a = 10; c = a -= 5; (ie. a = a – 5) c = 5
*=	Multiplication Assignment	a = 10; c = a *= 5; (ie. a = a * 5) c = 50
/=	Division Assignment	a = 10; c = a /= 5; (ie. a = a / 5) c = 2
%=	Modulus Assignment	a = 10; c = a %= 5; (ie. a = a % 5) c = 0

Discuss the differences between = and == operators

(5) Conditional Operator:

In C++, there is only one conditional operator. **?:** is a conditional Operator which is also known as Ternary Operator. This operator is used as an alternate to if … else control statement. We will learn more about this operator in later chapters along with if …. else structure.

Other Operators:

The Comma operator	Comma (,) is an operator in C++ used to string together several expressions. The group of expression separated by comma is evaluated from left to right.
Sizeof	This is called as compile time operator. It returns the size of a variable in bytes.
Pointer	* Pointer to a variable & Address of

Component selection	. Direct component selector -> Indirect component selector
Class member operators	:: Scope access / resolution .* Dereference ->* Dereference pointer to class member

Precedence of Operators:

Operators are executed in the order of precedence. The operands and the operators are grouped in a specific logical way for evaluation. This logical grouping is called as an Association.

The order of precedence:

( ) [ ]	Operators within parenthesis are performed first	Higher
++, --	Postfix increment / decrement	
++, --	Prefix increment / decrement	
*, /, %	Multiplication, Division, Modulus	
+, -	Addition, Subtraction	
<, <=, >, >=	Less than, Less than or equal to, Greater than, Greater than or equal to	
==, !=	Equal to, Not equal to	
&&	Logical AND	
||	Logical OR	
?:	Conditional Operator	
=	Simple Assignment	
+=, -=, *=, /=	Shorthand operators	
,	Comma operator	Lower

9.3.5 Punctuators

Punctuators are symbols, which are used as delimiters, while constructing a C++ program. They are also called as **"Separators".** The following punctuators are used in C++; most of these symbols are very similar to C and Java.

Separator	Description	Example
Curly braces { }	Opening and closing curly braces indicate the start and end of a block of code. A block of code containing more than one executable statement. These statements together are called as "compound statement"	int main ( ) { int x=10, y=20, sum; sum = x + y; cout << sum; }
Parenthesis ( )	Opening and closing parenthesis indicate function calls and function parameters.	clrscr( ); int main ( )
Square brackets [ ]	It indicates single and multidimensional arrays.	int num[5]; char name[50];
Comma ,	It is used as a separator in an expression	int x=10, y=20, sum;

Semicolon ;	Every executable statement in C++ should terminate with a semicolon	int main ( ) { int x=10, y=20, sum; sum = x + y; cout << sum; }
Colon :	It is used to label a statement.	private:
Comments // /* */	Any statement that begins with // are considered as comments. Comments are simply ignored by compilers. i.e., compiler does not execute any statement that begins with a // **// Single line comment** **/* …….. */ Multiline comment**	**/* This is written by me to learn CPP */** int main ( ) { int x=10, y=20, sum; **// to sum x and y** sum = x + y; cout << sum; }

In C++, one or two operators may be used in different places with different meaning.

For example: Asterisk (*) is used for multiplication as well as for pointer to a variable.

Evaluate Yourself

1. What is the use of operators?
2. What are binary operators? Give examples of arithmetic binary operators.
3. What does the modulus operator % do?
4. What will be the result of 8.5 % 2?
5. Given that i = 8, j = 10, k = 8, What will be result of the following expressions?
   (i) i < k (ii) i < j (iii) i > = k (iv) i = = j (v) j ! = k
6. What will be the order of evaluation for the following expressions?
   (i) i + 3 >= j - 9 (ii) a +10 < p - 3 + 2 q
7. Write an expression involving a logical operator to test, if marks are 75 and grade is 'A'.

9.4 I/O Operators

9.4.1 Input operator

C++ provides the operator >> to get input. It extracts the value through the keyboard and assigns it to the variable on its right; hence, it is called as **"Stream extraction"** or **"get from"** operator.

It is a binary operator i.e., it requires two operands. The first operand is the pre-defined identifier cin (pronounced as C-In) that identifies keyboard as the input device. The second operand must be a variable.

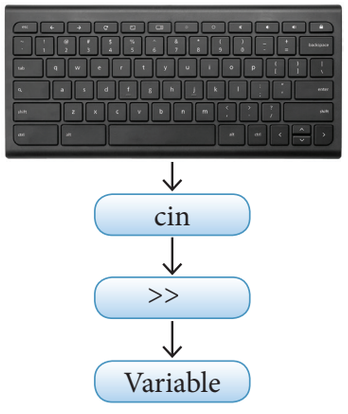


Figure 9.4 Working process of cin

To receive or extract more than one value at a time, >> operator should be used for each variable. This is called cascading of operator.

Example:

cin >> num ;	Pre-defined object cin extracts a value typed on keyboard and stores it in variable **num.**

cin >>x >> y;	This is used to extract two values. cin reads the first value and immediately assigns that to variable x; next, it reads the second value which is typed after a space and assigns that to y. Space is used as a separator for each input.

9.4.2 Output Operator

C++ provides << operator to perform output operation. The operator << is called the **"Stream insertion"** or **"put to"** operator. It is used to send the strings or values of the variables on its right to the object on its left. << is a binary operator.

The first operand is the pre-defined identifier cout (pronounced as C-Out) that identifies monitor as the standard output object. The second operand may be a constant, variable or an expression.

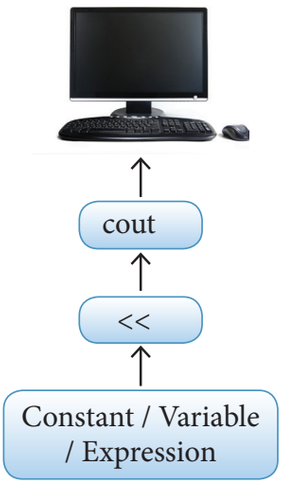


Figure 9.5 Working process of cout

To send more than one value at a time, << operator should be used for each constant/variable/expression. This is called **cascading of operator**.

Example:

cout << "Welcome";	Pre-defined object cout sends the given string "Welcome" to screen.
cout << "The sum = " << sum;	First, cout sends the string "The Sum = " to the screen and then sends the value of the variable sum; Usually, cout sends everything specified within double quotes or single quotes i.e., string or character constants, except non-graphic characters.
cout <<"\n The Area: " <<3.14*r*r;	First, cout sends everything specified within double quotes except \n to the screen, and then it evaluates the expression 3.14*r*r and sends the result to the screen. \n – is a non graphic character constant to feed a new line.
cout << a + b ;	cout sends the sum of a and b to the output console (monitor)

9.4.3. Cascading of I/O operators

The multiple use of input and output operators such as >> and << in a single statement is known as cascading of I/O operators.

Cascading cout:

```
int Num=20;
cout << "A=" << Num;
```

The Figure 9.6 is used to understand the working of Cascading cout statement

```
cout << "A=" << Num;
```

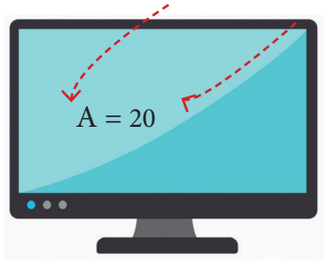


Figure 9.6 Cascading cout

Cascading cin - Example:

```
cout << "Enter two number: ";
cin >> a >> b;
```

The Figure 9.7 is used to understand the working of Cascading cin statement

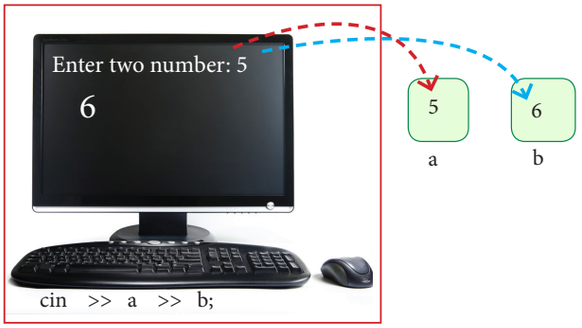


Figure 9.7 Cascading cin

9.5 Sample program – A first look at C++ program

Let us start our first C++ program that prints a string "Welcome to Programming in C++" on the screen.

Sample Prog 9_5.cpp

```
1  // C++ program to print a string
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      cout << "Welcome to Programming in C++";
7      return 0;
8  }
```

The above program produces, the following output:

Welcome to Programming in C++

This is very simple C++ program which includes the basic elements that every C++ program has. Let us have a look at these elements:

1	// C++ program to print a string
This is a comment statement. Any statement that begins with // are considered as comments. Compiler does not execute any comment as part of the program and it simply ignores. If we need to write multiple lines of comments, we can use /* ……. */.	
2	**# include <iostream>**
Usually all C++ programs begin with include statements starting with a # (hash / pound). The symbol # is a directive for the preprocessor. That means, these statements are processed before the compilation process begins. **#include <iostream> statement tells the compiler's preprocessor to include the header file "iostream" in the program.** The header file iostream should included in every C++ program to implement input / output functionalities. In simple words, **iostream header file contains the definition of its member objects cin and cout.** If you fail to include iostream in your program, an error message will occur on cin and cout; and we will not be able to get any input or send any output.	
3	**using namespace std;**
The line using namespace std; tells the compiler to use standard namespace. Namespace collects identifiers used for class, object and variables. Namespaces provide a method of preventing name conflicts in large projects. It is a new concept introduced by the ANSI C++ standards committee.	

4 int main ( )

C++ program is a collection of functions. Every C++ program must have a main function. The main( ) function is the starting point where all C++ programs begin their execution. Therefore, the executable statements should be inside the main( ) function.

```
5  {
6      cout << "Welcome to Programming in C++";
7      return 0;
8  }
```

The statements between the curly braces (Line number 5 to 8) are executable statements. This is actually called as a block of code. In line 6, cout simply sends the string constant "Welcome to Programming in C++" to the screen. As we discussed already, every executable statement must terminate with a semicolon. In line 7, return is a keyword which is used to return the value what you specified to a function. In this case, it will return 0 to main( ) function.

9.6 Execution of C++ program:

For creating and executing a C++ program, one must follow four important steps.

(1) Creating Source code

Creating includes typing and editing the valid C++ code as per the rules followed by the C++ Compiler.

(2) Saving source code with extension .cpp

After typing, the source code should be saved with the extension .cpp

(3) Compilation

This is an important step in constructing a program. In compilation, compiler links the library files with the source code and verifies each and every line of code. If any mistake or error is found, it will throw error message. If there are no errors, it translates the source code into machine readable object file with an extension .obj

(4) Execution

This is the final step of a C++ Program. In this stage, the object file becomes an executable file with extension .exe. Once the program becomes an executable file, the program has an independent existence. This means, you can run your application without the help of any compiler or IDE.

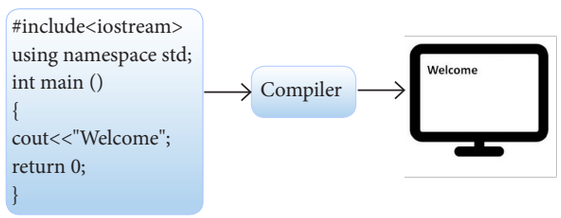


Figure 9.8 Execution

9.7 C++ Development Environment

There are lot of IDE programs available for C++. IDE makes it easy to create, compile and execute a C++ program. Most of the IDEs are open source applications (ie.) they are available free of cost.

9.7.1 Familiar C++ Compilers with IDE

Table 9.4 Open Source Compilers

Compiler	Availability
Dev C++	Open source
Geany	Open source
Code::blocks	Open source
Code Lite	Open source

Net Beans	Open source
Digital Mars	Open source
Sky IDE	Open source
Eclipse	Open source

9.7.2 Working with Dev C++

Among the dozens of IDEs, we take "Dev C++" compiler to create C++ programs. Programming techniques and illustrated programs of this book are based on "Dev C++" compiler.

Dev C++ is an open source, cross platform (alpha version available for Linux), full featured Integrated Development Environment (IDE) distributed with the GNU General Public License for programming in C and C++. It is written in Delphi. It can be downloaded from ***http://www.bloodshed.net/dev/devcpp.html***

1. After installation **Dev C++** icon is available on the desktop. Double click to open IDE. Dev C++ IDE appears as given below.

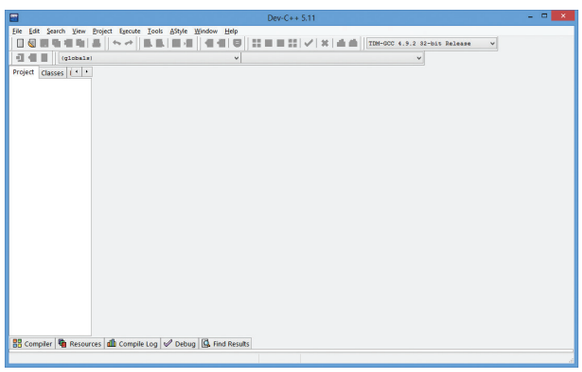


Figure 9.9 Dev C++ opening Window

2. To create a source file, Select **File** → **New** → **Source file** or **Press Ctrl + N**.
3. In the screen that appears, type your C++ program and save the file by clicking **File** → **Save** or Pressing **Ctrl + S.** It will add .cpp by default at the end of your source code file. No need to type .cpp along with your file name.

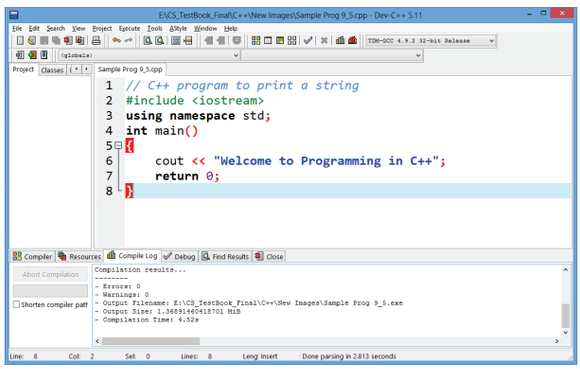


Figure 9.10 Dev C++ IDE with a program

4. **After save, Click Execute → Compile and Run or press F11 key.**

   If your program contains any error, it displays the errors under **compile log**. If your program is without any error, the display will appear as follows.

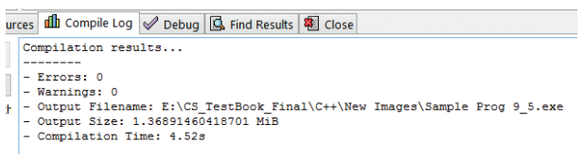


Figure 9.11 Dev C++ Compile Log

5. **After successful compilation, output will appear in output console, as follows**

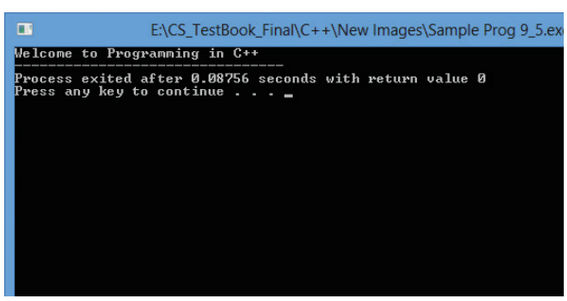


Figure 9.12 Dev C++ Output Window

9.8 Types of Errors

Some common types of errors are given below:

Type of Error	Description
Syntax Error	• Syntax is a set of grammatical rules to construct a program. Every programming language has unique rules for constructing the sourcecode. • Syntax errors occur when grammatical rules of C++ are violated. • Example: if you type as follows, C++ will throw an error. **cout << "Welcome to Programming in C++"** • As per grammatical rules of C++, every executable statement should terminate with a semicolon. But, this statement does not end with a semicolon.
Semantic Error	• A Program has not produced expected result even though the program is grammatically correct.It may be happened by wrong use of variable / operator / order of execution etc. This means, program is grammatically correct, but it contains some logical error. So, Semantic error is also called as **"Logic Error"**.
Run-time error	• A run time error occurs during the execution of a program. It occurs because of some illegal operation that takes place. • For example, if a program tries to open a file which does not exist, it results in a run-time error

Points to Remember:

- C++ was developed by Bjarne Stroustrup at AT & T Bell Labs during the year 1979.
- Character set is the set of characters which are allowed to write C++ programs.
- Individual elements are collectively called as Lexical units or Lexical elements or Tokens.
- Keywords are the reserved words that convey specific meaning to the C++ compiler.
- Identifiers are user-defined names given to different parts of the C++ program viz. variables, functions, arrays, classes etc.,
- Literals are data items whose values do not change during the execution of a program. Therefore, Literals are called as Constants.
- There are different kinds of literals used in C++ (Integer, Float, Character, String)
- The symbols which are used to do some mathematical, logical operations are called as "Operators".
- Punctuators are symbols, which are used as delimiters in constructing C++ programs. They are also called as "Separators".
- Extraction operator(>>) and Insertion operator (<<) are used to get input and send output in C++.