Unit 1 Overview of OOP Concepts

Structure:

1.1 Introduction

C++ is a general-purpose Object Oriented Programming (OOP) language. C++ includes features of object-oriented programming as well as conventional procedural programming. The major aim of developing object-oriented programming language is to remove some of the flaws encountered in programming a language of procedural approach. Object-Oriented Programming enables one to put data and functions in one container. This container is referred to as an object. An object enables you to model your application as closely as possible to the real world.

In this unit, we will discuss the basic features of object-oriented programming. In this unit, we also discuss the evolution of programming

methodologies, difference between C and C++, basic components of C++ programming. Finally, we discuss the process of compilation and execution of C++ programs.

Objectives:

After studying this unit, you should be able to:

- describe the role of object oriented programming approach over procedural languages
- discuss the basic features supported by OOP languages
- describe the data types of a C++ program
- define the terms ‘keyword’, ‘constant’, ‘identifiers’ and ‘operators’
- explain the process of compilation and execution

1.2 Evolution of Programming Methodologies

Computer performs a variety of tasks with the help of programming languages. The first version of a programming language was the machine-level language. During the1940s, the machine-level language, which used the binary codes of 0s and 1s to represent computer instructions, was developed. Machine-level language is also known as low-level programing language. Writing programs in machine language is symbolic, and had to be error prone (which is impractical). So, assembly languages were developed in the early 1950s. Assembly languages use mnemonics to write the instruction of a program.

First high level programming language known as FORTRAN was developed in the year 1957. During the 1960s, many high level languages were developed to support the needs of various disciplines like science and engineering, and business. The period between 1970 and 1980 was actually the golden era for the development of high-level programming languages. During 1970, a procedural programming language named PASCAL was developed. The C programming language was developed by Dennis Ritchie at the Bell Labs during the year 1973. In 1980, Bjarne Stroustrup, from the Bell labs, began the development of the C++ language.

The idea of object-oriented programming gained momentum in the 1970s and in the early 1980s. Bjorn Stroustrup integrated the object-oriented programming into the C language. C++ is a superset of C. Several features are similar in C and C++. At this time, many new features like object, class,

inheritance, virtual functions and function overloading were added. Therefore, there is a '++' symbol in C++. The ++ operator in the C++ means many new features were added around this time. The most notable features are - object, class, inheritance, virtual functions and function overloading.

Limitations of procedural languages:

Procedural languages focused on organizing program statements into procedures or functions. Larger programs were either broken into functions or modules which had defined purpose and interface to other functions.

Procedural approach for programming had several problems as the size of the software grew larger and larger. One of the main problems was that of the data being completely forgotten. The emphasis was on the action and the data was only used in the entire process. Data in the program was created by variables, and if more than one function had to access data, then global variables were used. The concept of global variables itself is a problem as it may be accidentally modified by an undesired function. This also leads to difficulty in debugging and modifying the program when several functions access a particular data.

The object Oriented approach overcomes this problem by modeling data and functions together, thereby allowing only certain functions to access the required data.

The procedural languages had limitations of extensibility as there was limited support for creating user-defined data types and defining how these datatypes are to be handled. For example, it would be difficult if the programmer had to define his own version of string, and define how this new datatypes will be manipulated. The Object Oriented Programming provides this flexibility through the concept of class.

Another limitation of the procedural languages is that the program model is not closer to real world objects. For example, if you want to develop a gaming application of car race, what data you would use and what functions you would require are difficult questions to answer in the procedural approach. The object oriented approach solves this further by conceptualizing the problem as group of objects which have their own specific data and functionality. In the car game, for example we would create several objects such as player, car, and traffic signal and so on.

Some of the languages that use object oriented programming approach are C++, Java, Csharp, Smalltalk etc. We will be learning C++ in this Self Learning Material (SLM) to understand the object oriented programming.

1.3 Difference between C and C++

C is a general purpose programming language. C has a flexible and compact structure. Programs written in C tend to consist of many small functions. C++ is derived from C Language. C++ is a procedural programing language with object-oriented extensions. That means, you can design and code programs using procedural approach.

The main difference between C and C++ is, that C++ is an object oriented programming language while C supports only procedure-oriented programming language. The main benefit of object oriented programming paradigm is that it focuses on writing programs that are more readable and maintainable. It also helps the reuse of code by packaging a group of similar objects, or using the concept of component programming model. The limitations of the procedural language are discussed in section 1.2.

The major difference between C and C++ are listed below:

- C has a top-down approach whereas c++ has a bottom-up approach.
- In C, a character constant is automatically elevated to an integer whereas in C++ this is not the case.
- In C++, identifiers are not allowed to contain two or more consecutive underscores in any position. C identifiers cannot start with two or more consecutive underscores, but may contain them in other positions.
- For an int main () in C++ we may not write a return statement but the return is mandatory in C if we are using an int main ().
- C input/output is based on library files, and the processes are carried out by including functions. The input and ouput (I/O) command is made through console commands ‘cin’ and ‘cout’ in C++.
- In C++, undeclared functions are not allowed. The function has to have a prototype defined before the main () before use in C++ although in C the functions can be declared at the point of use.
- C structures have a different behavior compared to C++ structures.
- After declaring structures and enumerators in C, we cannot declare the variable for the structure right after the end of the structure as in C++.

Self Assessment Questions

1. Who developed C++ language?
   a. Dennis Ritchie b. Bjorn Stroustrup
   c. James Gosling d. Charles Babbage
2. Assembly languages use _____ to write the instruction of a program.
3. In C, a character constant is automatically elevated to an _____ datatype, whereas in c++ this is not the case.

1.4 Introduction to OOP and its Basic Features

Objected Oriented Programming (OOP) is considered as important as the development of high level languages. As discussed earlier, one of the basic concepts in Object Oriented Programming is that it enables you to put data and functions in one unit, referred to as an object. The functions of a particular object can only access the data in the object providing high level of security for the data. The functions in the object are known as member functions or sometimes called 'methods'.

The basic features of OOP language are:

i) Objects
ii) Classes
iii) Inheritance
iv) Polymorphism
v) Encapsulation

i) Objects

An Object is the programming representation of the real world entity. According to Pressman, the Objects can be any one of the following:

a) External entities
b) Things
c) Occurrences or events
d) Roles
e) Organisational units
f) Places
g) Data Structures

Objects can have both *attributes (data)* and *behaviours* (functions or methods). Attributes describe the object with respect to certain parameters

and behavior or functions to describe the functionality of the object. Table 1.1 shows some examples of objects.

Table 1.1: Examples of Objects

Polygon Object		Bank Account	
Attributes	**Behaviour**	**Attributes**	**Behaviour**
Position	Move	Account number	Deduct Funds
Fill Color	Erase	Balance	Transfer funds
Border color	Change color		Deposit Funds
			Show balance

ii) Classes

Class is a group of objects that have the same properties and common behavior. A class is a category of things. An object is a specific item that belongs to a class called ‘instance of class’.

A class defines the characteristics of its objects and the methods that can be applied to its objects. Objects with the same data structure and behavior are grouped together as *class*. Classes are templates that provide definition to the objects of similar type. Objects are like variables created whenever necessary in the program. For example, student may be a class and John, James, Peter are objects of the class ‘student’. It is similar to the creation of variables of a default datatype such as integer- you can create any number of objects of a class.

iii) Inheritance

Inheritance is a technique through which one object acquires the properties of another. The technique also supports hierarchical classification. It is a powerful feature of OOP. It is a mechanism of deriving a new class from an old class, and of inheriting all the characteristics and behaviours of the old class.

The main advantage of Inheritance is "code reusability". The ability to reuse components of a program is an important feature for any programming language.

iv) Polymorphism

Polymorphism is the ability to present the same interface in different forms. Polymorphism means “many forms of a single object”. Operator overloading

is a kind of polymorphism. An operation may exhibit different behaviors in different instances. The behavior depends on the data types used in the operation.

The purpose of polymorphism is to implement a style of programming called 'message-passing'. When objects want to communicate with each other, they do so in terms of messages. Message passing is a method by which an object sends data to another object, or requests the other object to invoke a method.

v) Encapsulation

Encapsulation is used to bind code and data together. Wrapping of data and methods into a single unit is called 'encapsulation'. The result of encapsulation is data-hiding. It contains hiding information about an object. Ex: Private variable can be accessed only within the class. Encapsulation is also known as 'data hiding'.

OOP approach offers several advantages to programmers in these ways:

- The main advantage of an OOP is that it enables to model complex systems of real world into manageable software solutions.
- Object-oriented programming language can create extended and reusable parts of programs.
- OOP is easy to maintain and modify a code.
- OOP provides a good framework for code libraries.

Self Assessment Questions

4. A _____ defines the characteristics of its objects, and the methods that can be applied to its objects.
5. _____ is a technique in which one object acquires the properties of another object.
6. Encapsulation is also known as _____.
7. _____ is a method by which an object sends data to another object, or requests the other object to invoke method.

1.5 Basic Components of a C++ Program and Program Structure

We have already learnt that C++ is a superset of C. At the basic level, the programs look similar in both C and C++. Every statement in C++ ends with a semicolon (;). All the reserved words have to be written in the small case, and the C++ compiler is case sensitive.

1.5.1 Data types

A data type in a programming language is a set of data with values having predefined characteristics. Like C language, C++ also supports different data types. C++ language supports the following data types: char, int, float, double. The basic datatypes have various modifiers preceding them. The list of modifiers are: signed, unsigned, long and short. The datatypes supported in C++ are listed below:

Table 1.2: Basic Datatypes in c++

Data Type	Size (in bytes)	Values that can be taken
Signed int	2	-32768 to 32767
unsigned int	2	0 to 65535
Signed Char	1	-128 to 127
unsigned char	1	0 to 255
Float	4	3.4×10^{-38} to 3.4×10^{38} (Precision 7)
Double	8	1.7×10^{-308} to 1.7×10^{308} (Precision 15)
long double	10	3.4×10^{-4932} to 1.1×10^{4932} (Precision 19)
long int	4	-2147483648 to 2147483647
short int	2	-31768 to 32767

1.5.2 Variables

Variable is a container of data i.e. Variables are names used to refer to some memory location. Variables can be named according to the following rules:

- Variable name must begin with a letter.
- Variable name contains letters, numbers and underscore.
- Variable names are case sensitive.
- Reserve words of C++ cannot be used

Declaration and definition of variables:

Variables have to be declared before being used in the program. Variable declaration tells the compiler the name and datatype of the concerned variable. General syntax of declaration of variable is

Datatype variablename;

Example: int p;

The above declaration declares an integer variable named p. you can declare multiple variables of the same type in a single statement by separating them with commas. The Datatype of a variable optionally initializes the variable to a specific value. Values can be assigned to the variable during declarations, or can be initialized separately using the assignment operator equal (=).

Example: int P=4, Q=8;
Or
int P, Q;
P=4; Q=8

1.5.3 Token of C++: Identifiers, Keywords, Constants, Operators

A programmer uses C++ tokens to write a program. C++ supports the following types of tokens: identifiers, keywords, constants and operators.

1.5.3.1 Identifiers

Identifiers are used to define a name. They are mostly used to name variables (ex: array, structure), functions, class etc. Every language uses its own rules for naming the identifiers. In C++, identifiers can be named according to the following rules:

- Only alphabets, digits, characters and underscores are permitted.
- A name cannot start with a digit.
- An identifier cannot start with a symbol.
- Keywords cannot be used as an identifier.

Some examples for valid identifiers are:
Name, first_name, last_name

1.5.3.2 Keywords

Keywords are also known as ‘reserve words’. Many keywords are common to C as well C++. All Keywords have a specific meaning and purpose. Some of the C++ keywords are: auto, break, catch, class, delete, do, for, new, int, long, mutual, public, protected, return, size of, signed, void, while.

1.5.3.3 Constants

The constants in C++ can be a numeric, a character or string constants. The value of a constant does not change during the program. You should initialize a constant value when you create it.
Example: const int data=10;

Here 'const' allows you to create constants. The value of data is fixed during the entire program.

1.5.3.4 Operators

C++ has a rich set of operators. C++ supports all C operators. In C++, operators are used to perform a specific operation on a set of operands in an expression. Operators supported in C++ are listed below.

1. Arithmetic operators (+,-, *,/, %, ++,- -)
2. Relational operators (>,<, >=, <=, ==, !=)
3. Logical Operators (&&, ||, !)
4. Bit-wise operators (&, |. ^, ~)

There are a few other operators supported by C++ language: sizeof, the conditional operator and the cast operator. In later units, we will discuss in detail these additional operators.

1.5.4 Operators: expressions

An expression is a sequence of operators and operands, used for computation. Expression evaluation may produce a result (example: evaluation of 2+2 produces the result 4).

i. Arithmetic operators: C++ provides five simple arithmetic operators for creating arithmetic expressions. Table 1.3 below illustrates these arithmetic operators with example. If the value of X=6 and Y=3, results for the arithmetic operations are shown in Table 1.3.

Table 1.3: Arithmetic operators in C++

Operator	Description	Example
+	Used to add two operands	X+Y will give result 9
-	Used to subtract second operand from the first	X-Y will give result 3
*	Used to multiply both operands	X *Y will give result 18
/	Used to perform division operation	X/ Y will give result 2
%	Module operator, it gives reminder of integer Division	X%Y will give result 0
++	Increment operator, increase integer value by one	X++ will give value 7
--	Decrement operator, decreases integer value by one	X- - will give value 5

ii. Relational operators:

Relational operators are binary operators. Relational operators test data values against one other, and return the value as either 1 (true) or 0 (false).

The Table 1.4 illustrates relational operators with example. In example consider X=10 and Y=20.

Table 1.4: Relational operators in C++

Operator	Description	Example
>	Verifies whether the value of the left operand is greater than the value of the right operand; if yes, then the condition become true.	(X>Y) is not true
<	Verifies whether the value of the left operand is less than the value of the right operand; if yes, then the condition becomes true.	(X<Y)is true
>=	Verifies whether the value of the left operand is greater than or equal to the value of the right operand; if yes, then the condition result is true.	(X>=Y) is not true
<=	Verifies whether the value of the left operand is less than or equal to the value of the right operand; if yes, then the condition result is true.	(X<=Y) is not true
==	Verifies whether the values of two operands are equal or not; if equal, then condition result is true.	(X==Y) is not true
! =	Verifies whether the values of two operands are equal or not; if not equal, then condition result is true.	(X!=Y) is true

iii. Logical Operators:

Logical operators are used to combine multiple conditions, or to compare boolean expressions. The Table 1.5 illustrates logical operators with example. In the example given below, assume that variable X holds binary one (1) and variable Y holds binary zero (0).

Table 1.5: Logical operators in C++

Operator	Description	Example
&&	Logical AND operator. If both the operands are non-zero, Then condition becomes true.	(X && Y) Result is False
||	Logical OR operator. If any of the two operands are non-zero, then the condition becomes true.	(X || Y) Result is True
!	Logical NOT operator: used to reverse the logical states i.e. If the condition is true, then logical NOT operator will make it false. If the condition is false, then logical NOT operator will make it true.	(! X) Result is False

iv) Bitwise operators:

Bitwise operators modify variables considering the bit patterns that represent the values.

Bitwise AND (&) and bitwise OR (|) work similarly to their logical AND and logical OR counterparts. However, rather than evaluating a single boolean value, they are applied to each bit. Of all the bitwise operators, the bitwise NOT operator (~) is perhaps the easiest to understand. It simply flips each bit from a 0 to a 1, or vice versa. Bitwise exclusive-or (^) performs the exclusive-or operation on each pair of bits. Exclusive-or is commonly abbreviated as XOR. The exclusive-or operation takes two inputs and returns a 1 if either of the inputs is a 1, but not if both are. That is, if both inputs are 1 or both inputs are 0, it returns 0.

The Table 1.6 illustrates Bitwise logical operators with example. Let us assume that variable X holds binary value 5 (101) and variable Y holds binary 6 (110), then the result in the example column would be as given in table 1.6.

Table 1.6: Bitwise logical operators in c++

Operator	Description	Example
&	When evaluating bitwise AND, if all bits in a column are 1, the result for that column is 1.	(X & Y) Result is 100
|	When evaluating bitwise OR, if any bit in a column is 1, the result for that column is 1.	(X | Y) Result is 111
^	When evaluating bitwise XOR, if there is an odd number of 1 bits in a column, the result for that column is 1.	(X ^ Y) Result is 011
~	When evaluating bitwise NOT, it toggles each bit from 0 to a 1, or vice versa	(~ X) Result is 010

1.5.5 C++ I/O Methods: cin, cout

In every C++ program, it is compulsory to include “iostream.h” header file.

The header file “iostream.h” defines the cin, cout, objects, which correspond to the standard input stream and the standard output stream respectively. The cin statement is used for standard input or for input from keyboard, and the cout is used for standard output or for output- to- display screen.

The cin and cout are actually predefined objects in C++. Iostream.h file contains the declarations for the cin and cout statements.

cout (pronounced as C out) uses << or insertion operator to push data to the output stream.

cin (pronounced as c in statement) uses >> or extraction operator to feed the data to the input stream.

There are several such header files which have to be included, depending on the functions you are using. We will come across many such header files as we progress.

1.5.6 Structure of C++ programs

Every programming language consists of different sections. Some of the sections are optional and some are mandatory. The structure of C++ program supports different sections as shown in figure 1.1.

Documentation section
Header file section
Class declaration or definition section
Class function definition section
Main () function section

Figure 1.1: Basic Structure of C ++ program

1. First section, the documentation section, is optional, and is used to put comments for the program. Comment statements can be included in the program by prefixing the statement with "//" for single-line comments. Comments add clarity to the program. Multiple line comments can be added by enclosing the statements between /* and */.
2. Second section includes various header files required by C++ program.
3. Class declaration section: in this section declaration of class is done. User can declare class before or after main () section, but it is a good practice to declare it before main () function.
4. Class function definition: Class function definition can be done either inside or outside the class. When function is large, it is a good practice to define it outside the class.
5. Every C++ program starts with function main (). Program without main () function won't be executed; it is a compulsory section.

Like C, C++ too allows you to create your own functions in the program. However, the program execution always begins with the master function main (). Parentheses are used to group statements belonging to one function or program statement. Every opening parenthesis ({) should have a matching closing parenthesis (}).

We will discuss in detail class declaration and class function definition sections in unit 4.

Self Assessment Questions

8. All the reserved words have to be written in small case and the C++ compiler is case sensitive. (True/False)
9. Keywords are also known as _____.
10. _____ are used to combine multiple conditions or to compare boolean expressions.

11. The header file _____ defines the cin, cout, and objects.
12. _____ Statement is used for standard output or for output-to- display screen.
13. Documentation section is optional, and is used to put _____ for the program.
14. _____ are used to group statements belonging to one function or program statement.

1.6 Compiling and Executing C++ Program

There are three stages in executing a C++ program. They are: Compiling, Linking and Running the program.

Compiling:

Compiling is a process of converting source code to machine code. Each .cpp file is a program file which is also known as source file, and when it is compiled, you get an object file with extension .obj.

The c++ programs have to be typed in computer. Your program file is saved with an extension .cpp. This is known as 'source code'. The C++ compiler takes source code as its input and if there are no errors, it produces a machine code of your program, which is saved in a disk file with the same name, but with extension .obj. Compilation process translates program file in to object file, which is a machine-readable code.

C++ programs can be implemented on different Operating systems (ex: Windows, UNIX, DOS, etc.). In market, various types of compilers are available. Some of them are: GNU C++, Borland C++, Zortech C++, NDP C++, etc.

Linking:

Linking is an essential process when executing the program. The role of a linker is to link the files and functions required and used by programs. The object codes of these programs and the library files are linked to create a single executable file. For example, if the programmer uses clrscr (), then the object code of the function should be brought from "conio.h" header file. When the linking process is completed, the program code generates executable file.

Running the program:

Third and the last stage is running the executable file, where the statements in the program will be executed one by one.

Fig. 1.2 shows the entire process. When you execute the program, the compiler displays the output of the program and comes back to the program editor. To view the output, wait for the user to press any key to return to the editor and type getch() as the last statement in the program. Getch() is an inbuilt predefined library function which inputs a character from the user through standard input. However, you should include another header file named conio.h to use this function. Conio.h contains the necessary declarations for using this function. The include statement will be similar to iostream.h.

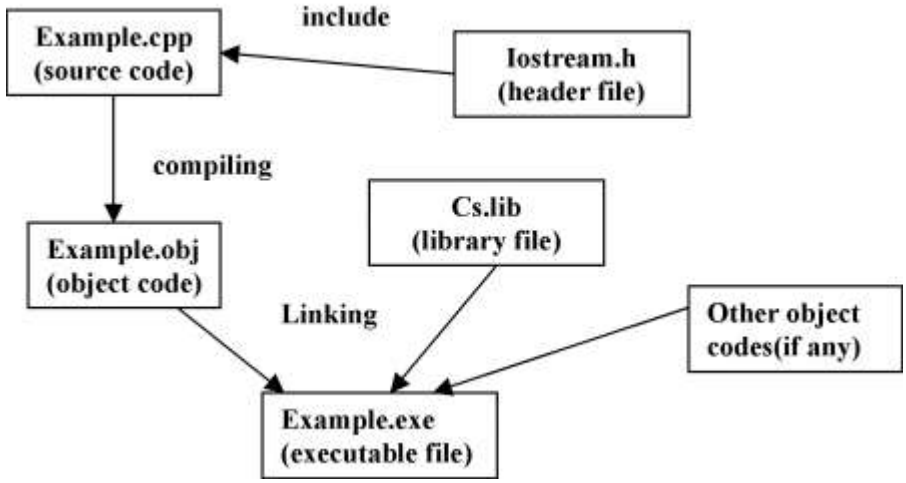

Figure 1.2: Execution process of C++ program

During compilation, if there are any errors, they will be listed by the compiler. The errors may be any one of the following:

1. Syntax error

   Syntax error is an error in the structure or spelling of a statement. It occurs when the program code breaks the basic syntax rules of C++ statement, as, for instance, in the wrong use of reserved words, improper variable names, variables without declaration etc. Further examples of syntax errors are- missing semi colon or parenthesis, type integer for int datatype etc. Often, syntax error message indicates the

approximate location of error; and the statement number is displayed to help the programmer. You can see the statement, make correction/s to the program file, save and recompile it.

2. Logical error
   This error occurs due to some flaw in the logic. Logical error occurs when you use a statement that is syntactically correct, but doesn't do what you need. The compiler doesn't identify when any logical error happens in the program. When compared to syntax error, finding a logical error is time-consuming for the programmer. However, it can be traced using the debug tool in the editor.
3. Linker error
   Linker error occurs when the files go missing during linking. Linker error implies that the linker is unable to build an executable program from the object code you provided. The following can be an example for a linker error-supposing you have used a function print, if the linker cannot find the function in any libraries that you have declared, then it is a linker error.
4. Runtime error
   Runtime error occurs during the execution of a program. If the program performs invalid operation during execution of the program, then runtime errors are detected (ex: division by zero, accessing a null pointer etc.)

1.6.1 Simple C++ program

In this section, you will learn how to write a program in the C++ programming language. To write and execute C++ programs, you need to have any text editor and C++ compiler. When a programmer writes a program for the first time, he or she must make sure that the flow of the program is followed by proper documentation. Example program is given below, where the description would help you to understand compilation and execution of programs. Only thing you need to do is just follow the steps in the given order.

```
//First simple C++ program
 #include <iostream.h>
 void main()
{
cout<< " Welcome to the programming language C++";
}
```

In the program shown above, #include<iostream.h> helps to stream programming features.

The statement cout displays the output enclosed between quotations. After finishing the writing of program, you have to compile and execute the program in order to get output. If you are compiling using UNIX environment, the program can be written in any text editor, and then you should save the program with extension .cpp (ex: first.cpp). Once you have saved the program, the next step is compilation. Compiler converts C++ program (first.cpp) to object code (first.obj). The General syntax of compiling the program in UNIX environment is:

g++ first.cpp –o first

where first.cpp is the name of source file.

To execute the program, type ‘./first.’ Where first is the name of the program. It will prompt the screen to command prompt and display the output.

Let’s go through the general steps followed to compile and execute the program:

Step 1: Download and install the C++ compilers.

Step 2: Type your C++ program and save your program (first.cpp)

Step 3: Next you need to compile your program.

Command to compile the program

g++ -c first.cpp

This will create an object file. To create an executable file type in:

g++ -o first.cpp

Step 4: To run the program, use the command below:

./first

Self Assessment Questions

15. _____ is a process of converting source code to machine code.
16. _____ is an error in the structure or spelling of statement.
17. Runtime error occurs during the _____ of program.
18. After finishing writing the program, in order to get output you have to _____ and then execute the program.
19. Each .cpp file that you compile produces an _____ file.

1.7 Summary

Let us recapitulate the important points discussed in this unit:

- Computer performs a variety of tasks with the help of programming languages. In 1980, Bjarne Stroustrup, from Bell labs, began the development of the C++ language.
- C++ is a superset of C. Several features are similar in C and C++. Object-oriented Programming (OOP) is considered as important as the development of high level languages. The basic features of OOP language are: Objects, Classes, Inheritance. Polymorphism, Encapsulation.
- Object-oriented Programming enables storing of data and functions together, which in turn enables hiding of data from unnecessary exposure.
- A class defines the characteristics of its objects and the methods that can be applied to its objects. The main advantage of Inheritance is "code reusability".
- C++ language supports following data types: char, int, float, double. A programmer uses C++ tokens to write a program.
- C++ supports the following types of tokens: identifiers, keywords, constants and operators. An expression is a sequence of operators and operands, used for computation.
- The header file “iostream.h” defines the cin, cout, objects, which correspond to the standard input stream and the standard output stream respectively.
- To write and execute C++ programs, you need to have a text editor and C++ compiler. In market various types of compilers are available: GNU C++, Borland C++, Zortech C++, NDP C++, etc.

1.8 Terminal Questions

1. Explain the following features of Object-oriented Programming.
   a. Encapsulation b. Inheritance c. Polymorphism
2. What do you understand by constant, keyword and identifier?
3. Describe the various datatypes available in C++?

4. List and explain any two types of operators available in C++ programming language.
5. Describe with example the basic structure of C++ program.
6. Write a program that accepts two numbers from the user and swaps the two numbers without using a temporary variable.
7. Write a program that accepts two numbers a and b, divides a by b, and displays the quotient and the remainder.

1.9 Answers

Self Assessment Questions

1. Bjarne Stroustrup
2. mnemonics
3. integer
4. class
5. Inheritance
6. data hiding
7. Message passing
8. False
9. reserve words
10. Logical operators
11. iostream.h
12. cout
13. comments
14. Parenthesis
15. Compiling
16. Syntax error
17. execution
18. compile
19. .obj

Terminal Questions

1. a. Encapsulation: Encapsulation used to bind code and data together. Wrapping of data and methods into a single unit is called ‘encapsulation’.
   b. Inheritance: Inheritance is a technique by which one object acquires the properties of another object. The technique also supports a hierarchical classification.

c. Polymorphism is the ability to present the same interface in different forms. Polymorphism means "many forms of a single object. “Operator Overloading” is a kind of polymorphism.

For more details, refer to section 1.4

2. Keywords are also known as reserve words. Many of C++ keywords are common with C. Keywords have a specific meaning and purpose. Some of the C++ keywords are: auto, break, catch, class, delete, do, for, new, int, long, mutual, public, protected, return, size of, signed, void, while.

   The constants in C++ can be numeric, character or string constants. The value of a constant does not change during the program.

   Identifiers are used to define a name. These are mostly used to name variables (ex: array, structure), function, class etc. Every language uses its own rules for naming the identifiers. For more details, refer to section 1.5.3.

3. A data type in a programming language is a set of data with values having predefined characteristics. Like C language, C++ also supports different data types. C++ language supports the following data types: char, int, float, double. The basic datatypes have various modifiers preceding them. The list of modifiers are: signed, unsigned, long and short. For more details, refer to section 1.5.1.
4. Arithmetic operators: C++ provides five simple arithmetic operators for creating arithmetic expression.
   Relational operators are binary operators. Relational operators test data values against one another, and return the value either 1 (true) or 0 (false).
   For more details, refer to section 1.5.4.
5. Every programming language consists of different sections. Some of the sections are optional and some are mandatory. Structure of C++ program contains the following sections: documentation section, header file section, class declaration, class function definition section and main function section. For more details, refer to section 1.5.5

6. Program to swap two numbers without using a temporary variable is-

```
# include<iostream.h>
void main()
    {
```

```
int num1,num2;
    cout <<" enter two numbers";
     cin>> num1>>num2;
  Num1=num1+num2;
          Num2=num1-num2;
  Num1=num1-num2;
  cout<< "numbers after swapping are "<<num1<<num2;
  }
```

7. Program to divide a and b and display quotient and remainder

```
# include<iostream.h>
void main()
{  int a,b, q, rem;
    cout <<" enter two numbers";
    cin>> a>>b;
    q=a/b;
    r =a%b;
          cout<< "Quotient = "<<q<<endl;
          cout<<"remainder="<<r;
    }
```

References:

- *Object-Oriented Programming Using C++*, By B. Chandra, CRC Press.
- *Starting Out with C++: From Control Structures Through Objects,* Global Edition, by Tony Gaddis, Pearson Education.