Chapter 4 - Overview of C++ Language

Introduction to C++ language

- C++ is a statically typed, compiled, general-purpose, case-sensitive, free-form programming language that supports procedural, object-oriented, and generic programming.
- C++ is regarded as a **middle-level language**, as it comprises a combination of both high-level and low-level language features.
- C++ was developed by **Bjarne Stroustrup** starting in 1979 at Bell Labs in Murray Hill, New Jersey, as an enhancement to the C language and originally named C with Classes but later it was renamed C++ in 1983.
- C++ is a superset of C, and that virtually any legal C program is a legal C++ program.

Structure of a C++ program

- Let us look at a simple code that would print the words Hello World.

```
#include <iostream.h>

// main() is where program execution begins.

void main()
{
        cout << "Hello World";
}
```

- Let us look various parts of the above program:
  - ✓ The C++ language defines several headers, which contain information that is either necessary or useful to your program. For this program, the header **<iostream.h>** is needed.
  - ✓ The next line // main() is where program execution begins. is a single-line comment available in C++. Single-line comments begin with // and stop at the end of the line.
  - ✓ The line int main() is the main function where program execution begins.
  - ✓ The next line cout << "This is my first C++ program."; causes the message "This is my first C++ program" to be displayed on the screen.

Concepts of compiling and linking

Let's look at how to save the file, compile and run the program. Please follow the steps given below:

- Open a text editor and add the code as above.
- Save the file as: hello.cpp
- Open a command prompt and go to the directory where you saved the file.
- Type 'g++ hello.cpp ' and press enter to compile your code. If there are no errors in your code the command prompt will take you to the next line and would generate a.out executable file.
- Now, type ' a.out' to run your program.
- You will be able to see ' Hello World ' printed on the window.

```
$ g++ hello.cpp
$ ./a.out
Hello World
```

Make sure that g++ is in your path and that you are running it in the directory containing file hello.cpp.

IDE and its features

Borland's Turbo C, first introduced in 1987, applied the same integrated development model used by the Silicon Valley software company to the C programming language. Enhanced with an assembler and debugger in 1989, Version 2.01 was the last release. In 1990 Borland introduced Turbo C++, ending the run of this memorable language.

Here are some of its best features:

- **Integrated Development Environment:** In the early days of PC development, a programmer would run one program to edit code, another to compile the program then the new program was run to test for errors. This process was repeated many, many times. The integrated development environment (IDE) that Borland first introduced with Turbo Pascal greatly simplified this by wrapping the entire development process into one program.
- **Optimized C Compiler:** Turbo C offered a number of optimization choices that enhanced size and speed at a time when memory and processor cycles were still limited resources.
- **Integrated Assembler Language:** Turbo C allows assembly code to be placed anywhere inside a C program.
- **Hardware Level Debugging:** The Turbo Debugger lets developers view computer memory and registers in real time as the program steps through the code. Breakpoints and watches can be set so the program runs and stops at predefined points or when memory locations or registers reach certain values.
- **Multiple Memory Models:** The early C languages solved this with a number of different memory models: tiny, small, compact and large.
- **Native Program Development:** Although most development is now targeted toward Windows, there are applications where the code needs to get down close to the bare metal. Device drivers, hard disk utilities, interfaces to specialized hardware and diagnostic programs all need low-level access.

Basic terminology –
Character set, Tokens, Identifiers, Keywords, Literal, Symbolic constants

Programming language is a set of rules, symbols, and special words used to construct programs. There are certain elements that are common to all programming languages. Now, we will discuss these elements in brief :

C++ Character Set

Character set is a set of valid characters that a language can recognize.

Letters	A-Z, a-z
Digits	0-9
Special Characters	Space + - * / ^ \ () [] {} = != <> ‘ “ $, ; : % ! & ? _ # <= >= @
Formatting characters	backspace, horizontal tab, vertical tab, form feed, and carriage return

Tokens

A token is a group of characters that logically belong together. The programmer can write a program by using tokens. C++ uses the following types of tokens: Keywords, Identifiers, Literals, Punctuators, Operators.

1. Keywords

These are some reserved words in C++ which have predefined meaning to compiler called keywords. The following list shows the reserved words in C++. These reserved words may not be used as constant or variable or any other identifier names.

asm	**else**	**new**	this
auto	enum	operator	throw
bool	explicit	**private**	**true**
break	export	**protected**	try
case	extern	**public**	typedef
catch	**false**	register	typeid
char	**float**	reinterpret_cast	typename
class	**for**	**return**	union
const	friend	**short**	unsigned
const_cast	goto	signed	using
continue	**if**	sizeof	virtual
default	inline	static	**void**
delete	**int**	static_cast	volatile
do	**long**	struct	wchar_t
double	mutable	**switch**	while
dynamic_cast	namespace	template	

2. Identifiers

Symbolic names can be used in C++ for various data items used by a programmer in his program. A symbolic name is generally known as an identifier. The identifier is a sequence of characters taken from C++ character set. The **rule for the formation of an identifier** are:

- An identifier can consist of alphabets, digits and/or underscores.
- It must not start with a digit
- C++ is case sensitive that is upper case and lower case letters are considered different from each other.
- It should not be a reserved word.

A C++ identifier is a name used to identify a variable, function, class, module, or any other user-defined item. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores, and digits (0 to 9).

C++ does not allow punctuation characters such as @, $, and % within identifiers. C++ is a case-sensitive programming language. Thus, Manpower and manpower are two different identifiers in C++.

Here are some examples of acceptable identifiers −

```
mohd       zara    abc   move_name  a_123
myname50   _temp   j     a23b9      retVal
```

3. Literals

Literals (often referred to as constants) are data items that never change their value during the execution of the program. The following types of literals are available in C++.

- Integer-Constants
- Character-constants
- Floating-constants
- Strings-constants
- **Integer Constants**

  Integer constants are whole number without any fractional part. C++ allows three types of integer constants.

  ✓ **Decimal integer constants :** It consists of sequence of digits and should not begin with 0 (zero). For example 124, - 179, +108.

  ✓ **Octal integer constants:** It consists of sequence of digits starting with 0 (zero). For example. 014, 012.

  ✓ **Hexadecimal integer constant:** It consists of sequence of digits preceded by ox or OX.
- **Character constants**

  A character constant in C++ must contain one or more characters and must be enclosed in single quotation marks. For example 'A', '9', etc. C++ allows nongraphic characters which cannot be typed directly from keyboard, e.g., backspace, tab, carriage return etc. These characters can be represented by using an escape sequence. An escape sequence represents a single character.
- **Floating constants**

  They are also called real constants. They are numbers having fractional parts. They may be written in fractional form or exponent form. A real constant in fractional form consists of signed or unsigned digits including a decimal point between digits. For example 3.0, -17.0, -0.627 etc.
- **String Literals**

  A sequence of character enclosed within double quotes is called a string literal. String literal is by default (automatically) added with a special character '\0' which denotes the end of the string. Therefore the size of the string is increased by one character. For example "COMPUTER" will re represented as "COMPUTER\0" in the memory and its size is 9 characters.

Fundamental data types

C++ offer the programmer a rich assortment of built-in as well as user defined data types. Following table lists down seven basic C++ data types:

Type	Keyword
Boolean	bool
Character	char
Integer	int
Floating point	float
Double floating point	double
Valueless	void

The following table shows the variable type, how much memory it takes to store the value in memory, and what is maximum and minimum value which can be stored in such type of variables.

Type	Typical Bit Width	Typical Range
char	1byte	-128 to 127 or 0 to 255
unsigned char	1byte	0 to 255
signed char	1byte	-128 to 127
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
unsigned short int	2bytes	0 to 65,535
signed short int	2bytes	-32768 to 32767
long int	8bytes	-2,147,483,648 to 2,147,483,647
signed long int	8bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
unsigned long int	8bytes	0 to 18,446,744,073,709,551,615
float	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	8bytes	+/- 1.7e +/- 308 (~15 digits)

Declaring variables, Initializing variables

A variable declaration provides assurance to the compiler that there is one variable existing with the given type and name so that compiler proceed for further compilation without needing complete detail about the variable. A variable declaration has its meaning at the time of compilation only, compiler needs actual variable definition at the time of linking of the program.

A variable declaration is useful when you are using multiple files and you define your variable in one of the files which will be available at the time of linking of the program. You will use extern keyword to declare a variable at any place. Though you can declare a variable multiple times in your C++ program, but it can be defined only once in a file, a function or a block of code.

Example

Try the following example where a variable has been declared at the top, but it has been defined inside the main function:

```
#include <iostream.h>

void main () {
   // Variable declaration
   int a, b, c;
   float f;
    // actual initialization
   a = 10;
   b = 20;
   c = a + b;
    cout << c << endl ;
   f = 70.0/3.0;
   cout << f << endl ;
}
```

Type modifiers

C++ allows the char, int, and double data types to have modifiers preceding them. A modifier is used to alter the meaning of the base type so that it more precisely fits the needs of various situations.

The data type modifiers are listed here:

- signed
- unsigned
- long
- short

The modifiers signed, unsigned, long, and short can be applied to integer base types. In addition, signed and unsigned can be applied to char, and long can be applied to double.

The modifiers signed and unsigned can also be used as prefix to long or short modifiers. For example, unsigned long int.

C++ allows a shorthand notation for declaring unsigned, short, or long integers. You can simply use the word unsigned, short, or long, without the int. The int is implied. For example, the following two statements both declare unsigned integer variables.

```
unsigned x;
unsigned int y;
```

To understand the difference between the way that signed and unsigned integer modifiers are interpreted by C++, you should run the following short program:

```
#include <iostream.h>

/* This program shows the difference between
 * signed and unsigned integers.
*/
void main() {
   short int i;           // a signed short integer
   short unsigned int j;  // an unsigned short integer

   j = 50000;

   i = j;
   cout << i << " " << j;
}
```

When this program is run, following is the output:

```
-15536 50000
```

The above result is because the bit pattern that represents 50,000 as a short unsigned integer is interpreted as -15,536 by a short.