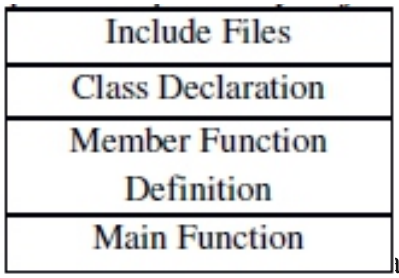
Chapter 6

Beginning with C++

6.1 Structure of C++ Program

A C++ program contains four sections as shown in figure 6.1. These sections may be placed in different source files and then compiled independently or jointly.



Fig[illegible]um

6.2 A Simple C++ Program

The following C++ program prints "Hello World" on the output screen.

Program 6.1 A Simple C++ Program

```
#include<iostream>          //include header file
using namespace std;
int  main()
{
        cout<<"Hello World";              //print "Hello World"
        return 0;
}
```

The output of the program 6.1 would be:

Hello World

Program features

- Like C, the C++ program is a collection of functions.
- Every C++ program must have a main function.
- Like C, the C++ statements terminate with semicolon(;).

Comments

- //(double slash) is used to comment a single line.
  For example-
  // This is my first C++ program.
- /*, */ used to comment multiple lines

For example-
/* This is my
first C++ program.*/

The iostream file

The statements preceded with # symbol are called pre-processor directive statements. They are placed at the beginning of the C++ program. The pre-processor processes these type of statements before the program is handed to the compiler. The #include<iostream> statement adds the contents of the iostream file to the program. It contains the declaration of the identifier cout and the insertion operator (<<).

Namespace

It defines scope of the identifiers used in the program. To use namespace scope we write using namespace std; std is the namespace where C++ standard class libraries are defined.

6.3 Compiling and Linking

The process of compiling and linking depends on the operating system.

- Linux OS

  The command g++ is used to compiling and linking a C++ program.

  For example-

  g++ abc.cpp

  g++ command compile the program written in abc.cpp file. The compiler produce an object file abc.o and then automatically link with the library functions to produce an executable file. The default executable file name is a.out.

- MS DOS

  Turbo C++ and Borland C++ compilers provide integrated development environment under MS DOS. They provide a built-in editor with File, Edit, Compile and Run options.

  -File option: To create and save the source file.
  -Edit option: To edit the source file
  -Compile option: To compile the program.
  -Run option: To compile, link and run the program in one step.

6.4 Tokens

The smallest individual unit in a program is called token. C++ has the following types of tokens:

- Keywords

- Identifiers
- Constants
- Operators
- Strings

6.5 Keywords

These are the reserved words whose meaning can't be changed by the programmer. These words can't be used as a name of the variable, constants or other user-defined program elements. The complete list of C++ keywords are shown in table 6.1. Many of them are common to both C and C++.

Table 6.1 C++ Keywords

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

6.6

Identifiers

The names of variables, functions, arrays, classes etc. that are created by the programmer are called identifiers. Each language has its own rules for naming these identifiers. The following rules are common to both C and C++:

- Only alphabetic characters, digits and underscore are allowed.
- The name can't begin with a digit.
- Uppercase and lowercase letters are distinct.
- Keywords can't be used as a variable name.

Constants

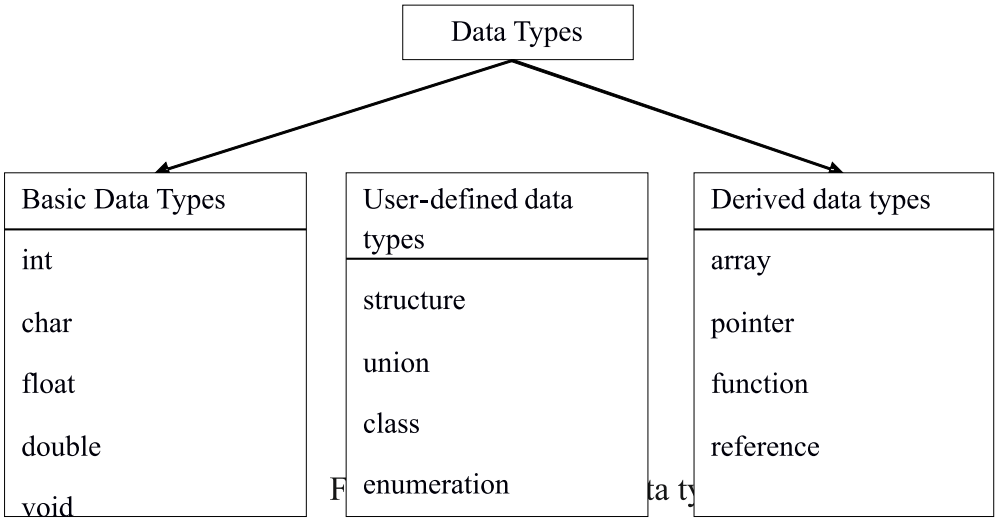
The fixed values that can't be changed during the execution of program

are called constants.
Examples:

```
65              // decimal integer
34.14           //floating point number
025             // octal integer
0x36            //hexadecimal integer
“Hello”          //string constant
'Z'             //character constant
```

6.7 Basic Data Types

Data types can be classified as shown in figure 6.2



F ta ty

The basic data types may have various modifiers which are placed before them, except the void data type. The modifiers may be applied to characters and integer basic data types. The modifier long can also be applied to double. The list of modifiers is as follows:

- signed
- unsigned
- short
- long

The list of all combinations of the basic data types and the modifiers along with their size and range is shown in table 6.2.

Table 6.2 Size and range of basic data types

Data type	Size in bytes	Range
char	1	-128 to 127
unsigned char	1	0 to 255
signed char	1	-128 to 127
int	2	-32768 to 32767
unsigned int	2	0 to 65535
signed int	2	-32768 to 32767
short int	1	-128 to 127
long int	4	-2147483648 to 2147483647
float	4	3.4E-38 to 3.4E+38
double	8	1.7E-308 to 1.7E+308
long double	10	3.4E-4932 to1.1E+4932

6

Structure

Basic data types are not sufficient to handle real world problems. A structure is a group of basic data types and other data types. The syntax of structure is:

```
struct name_of_structure
{
 data_type member1;
 data_type member2;
      - - - - - -
      - - - - - -
};
```

Let us take an example of a student which has several attributes such as name, age, percentage etc.

```
struct student
{
        char name[20];
        int age;
        float percentage;
};
struct student student1, student2;
```

Here student1 and student2 are variables of user-defined data type 'student'.

Union

Union is similar to structure, but there is a difference between structure and union. The size of structure type is equal to the sum of sizes of individual member types while the size of union type is equal to the size of its largest member's type. For example:

```
union item
{
        int m;
        float x;
        char c;
}item1;
```

This declares a variable item1 of type 'item'. This union contains three members each with different data type. However, only one of them can be used at a time. The variable item1 will occupy four bytes in memory as its largest size member is of floating type variable x. If we define item as a structure then the variable item1 will occupy seven bytes in memory. We can say union is memory efficient alternative of structure.

Class

Class is an important feature of C++. Just like any other basic data types class type variable can be declared. The class type variables are called objects. We will discuss classes in detail in chapter 9.

Enumerated Data Type

It is way of attaching names to numbers. The enum keyword assign the numbers 0,1,2, and so on to the list of names.

For example:

```
enum color{red, green, blue};
```

By default red is assigned 0, green is assigned 1 and blue is assigned 2. We can over-ride the default values by explicitly assigning integer values to the enumerators.

For example:

```
enum color{red, green=3, blue=8};
```

Here, red is assigned 0 by default.

6.9 Derived Data Types

Arrays

An array is collection of elements of same type.

For example:

```
int numbers[5]={2, 7, 8, 9, 11};
```

Here, the 'numbers' is an array of size five and containing the five

integer type elements.

Functions

A function is part of a program which is used to perform a task. Dividing a program in functions is one of the major principles of structured programming. It reduces the size of program by calling and using them at different places in the program. We will discuss functions in more detail in chapter 8.

Pointers

A pointer is variable which is used to store the address of another variable.
For example:

```
int x=5;        //integer variable
int *ptr;       // integer pointer variable
ptr= &x;        //address of x assigned to ptr
*ptr=10;        //the value of x is changed from 5 to 10
```

Reference Type

A variable of reference type is called reference variable. It provides an alternative name for the previously defined variable.
For example:

```
int x=10;
int & y=x;
```

Here, x is an integer variable and y is a reference type variable and it is an alternative name for already declared variable x.

```
cout<<x;
```

and

```
cout<<y;
```

both statements will print the value 10. The statement

```
x=x+5;
```

will change the values of both x and y to 15.

6.10 Type Compatibility

C++ is very strict to type compatibility as compared to C. int, short int and long int are three different types. They must be type cast when their values are assigned to one another. For an operation the type of operands must be type compatible with the operation. There are two mechanisms to achieve type compatibility.

Explicit type conversion

By using the type cast operator, explicit type conversion of variables and expressions can performed.

Program 6.2: Explicit type conversion

```
#include<iostream>
using namespace std;
int main()
{
    int i=5;
    float f=30.57;
    cout<<"i="<<i;
    cout<<"\nf="<<f;
    cout<<"\nfloat(i)="<<float(i);
    cout<<"\nint(f)="<<int(f);
    return 0;
}
```

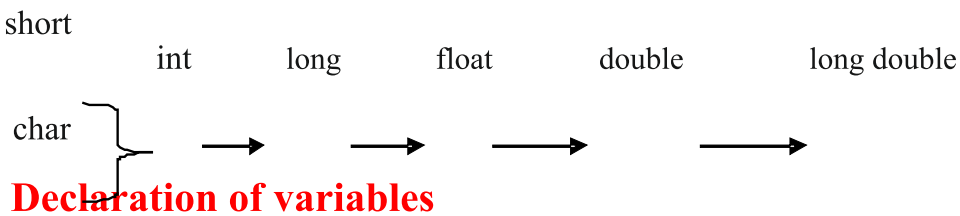
The output of the program 6.2 would be:

```
i=5
f=30.57
float(i)=5
int(f)=30
```

Implicit type conversion

When an expression consists of mix data types, the compiler performs the automatic type conversion by using the rule that smaller type is converted to the wider type. Whenever a char or short int appears in an expression, it is converted to int. This is called integral widening conversion.

The following diagram shows the implicit conversion rule:



6.11 Declaration of variables

In C, all variables are declared at the beginning of the program. In C++, this is true but it also allows the declaration of variables anywhere in the program. For example

```
int main()
{
```

```
        int x,y;                    //variable declaration
        cin>>x>>y;
        int sum=x+y; //variable declaration
        cout<<sum;
}
```

Important Points

- C++ program is a collection of functions.
- Every C++ program must have a main function.
- C++ program statements terminate with semicolon.
- Statements preceded with # symbol are called pre-processor directive statements.
- The smallest individual unit in a program is called token.
- Reserved words whose meaning can't be changed by the programmer are called keywords.
- Names of variables, functions, arrays, classes etc. that are created by the programmer are called identifiers.
- Fixed values that can't be changed during the execution of program are called constants.
- A structure is a group of basic data types and other data types.
- An array is collection of elements of same type.
- A pointer is variable which is used to store the address of another variable.
- C++ is very strict to type compatibility as compared to C.

Practice Questions

Objective type questions:

Q.1 Structure of a C++ program consist of

A. Class Declaration B. Member Function Definition
C. Main Function D. All of these

Q.2 Symbol used to comment a single line is

A. \\ B. //
C. || D. !!

Q.3 Pre-processor directive statements are preceded with the symbol

A. $ B. #
C. & D. *

Q.4 In Linux OS, the command used to compiling and linking a C++ program is

A. g++ B. a++
C. y++ D. z++

Q.5 Which is a token?

A. Keywords B. Identifiers
C. Operators D. All of these

Q.6 Which is NOT a basic data type
A. int B. char
C. float D. class

Very Short Answer Type Questions:

Q.1 What are tokens?
Q.2 What are keywords?
Q.3 What are identifiers?
Q.4 What are constants?
Q.5 What is difference between structure and union?

Short Answer Type Questions:

Q. 1 Explain the classification of data types.
Q.2 What is enumerated data type?
Q.3 What is reference type?

Essay Type Questions:

Q.1 Explain compiling and linking of a C++ program on Linux OS.
Q.2 Explain the implicit and explicit type conversions with suitable examples.

Answer Key

1. D 2. A 3. B 4. A 5. D 6. D