NEP NEW SYLLABUS

C++ PROGRAMMING

[For Practical]

Credits : 4

For B.Sc. (Computer Applications) : Semester - III

As Per New Syllabus, Effective from June 2025-26

Dr. Ms. Manisha Bharambe

M.Sc. (Comp. Sci), M.Phil, Ph.D. (Comp. Sci)

Vice Principal and Associate Professor

Computer Science Department

MES Abasaheb Garware College

Pune 4

Price ₹ 225.00

NIRALI PRAKASHAN

ADVANCEMENT OF KNOWLEDGE

N2557

C++ Programming **ISBN 978-93-6993-517-8**

First Edition : July 2025

© : Author

The text of this publication, or any part thereof, should not be reproduced or transmitted in any form or stored in any computer storage system or device for distribution including photocopy, recording, taping or information retrieval system or reproduced on any disc, tape, perforated media or other information storage device etc., without the written permission of Author with whom the rights are reserved. Breach of this condition is liable for legal action.

Every effort has been made to avoid errors or omissions in this publication. In spite of this, errors may have crept in. Any mistake, error or discrepancy so noted and shall be brought to our notice shall be taken care of in the next edition. It is notified that neither the publisher nor the author or seller shall be responsible for any damage or loss of action to any one, of any kind, in any manner, therefrom. The reader must cross check all the facts and contents with original Government notification or publications.

Published By :
NIRALI PRAKASHAN
Abhyudaya Pragati, 1312, Shivaji Nagar,
Off J.M. Road, Pune – 411005
Tel - (020) 25512336/37/39
Email : niralipune@pragationline.com

Polyplate

Printed By :
YOGIRAJ PRINTERS AND BINDERS
Survey No. 10/1A, Ghule Industrial Estate
Nanded Gaon Road
Nanded, Pune - 411041

DISTRIBUTION CENTRES

PUNE

Nirali Prakashan
(For orders outside Pune)
S. No. 28/27, Dhayari Narhe Road, Near Asian College
Pune 411041, Maharashtra
Mobile : 9049672225

Nirali Prakashan
(For orders within Pune)
119, Budhwar Peth, Jogeshwari Mandir Lane
Pune 411002, Maharashtra
Mobile : 9657703145, 9890997937
Email : niralilocal@pragationline.com

MUMBAI

Nirali Prakashan
Rasdhara Co-op. Hsg. Society Ltd., 'D' Wing Ground Floor, 385 S.V.P. Road
Girgaum, Mumbai 400004, Maharashtra
Mobile : 7045821020, Tel : (022) 2385 6339 / 2386 9976
Email : mumbai@niralibooks.com

DISTRIBUTION BRANCHES

DELHI

Nirali Prakashan
Room No. 2 Ground Floor
4575/15 Omkar Tower, Agarwal Road
Darya Ganj, New Delhi 110002
Mobile : 9818561840
Email : delhi@niralibooks.com

BENGALURU

Nirali Prakashan
Maitri Ground Floor, Jaya Apartments,
No. 99, 6th Cross, 6th Main,
Malleswaram, Bengaluru 560003
Karnataka; Mob : 9686821074
Email : bengaluru@niralibooks.com

NAGPUR

Nirali Prakashan
Above Maratha Mandir, Shop No. 3,
First Floor, Rani Jhanshi Square,
Sitabuldi Nagpur 440012 (MAH)
Tel : (0712) 254 7129
Email : nagpur@niralibooks.com

KOLHAPUR

Nirali Prakashan
438/2, Bhosale Plaza, Ground Floor
Khasbag, Opp. Balgopal Talim
Kolhapur 416 012, Maharashtra
Mob : 9850046155
Email : kolhapur@niralibooks.com

JALGAON

Nirali Prakashan
34, V. V. Golani Market, Navi Peth,
Jalgaon 425001, Maharashtra
Tel : (0257) 222 0395
Mob : 94234 91860
Email : jalgaon@niralibooks.com

SOLAPUR

Nirali Prakashan
R-158/2, Avanti Nagar, Near Golden
Gate, Pune Naka Chowk
Solapur 413001, Maharashtra
Mobile 9890918687
Email : solapur@niralibooks.com

For Order Enquiries: marketing@pragationline.com | www.niralibooks.com

Also find us on www.facebook.com/niralibooks

Preface ...

I take this opportunity to present this book entitled as **"C++ Programming"** to the students of Third Semester - B.Sc. (Computer Applications). The object of this book is to present the subject matter in a most concise and simple manner. The book is written strictly according to the New Syllabus (NEP Pattern).

The book has its own unique features. It brings out the subject in a very simple and lucid manner for easy and comprehensive understanding of the basic concepts, its intricacies, procedures and practices. This book will help the readers to have a broader view on C++ Programming. The language used in this book is easy and will help students to improve their vocabulary of Technical terms and understand the matter in a better and happier way.

I sincerely thank Shri. Dineshbhai Furia and Shri. Jignesh Furia of Nirali Prakashan, for the confidence reposed in me and giving me this opportunity to reach out to the students of management studies.

I have given my best inputs for this book. Any suggestions towards the improvement of this book and sincere comments are most welcome on niralipune@pragationline.com.

Author

■■■

Syllabus ...

1. **Object Oriented Programming and Functions in C++ [20 Hrs.]**

   Assignment No 1: Creation of Classes, Objects, Methods, Access Specifiers, Input-Output.

   Assignment No 2: Scope Resolution Operator, Static Members, Call by Reference.

   Assignment No 3: Inline Function, Friend Class and Function.

2. **Inheritance and Polymorphism [20 Hrs.]**

   Assignment No 4: Constructor and Destructor

   Assignment No 5: Single Inheritance and Multiple Inheritance

   Assignment No 6: Multilevel Inheritance and Hierarchical Inheritance, Hybrid Inheritance

   Assignment No 7: Polymorphism (Function Overloading)

   Assignment No 8: Polymorphism (Operator Overloading)

3. **File Handling and Exception Handling [20 Hrs.]**

   Assignment No. 9: Operations on Files (Read, Write, Open, Close), Random Access File Functions

   Assignment No 10: Exception Handling

   Assignment No.11: Hash Tables and Dictionaries

■■■

Contents ...

1. **Object Oriented Programming and Functions in C++** **1.1 – 1.60**

2. **Inheritance and Polymorphism** **2.1 – 2.80**

3. **File Handling and Exception Handling** **3.1 – 3.34**

■■■

CHAPTER 1

Object Oriented Programming and Functions in C++

Objectives ...

After learning this chapter you will be able:

- To learn How C++ evolved from C?
- To understand Features of C++.
- To study about compiling, linking and running a C++ program.
- To study scope resolution operator, static members, call by reference.
- To learn about Inline function, Friend class and function.

1.1 INTRODUCTION

- C++ is an extension of C language, developed by Bjarne Stroustrup at AT&T's Bell Labs in the early 1980's.
- C++ is a general purpose programming language which has been derived from C programming language.
- C++ could be considered a superset of C language with extensions and improvements with object oriented features included in it. C++ runs on a variety of platforms, such as Windows, Mac OS, UNIX, LINUX etc.
- C++ is regarded as a middle-level language, as it comprises a combination of both high-level and low-level language features.
- C++ is popular languages because of following reasons:
  1. C++ is ideally suited for development of reusable software.
  2. C++ is highly flexible language with versatility.

1.1.1 Features of C++

- C++ language provides a lot of features that are given in Fig. 1.1.

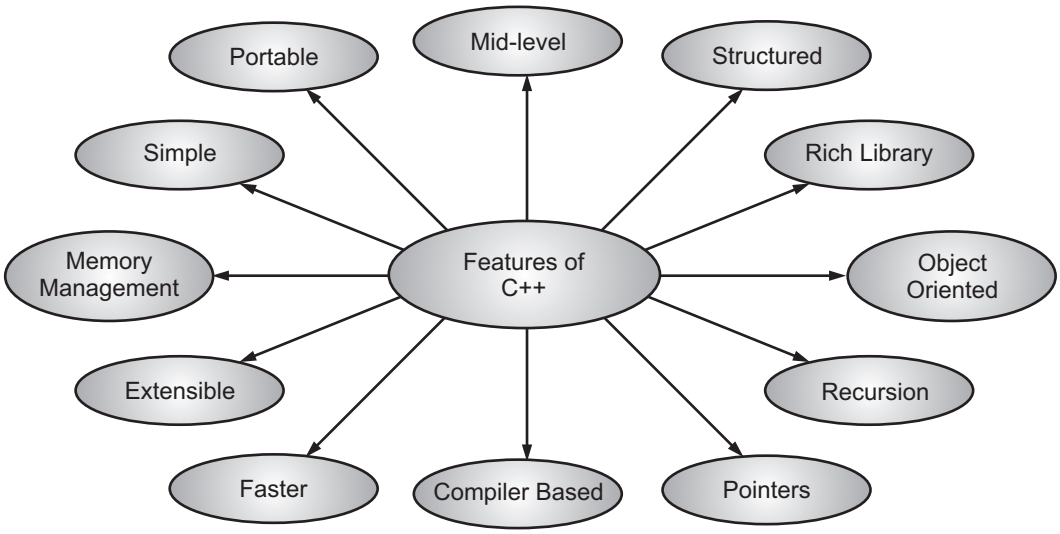


Fig. 1.1: Features of C++

- Various features of C++ are explained below:
  1. **Simple:** C++ is a simple language because it provides structured programming approach, rich set of library functions, data types etc.
  2. **Machine Independent or Portable:** C++ programs can be executed in different computer machines with little bit or no change.
  3. **Middle-level Programming Language:** C++ is regarded as a middle-level language, as it comprises a combination of both high-level and low-level language features.
  4. **Structured Programming Language:** C++ is a structured programming language in the sense that we can break the program into parts using functions. So, it is easy to understand and modify.
  5. **Rich Library:** C++ provides a lot of inbuilt functions that makes the development fast.
  6. **Memory Management:** C++ supports the feature of dynamic memory allocation. In C++ language, we can free the allocated memory at any time by calling the free() function.
  7. **Speed:** The compilation and execution time of C++ language is fast.
  8. **Pointer:** C++ provides the feature of pointers. We can directly interact with the memory by using the pointers. We can use pointers for memory, structures, functions, array etc.
  9. **Recursion:** In C++, we can call the function within the function. It provides code reusability for every function.
  10. **Extensible:** C++ language is extensible because it can easily adopt new features.
  11. **Object Oriented:** C++ is object oriented programming language. OOPs make development and maintenance easier.

12. **Compiler Based:** C++ is a compiler based programming language, it means without compilation no C++ program can be executed. First we need to compile the program using compiler and then we can execute the program.

1.1.2 Structure of a C++ Program

- A computer program is a sequence of instructions that tell the computer what to do.
- A typical C++ program would contain four sections:
  1. Include file (Header file).
  2. Class declaration.
  3. Member function definition.
  4. Main function.
- Fig. 1.2 shows structure of a C++ program.

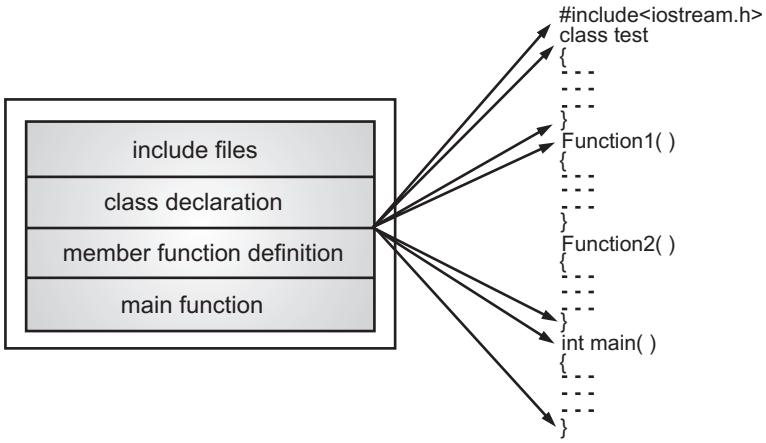


Fig. 1.2: Structure of C++ Program

- C++ program is a collection of functions.

Program 1.1: Simple "Hello World" program.

```
1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5.     cout << "Hello World!" << endl;
6.     return 0;
7. }
```

Output:

```
Hello World!
```

- Line 1 `#include<iostream>` is a special type of statement called a preprocessor directive and it start from #. Preprocessor directives tell the compiler to perform a special task. In this case, we are telling the compiler that we would like to use the iostream library. The iostream library contains code that tells the compiler what cout and endl do. In other words, we need to include the iostream library in order to write to the screen.
- Line 2 `using namespace std;` defines a scope for the identifiers that are used in the program. As you learned in the explanation for line 1, cout and endl live inside the iostream library. However, within iostream, they live inside a special compartment named std (short for standard). This using statement tells the compiler to look inside a compartment named std if it cannot find cout or endl defined anywhere else. In other words, this statement is also necessary so the compiler can find cout and endl, which we use on line 5.
- Lines 3 `int main()` declares the main() function which is the point from where all C++ program begin their execution. Every program must have a main() function.
- Line 4 ({ is the opening curly brace marks the start of a code block) and 7 (} is the closing curly brace, marks the end of function main()) tell the compiler which lines are part of the main function. Everything between the opening curly brace on line 4 and the closing curly brace on line 7 is considered part of the main() function.
- Line 5 `cout <<"Hello World!" << endl;` is our output statement. cout is a special object that represents the console/screen. The << symbol is an operator (much like + is an operator) called the output operator. cout understands that anything sent to it via the << operator should be printed on the screen. endl is a special symbol that moves the cursor to the next line.
- Line 6 `return 0;` is a new type of statement, called a return statement. When an executable program finishes running, it sends a value to the operating system that indicates whether it was run successfully or not. The return value of main () is used for this purpose. This particular return statement returns the value of 0 to the operating system, which means "everything went okay!".

1. Comments:

- Comments are parts of the source code disregarded by the compiler. They simply do nothing.
- Their purpose is only to allow the programmer to insert notes or descriptions embedded within the source code.
- C++ supports two ways to insert comments:

  (i) // line comment, and

  (ii) /* */ block comment.
- The first of them, known as line comment, discards everything from where the pair of slash signs (//) is found up to the end of that same line.
- The second one, known as block comment, discards everything between the /* characters and the first appearance of the */ characters, with the possibility of including more than one line.

Program 1.2: Program with comments.

```
#include <iostream>
using namespace std;
int main()
{
   cout << "Hello World!" << endl;        //Prints Hello World!
   cout << "Welcome to C++ !" << endl;    //Prints Welcome to C++!
   return 0;
}
```

Output:

```
Hello World!
Welcome to C++ !
Press any key to continue . . .
```

- If you include comments within the source code of your programs without using the comment characters combinations //, /* */, the compiler will take them as if they were C++ expressions, most likely causing one or several error messages when you compile it.

2. Header Files:

- Each standard library has a corresponding header file containing the function prototypes for all the functions in that library and definitions of various data types and constants needed by those functions.
- Some common C++ header files that may be included in standard C++ program, are given below:

Table 1.1: Standard Library Header Files

Standard library header files	Explanation
<assert.h>	Contains macros and information for adding diagnostics that aid program debugging.
<ctype.h>	Contains function prototypes for functions that test characters for certain properties and that can be used to convert lowercase letters to uppercase letters.
<float.h>	Contains the floating point size limits of one program.
<limits.h>	Contains integral size limits of the system.
<math.h>	Contains function prototype for math library function.

contd. …

<stdio.h>	Contains function prototype for the standard input/output library functions and information used by them.
<stdlib.h>	Contains function prototype for conversions of numbers to text, text to numbers, memory allocation, random numbers and various other utility functions.
<string.h>	Contains function prototype for C-style string processing function.
<time.h>	Contains function prototype for manipulating time and date.
<iostream.h>	Contains function prototype for standard input and output functions.
<iomanip.h>	Contains function prototype for the stream manipulators that enable formatting of streams of data.
<fstream.h>	Contains function prototype for functions that perform input from files on disk and output to files on disk.

- The programmer can create custom header files. Programmer-defined files should end in .h, which can be included by using the # include preprocessor directive.
- For example, the header file square.h can be included in our program by the directive,

```
# include "square.h"
```

at the top of the program.

3. Streams in C++:

- C++ supports a set of functions for performing input and output operations. The new features of C++ are called **streams** which are used to handle I/O operations.
- C++ streams are of two types:
  (i) Input stream.
  (ii) Output stream.
- Stream refers to the flow of data from a particular source to a specified destination.
- There are classes representing each stream **istream** is the class representing the input stream (istream) and **ostream** is the class representing the output stream.
- To achieve the console input/output operations, we use the objects of these stream classes.

(i) Input Stream:

- The input stream is handling all input operations or reading operations with input devices like keyboard, disk etc. **"cin"** is the predefined object of the istream class. It is used for all console input operations.
- Operator **>>**, (or the right shift operator in C) is called the **extraction** operator, it gets the value from the stream object **cin** on its left and places it in the variable on its right.

  Syntax:

```
cin>>variable;
```

- The use of `cin` is shown in Fig. 1.3.

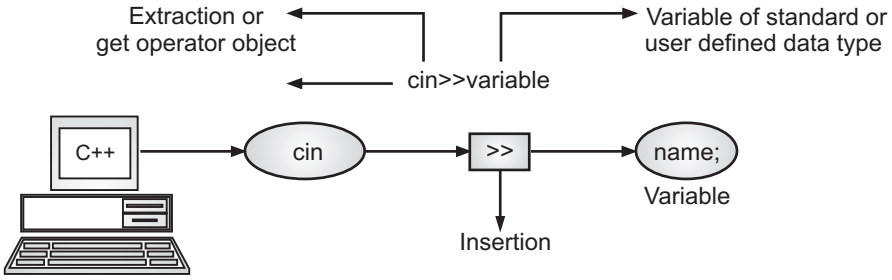


Fig. 1.3: Input with cin Operator

Examples:

1. `int a;`
   `cin>>a;`
2. `float salary;`
   `cin>>salary;`
3. `double area;`
   `cin>>area;`
4. `char name[15];`
   `cin>>name;`

- We can input more than one item using cin object. Such input operations are called cascaded input operations.
- cin will read all the items from left to right.

Syntax for more than one variable as:

```
cin>>var1>>var2>>var3........ >>varn;
```

For examples,

1. `cin>>x>>y;  //x read first then y`
2. `cin>>name>>address;`
3. `cin>>roll>>name>>marks;`

(ii) Output Stream:

- The output stream handles the write operations on output devices like screen, disk etc. **cout** is the predefined object of the **stream** class and is used for all console output operations.
- Operator **<<,** (or the left shift operator in C) is called **insertion** operator. It directs the contents of the variable on its right to the object (cout) on its left.

Syntax:

```
cout<<variable;
```

- The use of cout is shown in Fig. 1.4.

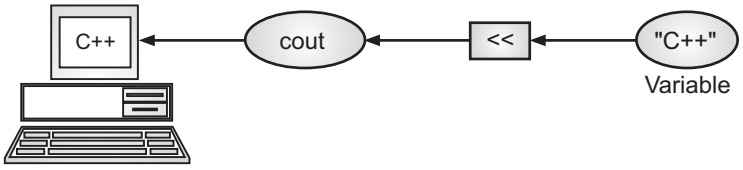


Fig. 1.4: Output with cout Operator

For examples,

1. `int a;`
   `cout<<a;`
2. `float salary;`
   `cout<<salary;`
3. `char name[15];`
   `cout<<name;`

- We can display more than one outputs called cascaded outputs.

Syntax:

```
cout<<var1<<var2<<.....<<varn;
```

For examples,

1. `cout<<"x="<<x;`
2. `cout<<roll<<"  "<<name;`
3. `cout<<roll<<"\t"<<name;`
4. `cout<<name<<endl;`

1.1.3 Data Types

- A data type determines the type and the operations that can be performed on the data. A data type is defined as, "the set/collection of values and the set of operations that operate on those values".
- In short, data type is analyzed as follows:

  Data Type = Permitted Data Values + Operations
- The various data types provided by C++ are built-in data types, derived data types and user-defined data types as shown in Fig. 1.5.

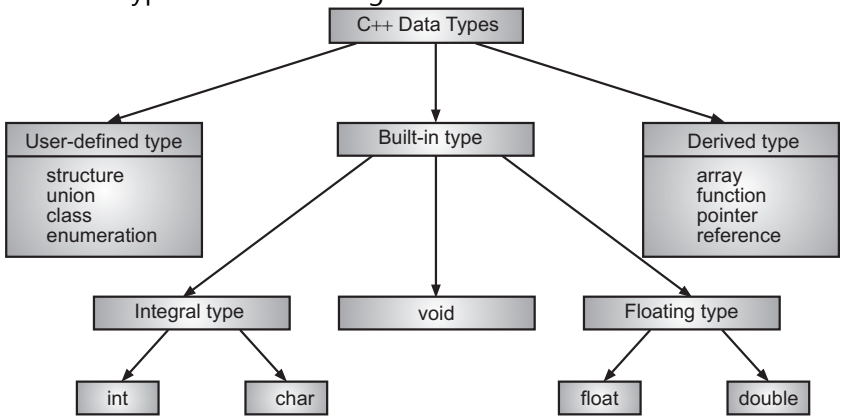


Fig. 1.5: Hierarchy of C++ Data Types

1. Built-in Data Types:

- Built-in data types are atomic data type which is not composed of other data types. C++ compilers support all the built-in (also known as basic or fundamental) data types as shown in Fig. 1.5.
- C++ basic data types are integer, float, character and boolean. C++ uses two more data types, bool and wchar_t.
- The Table 1.2 shows the description of C++ basic data types.

Table 1.2: Basic Data Type, Size and Ranges

Type	Bytes	Range
char	1	signed char: – 128 to 127
		unsigned char: 0 to 255
int	2	signed int: – 32768 to 32767
		unsigned int: 0 to 655535
short int	2	signed: – 32768 to 32767
		unsigned: 0 to 655535
long int	4	signed: – 2147483648 to 2147483647
		unsigned: 0 to 4294967295
float	4	3.4 E – 38 to 3.4 E + 38
double	8	1.7 E – 308 to 1.7 E + 308
long double	10	3.4 E – 4932 to 1.1 E + 4932
		(Compiler dependent)

- The short, long and unsigned qualifiers are used with int types. We can used unsigned qualifier when we want the variable to have positive values only.

For example,

```
short number;   //short int
long area;      //long int
unsigned i,j;   //unsigned int
double result;
```

2. User-defined Data Types:

- User has the facility in the programming language to create data types according to his/her own needs.
- The data types defined by user according to his/her requirements known as user-defined data types.

(i) Enumerated:

- In C++, enumerated data type helps in enhancing the reliability of the code. Enumerations are a way of defining constants.

Syntax:

```
enum tag {member1, member2, ……., membern};
```

- Here, enum is the keyword and tag is the name of enumeration and member1, member2, … membern are identifiers which represents integer constant values. The member should be unique. Once, the enumeration is define then variable or object of that type is defined.

Syntax:

```
enum tag {member1, member2 ……} variables;   OR
enum tag {member1, member2 ……} object_name;
```

- Here, all members take integer values starting with zero i.e. member1 is assigned 0, member2 is assigned 1 …… and so on.

For example,

```
enum colors {black, green, red, yellow, blue};
enum colors background;
```

Here ,

```
black = 0
green = 1
red = 2
  .
  .
  .
```

- Enumeration variables are particularly useful as flags to indicate various options are to identify various conditions. It increases clarity of the program. If we want to have blue colors for the background, it is easier to understand with,

```
background = blue;
```

Than with,

```
background = 4;
```

It makes program more readable.

(ii) Structure:

- In 'C' structure is a collection of data items of different data types. Structure in C++ is a collection of variables of different data types in a single name.

Syntax:

```
struct name of structure
{
   data members;
   functions
}
```

For example,

```
struct student
{
   private;
      char name[20];
      long phone;
   public:
      void display(void)
         {
            cout<<phone;
         }
};
```

- By default members of the C++ structures are public.

For example,

```
struct student s;
printf ("%ld",  s.phone);    //Invalid
```

- We cannot print phone like this, we have to invoke public function display to print phone i.e. Instead of printf statement we can write,

```
s.display();
```

(iii) Union:

- Union in C++ is a user defined data type that uses the same memory as other objects from a list of objects. At an instance it contains only a single object.

Syntax:

```
union tag
{
   member1;
   member2;
   '
   '
   member n;
} objectname;
```

For example,

```
union example
{
   int a;
   char b;
   float c;
} e;
```

- There are three elements of different type's int, char and float respectively. Here, int takes two bytes memory, char takes one byte and float takes four bytes. Union allows the same storage is shared with its members. Hence, total memory required is four bytes. Unions are very useful for applications involving multiple members, where values need not be assigned to all members at one time. While in case of structure for same example, memory required is 2 + 1 + 4 = 7 bytes to store its contents.

(iv) Class:

- C++ also permits to define another user-defined data type known as class which can be used, just like any other basic data type, to declare variables. The class variables are known as objects, which are the focus of object oriented programming. Generally, class is used in C++ instead of struct. Class is user define data type with data elements and functions.
- **Difference between a class and a structure in C++**: There is not much difference between a class in C++ and a structure in C++. Everything which is defined in structure in C++ is also applicable to the classes in C++.
- **The two differences are:**
  - The keyword class is used instead of struct.
  - All the members in a class are by default private.

For example,

```
class student
{
   int roll;          }
   char name[20];     }  By default both are private
   public:
      void getdata(void);
      void putdata();
};
```

3. Derived Data Types:

- The data types, which are extracted/derived from fundamental data types are called derived data types.

(i) Arrays: An array in C++ is a collection of fixed-size finite number of objects, data value that belong to the same type. An array is a collection of data storage locations. Each of the locations of an array in C++ holds same type of data. The storage location is called an element or component of the array. The individual element within the array is called subscript. In C, array size is exact same length of the string constant.

For example, `char string[5] = 'abcde";//valid in C`

But in C++, the array size is one larger than the number of characters present in the string. Here, "\0" is treated as a one character.