Business Application Programming

Introduction:

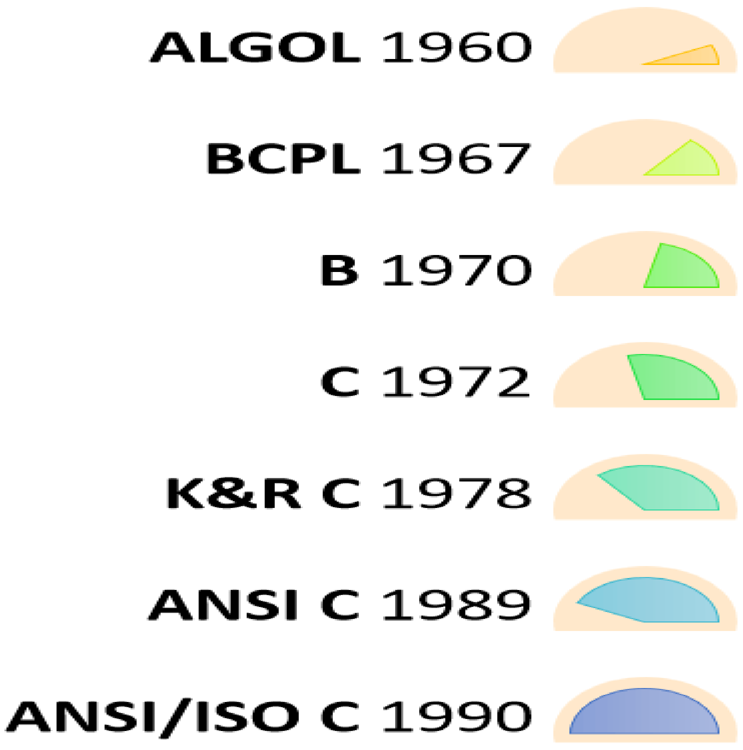
C is a general-purpose programming language that is extremely popular, simple and flexible. It is machine-independent, structured programming language which is used extensively in various applications.

C was the basic language to write everything from operating systems (Windows and many others) to complex programs like the Oracle database, Git, Python interpreter and more.

It is said that 'C' is a god's programming language. One can say, C is a base for the programming. If you know 'C,' you can easily grasp the knowledge of the other programming languages that uses the concept of 'C'

It is essential to have a background in computer memory mechanisms because it is an important aspect when dealing with the C programming language.

History of C language



1. **1960 - ALGOL - Developed by International Group**
2. **1967 - BCPL - Developed by Martin Richards**
3. **1970 - B - Developed by Ken Thompson**
4. **1972 - C - Developed by Dennis M. Ritchie**
5. **1978 - K&R C - Developed by Brian w. Kernighan and Dennis M. Ritchie**
6. **1989 - ANSI C - Developed by ANSI Committee**
7. **1990 - ANSI/ISO C - Standardaized by ISO**

Languages such as C++/Java are developed from 'C'. These languages are widely used in various technologies. Thus, 'C' forms a base for many other languages that are currently in use.

The C programming language is developed by **Dennis Ritchie at AT&T Bell Laboratories in 1972.** It is named C because many **features of C were derived from an earlier language called B.**

The **history of C language goes back to 1960's**, when a number of computer languages were being used for various purposes. **COBOL** (Common Business-Oriented Language) was being used for commercial purposes, **FORTRAN** (Formula Translation) was being used for scientific and engineering applications and so on.

Most of the modern languages including ANSI (American National Standards Institute)/ISO (International Organization for Standardization) C are derived from the algorithmic language called **ALGOL** which was developed by international group and introduced in 1960's.

Martin Richards in 1967 developed programming language called **BCPL (Basic Combined Programming Language)** which was derived from ALGOL. Similarly, **BCPL influenced development of programming language called B by Ken Thompson in 1970.**

In 1972, Dennis Ritchie introduced "Traditional C" and it was confined to use within Bell Laboratories until 1978.

In 1978, Brian Kernighan and Dennis Ritchie published a book called "The C Programming Language". The book was so popular and the use of C started spreading. **C Language at that time is commonly referred to as "K&R C".**

In 1983, the American National Standards Institute formed a committee to produce a C programming language standard. This **"ANSI C" was completed in 1988**, and was approved in 1989. It was then approved by the International Organization for Standardization (ISO) in 1990.

Importance of C Programming

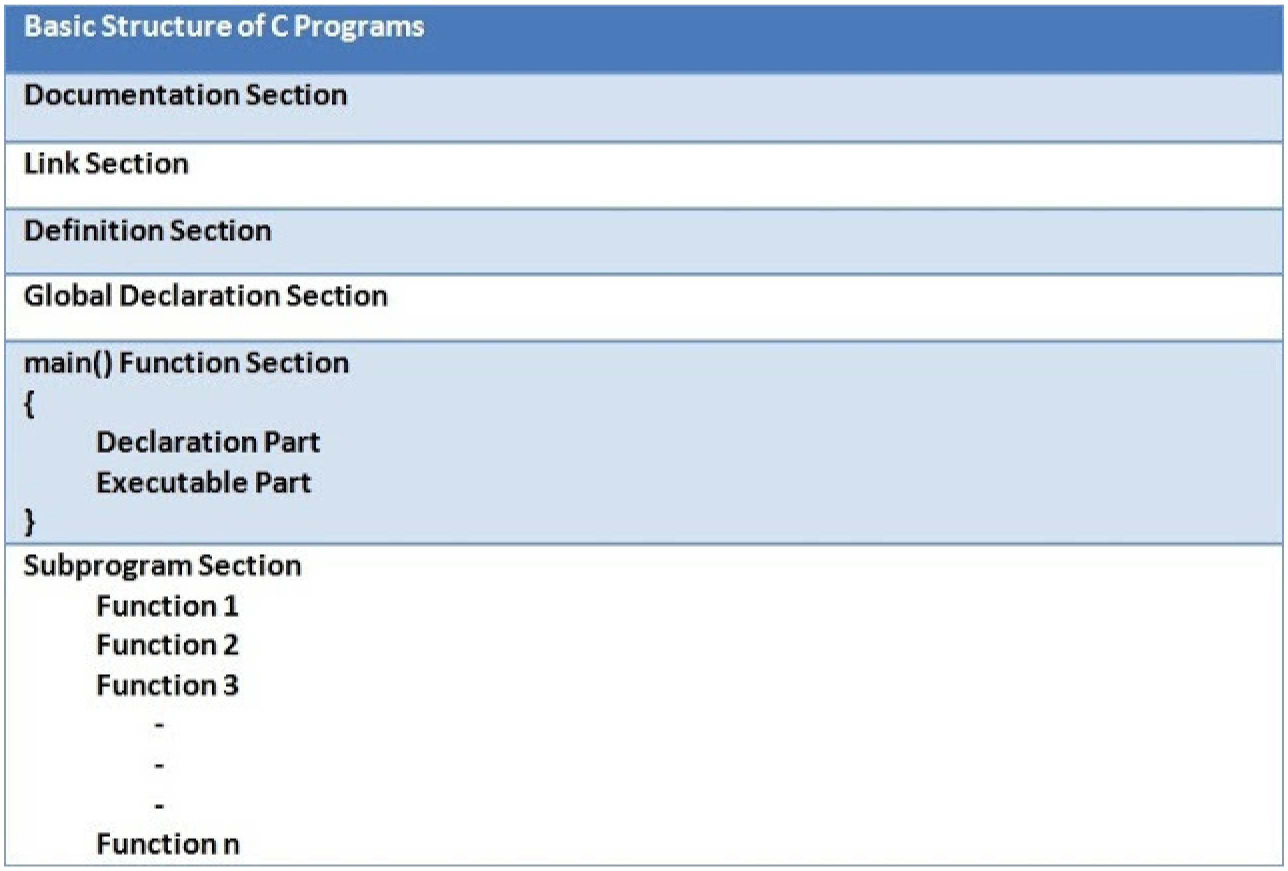
C programming language has following importances:

1. C is robust language and has **rich set of built-in functions, data types and operators** which can be used to write any complex program.
2. Program written in C are efficient due to availability of several data types and operators.
3. C has the capabilities of an assembly language (low level features) with the feature of high level language so it is well suited for writing both **system software and application software.**
4. **C is highly portable language** i.e. code written in one machine can be moved to other which is very important and powerful feature.
5. C supports low level features like **bit level programming and direct access to memory using pointer** which is very useful for managing resource efficiently.
6. C has high **level constructs** and it is more user friendly as its syntaxes approaches to English like language.

Basic Structure of C Program

Here you will learn about basic structure of C program.

Any C program is consists of 6 main sections. Below you will find brief explanation of each of them.



Basic Structure of C Programs
Documentation Section
Link Section
Definition Section
Global Declaration Section

main() Function Section
{
Declaration Part
Executable Part
}

Subprogram Section Function 1 Function 2 Function 3 - - - Function n

Basic Structure of C Program

Documentation Section

This section consists of comment lines which include the name of programmer, the author and other details like time and date of writing the program. Documentation section helps anyone to get an overview of the program.

Link Section

The link section consists of the header files of the functions that are used in the program. It provides instructions to the compiler to link functions from the system library.

Definition Section

All the symbolic constants are written in definition section. Macros are known as symbolic constants.

Global Declaration Section

The global variables that can be used anywhere in the program are declared in global declaration section. This section also declares the user defined functions.

main() Function Section

It is necessary have one main() function section in every C program. This section contains two parts, declaration and executable part. The declaration part declares all the variables that are used in executable part. These two parts must be written in between the opening and closing braces. Each statement in the declaration and executable part must end with a semicolon (;). The execution of program starts at opening braces and ends at closing braces.

Subprogram Section

The subprogram section contains all the user defined functions that are used to perform a specific task. These user defined functions are called in the main() function.

Variable, Data types, Constants

Variable

A variable is an identifier which is used to store some value. Constants can never change at the time of execution. Variables can change during the execution of a program and update the value stored inside it.

A single variable can be used at multiple locations in a program. A variable name must be meaningful. It should represent the purpose of the variable.

Example: Height, age, are the meaningful variables that represent the purpose it is being used for. Height variable can be used to store a height value. Age variable can be used to store the age of a person

A variable must be declared first before it is used somewhere inside the program. A variable name is formed using characters, digits and an underscore.

Following are the rules that must be followed while creating a variable:

1. A variable name should consist of only characters, digits and an underscore.
2. A variable name should not begin with a number.
3. A variable name should not consist of whitespace.
4. A variable name should not consist of a keyword.
5. 'C' is a case sensitive language that means a variable named 'age' and 'AGE' are different.

Following are the examples of valid variable names in a 'C' program:

height or HEIGHT

_height

_height1

My_name

Following are the examples of invalid variable names in a 'C' program:

1height

Hei$ght

My name

For example, we declare an integer variable **my_variable** and assign it the value 48:

int my_variable;

my_variable = 48;

By the way, we can both declare and initialize (assign an initial value) a variable in a single statement:

int my_variable = 48;

Data types

' C' provides various data types to make it easy for a programmer to select a suitable data type as per the requirements of an application. Following are the three data types:

1. Primitive data types
2. Derived data types
3. User-defined data types

There are five primary fundamental data types,

1. int for integer data
2. char for character data
3. float for floating point numbers
4. double for double precision floating point numbers
5. void

Array, functions, pointers, structures are derived data types. 'C' language provides more extended versions of the above mentioned primary data types. Each data type differs from one another in size and range. Following table displays the size and range of each data type.

Data type	**Size in bytes**	**Range**
Char or signed char	1	-128 to 127
Unsigned char	1	0 to 255
int or signed int	2	-32768 to 32767
Unsigned int	2	0 to 65535
Short int or Unsigned short int	2	0 to 255
Signed short int	2	-128 to 127
Long int or Signed long int	4	-2147483648 to 2147483647
Unsigned long int	4	0 to 4294967295
float	4	3.4E-38 to 3.4E+38
double	8	1.7E-308 to 1.7E+308
Long double	10	3.4E-4932 to 1.1E+4932

Note: In C, there is no Boolean data type.

Integer data type

Integer is nothing but a whole number. The range for an integer data type varies from machine to machine. The standard range for an integer data type is -32768 to 32767. An integer typically is of 2 bytes which means it consumes a total of 16 bits in memory. A single integer value takes 2 bytes of memory. An integer data type is further divided into other data types such as short int, int, and long int.

Each data type differs in range even though it belongs to the integer data type family. The size may not change for each data type of integer family.

The short int is mostly used for storing small numbers, int is used for storing averagely sized integer values, and long int is used for storing large integer values.

Whenever we want to use an integer data type, we have place int before the identifier such as, int age; Here, age is a variable of an integer data type which can be used to store integer values.

Floating point data type

Like integers, in 'C' program we can also make use of floating point data types. The 'float' keyword is used to represent the floating point data type. It can hold a floating point value which means a number is having a fraction and a decimal part. A floating point value is a real number that contains a decimal point. Integer data type doesn't store the decimal part hence we can use floats to store decimal part of a value.

Generally, a float can hold up to 6 precision values. If the float is not sufficient, then we can make use of other data types that can hold large floating point values. The data type double and long double are used to store real numbers with precision up to 14 and 80 bits respectively.

While using a floating point number a keyword float/double/long double must be placed before an identifier. The valid examples are,

float division;

double BankBalance;

Character data type

Character data types are used to store a single character value enclosed in single quotes.

A character data type takes up-to 1 byte of memory space.

Example,

```
Char letter;
```

Void data type

A void data type doesn't contain or return any value. It is mostly used for defining functions in 'C'.

Example,

```
void displayData()
```

Type declaration of a variable

```
int main() {
int x, y;
float salary = 13.48;
char letter = 'K';
x = 25;
y = 34;
int z = x+y;
printf("%d \n", z);
printf("%f \n", salary);
printf("%c \n", letter);
return 0;}
```

Output:

```
59
13.480000
K
```

We can declare multiple variables with the same data type on a single line by separating them with a comma. Also, notice the use of format specifiers in **printf** output function float (%f) and char (%c) and int (%d).

Constants

Constants are the fixed values that never change during the execution of a program. Following are the various types of constants:

Integer constants

An integer constant is nothing but a value consisting of digits or numbers. These values never change during the execution of a program. Integer constants can be octal, decimal and hexadecimal.

1. Decimal constant contains digits from 0-9 such as,

Example, 111, 1234

Above are the valid decimal constants.

2. Octal constant contains digits from 0-7, and these types of constants are always preceded by 0.

Example, 012, 065

Above are the valid decimal constants.

3. Hexadecimal constant contains a digit from 0-9 as well as characters from A-F. Hexadecimal constants are always preceded by 0X.

Example, 0X2, 0Xbcd

Above are the valid hexadecimal constants.

The octal and hexadecimal integer constants are very rarely used in programming with 'C'.

Character constants

A character constant contains only a single character enclosed within a single quote (''). We can also represent character constant by providing ASCII value of it.

Example, 'A', '9'

Above are the examples of valid character constants.

String constants

A string constant contains a sequence of characters enclosed within double quotes ("").

Example, "Hello", "Programming"

These are the examples of valid string constants.

Real Constants

Like integer constants that always contains an integer value. 'C' also provides real constants that contain a decimal point or a fraction value. The real constants are also called as floating point constants. The real constant contains a decimal point and a fractional value.

Example, 202.15, 300.00

These are the valid real constants in 'C'.

A real constant can also be written as,

Mantissa e Exponent

For example, to declare a value that does not change like the classic circle constant PI, there are two ways to declare this constant

1. By using the **const** keyword in a variable declaration which will reserve a storage memory

```
 #include <stdio.h>
int main() {
const double PI = 3.14;
printf("%f", PI);
//PI++; // This will generate an error as constants cannot be changed
return 0;}
```

2. By using the **#define** pre-processor directive which doesn't use memory for storage and without putting a semicolon character at the end of that statement

```
#include <stdio.h>
#define PI 3.14
int main() {
printf("%f", PI);
return 0;}
```

Object-Oriented Programming

OOP stands for **Object-Oriented Programming**.

Procedural programming is about writing procedures or functions that perform operations on the data, while **object-oriented programming** is about creating objects that contain both data and functions. ...

OOP provides a clear structure for the programs.

Object-Oriented Programming (OOP) is a programming language model that revolves around objects and not actions.

Historically, it was viewed as a procedure that takes input, processes the data and gives output.

Web developers across the world learn Object-Oriented Programming with Python to achieve many goals.

Some features of OOP

- Emphasis on data rather than procedure
- Programs are divided into entities known as objects
- Data Structures are designed such that they characterize objects
- Functions that operate on data of an object are tied together in data structures
- Data is hidden and cannot be accessed by external functions
- Objects communicate with each other through functions
- New data and functions can be easily added whenever necessary
- Follows bottom up design in program design

Basic Concepts of Object Oriented Programming

Abstraction

It is the process of picking out (abstracting) similar characteristics of procedures and objects.

Class

It means categorizing objects. A class defines all the common traits of the numerous objects that fall under it.

Encapsulation

It is defined as wrapping the data under a single, consolidated unit. In OOP, it is defined as binding data with a function that manipulates it.

Inheritance

It is defined as the ability of one class to derive its characteristics from another class.

Interface

It comprises the languages and the codes used by various applications to communicate with each other.

Object

It is an entity that is self-contained. It consists of data as well as procedures.
Polymorphism
It refers to a programming language's ability to process objects uniquely according to their data type and/or class.
Procedure
It is a part of a program performing a specific task.
Message Passing
It is a form of communication that is used in parallel programming and OOP.

Advantages of OOP

Re-usability:
"Write once and use it multiple times" you can achieve this by using class.

Redundancy:
Inheritance is the good feature for data redundancy. If you need a same functionality in multiple class you can write a common class for the same functionality and inherit that class to sub class.

Easy Maintenance:
It is easy to maintain and modify existing code as new objects can be created with small differences to existing ones.

Security:
Using data hiding and abstraction only necessary data will be provided thus maintains the security of data.

Disadvantages of OOP

Size:
Object Oriented Programs are much larger than other programs.

Effort:
Object Oriented Programs require a lot of work to create.

Speed:
Object Oriented Programs are slower than other programs, because of their size.

Advantages of OOP
1. Re-usability
2. Data Redundancy
3. Code Maintenance
4. Security
5. Design Benefits
6. Better productivity
7. Easy troubleshooting
8. Polymorphism Flexibility
9. Problems solving

Object-Oriented Programming Paradigm

Object-oriented programming (OOP) is a programming paradigm based upon objects (having both data and methods) that aims to incorporate the advantages of modularity and reusability. Objects, which are usually instances of classes, are used to interact with one another to design applications and computer programs.

The important features of object–oriented programming are −

- Bottom–up approach in program design
- Programs organized around objects, grouped in classes
- Focus on data with methods to operate upon object's data
- Interaction between objects through functions
- Reusability of design through creation of new classes by adding features to existing classes

Some examples of object-oriented programming languages are C++, Java, Smalltalk, Delphi, C#, Perl, Python, Ruby, and PHP.

Grady Booch has defined object–oriented programming as ***"a method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships".***

Object-Oriented Programming Paradigm

- The major motivating factor in the invention of object-oriented approch is to remove some of the flaws encountered in the procedural approch.
- OOP treats data as a critical element in the program development and does not allow it to flow freely around the systems.
- It ties data more closely to the functions that operate on it, and protects it from accidental modification from outside functions.
- OOP allows decomposition of a problem into a number of entities called objects and then builds data and functions around these objects.
- The data of an object can be accessed only by the function associated with that object.
- However, functions of one object can access the the functions of other objects.

Some of the striking features of object-oriented programming are

- Emphasis is on data rather than procedure.
- Programs are divided into what are known as objects.
- Data structures are designed such that they characterize the objects.
- Data is hidden and cannot be accessed by external functions.
- Objects may communicate with each other through functions.
- New data and functions can be easily added whenever necessary.
- Follows bottom-up approch in program design.

Applications of Object Oriented Programming

Main application areas of OOP are:

- User interface design such as windows, menu.
- Real Time Systems
- Simulation and Modeling
- Object oriented databases
- AI and Expert System
- Neural Networks and parallel programming
- Decision support and office automation systems etc.

Benefits of OOP

- It is easy to model a real system as real objects are represented by programming objects in OOP.
- The objects are processed by their member data and functions.
- It is easy to analyze the user requirements.
- With the help of i**nheritance**, we can reuse the existing class to derive a new class. This saves time and cost of program.
- In OOP, data can be made private to a class such that only member functions of the class can access the data.
- This principle of **data hiding** helps the programmer to build a secure program .
- With the help of **polymorphism**, the same function or same operator can be used for different purposes. This helps to manage software complexity easily.
- Large problems can be reduced to smaller and more manageable problems.
- It is easy to partition the work in a project based on objects.
- It is possible to have multiple instances of an object to co-exist without any interference i.e. each object has its own separate member data and function.

Structure of C++ program

Probably the best way to start learning a programming language is by writing a program. Therefore, here is our first program:

Syntax:

```
#include<iostream.h>
int main()
{
   .........;
   return 0;
}
```

Example:

```
#include<iostream.h>
//main() is where program execution begins.
int main()
{
   cout << "This is my first C++ Program!";
   return 0;
}
```

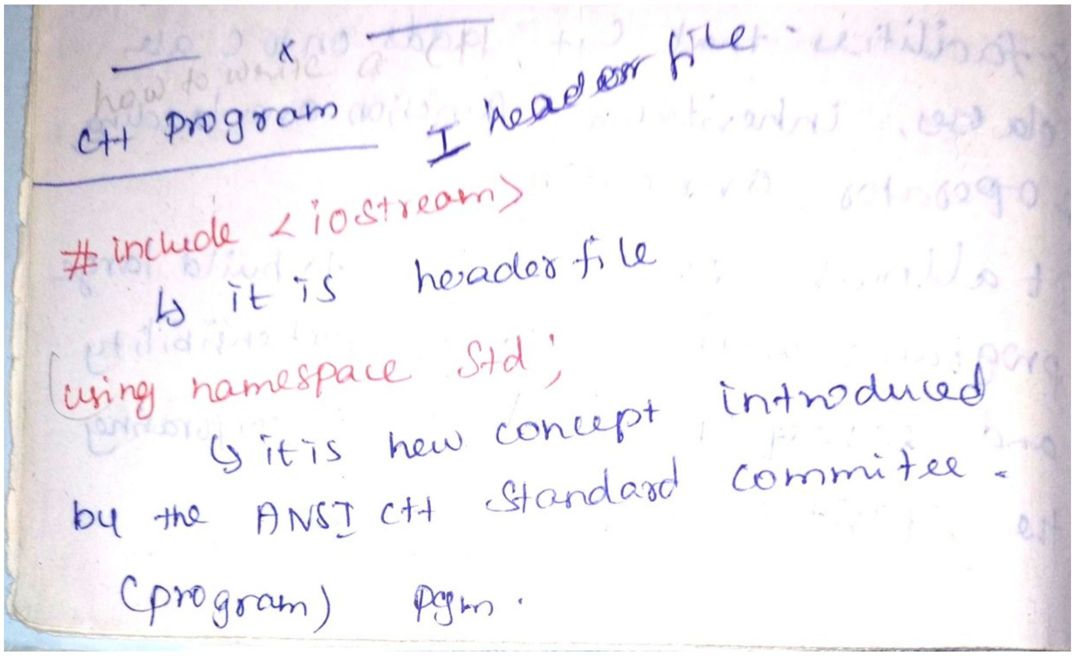
Output:

```
Command Prompt
This is my first C++ Program!
```

1. The C++ language defines several headers, which contain information that is either necessary or useful to your program. For this program, **#include<iostream.h>** header is needed. this header file is included for Standard I/O (input output) function.
2. 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.
3. The line **int main()** is the main function where program execution begins.
4. 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.
5. The next line **return 0;** terminates main() function and causes it to return the value 0 to the calling process.

Applications of C++

1. C++ is used by hundreds of thousands of programmers in essentially every **application domain.**
2. C++ is being highly used to write device drivers and other softwares that rely on direct manipulation of hardware under realtime constraints.
3. C++ is widely used for teaching and research because it is clean enough for successful teaching of basic concepts.
4. Anyone who has used either an Apple Macintosh or a PC running Windows has indirectly used C++ because the primary user interfaces of these systems are written in C++.
5. C++ is used to create computer programs! Anything from art applications, music players and even video games.



how to write a

C++ program

I header file

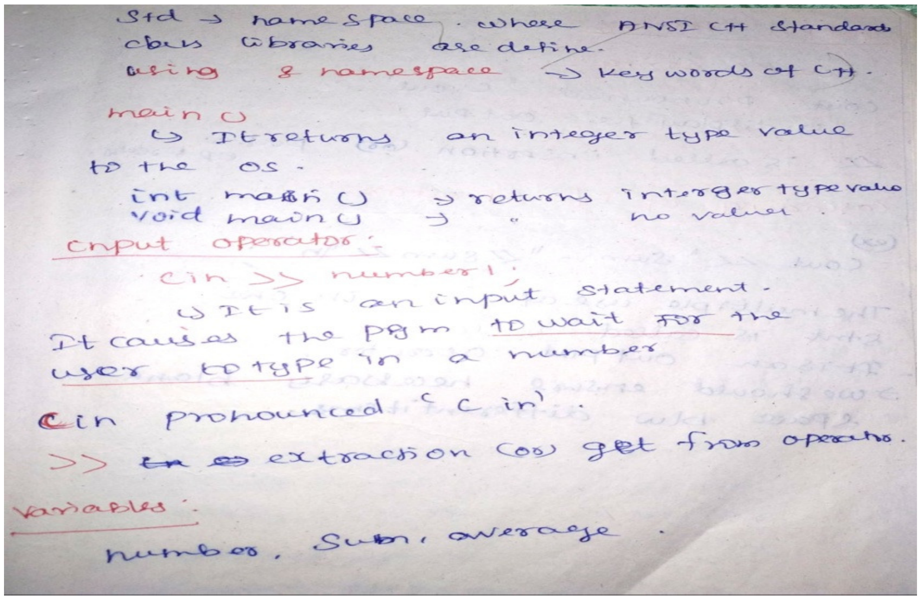
#include <iostream>

↳ it is header file

using namespace std;

↳ it is new concept introduced by the ANSI C++ Standard commitee.

(program) pgm.


std → namespace where ANSI C++ standard
class libraries are define.
using & namespace → keywords of C++.

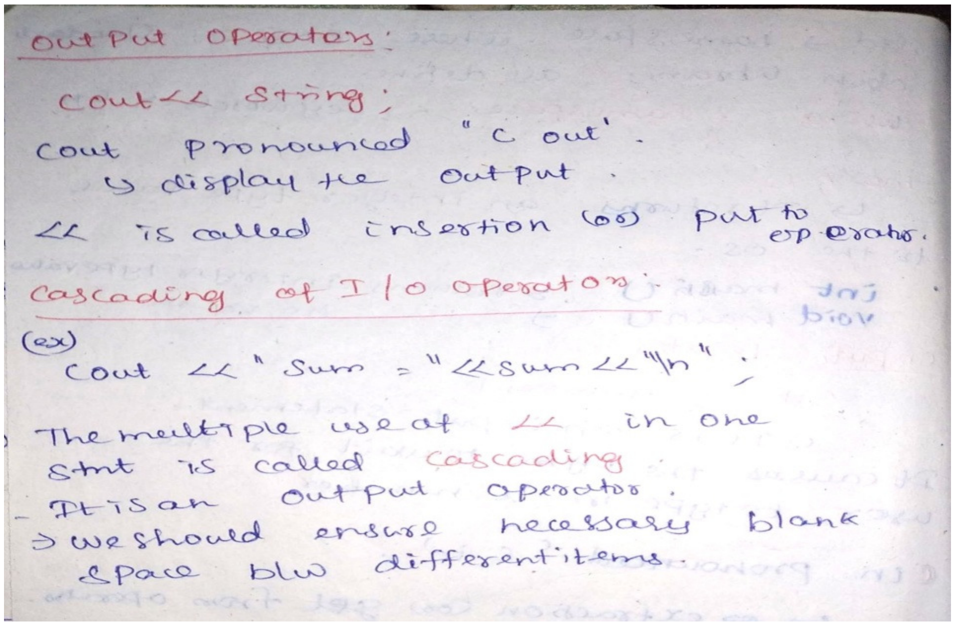
main ()
↳ It returns an integer type value
to the OS.
int main () → returns interger type value
void main () → " no value.

Input operator:

cin >> number1;
↳ It is an input statement.
It causes the pgm to wait for the
user to type in a number

Cin pronounced 'c in'.
>> ~~in~~ extraction (or) get from operator.

Variables:
number, sum, average.


Output Operators:

cout << string;
cout pronounced "c out'.
↳ display the output.
<< is called insertion (or) put to operator.

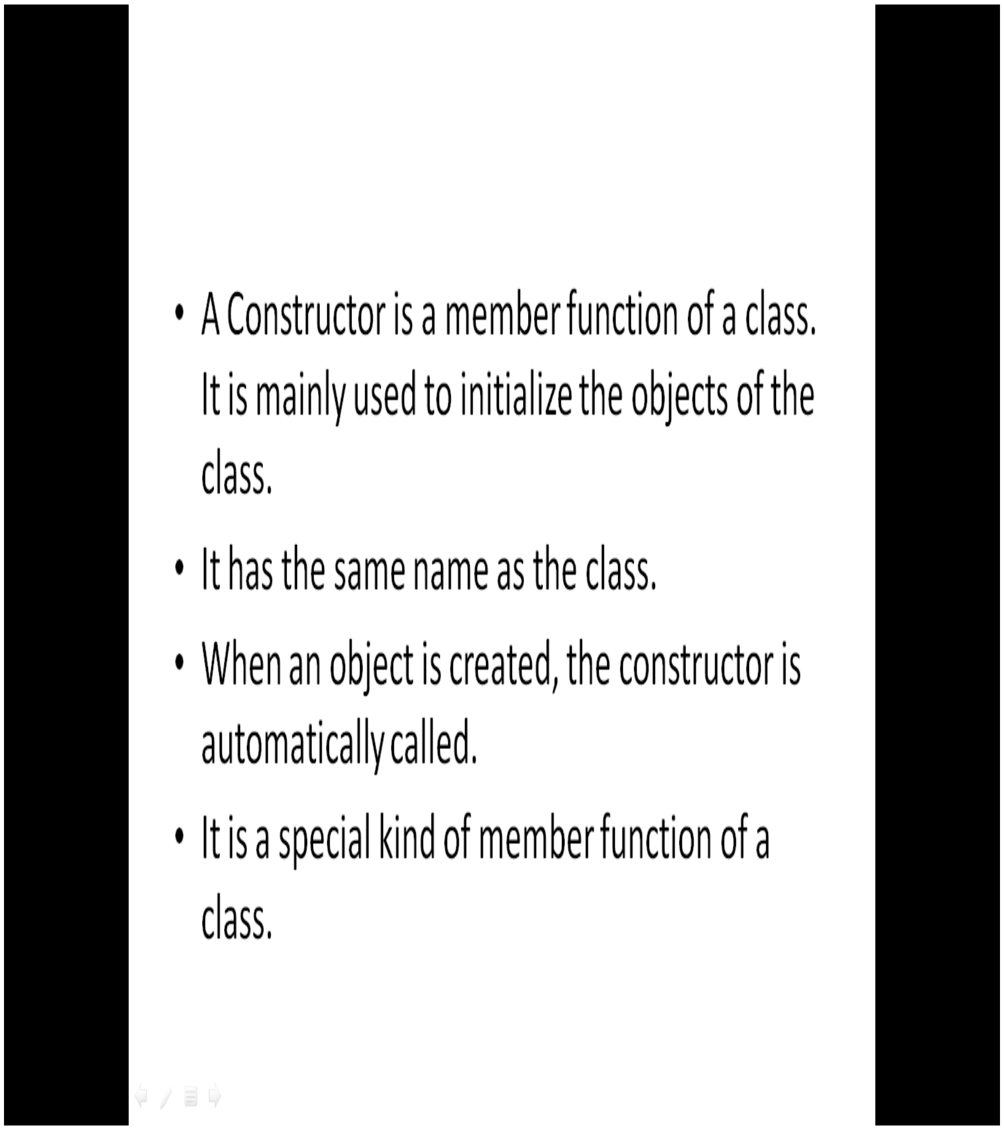
Cascading of I/O operators:
(ex)
cout << " sum = " << sum << "\n";

The multiple use of << in one
stmt is called cascading
It is an output operator.
→ we should ensure necessary blank
space b/w different items.

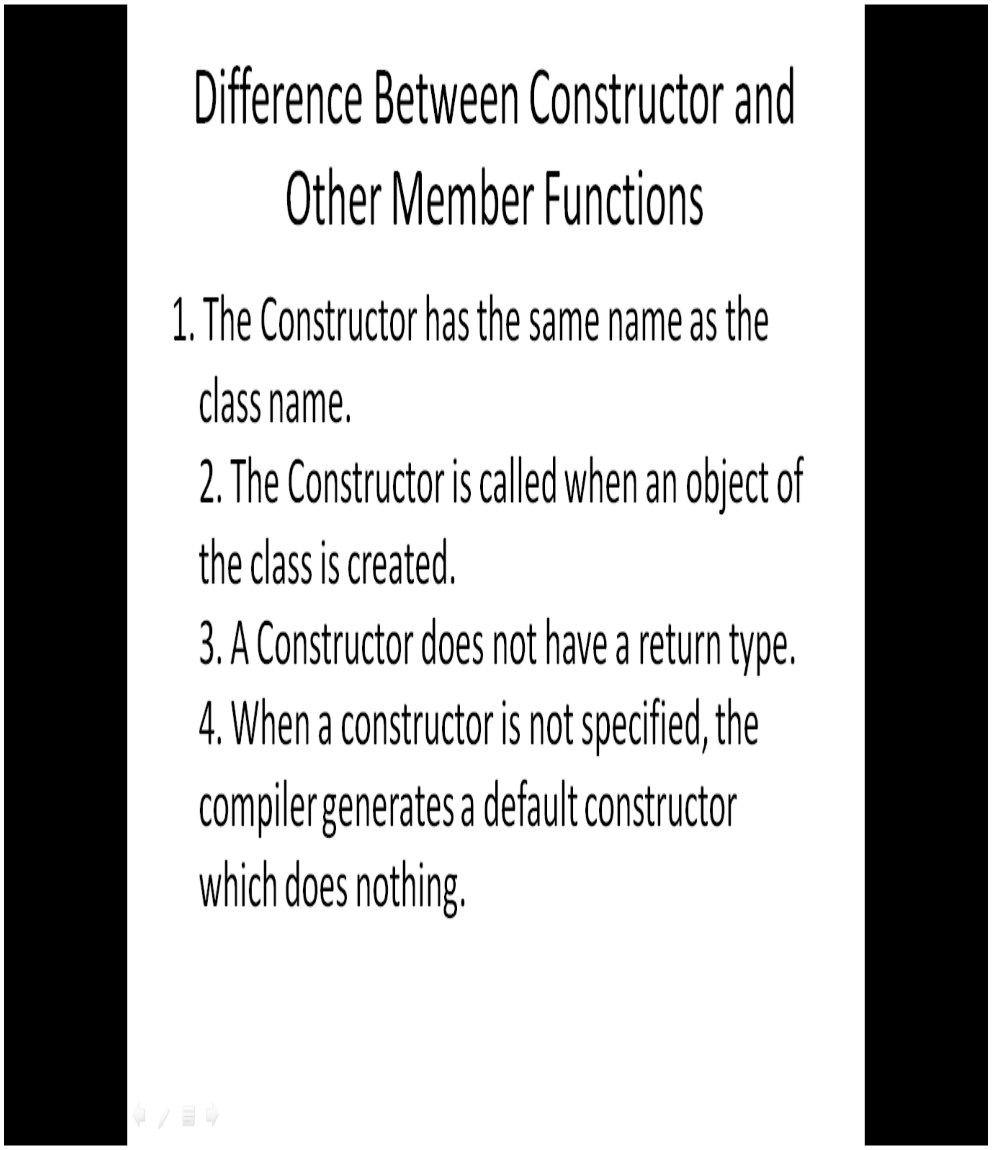
PPT CONSTRUCTOR and DESTRUCTOR

Constructor

- A class constructor is a special member function of a class that is executed whenever we create new objects of that class.
- A constructor will have exact same name as the class.
- it does not have any return type at all, not even void.
- Constructors can be very useful for setting initial values for certain member variables.



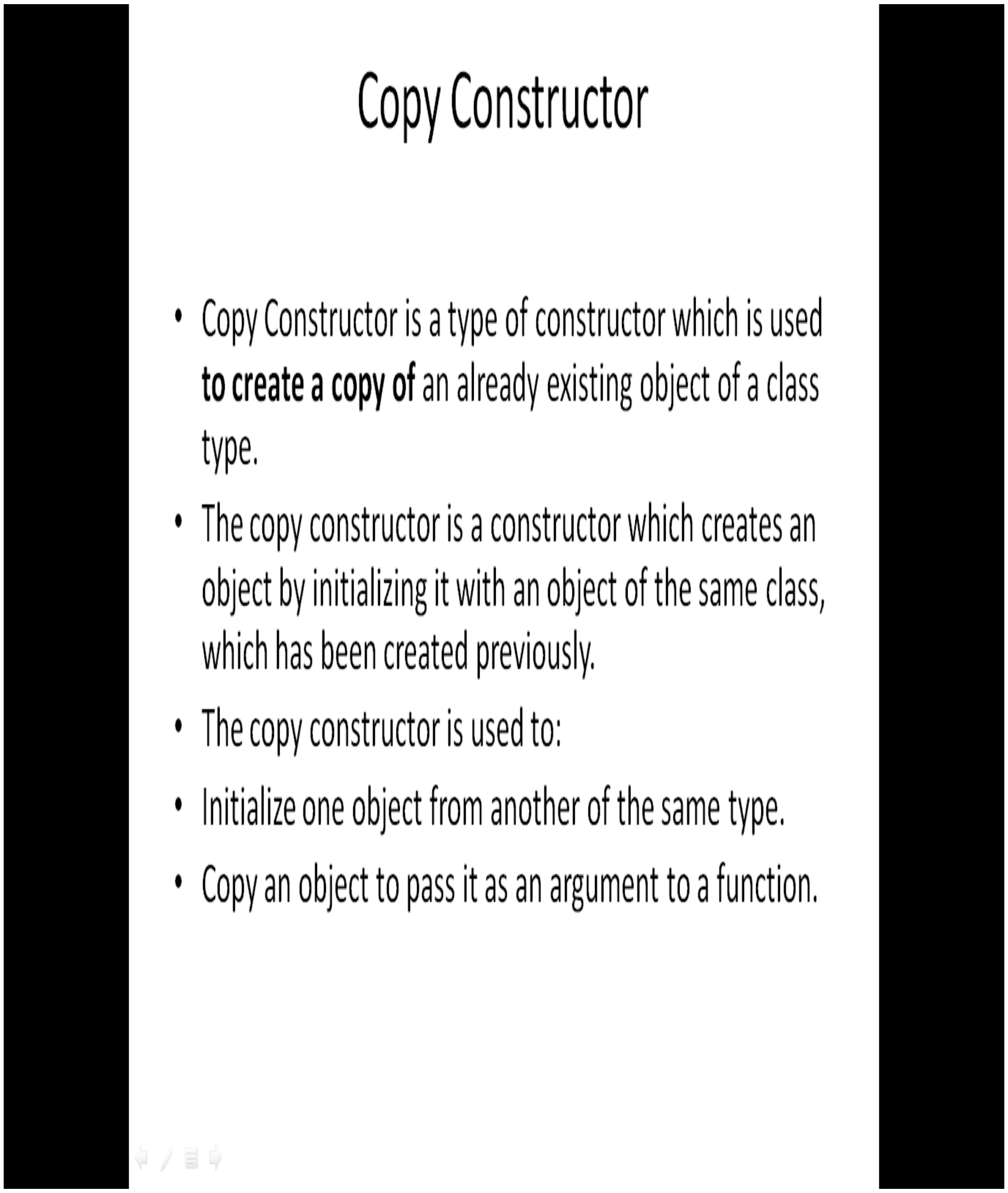
- A Constructor is a member function of a class. It is mainly used to initialize the objects of the class.
- It has the same name as the class.
- When an object is created, the constructor is automatically called.
- It is a special kind of member function of a class.



Difference Between Constructor and Other Member Functions

1. The Constructor has the same name as the class name.
2. The Constructor is called when an object of the class is created.
3. A Constructor does not have a return type.
4. When a constructor is not specified, the compiler generates a default constructor which does nothing.

- *Syntax:*
- **class_name ()**
- **{ Constructor Definition**
- Types of Constructor
  - 1. Copy Constructor
  - 2. Dynamic Constructor



Copy Constructor

- Copy Constructor is a type of constructor which is used **to create a copy of** an already existing object of a class type.
- The copy constructor is a constructor which creates an object by initializing it with an object of the same class, which has been created previously.
- The copy constructor is used to:
- Initialize one object from another of the same type.
- Copy an object to pass it as an argument to a function.

- Copy an object to return it from a function.
- If a copy constructor is not defined in a class, the compiler itself defines one.
- the class has pointer variables and has some dynamic memory allocations, then it is a must to have a copy constructor.
- it is usually of the form **X (X&)**, where X is the class name.
- The compiler provides a default Copy Constructor to all the classes.

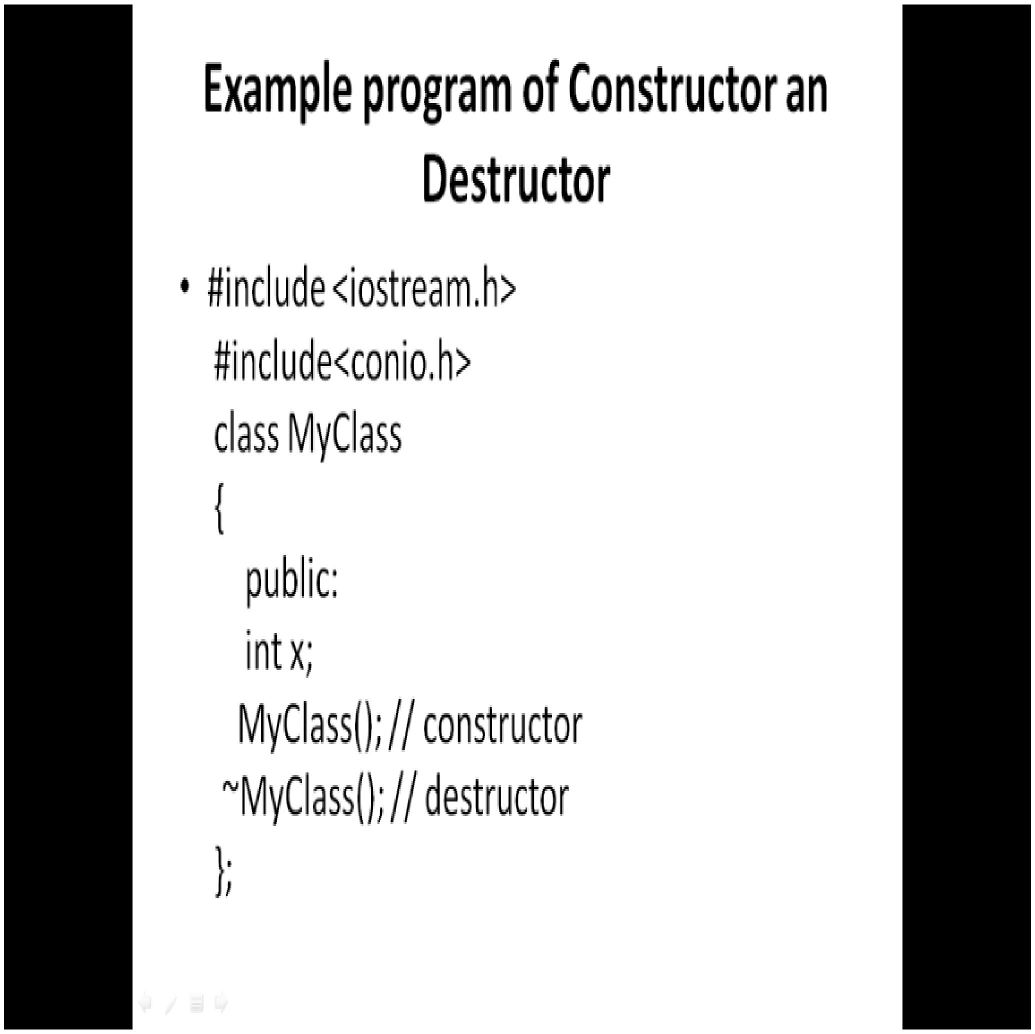
Dynamic Constructor

- Dynamic constructor is used to **allocate the memory** to the objects at the run time.
- Memory is allocated at run time with the help of 'new' operator.
- By using this constructor, we can dynamically initialize the objects.

Destructor

- **Destructor** is the opposite of **constructor**.
- **Destructor** is used to destroy the objects created by **constructor**
- Destructors are another type of member function that is responsible for destroying or deleting the object.

- It frees up space occupied by the object after it is no longer needed.
- A Destructor is called automatically when the object is out of scope and no longer needed.
- A Destructor has the name same as the class name, but the only difference is that the name is preceded by a tile(~).



Example program of Constructor an Destructor

```
• #include <iostream.h>
  #include<conio.h>
  class MyClass
  {
     public:
     int x;
    MyClass(); // constructor
   ~MyClass(); // destructor
  };
```

•

```
// Implement MyClassconstructor.
        MyClass::MyClass()
     {
             x = 10;
     }
// Implement MyClass destructor.
         MyClass::~MyClass()
     {
         cout<< "Destructing ...\n";
     }
```

-
  ```
  void main()
      {
          clrscr();
          MyClass ob1;
          MyClass ob2;
          cout <<ob1.x<< " " <<ob2.x<<"\n";
          getch();
      }
  ```

  Output:

  ```
  10 10
  ```

Question Bank

1. The first computer language to use a block structure was **ALGOL.**
2. C programming is a **procedure-oriented** language.
3. A **Function** is a subroutine that may include one or more statement.
4. The **Document** section consists of a set of comment line giving the name of the program, author and other details.
5. The **Definition** Section defines all symbolic constants.
6. The **Link** section provides instruction to the compiler to link function from the system library.
7. An **Octal integer** constant consists of any combination of digits from the set 0 through 7, with a leading 0.
8. **Single Character Constant** contains one character within a pair of single quote marks.
9. **'\t'** constant is used to create horizontal tab.
10. **32** Keyword are used in C programming.
11. Floating point number is stored in **32 bits.**
12. Double data type number uses **64 bits.**
13. **Void** data type has no return type values.
14. **Variable**is a data name that may be used to store a data value.
15. **Global**variables are used in more than one function.
16. The **Declaration** part declares all the variables used in the executable part.
17. The **Closing Brace** of the main function section is the logical end of the program.
18. The structure and unions are referred as **user defined** data type.

19. The subprogram section contains all the user defined function that is called in **main** function.
20. Array and function are referred as **derived**data type.

1. A built in multiway decision statement used in C language is known as **Switch.**
2. The body of the loop is executed at least once when **do..while** statement is used.
3. **Break** statement at the end of each block signal the end of a particular case and causes an exit from the switch statement.
4. The While statement is an **Entry-Controlled Loop** statement.
5. The Do..While statement is an **Exit controlled loop** statement.
6. **Go to** statement helps to branch unconditionally from one point to another point in the program.
7. **If Statement** is basically a two-way decision statement.
8. **Array** is a collection of similar data type in which each element is located in special memory allocation.
9. Single variable name with single subscript value is called **One-dimensional array.**

10 The array that forms more than three dimensions is called **Multi-dimensional array.**

11 A String is a sequence of characters that is treated as **a single data item**.

12 **Double Quotation** is used to define group of characters.

13 In array individual value is referred as **element.**

14 The header file that contains many string manipulation functions is **<string.h>**

15 **strcat()** function is used to concatenates two strings.

16 **Strcmp()** function is used to compare two strings.

17 **Strcpy()** function is used to copy one string over another.

18 **Strlen()**function is used to find the length of a string.

19 The header file used for standard I/O functions is**<stdio.h>.**

20 **Strings** cannot be manipulated with operators.

1. **Objects** are the basic run-time entities an object-oriented system.
2. **Class** is a collection of object.
3. The wrapping up of data and function into a single unit is called **Encapsulation.**
4. The act of representing essential features without including the background details is referred as **Data Abstraction.**
5. Attributes are sometimes called **Data Member**.
6. Data are sometimes called **Member Function**.
7. Inheritance provides the idea of **Reusability**.
8. New classes derived from old classes are called **Inheritance.**
9. A single function name to perform different types of tasks is known as **Function Overloading.**
10. **Polymorphism** is extensively used in implementing inheritance.
11. **Message Passing** involves specifying the name of the object, the name of the function and the information to be sent.
12. Encapsulation and object identity primarily supports **Object-Based programming.**
13. **<string.h>**header file is used to string processing function.
14. **<iostream.h>**header file is used for standard input and standard output function.
15. The **<<**operator is called insertion or put to operator.
16. The**>>**operator is called extraction or get from operator.

17. The smallest individual units in a program are called **Tokens**.
18. The declared keyword can't be used as a **Variable Name**.
19. A **Constant** refers to fixed values that do not change during the execution of a program.
20. Fundamental data types are used to support all **Built-In** data type.

1 **::**operator refers to scope resolution operator.
2 **::***operator refers to pointer-to- member declarator.
3 **->*** operator refers to pointer-to- member operator.
4 **Endl** operator refers to line feed operator.
5 **New** operator refers to memory release operator.
6 **Setw** operator refers to field width operator.
7 **Delete** operator refers to memory release operator.
8 **Manipulators** operators are used to format the data display.
9 Relational Expression is known as **Boolean expression.**
10 **Logical** expression combines two or more relational expression.
11 Bitwise expressions are used to manipulate data at **bit level.**
12 **Called function** creates a new set of variables and copies the values of arguments.
13 **Function prototype** is a declaration statement in calling program.
14 Char data type holds the values in **one** bytes.
15 Int data type holds the values in **two** bytes.
16 Float data type holds the values in **four** bytes.
17 A **minus** operator when used as a unary, take just one operand.
18 The **friend** function will have only one argument for unary operators.
19 The friend function will have **two arguments** for binary operators.
A **structure** is a convenient tool for handling a group of logically related data items.

1. A **constructor** is a 'special' member function whose task is to initialize the objects of its class.
2. The constructor name is the same as the **class** name.
3. The constructors that can take arguments are called **parameterized** constructors.
4. The constructor functions should be declared in the **public** section
5. A **copy** constructor is used to declare and initialize an object from another object.
6. The **constructor** is used to allocate memory while creating objects.
7. A **destructor** is used to destroy the objects.
8. A destructor never takes any **argument** nor does it return any value.
9. A derived class with only one base class is called **single** inheritance.
10. A derived class with several base classes is called **multiple** inheritance .
11. The mechanism of deriving a class from another derived class is known as **multilevel** inheritance.
12. The combination of multilevel and multiple inheritance is called **hybrid inheritance**.
13. The old class is referred as the **base** class.
14. The new class is referred as the **derived** class.
15. The virtual function has **destructor**.
16. A **file** is a collection of related data stored in a particular area on the disk.
17. The filename is used to **initialize** the file stream object.
18. The **if stream** provides support for input operations.
19. The **of stream** provides support for output operations.
20. The **end-of-file** condition is necessary for preventing any further attempt to read data from the file.