Subject :- Object Oriented Programming

First Lecture Activity what you should know before beginning of the course?

Course Name:-	Object Oriented Programming
Course Code:-	210243
Syllabus:-	Click here
Course Objective:	Click here
Course Outcomes:-	Click here
Google classroom Details:-	[hcpmf7q]- 2021-22 [homps27]- (2020-21) Last Year
Blog:-	Click Here

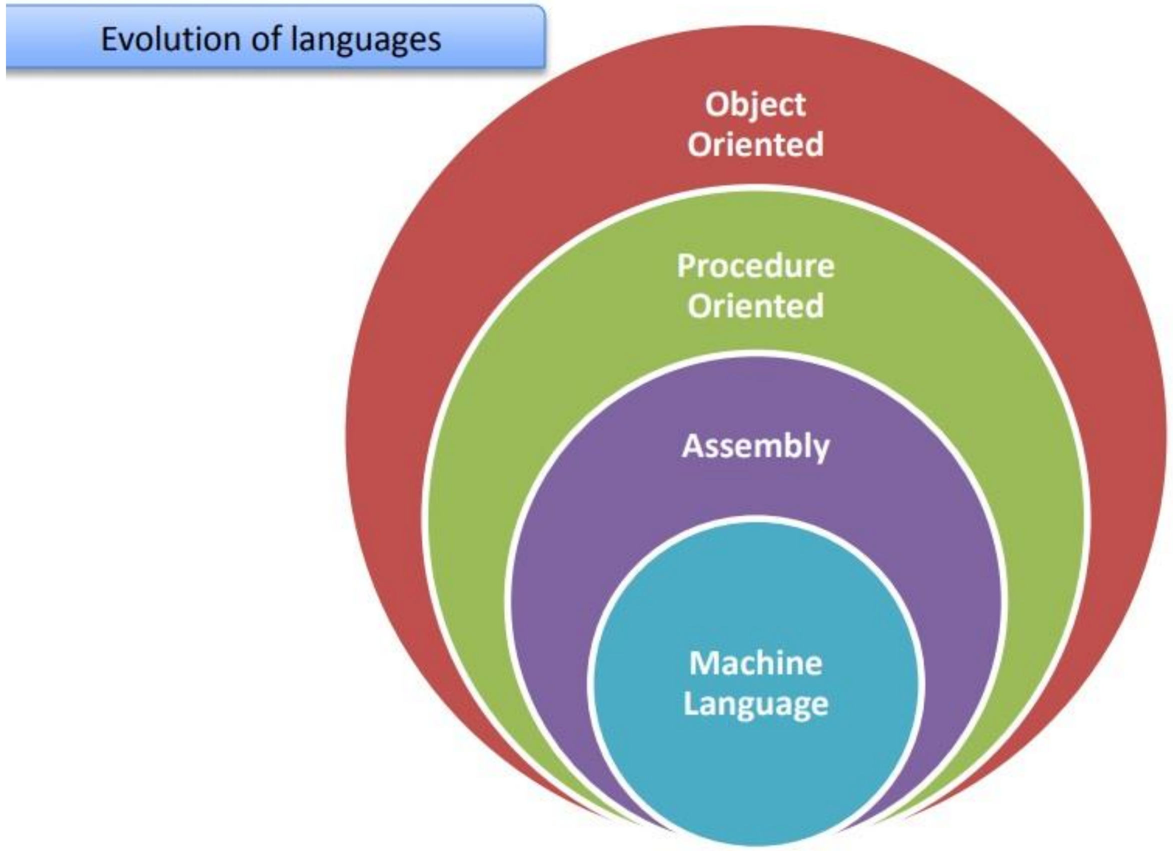
Subject :- Object Oriented Programming

Unit 1

Why Programming?

Subject :- Object Oriented Programming

Unit 1



Subject :- Object Oriented Programming

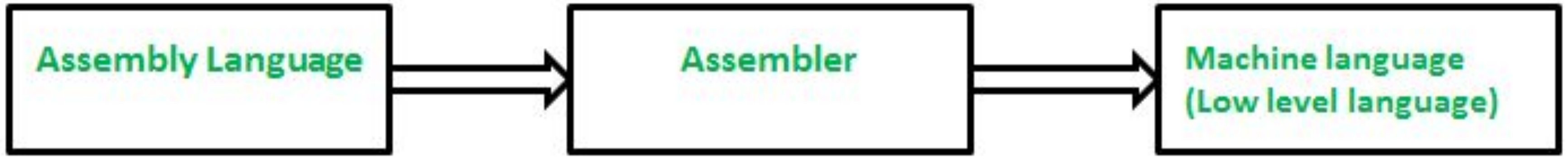
Unit 1

Machine language is the low level programming language. Machine language can only be represented by 0s and 1s. In earlier when we have to create a picture or show data on the screen of the computer then it is very difficult to draw using only binary digits(0s and 1s). For example: To write 120 in the computer system its representation is 1111000. So it is very difficult to learn. To overcome this problem the assembly language is invented.

Assembly language is the more than low level and less than high-level language so it is intermediary language. Assembly languages use numbers, symbols, and abbreviations instead of 0s and 1s.For example: For addition, subtraction and multiplications it uses symbols likes Add, sub and Mul, etc.

Subject :- Object Oriented Programming

Unit 1



Subject :- Object Oriented Programming

Unit 1

Procedure Oriented Programming:-

- Programming in the high-level languages such as COBOL, FORTRAN, C, etc. is known as procedure-oriented programming.
- Procedure-oriented programming basically contains **group of instructions known as function**. There are multiple functions into the program.

Subject :- Object Oriented Programming

Unit 1

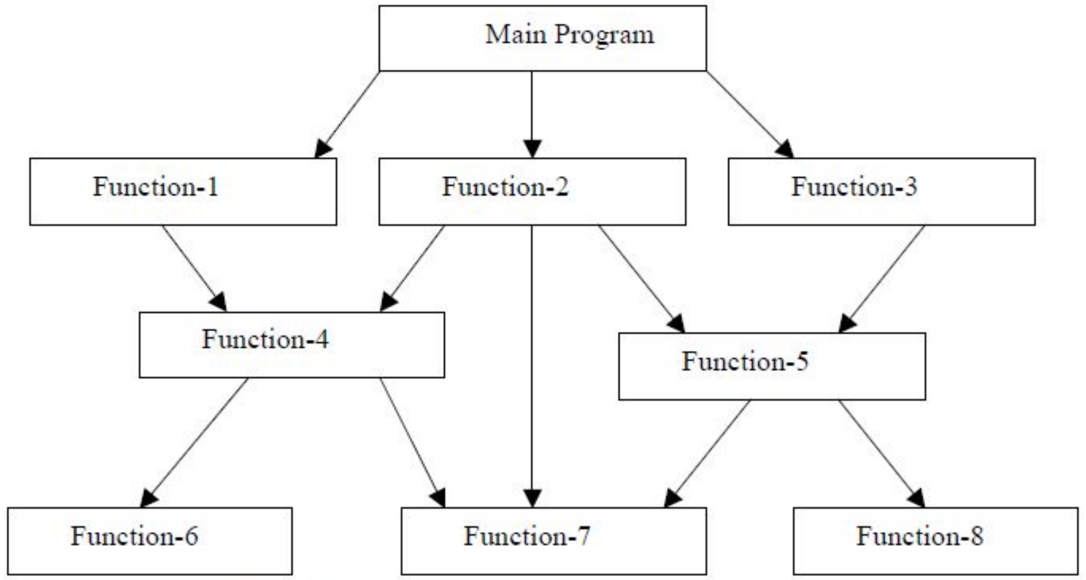
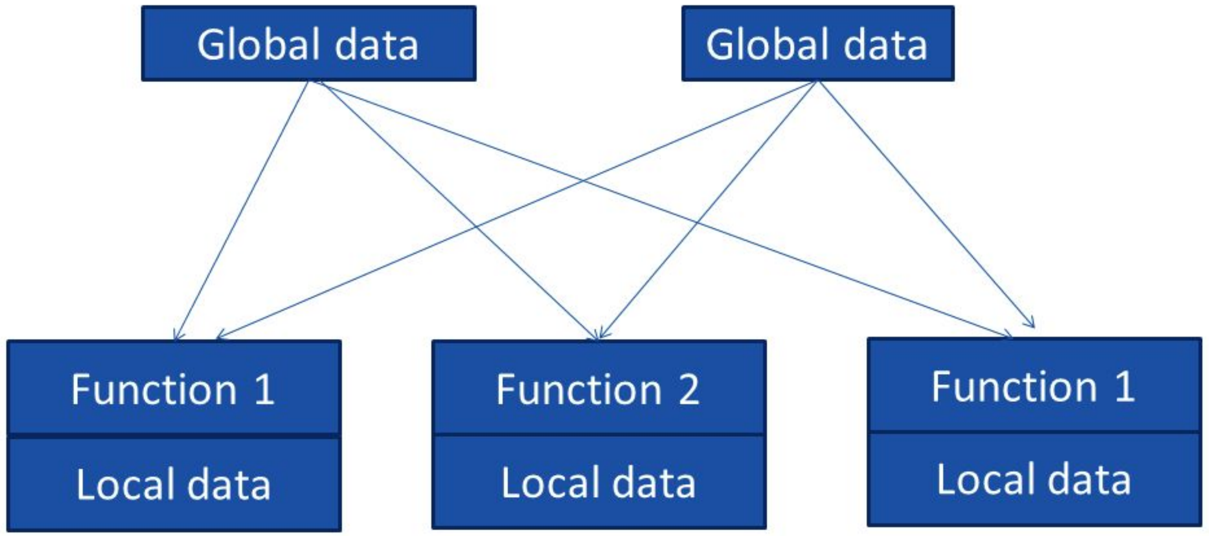


Fig. 1.2 Typical structure of procedural oriented programs



Subject :- Object Oriented Programming

Unit 1

Characteristics of procedure-oriented programming language:

1. It emphasis on algorithm (doing this).
2. Large programs are divided into smaller programs known as functions.
3. Function can communicate by global variable.
4. Data move freely from one function to another function.
5. Functions change the value of data at any time from any place. (Functions transform data from one form to another.)
6. It uses top-down programming approach.

Subject :- Object Oriented Programming

Unit 1

Modular Programming:-

Modular programming emphasis on breaking of large programs into small problems to increase the maintainability, readability of the code and to make the program handy to make any changes in future or to correct the errors.

Subject :- Object Oriented Programming

Unit 1

Generic Programming:-

Generic programming is one popular type of **computer programming** written in such a way that it creates the most efficient code possible while allowing the code to apply to as many situations as possible without requiring any changes to the original code itself.

Once the code is written, it can only perform the exact functions it was written for. By using generic programming to create codes that work in a number of different situations, while still performing the same basic, overall function, programmers can use a single piece of code in different programs without ever making changes to the original.

Subject :- Object Oriented Programming

Unit 1

Object Oriented Programming(OOP) :-

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.

The Main Focus is on object not on the Functions

Subject :- Object Oriented Programming

Unit 1

Need of OOP:-

POP	OOP
In POP, program is divided into small parts called functions.	In OOP, program is divided into parts called objects.
POP does not have any access specifier	OOP has access specifiers named Public, Private, Protected, etc
POP does not have any proper way for hiding data so it is less secure.	OOP provides Data Hiding so provides more security.
In POP, Overloading is not possible.	In OOP, overloading is possible in the form of Function Overloading and Operator Overloading
In POP, Most function uses Global data for sharing that can be accessed freely from function to function in the system.	In OOP, data can not move easily from function to function, it can be kept public or private so we can control the access of data.
Example of POP are : C, VB, FORTRAN, Pascal.	Example of OOP are : C++, JAVA, VB.NET, C#.NET

Subject :- Object Oriented Programming

Unit 1

Fundamentals of Object Oriented Programming:-

1. Namespaces:-

Namespace is a feature added in C++ and not present in C. A namespace is **a declarative region that provides a scope to the identifiers** (names of the types, function, variables etc) inside it. Multiple namespace blocks with the same name are allowed.

Ex. Same name in college classroom

Subject :- Object Oriented Programming

Fundamentals of Object Oriented Programming:-nit 1

```
#include <iostream>
using namespace std;

// first name space
namespace first_space {
   void func() {
      cout << "Inside first_space" << endl;
   }
}

// second name space
namespace second_space {
   void func() {
      cout << "Inside second_space" << endl;
   }
}

int main () {
   // Calls function from first name space.
   first_space::func();

   // Calls function from second name space.
   second_space::func();

   return 0;
}
```

Subject :- Object Oriented Programming

Unit 1

Fundamentals of Object Oriented Programming:-

2. Class & Objects:-

Everything in C++ is associated with classes and objects, along with its attributes and methods. For example: in real life, a car is an object. The car has attributes, such as weight and color, and methods, such as drive and brake.

- **Attributes and methods are basically variables and functions that belongs to the class. These are often referred to as "class members".**
- **A class is a user-defined data type that we can use in our program, and it works as an object constructor, or a "blueprint" for creating objects.**

Subject :- Object Oriented Programming

Fundamentals of Object Oriented Programming:-nit 1

Creating a Class

```
class MyClass {         // The class
  public:               // Access specifier
    int myNum;          // Attribute (int variable)

};
```

Creating a Object

```
class MyClass {         // The class
  public:               // Access specifier
    int myNum;          // Attribute (int variable)

};

int main() {
  MyClass myObj;  // Create an object of MyClass

  // Access attributes and set values
  myObj.myNum = 25;
   // Print attribute values
  cout << myObj.myNum<<"\n";
    return 0;
}
```

Subject :- Object Oriented Programming

Unit 1

Fundamentals of Object Oriented Programming:-

3. Data members and Member functions in C++

The variables which are declared in any class by using any fundamental data types (**like int, char, float etc**) or derived data type (**like class, structure, pointer etc.**) are known as **Data Members**. And the functions which are declared either in private section or public section are known as **Member functions**.

There are three **types of data members/member functions in C++**:

1. Private members
2. Public members
3. Protected Members (Inheritance)

Fundamentals of Object Oriented Programming:-

1) Private members

The members which are declared in private section of the class (using private access modifier) are known as private members. Private members can also be accessible within the same class in which they are declared.

2) Public members

The members which are declared in public section of the class (using public access modifier) are known as public members. Public members can access within the class and outside of the class by using the object name of the class in which they are declared.

3) Protected members

Normally used in Inheritance for intermediate level of security

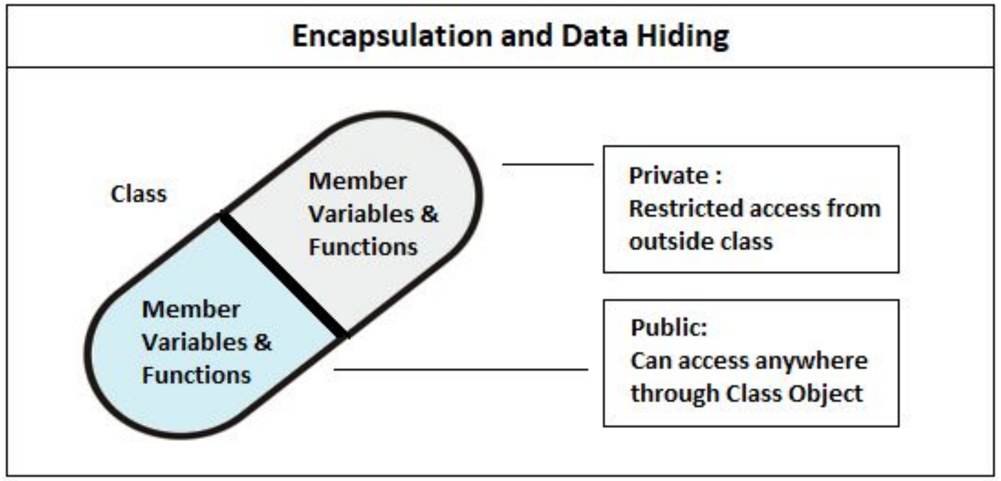
Subject :- Object Oriented Programming

Unit 1

Features of OOP:-

1. Data Encapsulation

Binding (or wrapping) code and data together into a single unit is known as encapsulation. For example: capsule, it is wrapped with different medicines.


Encapsulation and Data Hiding
Class
Member Variables & Functions
Member Variables & Functions
Private : Restricted access from outside class
Public: Can access anywhere through Class Object

Subject :- Object Oriented Programming

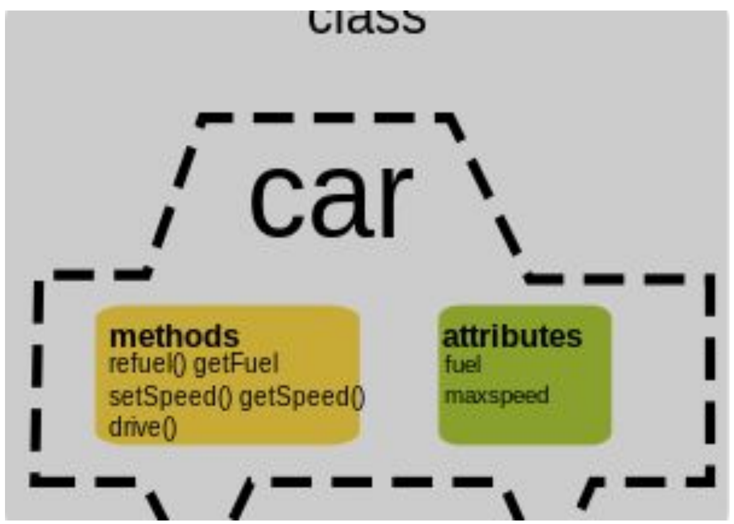
Unit 1

Features of OOP:-

2. Data Abstraction

Hiding internal details and showing functionality is known as abstraction. For example: phone call, we don't know the internal processing.

In C++, we use abstract class and interface to achieve abstraction.



Subject :- Object Oriented Programming

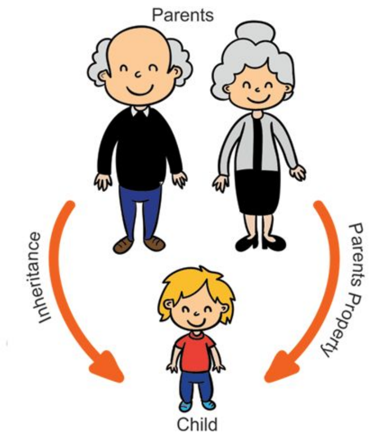
Unit 1

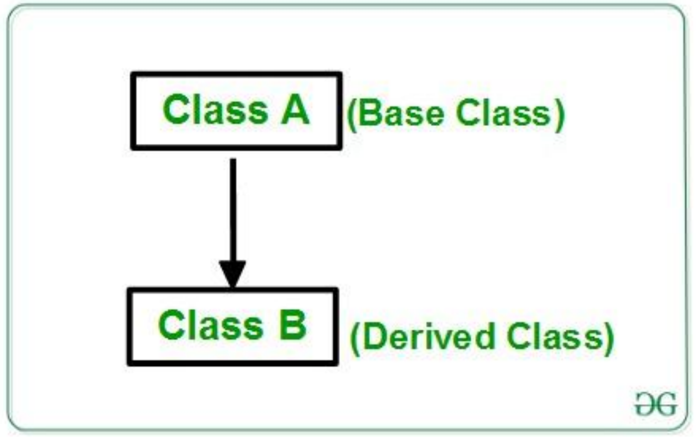
Features of OOP:-

3. Inheritance

When one object acquires all the properties and behaviours of parent object i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

.





Subject :- Object Oriented Programming

Unit 1

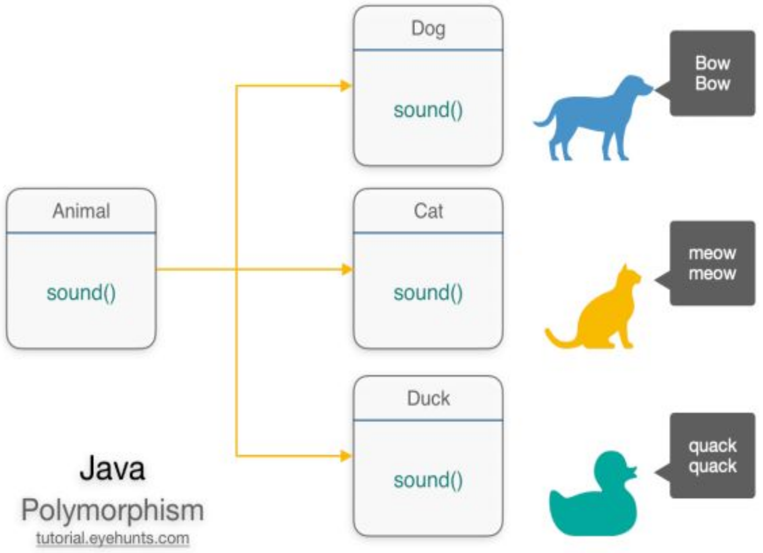
Features of OOP:-

4. Polymorphism

When **one task is performed by different ways** i.e. known as polymorphism. For example: to convince the customer differently, to draw something e.g. shape or rectangle etc.

In C++, we use Function overloading and Function overriding to achieve polymorphism

.

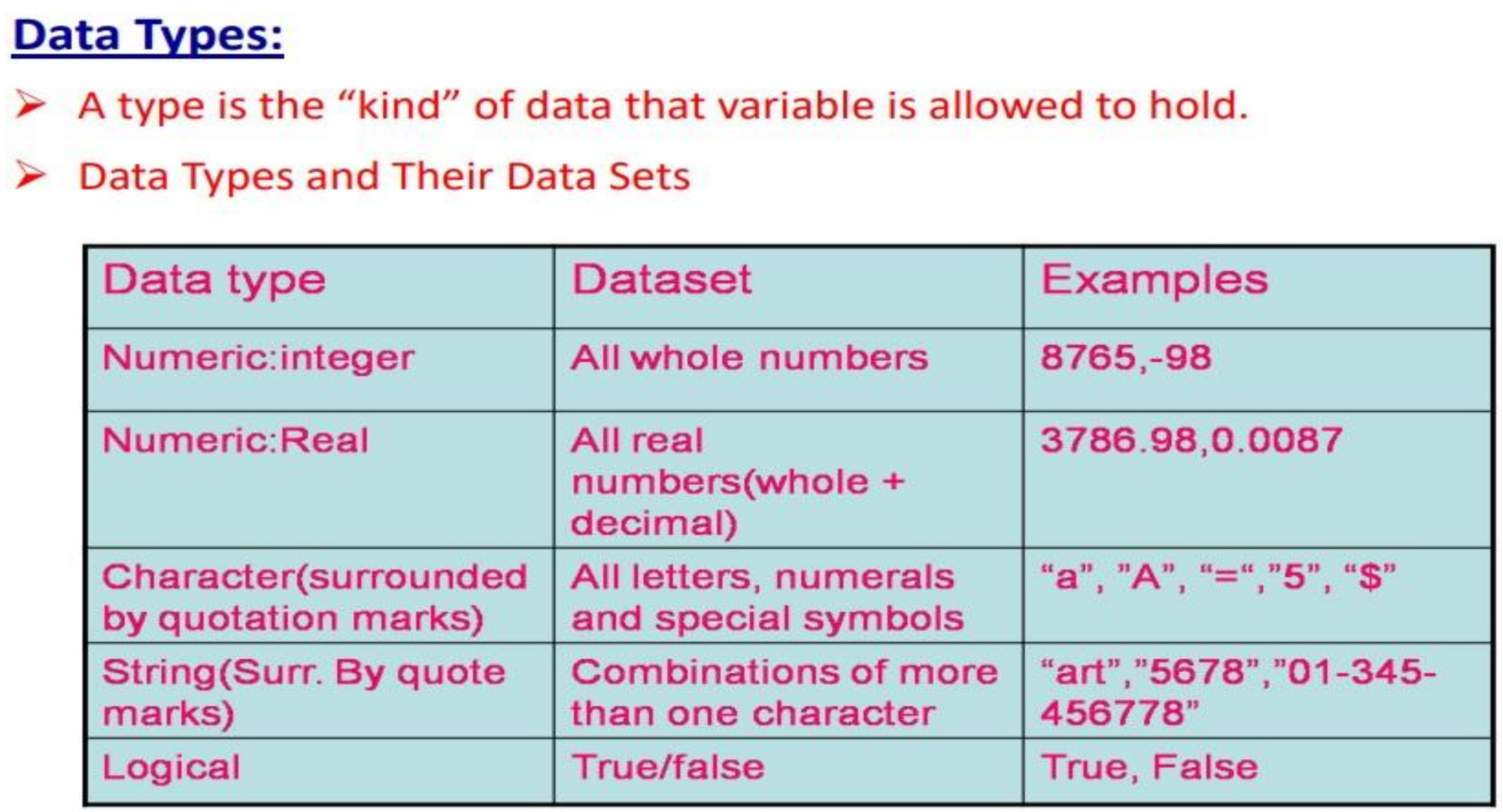


Subject :- Object Oriented Programming

Unit 1

CPP Programming :-

1. DataType
2. Strctures
3. Enumerations
4. Control Structures
5. Arrays and Strings
6. Access Specifiers
7. separating interfaces and implementation



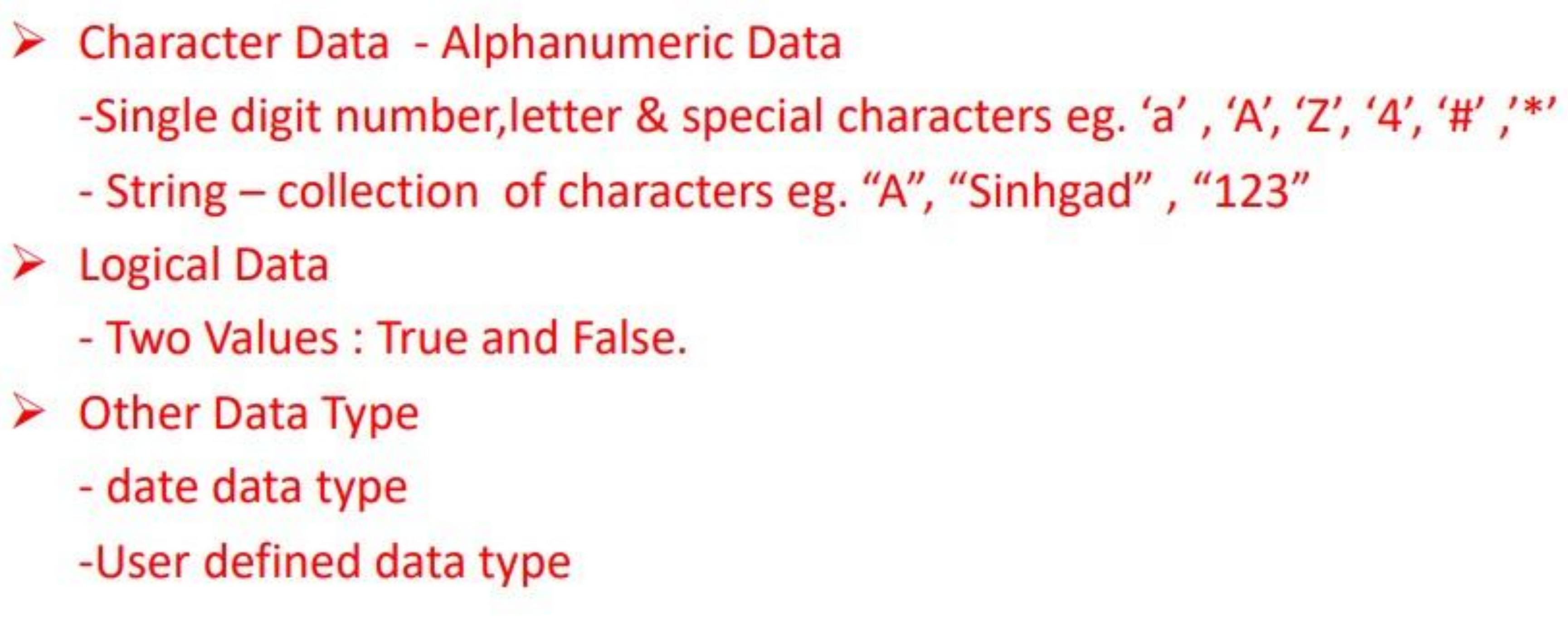
Data Types:

- A type is the “kind” of data that variable is allowed to hold.
- Data Types and Their Data Sets

Data type	Dataset	Examples
Numeric:integer	All whole numbers	8765,-98
Numeric:Real	All real numbers(whole + decimal)	3786.98,0.0087
Character(surrounded by quotation marks)	All letters, numerals and special symbols	“a”, ”A”, “=“,”5”, “$”
String(Surr. By quote marks)	Combinations of more than one character	“art”,”5678”,”01-345-456778”
Logical	True/false	True, False

Subject :- Object Oriented Programming

Unit 1



- Character Data - Alphanumeric Data

  -Single digit number,letter & special characters eg. ‘a’ , ‘A’, ‘Z’, ‘4’, ‘#’ ,’*’

  - String – collection of characters eg. “A”, “Sinhgad” , “123”
- Logical Data

  - Two Values : True and False.
- Other Data Type

  - date data type

  -User defined data type

Subject :- Object Oriented Programming

Unit 1

CPP Programming :-

2. Control Structures

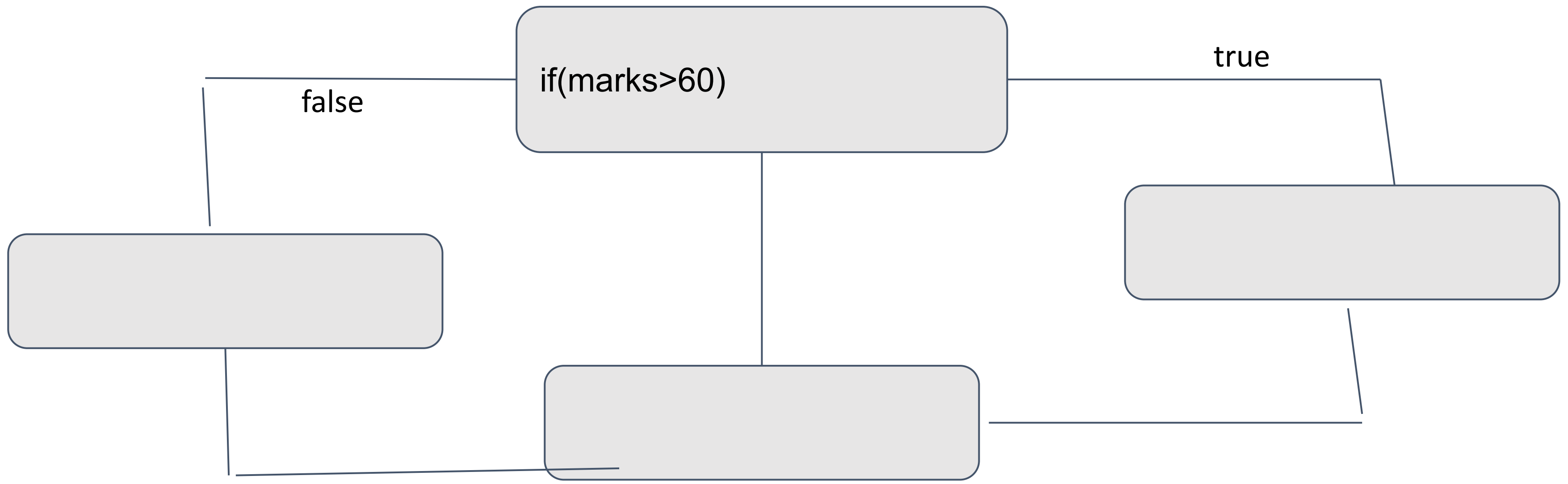
Control Structure Normally, a program is executed in a sequential manner.However, in many cases, a program has to choose among alternative statements,C++ provides constructs that enable the programmer to select which statement in the program to be executed next. This is called transfer of control.

following are Control structures-

1. **if**
2. **if ..else**
3. **nested if-else**
4. **switch**

Subject :- Object Oriented Programming

Unit 1



C programming has three types of loops:

- for loop
- while loop
- do...while loop

for Loop
The syntax of the for loop is:

- for (initialization Statement; test Expression; update Statement)
- { // statements inside the body of loop }

How for loop works?

- The initialization statement is executed only once.
- Then, the test expression is evaluated. If the test expression is evaluated to false, the for loop is terminated.
- However, if the test expression is evaluated to true, statements inside the body of the for loop are executed, and the update expression is updated.
- Again the test expression is evaluated.
- This process goes on until the test expression is false. When the test expression is false, the loop terminates.

- Print numbers from 1 to 10
- #include <stdio.h>
- int main()
- {
- int i;
- for (i = 1; i < 11; ++i)
- { printf("%d ", i);
- }
- return 0;
- }

while loop

- The syntax of the while loop is:
- while (test Expression)
- { // the body of the loop }

- **How while loop works?**
- The while loop evaluates the test Expression inside the parentheses ().
- If test Expression is **true**, statements inside the body of while loop are executed. Then, test Expression is evaluated again.
- The process goes on until test Expression is evaluated to **false**.
- If test Expression is **false**, the loop terminates (ends).

- Print numbers from 1 to 5
- #include <stdio.h>
- int main()
- {
- int i = 1;
- while (i <= 5)
- {
- printf("%d\n", i);
- ++i;
- }
- return 0;
- }

do...while loop

- The do..while loop is similar to the while loop with one important difference. The body of do...while loop is executed at least once. Only then, the test expression is evaluated.
- The syntax of the do...while loop is:
- do
- { // the body of the loop }
- while (test Expression);

How do...while loop works?

- The body of do...while loop is executed once. Only then, the testExpression is evaluated.
- If testExpression is **true**, the body of the loop is executed again and testExpression is evaluated once more.
- This process goes on until testExpression becomes **false**.
- If testExpression is **false**, the loop ends.

- Program to add numbers until the user enters zero
- #include <stdio.h>
- int main()
- { double number, sum = 0; // the body of the loop is executed at least once
- do {
- printf("Enter a number: ");
- scanf("%lf", &number);
- sum += number;
- } while(number != 0.0);
- printf("Sum = %.2lf",sum);
- return 0;
- }

output

- Enter a number: 1.5
- Enter a number: 2.4
- Enter a number: -3.4
- Enter a number: 4.2
- Enter a number: 0
- Sum = 4.70

3. Arrays

1.Array is a collection of same data types.

2. Collection of items stored at contiguous memory locations and elements can be accessed randomly using indices of an array

3. data type must be the same for all elements.

There are two types of arrays:

- One Dimensional Array
- Multi Dimensional Array

Subject :- Object Oriented Programming

Unit 1

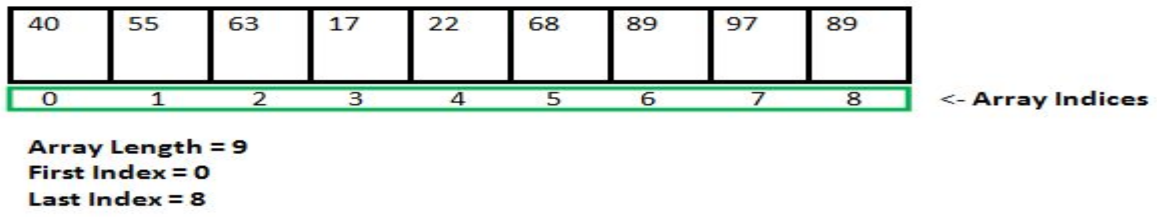
One Dimensional Array: A one dimensional array is a collection of same data types. 1-D array is declared as:

data_type variable_name[size]

data_type is the type of array, like int, float, char, etc.

variable_name is the name of the array.

size is the length of the array which is fixed.   Note: The location of the array elements depends upon the data type we use.



Subject :- Object Oriented Programming

Unit 1

Multi Dimensional Array:-

A multidimensional array is also known as array of arrays. Generally, we use a two-dimensional array.

It is also known as the matrix. We use two indices to traverse the rows and columns of the 2D array. It is declared as:

```
data_type variable_name[N][M]
```

data_type is the type of array, like int, float, char, etc.

variable_name is the name of the array.

N is the number of rows.

M is the number of columns.

Subject :- Object Oriented Programming

Unit 1

Strings:-

- This string is actually a one-dimensional array of characters which is terminated by a null character '\0'.
- Thus a null-terminated string contains the characters that comprise the string followed by a null.
- The following declaration and initialization create a string consisting of the word "Hello". To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word "Hello."

```
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

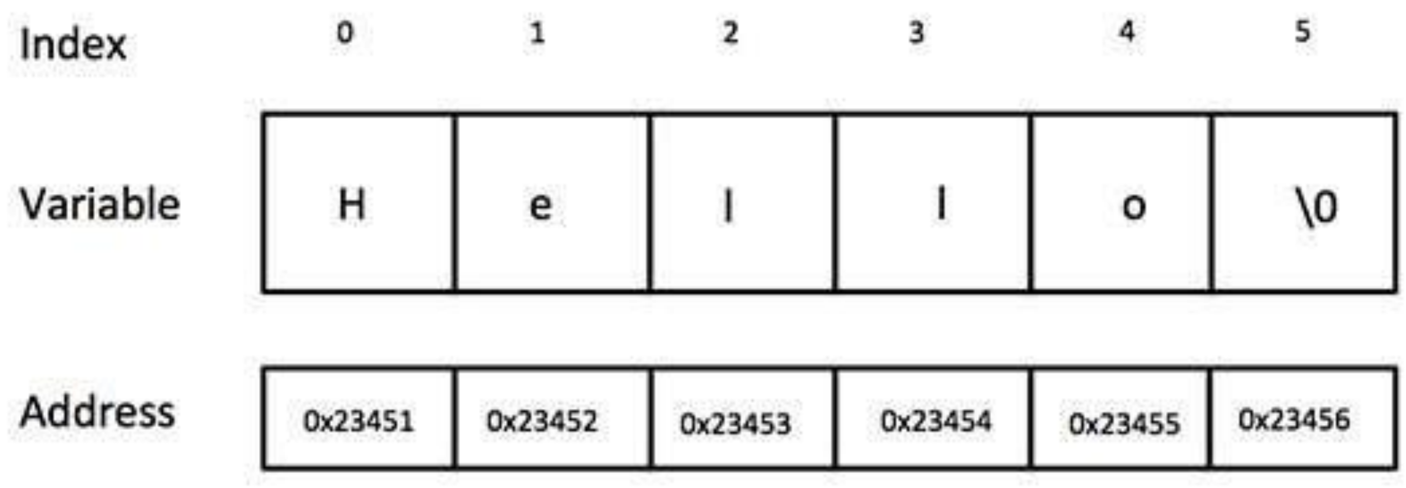
If you follow the rule of array initialization, then you can write the above statement as follows −

```
char greeting[] = "Hello";
```

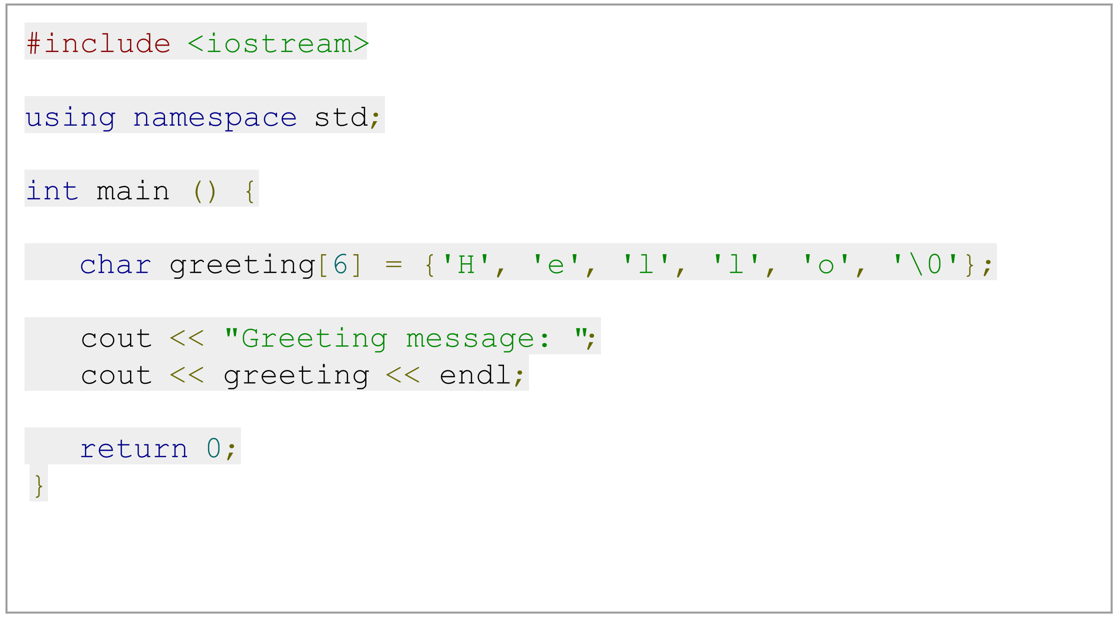
Subject :- Object Oriented Programming

Unit 1

Following is the memory presentation of above defined string in C/C++ −



Actually, you do not place the null character at the end of a string constant.
The C++ compiler automatically places the '\0' at the end of the string when it initializes the array. Let us try to print above-mentioned string –



```
#include <iostream>

using namespace std;

int main () {

   char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};

   cout << "Greeting message: ";
   cout << greeting << endl;

   return 0;
}
```

Subject :- Object Oriented Programming

Unit 1

Function Name	**Use**
strcpy(s1, s2);	Copies string s2 into string s1
strcat(s1, s2);	Concatenates string s2 onto the end of string s1. Ex. s1=”ABC” and s2=”PQR” after contact - ABCPQR
strlen(s1);	Returns the length of string s1. Ex. s1=”SNJB” length=4
strcmp(s1, s2);	Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2

Functions:-

A function is a group of statements that together perform a task. Every C++ program has at least one function, **which is main()**, and all the most trivial programs can define additional functions.

You can divide up your code into separate functions. How you divide up your code among different functions is up to you, but logically the division usually is such that each function performs a specific task.

Defining a Function

The general form of a C++ function definition is as follows −

```
return_type function_name( parameter list ) {
   body of the function
}
```

Examples of Function:-

Subject :- Object Oriented Programming
Unit 1

1. Inside of class

```
#include<iostream.h>
using namespace std;
class Demo
{
public:
            void getdata()
            {
           cout<<"\n i am in getdata";
            }
};
int main()
{
Demo obj;
obj.getdata();
return 0;
}
```

2. Outside of class

```
#include<iostream.h>
using namespace std;
class Demo
{
public:
            void getdata();
};
void Demo::getdata()
{
 cout<<"\n i am in getdata";
}

int main()
{
Demo obj;
obj.getdata();
return 0;
}
```

Subject :- Object Oriented Programming

Unit 1

Constructors:-

Constructors are special **class functions** which performs initialization of every object. **The Compiler calls the Constructor whenever an object is created**. Constructors initialize values to object members after storage is allocated to the object.

Whereas, Example to create the constructor:-

```
class A
{
    public:
    int x;
    // constructor
    A()
    {
        // object initialization
    }

    };
```

Subject :- Object Oriented Programming

Unit 1

Types of Constructors in C++

Constructors are of three types:

1. **Default Constructor**
2. **Parameterized Constructor**
3. **Copy COnstructor**

Subject :- Object Oriented Programming

Unit 1

1. Default Constructor:-

- A default constructor is a **0 argument constructor** which contains a no-argument
- To assign default values to the newly created objects is the main responsibility of default constructor.

Note:-

The compiler adds a default constructor to the code only when the programmer writes no constructor in the code.
If the programmer writes any constructor in the code, then the compiler doesn't add any constructor.

2. Parameterized Constructor:-

- The parameterized constructors are the constructors having a **specific number of arguments** to be passed.
- The purpose of a parameterized constructor is to assign user-wanted specific values to the instance variables of different objects.
- A parameterized constructor is written explicitly by a programmer.

Subject :- Object Oriented Programming

Unit 1

3. Copy Constructor:-

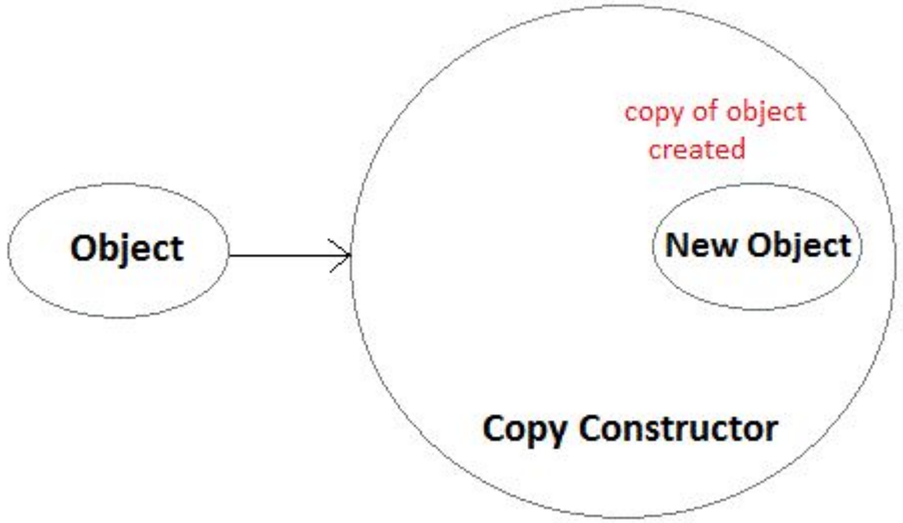
- Copy Constructor is a type of constructor which is used to create a copy of an already existing object of a class type.

```
Classname(const classname & objectname)
{
 . . . .
}
```

As it is used to create an object, hence it is called a constructor. And, it creates a new object, which is exact copy of the existing copy, hence it is called **copy constructor**.

Subject :- Object Oriented Programming

Unit 1



```
SampleCopy(const SampleCopy &obj)
        {
            a=obj.a;
            b=obj.b;
        }


 SampleCopy obj2=obj1;
```

Subject :- Object Oriented Programming

Unit 1

Characteristics of Constructors:-

1. They are invoked automatically when the objects are created.
2. They do not have return (data type) type not even void and therefore they cannot return any values.
3. They can not be inherited, the a derived class can call the base class constructor.
4. They make implicit calls to the operator new and delete when memory allocation is required
5. They should be declared in the public section.

Subject :- Object Oriented Programming

Unit 1

Destructors:-

1. Destructors in C++ are members functions in a class that delete an object.
2. They are called when the class **object goes out of scope** such as when the function ends, the program ends, a delete variable is called etc.
3. Destructors are different from normal member functions as **they don't take any argument** and **don't return anything**. Also, **destructors have the same name as their class and their name is preceded by a tilde(~).**

Syntax of Destructor:-

```
~Classname()
{
}
```

Constructor	**Destructor**
Constructor helps to initialize the object of a class.	Whereas destructor is used to destroy the instances.
It is declared as Classname(arguments if any){Constructor's Body }.	Whereas it is declared as ~ ClassName(no arguments){ };.
Constructor can either accept the arguments or not.	While it can't have any argument.
A constructor is called when the instance or object of the class is created.	It is called while object of the class is freed or deleted.
Constructor is used to allocate the memory to an instance or object.	While it is used to deallocate the memory of an object of a class.
Constructor can be overloaded.	While it can't be overloaded.
The constructor's name is same as the class name.	Here, it's name is also same as the class name preceded by tiled (~) operator.
There is a concept of copy constructor which is used to initialize a object from another object.	While here, there is no copy destructor concept.

Subject :- Object Oriented Programming

Unit 1

Memory Management in CPP

1. **Why we need a memory in CPP?**

   Ex

   int x=10; // reserves 2 bytes of the memory to store value 10

   char a='A'; //reserves 1 bytes of the memory to store value 'A'

Types of Memory Allocation :-

1. **Compile Time / Static Memory Allocation**
2. **Run Time / Dynamic Memory Allocation**

Subject :- Object Oriented Programming

Unit 1

Memory Management in CPP

1. Compile Time / Static Memory Allocation:-

For example, if you need to declare a variable to store marks of `N` students, in that case `N` is undefined at compilation stage. Therefore you will end up with following declaration

```
int marks[100];
```

Above declaration will occupy memory for 100 students (reserved memory bytes will be `100 * sizeof(int)`). Now, there are few issues with above declaration.

1. **Memory wastage**: If you are providing input of 10 students, then `90 * sizeof(int)` memory gets wasted and cannot be used for other purposes. It will be reserved until scope of `marks` variable, constants will be there.
2. **Less flexible**: You cannot alter memory size once allocated. For an array of size 100, at any stage it is not possible again to input marks for 120 students.
3. **No control**: You do not have any control on memory allocation and deallocation. You cannot clear allocated bytes from memory if you don't require at any stage. The C compiler controls memory allocation and deallocation.

Subject :- Object Oriented Programming

Unit 1

Memory Management in CPP

1. Run Time / Dynamic Memory Allocation:-

- C++ allows us to allocate the memory of a variable or an array in run time. This is known as dynamic memory allocation.
- We can allocate and then deallocate memory dynamically using the `new` and `delete` operators respectively.

Dynamically Memory is allocated with Keyword - new

Dynamically Memory is deallocated with Keyword - delete

Subject :- Object Oriented Programming

Unit 1

Memory Management in CPP

1. New Keyword

The new operator allocates memory to a variable. For example,

```
// declare an int pointer
int* pointVar;
// dynamically allocate memory
// using the new keyword
pointVar = new int;
// assign value to allocated memory
*pointVar = 45;  // int *pointVar =new int(10)
```

or

Notice that we have used the pointer `pointVar` to allocate the memory dynamically. This is because the `new` operator returns the address of the memory location.

Subject :- Object Oriented Programming

Unit 1

Memory Management in CPP

2. Delete Keyword

Once we no longer need to use a variable that we have declared dynamically, we can deallocate the memory occupied by the variable. For this, the **delete** operator is used. It returns the memory to the operating system. This is known as memory deallocation.

The syntax for this operator is

```
delete pointerVariable;
```

```
// declare an int pointer
int* pointVar;

// dynamically allocate memory
// for an int variable
pointVar = new int;

// assign value to the variable memory
*pointVar = 45;

// print the value stored in memory
cout << *pointVar; // Output: 45

// deallocate the memory
  delete pointVar;
```

Subject :- Object Oriented Programming

Unit 1

Inline Function

1. The inline functions are a C++ enhancement feature to increase the execution time of a program.
2. Functions can be instructed to compiler to make them inline so that compiler can replace those function definition wherever those are being called.
3. Compiler replaces the definition of inline functions at compile time instead of referring function definition at runtime.

Subject :- Object Oriented Programming

Unit 1

Inline Function

Inline Functions

To create an inline function, we use the `inline` keyword. For example,

```
inline returnType functionName(parameters)
{
   // code
}
```

Notice the use of keyword `inline` before the function definition.

The compiler may not perform inlining in such circumstances as:

- If a function contains a loop. (*for, while and do-while*)
- If a function contains static variables.
- If a function is recursive.
- If a function return type is other than void, and the return statement doesn’t exist in a function body.
- If a function contains a switch or goto statement.

Inline Function

```
##include <iostream>
using namespace std;
inline int cube(int s)
{
return s * s * s;
}


int main()
{
    cout << "The cube of 3 is: " << cube(3) << "\n";
    return 0;
}
```

Output:

```
The cube of 3 is: 27
```

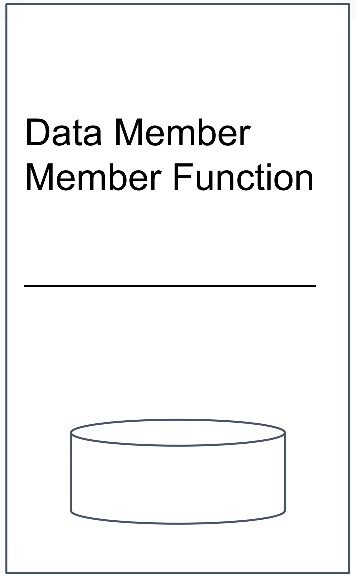
Subject :- Object Oriented Programming

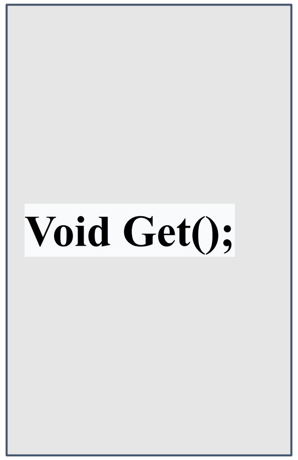
Unit 1

Friend Function

One of the important concepts of OOP is data hiding,
i.e., a nonmember function cannot access an object's private or protected data.

Class:- **Non Member Function:-**





Subject :- Object Oriented Programming

Unit 1

Friend Function

Friend Function

- If a function is defined as a friend function then, the private and protected data of a class can be accessed using the function.
- The compiler knows a given function is a friend function by the use of the keyword friend.
- For accessing the data, the declaration of a friend function should be made inside the body of the class (can be anywhere inside class either in private or public section) starting with keyword friend.

Subject :- Object Oriented Programming

Unit 1

Friend Function

```
class class_name
{
    ... .. ...
    friend return_type function_name(argument/s);
    ... .. ...
}
```

Subject :- Object Oriented Programming

Unit 1
Friend Function

```
#include <iostream>

class Distance
{
    private:
        int meter;
    public:
        Distance(): meter(0){ }
        friend int func(Distance);  //friend function
};
int func(Distance d)            //function definition
{
    d.meter=5;         //accessing private data from non-member function
    return d.meter;
}
void main()
{
    Distance D;
    cout<<"Distace: "<<func(D);
    getch();
}
```

Subject :- Object Oriented Programming

Unit 1

Static Data Member

- Static data members are class members that are declared using the static keyword.
- There is only one copy of the static data member in the class, even if there are many class objects.
- This is because all the objects share the static data member.
- The static data member is always initialized to zero when the first class object is created.

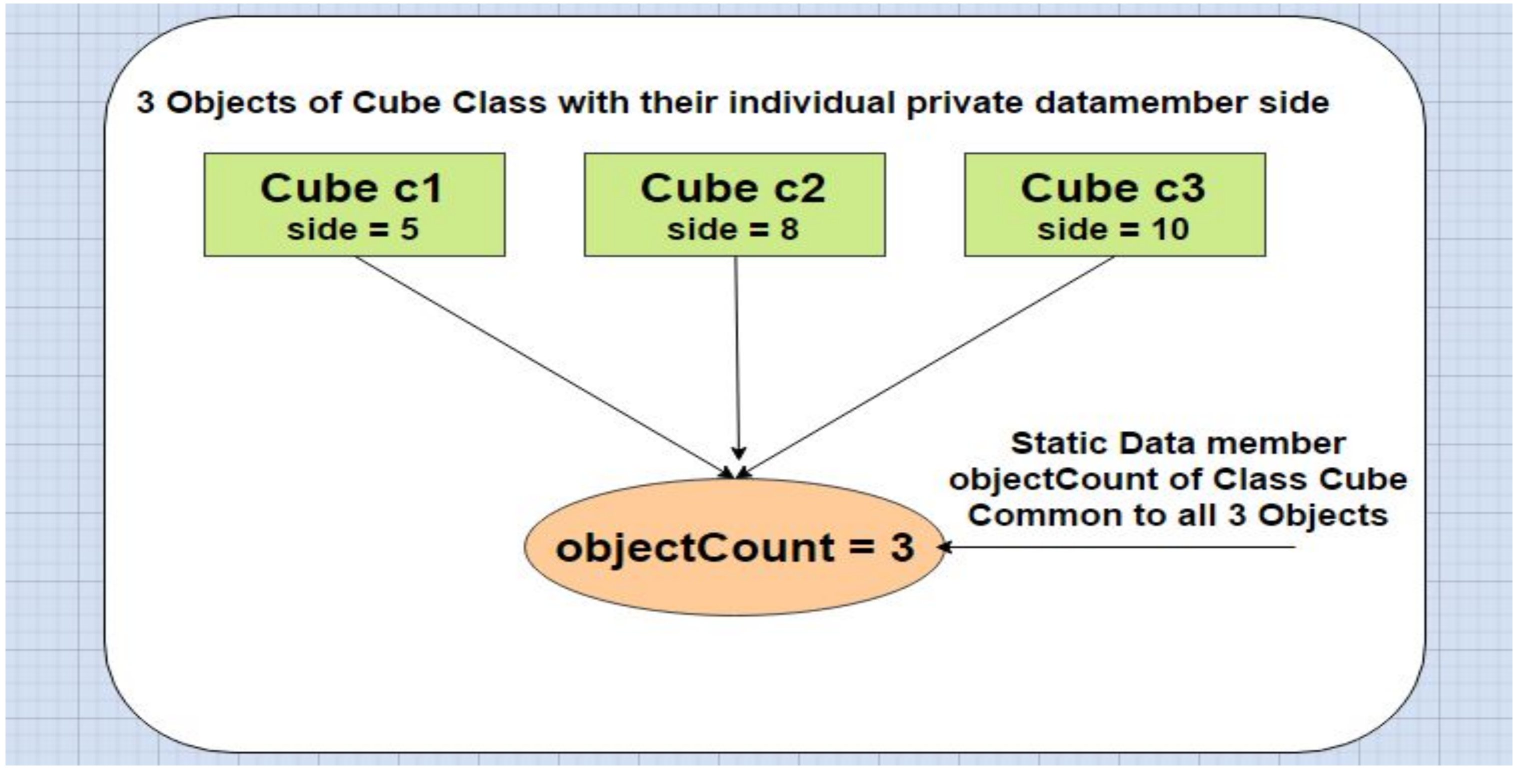
The syntax of the static data members is given as follows −

```
static data_type data_member_name;
```

Subject :- Object Oriented Programming

Unit 1

Static Data Member



```
class Demo
{
 public: static int ABC;
};
 int Demo :: ABC =10; //defining
 int main()
{
cout<<"\nValue of ABC: "<<Demo::ABC;
 return 0;
}
```

Subject :- Object Oriented Programming

Unit 1

Static Member Function

- By declaring a function member as static, you make it independent of any particular object of the class.
- A static member function can be called even if no objects of the class exist and the static functions are accessed using only the class name and the scope resolution operator ::.
- A static member function can only access static data member.

Synatx: -

```
static return type function-name()
{

    }
   And For Accessing the Function:-

   Syntax:-

      Classname::Functionname();
```

- **class** Demo
- {
- **private**:
- **static int** X;
- **public**:
- **static void** fun()
- {
- cout <<"Value of X: " << X << endl;
- } };
- **int** Demo :: X =10; //defining
- **int** main()
- { Demo X; X.fun();
- **return** 0; }