Introduction to C++

Object oriented concepts:

Object-Oriented Programming or OOPs refers to languages that uses objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism etc in programming.

The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

Principles (Features) of Object Oriented Programming:

1. Encapsulation
2. Data abstraction
3. Polymorphism
4. Inheritance
5. Dynamic binding
6. Message passing

Encapsulation: Wrapping of data and functions together as a single unit is known as encapsulation. By default data is not accessible to outside world and they are only accessible through the functions which are wrapped in a class. Prevention of data direct access by the program is called data hiding or information hiding

Data abstraction: Abstraction refers to the act of representing essential features without including the back ground details or explanation. Classes use the concept of abstraction and are defined as a list of attributes such as size, weight, cost and functions to operate on these attributes. They encapsulate all essential properties of the object that are to be created. The attributes are called as data members as they hold data and the functions which operate on these data are called as member functions.
Class use the concept of data abstraction so they are called **abstract data type** (ADT).

Polymorphism: Polymorphism comes from the Greek words “poly” and “morphism”. “poly” means many and “morphism” means form i.e.. many forms. Polymorphism means the ability to take more than one form. For example, an operation have different behavior in different instances. The behavior depends upon the type of the data used in the operation.
Different ways to achieving polymorphism in C++ program are:
1) Function overloading 2) Operator overloading

Inheritance: Inheritance is the process by which one object can acquire the properties of another. Inheritance is the most promising concept of OOP, which helps realize the goal of constructing software from reusable parts, rather than hand coding every system from scratch. Inheritance not only supports reuse across systems, but also directly facilitates extensibility within a system. Inheritance coupled with polymorphism and dynamic binding minimizes the amount of existing code to be modified while enhancing a system.

When the class child, inherits the class parent, the class child is referred to as derived class (sub class) and the class parent as a base class (super class). In this case, the class child has two parts: a derived part and an incremental part. The derived part is inherited from the class parent. The incremental part is the new code written specifically for the class child.

Dynamic binding: Binding refers to linking of procedure call to the code to be executed in response to the call. Dynamic binding(or late binding) means the code associated with a given procedure call in not known until the time of call at run time.

Message passing: An object oriented program consists of set of object that communicate with each other. Objects communicates with each other by sending and receiving information. A message for an object is a request for execution of a procedure and therefore invoke the function that is called for an object and generates result.

Advantages and Applications of OOPS

Advantages (Benefits) of Object Oriented Programming (OOPs)

- **Reusability:** In OOP‟ s programs functions and modules that are written by a user can be reused by other users without any modification.
- **Inheritance**: Through this we can eliminate redundant code and extend the use of existing classes.
- **Data Hiding:** The programmer can hide the data and functions in a class from other classes. It helps the programmer to build the secure programs.
- **Reduced complexity of a problem:** The given problem can be viewed as a collection of different objects. Each object is responsible for a specific task. The problem is solved by interfacing the objects. This technique reduces the complexity of the program design.
- **Easy to Maintain and Upgrade**: OOP makes it easy to maintain and modify existing code as new objects can be created with small differences to existing ones. Software complexity can be easily managed.
- **Message Passing:** The technique of message communication between objects makes the interface with external systems easier.
- **Modifiability:** it is easy to make minor changes in the data representation or the procedures in an Object Oriented program. Changes inside a class do not affect any other part of a program, since the only public interface that the external world has to a class is through the use of methods.

Applications of OOP:

- Real-time systems
- Simulation and modeling
- Object-oriented databases
- Hypertext, hypermedia and expertext
- AI and expert systems

- Neural networks and parallel programming
- Decision support and office automation systems
- CIM/CAM/CAD systems.

Applications of C++

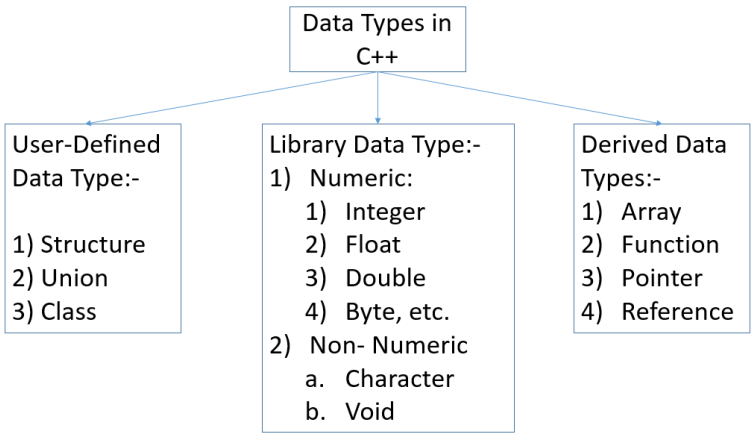
C++ is a versatile language for handling very large programs. It is suitable for virtually any programming task including development of editors, compilers, databases, communication systems and any complex real-life application systems.

- Since C++ allows us to create hierarchy-related objects, we can build special object oriented libraries which can be used later by many programmers.
- While C++ is able to map the real-world problem properly, the C part of C++ gives the language the ability to get close to the machine-level details.
- C++ programs are easily maintainable and expandable. When a new feature needs to be implemented, it is very easy to add to the existing structure of an object.

Data types, new operators and keywords, type conversion in C++

Data Types:

The language C++ supports all the data types available in 'C' language. The data types of language C++ can be viewed as given bellow.



As shown in above figure,

The user-define data type Structure and union are same as that of the implementation in C, there is small advantage in implementation of structure in C++, that we will discuss later. The new data type which is not in C is the "class". The concept of the class is the main concept in C++ programming.

The Library / Built-In data type consist of all the data types as that of the data types used in C, the range and keyword used for these data type is similar to C. There is little bit difference in size used in C and in C++.

The derived data types, array, function. Pointer are also similar to the language used in C. The language C++ introduces new data type as "reference"

Operators and Keywords:

C++ has a rich set of operators. All operators in C are used as it is with their meaning in C++ also.
The various operators are:-
a. Assignment Operator
b. Arithmetic Operator
c. Relational Operator
d. Logical Operator
d. Bit wise Operator
e. Increment/Decrement operator
f. Shortcut Operator
g. Conditional Operator
h. Special operator

Assignment operator:
The assignment operator is used to store the values of constants, variables, and result of an expression and return value of a function into variable placed in left side.

Operator Symbol: **=**

Ex : 1) a = 6; 2) a = b; 3) a = b+c;

Arithmetic operators:
The arithmetic operators are used to perform the basic operations like addition, subtraction, multiplication, division and modulus.
The operators are: Addition (**+**) **,** subtraction (**–**) **,** multiplication (*) **,** Division (**/**)**,** modulus division (**%**)

Ex:
1) **x=a+b;**
2) x=a-b; 3) x=a*b; 4) x=a/b; 5) x=a%b;

Increment/Decrement operators:
The Increment operators **++** increase the value of variable by one.
The Decrement operators **--** decrease the value of variable by one.
Ex :
i++,
++i,
i--,
--i.

Shortcut operators:
When the arithmetic operation on value of a variable is to be performed and assigned to the same variable, we can use shortcut assignment operator.
The operators are: **+=, – =, *=, /=, %=**

Ex:
A += 2; A -= 3'

A *= 2; **A /= 2**
A %= 4;

Sizeof:
The **sizeof** operator is used to find the size of the memory occupied by an variable in terms of bytes.
Ex : **a = sizeof(int)**;

Relational operators:
The operators that do relationship tests with the operands are called relational operators.
The relational operators are:
< (is less than)
<= (is less than or equal to)
> (is greater than)
>= (is greater than or equal to)
== (is equal to)
!= (is not equal to)

Logical operators:
Logical operators are used when more than one relationship should be considered in evaluation.
It gives either True value or False value.
The operators are
AND operator: **&&**
OR operator: **||**
NOT operator: **!**

The language C++ introduces some new operators which are:

1) **<<** :- insertion operator (or output operator)
2) **>>** :- extraction operator (or input operator)
3) **::** :- Scope resolution operator
4) **::*** :- pointer-to-member declarator
5) **->*** pointer-to-member operator
6) **.*** pointer-to-member operator
7) **endl** line feed operator
8) **new** memory allocation operator

Keywords:
The keywords are the reserved words with specific meaning, these cannot be used and programe names or variable names in programe or user define program name.
Some of the keywords are:

auto	break	case	catch	char	class
const	continue	default	delete	do	double
else	enum	extern	float	for	friend
goto	if	inline	int	long	new

operator	private	protected	public	register	return
short	signed	sizeof	static	struct	switch
template	this	throw	try	typedef	union
unsigned	virtual	void	volatile	while	

Mostly used new keywords in C++ are **new, class, inline, private, protected, public, virtual**, etc. These keywords plays very important role in C++ programming.

The general program structure in C++ can be given as

Include Section
Class declaration
Class function definition
Main function programe
User-define programe

The simple programe in C++ can be written as

```
#include<iostream.h>
#include<conio.h>
void main()
{
  int a,b,c;
  clrscr();
  cout<<"Enter first number ";
  cin>>a;
  cout<<"Enter second number ";
  cin>>b;
  c = a + b;
  cout<<"Addition is "<<c;
  getch();
}
```

In the programe above

iostream.h:- is an header file containing the information about input and output stream. i.e. input and output statements used in C++.

Input statement:- The “cin” is the statement used to read the values from keyboard. The general syntax of cin statement is given as follows:

Syntax:

cin>>variable_name[>>var_name>>var_name…..];

here cin is the statement used to read values from keyboard. The operator >> is known as **"Extraction"** *or **"get from"*** operator. It extracts (or takes) the value form the keyboard and assign it to the variable.

```
int a,b,c;
        cin>>a;
        cin>>b>>c;
```

Output statement:- The "cout" is the statement used to display values on to the monitor. The general syntax of cout statement is given as follows:

Syntax:

1) cout<<variable_name[<<var_name<<var_name…..];
2) cout<<"Any data in double quot.."[<<"……."<<"var_name"<<…..];
3) cout<<"string information"<<variable_name;

here cout is the statement used to display values on monitor. The operator << is known as **"Insertion"** *or **"put to"*** operator. It inserts (or sends) the value of variable to the monitor.

```
int a,b;
a = 90; b = 20;
cout<<a;
cout<<a<<b;
cout<<"The value of A = "<<a<<" B = "<<b;
cout<<"The addition is "<<a+b;
```

Classes & Objects

Class:

A Class is an user define data type that can be treated like any other built-in data type. The class is a way to bind data and its associated functions together. The class allows the data and functions to be hidden form external user i.e. form user, if necessary. When we are defining a class it means we are creating new user define data type also known as abstract data type.

General form of a class declaration is:

```
class class_name
{
        private:
                variable declarations;
                function declarations;
        public:
                variable declarations;
                function declarations;
};
```

In above declaration

Class is a keyword used to declare class.

Private, public are the two visibility modes, these are also called as access specifiers. These are used to decide whether to allow user to make use of variables and or functions.

The public access specifier is allows user to make use of variables / functions, whereas private access specifier is used to protect variables / functions.

The variables declared under public / private / protected access specifier are known as **data members / properties** of class. The functions declared in class are called as **method members** of class. The method members are used to make operations on data members of class.

Object:

Objects are the basic run-time entities in an object-oriented system. They may represent a person, a place, a bank account, a table of data or any item the program has to handle. They may be represents user-defined data such as vectors, time and lists.

To make use of the data members and method members of class one has to create and instance / variable of a class. The class instance / variable is called as an object of that class. The object of a class contains all the properties and methods of a class. It is also termed as blue print of a class.

The general syntax of creating object is

class_name variable_name;

Example of creating class and object:

```
#include<iostream.h>
#include<conio.h>
class student
{
  private:
        int rollno;                     // Data Member
        char sname[20];                 // Data Member
  public:
        void getdata()                  // Method Member
        {
          cout<<"Enter roll Number ";
          cin>>rollno;
          cout<<"Enter name of Student ";
          cin>>sname;
        }
        void putdata()                  // Method Member
        {
         cout<<" The Roll Number is "<<rollno;
         cout<<" \n The Name is "<<sname;
        }
};
void main()
{
  student obj;          // Creating an object of class
  clrscr();
```

```
  obj.getdata();        // accessing members of class using object
  obj.putdata();        // accessing members of class using object
  getch();
}
```

The operator dot (.) is used to access class members using object of the class.