C++ (1979)

Bjarne Stroustrup
Middle level language.
C with Classes. (Cpp is superset of C lang)
Cpp joins three seperate programming traditions i.e procedural, object-oriented, generic programming(C++ templates).
C follows top down approach of programming, wheras Cpp follows bottom up approach of programming.

Object oriented programming

Object - Any entity in the system that can be defined as a set of operations performed using enitity's property set.

Principles
- Encapsulation -> Each entity information associated with single object.
- Data Hiding -> Hiding the data from any other class and can only accessed by any member function.
- Abstraction -> Hiding the implementation and show the essential properties.
- Polymorphism -> Existing in many forms such that we can use each of them by its behaviour.
- Inheritance -> It is a process of inheriting properties and behaviours of existing class into a new class.

Class
- Blueprint of Object, Description of Object's property set and set of operations.
- Creating class is as good as defining a new non-primitive/user defined data type.
- Class is a means to achieve encapsulation.
- Object is a run time entity.
- Object is an instance of a class.

Identifiers

Constants
- Any information is constant
- Primary constants (Integer, Real, Character).
- Secondary constants (Array, String, Pointer, Union, Structure, Enumerator, Class). Build using primary constants.

Varriables
- Names of memory locaations where we store data. It can be any combination of alphaber, underscore, number.
- Valid varriable name cannot start with digit.

Keywords
- Reserve words/ Predefined words.

Data types
- int
- char
- float
- void

* Unlike C, you can declare varriables even after action statements.
* example{
* clrscr();
* int x=4;
* }

Input and Output

Output Instruction
- cout is a predefined object whereas in C printf() is function to send message to monitor.
- The operator << is called the insertion or put to operator (in the output stream).
- printf("Sum of %d and %d is %d",a,b,c);
- cout<<"Sum of "<<a<<" and "<<b<<" is "<<c;

Input Instruction
- cin is predefined object to input data from keyboard
- The operator >> is known as extraction or get from operator
- scanf("%d",&a);
- cin>>a;

According to the ANSI standards for C language, explicit declaration of function is recommended but not mandatory but whereas C++ language says explicit declaration of function is mandatory.

Predefined functions are declared in header files, so whenever using any predefined function in code have to include specific header file contains its declaration. Library files have the code and header files have the declaration of the function.

iostream.h is a header file it contains declarations for the identifier cout and cin, operators << and >>.

Identifers can be function names, varriables, objects, macros.

endl - newline which is declared in iostream.h header file.

Types of varriables:

Ordinary varriable
- int x=5;

Pointer varriable
- int *p;
- p=&x;

Reference varriable
- int &y=x;
- y++; // x value is incremented the same block has two names x and y

Reference means address.
Reference varriable is an internal pointer.
Declaration of reference varriable is preceded with '&' symbol.
Reference varriable must be intialized during declaration.
It can be intialized with already declared varriables only.
Reference varriables can not be updated.

Functions

- Is a block of code performing a unit task.
- Function has name, return type and arguments.
- It is a way to achieve modularization.
- Functions are predefined and user-defined.
- Predefined functions are declared in header files and defined in library files.
- Function can be declared globally and locally.
- ReturnType functionName(argumentList);
- Functionns need to be declared before use just like varriables.
- Function declaration is also known as Function prototype.
- Function definition is a block of code.
- The passed varriables to a function is actual arguments.
- The copied values of actual arguments to the function arguments are known as Formal arguments

Formal arguments can be of three types
- Ordinary var of any type
- Pointer varriables
- Reference varriables

Call by Value - When formal arguments are ordinary varriables.
Call by address - When formal arguments are pointer varriables.
Call by Reference - When formal arguments are reference varriables.

Advantages of functions
- Easy to read.
- Easy to Modify.
- Avoids rewriting of same code.
- Easy to debug.
- Better memory utilization.

Dis-Adavantages:
- Functions is time consuming.(because of jumping of control flow and saving PCB registers).
- To overcome the function time consuming overhead small block of code is not written as function whereas large section of code should be written in Function such that we can reduce the function overhead.

To eliminate the cost of call to small functions, C++ proposes a new feature called inline funtion.

INLINE FUNCTION

- An inline function is a function that is expanded in line when it is invoked.
- Compiler replaces the function call with corresponding function code.
- inline is a request not a command.
- The benfit of speed of inline functions reduces as the function grows in size
- So the compiler may ignore the request in some situations like recursion, loops, goto, switch, containing static var.

- inline declaration of function.
- inline void fun(int, int);

Default arguments while declaration we can make a argument intialized a value by default with a value. while calling function if the argument does not get value then the default argument gets passed.

Polymorphishm

* Compiletime Polymorphism
    - Function overloading.
    - Operator overloading.
* Runtime Polymorphism
    - Virtual Function. (Runtime binding)

Function overloading (Compiletime binding the function name with call)
- More than one function definiton has same name.

Structures in C++ are almost same as C language but the main difference are
- while declaring a struct varriable there is no mandatory for the struct keyword when using tag name.
- we can define the function in structure definiton that is encapsulation (act of combining varriables that is properites and functions that is methods combining together) property. (which is the property of object oriented language is Cpp feature).
- data security the data will not be corrupted using access specifiers (private, public, protected).
- by default structure members are public

Classes and Objects

* The only difference between Classes and strucuture is that, the members of class are by default private.
* Class is a description of an object.
* Object is an instance of class.
* Instance member varriable - attributes, data members, properties.
* Instance member functions - Methods, operations, procedures, actions.

Static Members

static local varriable (static int a)
- varriable which is intialized to zero and the lifetime is throughout the program.

static member varriable / class member varriable
- varribale which is declared inside the class.
- must be defined oustside the class.
- static member varriable does not belong to any object, but to the whole class.
- There will be only one copy of static member varriable for the whole class.
- Any object can use the same copy of class varriable.
- They can also be used with class name.

static member functions / class member functions
- They are qualified with the keyword static.
- They are also called member functions.
- They can be invoked with or without object.
- They can only access static members of the class.

Constructor

- Constructor is a member function of a class.
- The name of the constructor is same as the name of the class.
- It has no return type, so cannot use return keyword.
- It must be an instance member function, that is it can never be static.
- Constructor is implicitly invoked when an object is created.
- Constructor is used to solve problem of intialization. (member varriable intialization)
- * compiler creates a default and copy constructors in class.

Default Constructor
- If and only if the constructor is not defined in the class, compiler adds the default constructor to the class with no parameters and no code block in the braces.

Parameterized Constructor

- The user defined constructor in which parameters are passed.

Constructor Overloading

- If more than one constructors are defined with different parameters then constructor overloading happens.

Copy Constructor

- Compiler defaulty adds copy constructor unless one copy constructor is defined in the class even if there is a parameterized constructor or default constructor defined in the class.
- If the copy constructor is defined by the user both copy and default constructors are not added by the compiler to the class.

Destructor

- Destructor is an instance member function of a class.
- The name of the destructor is same as the name of a class but preceded by tilde(~) symbol.
- Destructor can never be static.
- Destructor has no return type.
- Destructor takes no argument (No overloading is possible).
- Defaulty compiler adds the destructor with empty block of code.
- It is invoked implicitly when object is going to destroy.
- It should be defined to release resources(pointer to a memory) alloacted to an object.

Operator Overloading

- When an operator is overloaded with multiple jobs, it is known as operator overloading.
- It is a way to implement compile time polymorphism.
- Any symbol can be used as function name
    - If it is a valid operator in C language.
    - If it is preceded by operator keyword.
- sizeof and ?: operators cannot be overloaded.

Friend Function

- Friend Function is not a member function of class to which it is a friend.
- Friend Function is declare in the class with friend keyword.
- It must be defined outside the class to which it is friend.
- Friend function can acess any member of the class to which it is friend.
- Friend function cannot access members of the class directly.
- It has no caller ojbect.
- It should not be defined with membership label.
- *Friend function can become friend to more than one class.

- Operator overaloading can be done with friend function with adding one more argument than member function for operator overloading.
- Member function of one class can become friend function to another class.

Inheritance

- It is a process of inheriting properties and behaviours of existing class into a new class.
- Existing class = Old class = Parent class = Base class.
- New class = Child class = Derived class.

Types of Inheritance

- Single Inheritance. A->B
- Multi level Inheritance. A->B->C (transitive relation)
- Multiple Inheritance. A,B->C
- Hierarchical Inheritance. A->B,C
- Hybrid Inheritance. Combination of two Inheritance

Class contains Visibility Modes / Acess specifiers
- Private
- Protected
- Public

Types of users of a class

- User1 will create Object of your class.
- User2 will derived class from your class.

* For user1 and user2 while object creation public, private, protected members get memory allocation and available for the object but not all members are accessable,only accessable members are public.

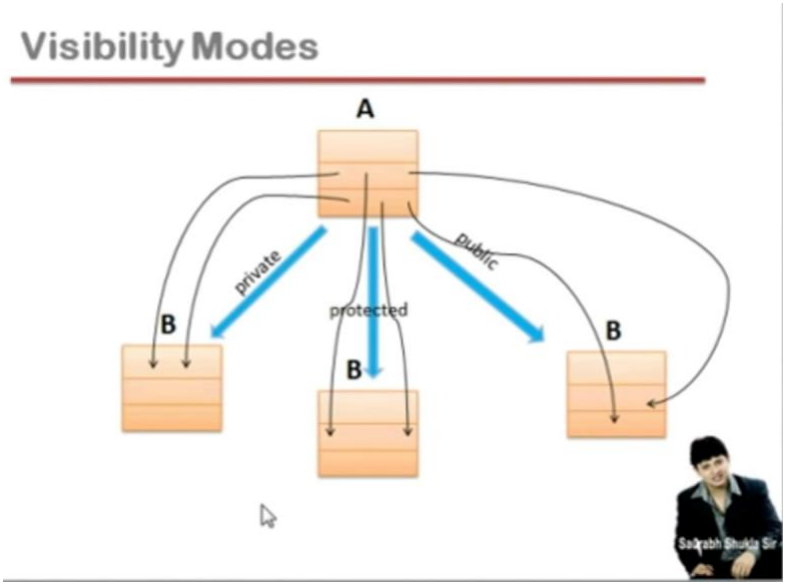
* User1 doesnt have access to Protected members but User2 have acess.

Is - a relationship

Banana is a fruit.
Fruit - more genernalized (parent class).
Banana - more specific (child class).
All bananas are fruits, but not all bananas are not fruits.

Association (Relationship between two Entities i.e Classes)
- Aggregation
- Composition (has-a)

- Inheritance (is-a)



* Is a relaitionship is always implemented as public inheritance.
* Why? because when private and protected inheritance is used then there is no access to the object for parent methods. (is a relationship provides the child class obj to acess the parent class methods)
* When to use private and protected inheritance?

Constructor and Destructor in Inheritance

* We know that constructor is inovked implicitly when an object is created.
* In inheritance, when we create object of derived class, what will happen?
* The child object calls the child class constructor.
* The child class constructor calls the parent call constructor.
* After the parent class constructor execution the child class constructor execution happens.
* If user defined constructor is there in parent class then the user should pass the value while calling the constructor in child class by default the compiler calls default constructor which doesnt have any arguments so, define the constructor in child class and call the parent class by passing value for the argument.

* After executing child class destructor only the parent class destructor will be called and exectued which free's the resources occupied by the object.

---

this pointer

A pointer contains address of an object is called object pointer.

* this is a keyword.

* this is a local object pointer in every instance member function containing address of the caller object.
* this pointer can not be modify.
* It is used to refer the caller object in member function.

---

new and delete keywords

SMA : Static Memory Allocation (compile time memory allocation, lifetime of the sma var is in the block only).
DMA : Dynamic Memory Allocation (malloc, calloc alternative is used new, lifetime of the dma var throughout the program untill we use delete and free the allocated memory).

```
int *p = new int;
float *q = new float; // the pointer q will catch the address of newly allocated memory of size float type
Complex *ptr = new Complex;
float *a = new float[5];
int x;
cin>>x;
int *p = new int[x];

delete p;
delete []p;
```

Method Overriding

* It is useful when the parent class implementation should be modified for the child class then go for overriding.

Method Hiding

* Method hiding is used when the parent class methods should not be visible from the child class object.

Virtual function

Base class Pointer (Inheritance)
* Base class pointer can point to the object of any of its descendant(child) class.
* But its converse is not true.

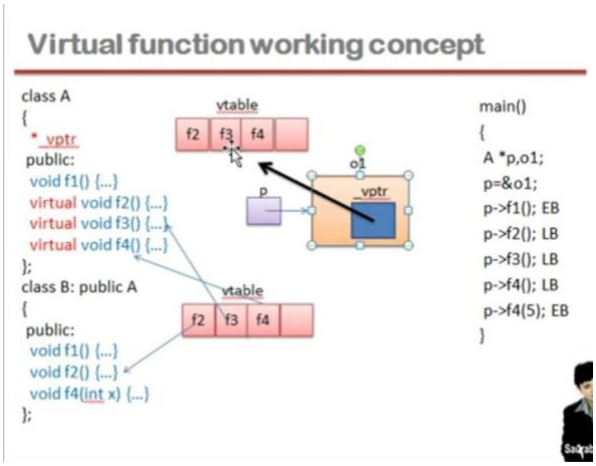
working of virtual function

```
class A{
```

*vptr; // created by compiler which points to vtable(static array createc by compiler) which consits of the function pointers (only created for virtual functions).

```
public:
   void f1() { }
   virtual void f2() { }
   virtual void f3() { }
   virtual void f4() { }
};

class B: public A{
public:
   void f1() { }
   void f2() { }
   void f4(int x) { }
};
```



Abstract class

Pure virtual function - A do nothing function is pure virtual function.

* A class containing atleast one pure virtual function is an abstract class.
* We can not instantiate abstract class. (we cannot use the object)

Template

* The keyword template is used to define function template and class template.
* It is way to make your function or class generalize as far as data type is concern.

Function template is also known as generic function
Syntax:
tempalte <class type> type func_name(type arg1, ...);

Class template is also known as generic class.
Syntax:
template <class type> class class_name {.....};

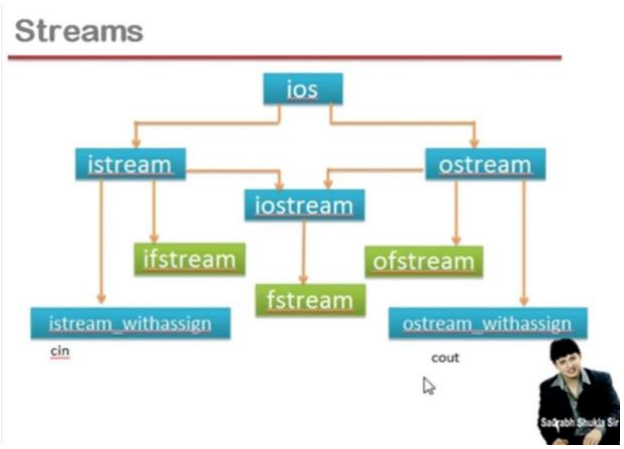
File Handling

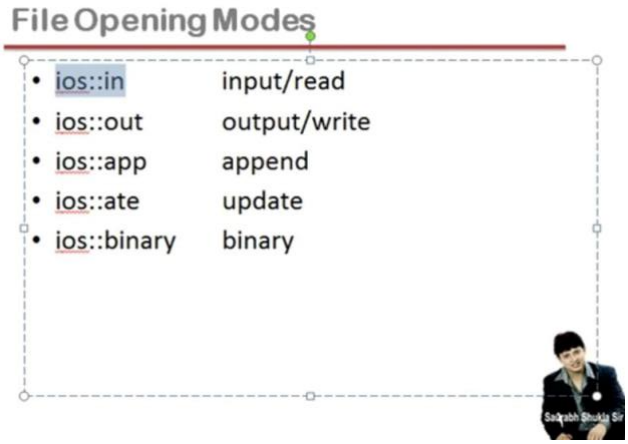
Data Persistence/ Data Existence - Life of Data (life of progam > = life of varriable).

* Permenant storing of a varriable after the execution of the program for performing some operations.
* For the above application file handling is used we will store the varriable in the file after the exection of the program.
* Again by reading the input from the file we can perform operations on that data stored in file.

Streams

* Input stream - File data is transferred to the varriable.
* Output stream - Varriable data is transferred to the file for storing.





tellg()

* Defined in istream class.
* Its prototype is - streampos tellg();
* Returns the position of the current character in the input stream.

tellp()

* Defined in ostream class.
* Its prototype is - streampos tellp();
* Return the positon of current character in the output stream.

seekg()

* Defined in istream class
* Its prototype is
  - istream& seekg(streampos pos);

- istream& seekg(streamoff off, ios_base::seekdir way);
* pos is new absolute position within the stream (relative to the begining).
* off is offset value, relative to the way parameter.
* way values ios_base::beg, ios_base::cur and ios_base::end.

seekp()

* Defined in ostream class
* Its prototype is
    - istream& seekp(streampos pos);
    - istream& seekp(streamoff off, ios_base::seekdir way);
* pos is new absolute position within the stream (relative to the begining).
* off is offset value, relative to the way parameter.
* way values ios_base::beg, ios_base::cur and ios_base::end.

Intializers

* Intializer List is used to intialize data member of a class.
* The list of members to be intialized is indicated with constructor as a comma seperated list followed by a colon.
* There are situations where intialization of data members inside constructor doesn't work and Intializer List must be used.
    - For intialization of non-static const data members.
    - For intialization of reference members.

Deep copy and Shallow copy

* How can we create a copy of object?
    - Copy constructor
    - Implicit copy assignment operator

Shallow copy - Creating copy of object by copying data of all member varriable as it is.

Deep copy - Creating an object by copying data of another object along with the values of memory resources resides outside the object but handled by that object.

Dangling pointer
    - pointer which points to the invalid address that is which was deleted.(accessing illegal memory)

Type Conversion

* Primitive type to Class type - It can be implemented through consructor.
* Class type to Primitve type - It can be implemented with the casting operator.

* Class type to Class type - It can be implemented through constructor(inside the LHS class)/ casting operator(inside the RHS class).

Exception Handling

* Exception is any abnormal behaviour, run time error.
* Exception are off beat situation in your program where your program should be ready to handle it with appropriate response.
* Cpp provides a built-in error handling mechanism that is called exception handling.
* Using Exception handling, you can more easily manage and respond to runtime errors.

* try catch throw keywords

- Program statements that you want to monitor for exceptions are contained in a try block.
- If any exception occurs within the try block, it is thrown (using throw).
- The exception is caught, using catch, and processed.

Syntax:

```
try{
}
catch(type1 arg){
}
catch(type2 arg){
}
.
.
catch(typeN arg){
}
```

catch
- When an exception is caught, arg will receive its value.
- If you dont need access to the exception itself, specify only type in the catch clause - arg is optional.
- Any type of data can be caught, including classes that you create.

throw
- The general form of the throw statement is: throw exception;
- Throw must be executed either within the try block proper or from any function that the code within the block calls.

Dynamic Constructor

- Constructor can allocate dynamically created memory to the object.
- Thus, object is going to use memory region, which is dynamically created by constructor.

STL Introduction

* STL is Standard Template Library.
* It is a powerful set of C++ template classes.
* At the core of the C++ Standard Template Library are following three well-structured components
  - Containers.
  - Algorithms.
  - Iterators.

Containers

- Containers are used to manage collections of objects of a certain kind.
- Container library in STL provide containers that are used to create data structures like arrays, linkedlist, trees..etc
- These container are generic, they can hold elements of any data types.

Example
vector can be used for creating dynamic arrays of char, integer, float and other types.

Algorithms

- Algorithms act on containers. They provide the means by which you will perform intialization, sorting, searching and transforming of the contents of containers.
- Algorithms library contains built in functions that performs complex algorithms on the data structures.

Example
One can reverse a range with reverse() function, sort a range with sort() function, search in range with binary_search() and so on.

Algorithms library provides abstraction, i.e you dont necessarily need to know how the algorithm works.

Iterators

- Iterators are used to step through the elements of collections of objects. These collections may be containers or subsets of containers.
- Iterators in STL are used to point to the containers.
- Iterators acutally acts as bridge between containers and algorithms.

example

sort() algorithm have two parameters, starting iterator and ending iterator, now sort() compare the elements pointed by each of these iterators and arrange them in sorted order, thus it does not matter what is the type of the container and same sort() can be used on different types of containers.

CONTAINERS

- Container library is a collection of classes.
- The containers are implemented as generic class templates.
  Containers help us to implement and replicate simple and complex data structures very easily like arrays,
  list, arrays, trees, associative arrays and many more.
- Containers can be used to hold different kind of objects.

Common containers
- vector : replicates arrays
- queue : replicates queues
- stack : replicates stack
- priority_queue : replicates heaps
- list : replicates linked list
- set : replicates trees
- map : replicates associative arrays

Classifications of Containers
- Sequence Containers : arrays, linked list, etc (lineary data structures).
- Associative Containers : sorted data structures like map, set, etc.
- Unordered Associative Containers : Unsorted Data Structures
- Containers Adapters : Interfaces to sequence containers

How to use container library?
When we use list containers to implement linked list we just have to include the list header file and use list constructor to intialize the list.

```
#include<iostream>
#include<list>
using namespace std;
int main(){
   list <int> mylist;
}
```

Array container

- Array is a linear collection of similar elements.
- Array container in STL provides us the implementation of static array.
- Use header array #include<array>

Member funtions

Following are the important and most used member functions of array template.
- at : This method returns the value in the array at the given range(if out of range then throws an exception).
- [] operator : Used to access the value of the array at the given range.
- front() : Returns the first element in the array.
- back() : Returns the last element in the array.
- fill() : This method asssigns the given value to every element of the array.
- swap() : This method swaps the content of two arrays of same type and same size(performs index wise swap)
- size() : This method returns the number of element present in the array.
- begin() : This method returns the iterator pointing to the first element of the array.
- end() : method returns an iterator pointing to an element next to the last element in the array.

Pair (It is a part of utility library).

- It is a template class in Standard template library.
- pair is not a part of container.
- Syntax : pair <T1,T2> pair1;

Member functions
- make_pair()
- first
- second

We can compare two paris by using comparision operators (==, !=, <, >, <=, >=)

Tuple - Just like in pair, we can pair two heterogeneous objects, in tuple we can multiple objects.

Sytntx: tuple <T1,T2,T3> tuple 1;

Member functions
- make_tuple()
- we can compare two tuples by using comparisin operators.

Vectors container

- The most general purpose container is the vector.
- It supports a Dynamic Array.
- Array size is fixed, no flexibility to grow or shrink during execution.
- Vector can provide memory flexibility.

- Vector being a dynamic array, doesn't needs size during declaration.
- Code below will create a blank vector.

Member funcitons
- Subscript operator[] is also defined for vector.
- push_back() is a member function, which can be used to add value to the vector at the end.
- pop_back() is member function, It removes the last element in the vector
- capacity() returns the capacity of the vector, this is the number of elements it can hold before it will need to allocate more memory.
- size() returns the number of elements currently in the vector.
- clear() removes all elements from the vector.
- at() returns the element at ith index in the vector.
- inset() inserts an element in between the vector using iterator.
- front()
- back()
- begin()
- end()

List container

- List class supports a bidirectional, linear list.
- vector supports random acess but a list can be acessed sequentially only.
- List can be acessed front to back or back to front.
- Syntax : list <type> obj_name;

Member functions
- sort()
- size()
- push_back()
- pop_back()
- pop_front()
- reverse()
- remove()
- clear()

Map

- Maps are used to replicate associative arrays.
- Maps contain sorted key-value pair, in which each key is unique and cannot be changed, and it can be inserted or deleted but cannot be altered.
- value associated with keys can be altered.
- Maps alwats arrange its keys in sorted order.
- In case the keys are of string type, they are sorted in dictionary order.

Member functions

- at()
- []
- size()
- empty()
- insert()
- clear()

Strings
- Using null-terminated character arrays arenot technically data types.
- So, C++ operators cannot be applied to them. (+, >, =) it can be applied using strcat, strcpy, strcmp

string class - The string class is a specialization of a more general template class called basic_string.
*Since defining class in C++ is creating new data type, string is derived data type.
*This means operators can be overloaded for the class.

string is safe than char array
- careless programmer can overrun the end of an array that holds a null terminated string.
- For example, using strcpy().
- string class handles such issues.
- string is an another container class.
- To use string class, you have to include string header.

Mixed operations
- You can mix string objects with another string object c style string.
- C++ string can also concatenated with character constant.

Member functions

- assign()
- append()
- insert()
- replace()
- erase()
- find()
- rfind()
- compare()
- c_str()
- size()
