C++

Example.cpp → (Preprocessor) → Example.i → (Compiler) → Example.obj
remove
Example.exe ← (Linker)
software

Cout ✓ Predefined object
× not fxn.

\n == endl

Ex 1) Cout << "Hello world";
2) Cout << "Sum of" << a << "and" << b << " is" << c;

cin >> a;
Ex Cin >> a >> b;

cout << " " << endl

inline fxn

Reference variable or (Pointer)
* int &y = x;
* Can't be updated.

• Struct is not necessary to use variable made by structure
• May use Public: / Private: Keywords in fxn & variable to make data excessible or not out of this struct Block.

Distructor fxn [~Destructor()] {Class-Name}

• The members of class are by default private

Lec 7 Part-1 membership of class.

Constructor ⇒ it is an (no return type) undefined fxn in a class which run's automaticly when we make an obj using this class.
class-Name
Ex constructor() { }

data type ClassName :: datavariable(); (scope resolution oper.)

Scanned with CamScanner

Friend fxn keyword (friend)
in class → friend void fun();
- defined out of class.
- called in main like fun(c1);

Constructor in Inheritance :-
in Const & destruc
- Call order : child to parent class
- Run order : parent to child class. (Reverse in Destructor)

→ in child class Constructor B() : Constructor A()
{ }

DMA (new) :→ Eg: complex *ptr = new Complex;
Eg: float *ptr = new float[5];
* malloc/calloc
→ (delete) :→ Eg: delete p; Eg: delete []p;
Pointer

Virtual fxn *

LEC 18 - Part-2

Template :- template <class X, > (Placeholder — Data type token)
X fxn_name (X a, X b)
{ }

Note → In Constructor (int x, int y)
class B { int x,y; Public: } { This→x=x; This→y=y; }
main { B B2(37,90); B B4(B2); }
Can run.

Lect # File Handling :- #include<fstream.h>

```
main()
{
  ifstream fin;   // xyz
  // Reading from file
  char ch;
  fin.open("f1.dat");
  fin>>ch;   → OR [ch = fin.get();]
  while(!fin.eof())
  {  cout<<ch;
     fin>>ch;
  }
  fin.close();
}
```
// Reading from file
* fin.eof() // end of file
[Makes space appears]

```
main()
{ ofstream fout;   // outputstream
  fout.open("f1.dat");   // fopen
  fout<<"Hello";
  fout.close();   // fclose
```
// Writing in file

file opening modes   fout. open("f1.dat", ios::app | ios::in);
// reads line with Hello   // f1.txt → Hello

# * tellp(); * tellg();   // returns position
int pos;   pos = fin.tellp();   // g → os
or fout

* seekg(); → seek the tellg (get pointer) point to any position
1) fin or fout. seekg (int pos);
2) f // . seekg (int pos, ios_base :: ^) (curr, beg, end)

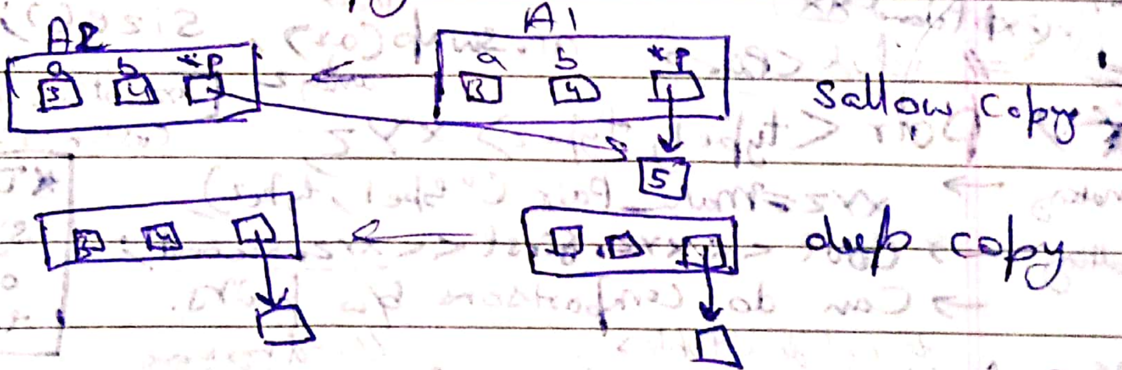
pos is taken

* seekp(); → seek the tellp (o/p points) point to any position
1) fout. seekp (int pos);
2)

Initialize list   constructor () : a(5), b(6)
// a, b variable of class

Deep copy & Shallow copy :-

Class A
A1(3, 4, r)
A2 = A1;



Type conversion:
1) Class type = Primitivetype (Using Constructor)
2) Primitive type = Class type (Casting operator   operator type()
{ return(a);
} member of class
3) class type ① = Class type ②
* (Using Constructor in class ① a, b) ② x, y
① (② Product)
{ a = Product. getX();
b = Product. getY();
}

* (Using Casting operator)

Lec 8

Exception Handeling :

```
try {      // throw
}          // throw 10;
Catch (type1 arg) {
}
or Catch (type2 arg) {
}
```

Lec 1

namespace XYZ

```
{
    // Declarations
}
```

* namespace xyz = XYZ;
// Name change

// Using Namespace XYZ;

Virtual Distructor()
{ }

Nested Class →

STL → Std. template lib.

*3 well-structured compo:-
- ① Containers // Contains Data
- ② Algorithms // Operations on data
- ③ iterators // Bridge b/w ① & ②

① Container → Array   #include <array>
array <type, size> xyz.
→ Member fx^n at(,), front(), back(), fill (value)
at. swap(a2) (index, array 2), size(); begin(), end() (iterators)

xyz.at(i).show_data(); (Customise for class)

(explained xx
#include <Pair>)

* → Pair <type1, type2> xyz;
// Making → xyz = Make_Pair ("type1", type2);
// Calling → Cout << xyz.first << xyz.second;
→ Can do comparisons b/w pairs.

*Dynamic Array → size of arr can increase on run time acc to utility

* → tuple <type1, type2, type3> xyz; (#include <tuple>) // and many more
→ xyz = make_tuple (type1, type2, type3); (0, 1, 2)
→ Cout << get<0> (xyz);

* → Vector Class → Arr. that supports Dynamic arr.
* Capacity only doubles (x2)

#include <Vector>

// Declaration
- vector <type> xyz = {10, 30, 20};
- vector <char> v2 (size);
- vector <char> v3 (size, x); (3) [x | x | x]
- vector <type> xyz;

→ xyz.capacity(), xyz.push_back(x);, xyz.pop_back(); // last value
xyz.size(); // (No. of elements), xyz.clear();, front() & back();

//(iterator) vector <int> :: iterator it = xyz.begin();
c++ v1.insert(it+3, x);

#include <list>

★ → list class → list <int> l1; front(), Back()
// (traverse using iterator) → l1.end(), push_back(x); push_front(x)
pop_back(), pop_front(), sort(), reverse(), remove(x), clear(),
// (elmt of list)

Numeric arry :- indexes are numbers
Key → 0, 1, 2, 3 ...
value ←

Associative arr :- Customized indexes. (key)
i.e. name etc.
Amit raj Badal

★ → map class → (associative arry)
#include <map>
→ map <Key type, value type> xyz;
// (Assignment) // xyz{{key, value}, {key, value}, ...}
OR xyz[key] = value;
// (traversal) → iterator p = xyz.begin;
p→first (key) OR p→second (value)

→ clear(), empty(), size(),
xyz.insert (pair <type1, type2> (x, y));

include <string>

★ → String Class → string s1 = "Hello";
→ // (mixed operations s1 = s2 + "123" / s2 + (char arr))
→ xyz. assign ("..."), append ("..."),
xyz .insert (index, "..."), replace (index, length, "...")
// (if (2,2,"A") Hello → HeAo)
.erase(); .find("...") // (return index)
.rfind("...") // (from reverse), s1.Compare(s2); // (return 0 if same, -1 if opp to dic order, 1 if in dictionary order)
, .c_str(); and (& return) , .size()

// operators (+, =, +=, ==, etc. >=, <=, >, < != etc. [ ]) >>, <<

★ ★ ★ ★