KLiC C++ Programming

Detailed Syllabus:

KLiC C++ Programming

Intro to OOP

- The Beginning
- Structured Programming
- Object Oriented Programing
- Characteristics of Object Oriented Programing

Before We Begin

Grad Function Prototypes

- Comments
- Flexible Declarations
- Structure, union and enum Syntax
- Anonymous unions and enums
- Typecasting
- Void Pointers
- The: Operator
- References
- The const Qualifier
- Constructors for Intrinsic Data Types
- The bool Data Typecasting to C++

Functions

- Function Prototypes
- Function Overloading
- Default Arguments in Functions
- Operator Overloading
- Inline Functions
- Static, virtual and friend Functions

Classes in C++

- Classes and Constructors
- Destructors
- A Complex Class
- Overloaded Operators Revisited
- The this Pointer
- Overloading Unary Operators
- Function Definition Outside The Class
- Function Definition outside The Class
- New and delete Operators

- Using new and delete
- malloc ( )/free( ) versus new/ delete
- The Matrix Class
- Classes, Objects and Memory
- Structures and Classes

The C++ Free Store

- Free Store Exhaustion
- Custom new and delete Operators
- Overloading new/delete in Classes
- Understanding The sequence
- Construction at Predetermined Location
- One Last Issue

Miscellaneous Class Issue

- Static Class Data
- Static Member Functions
- const and Classes
- Overloaded Assignment Operator, Copy Constructor
- Data Conversion
- Data Conversion between Object of Different Classes

Data Structures through C++

- Stacks and Queues
- The Linked List
- Stacks and Queues Revisited
- Trees
- Binary Trees
- Traversal of a Binary Tree
- Deletion from a Binary Tree

Inheritance

- More Inheritance
- Some More Inheritance
- Multiple Levels of Inheritance
- Multiple Inheritance
- Constructors in Multiple Inheritance
- A Word of Caution
- Private Inheritance
- Protected Inheritance
- Functions That Are Not Inherited
- Incremental Development

Virtual Functions

- Pure Virtual Functions
- Virtual Functions under the Hood
- Why virtual Functions?
- Virtual Functions in Derived Classes
- Object Slicing
- Virtual Functions and Constructors
- Destructors and virtual Destructors
- Virtual Base Classes
- Putting it All Together

Input / Output in C++

- The iostream Library
- The ios Class
- Manipulators
- Creating Our Own Manipulators
- User-defined Manipulators with Arguments
- Come GUI and…
- The istream Class
- The ostream Class
- The iostream Class
- The with assign Classes
- Predefined Stream Objects
- Outputting Strings
- A Brief Review
- File I/O with Streams
- A Better way
- A File copy Program
- File Opening Modes
- Binary I/O
- Elementary Database Management
- Class That Read/Write Themselves
- Errors during I/O
- File copy Program Revisited
- Overloading <<and>>
- Str streams
- Automatic Storage Allocation
- Sending Output to Printer

Advanced Features

- Classes Within Classes
- friend Functions
- Overloading << and >>
- One More Use Of friend Function
- friend Classes

- A Word of Caution
- Smart Pointers
- More Smart Pointers
- Pointers to Members
- The explicit Keyword
- The mutable Keyword
- Namespaces
- Using A Namespace
- RTTI
- When to Use RTTI
- Typecasting in C++

Templates

- Function Templates
- A Template Based Quick Sort
- Class Templates
- A Linked List Class Template
- Tips about List Class Template

Exception Handling

- Checking Function Return Value
- setjmp( ) and longjmp( )
- Exception Handling in C++
- Exception with Arguments

Case Studies

- Tic Tac Toe Game
- Student Management System
- Student Attendance Management System
- Event Management System
- Hangman Game
- Employee Leave Management System
- Furniture Business System
- Society Management System

```cpp
#include <iostream>
using namespace std;
class one
{
      public:
            void display( )
            {
               cout<<endl<<"ln base class"<<endl;
             }
} ;
class oneofone : public one
{
        public :

              void disply( )
              {
                    cout<<endl<<"ln oneofone class"<<endl;
               }
} ;

class  twoofone : public one
{
             public :

                    void display( )
                    {
                          cout<<endl<<"ln twoofone class"<<endl;
                     }
} ;
```