# CHI TIẾT

# LỊCH VÀ CHI TIẾT NỘI DUNG KHÓA ĐÀO TẠO LẬP TRÌNH C++

*(Khóa đào tạo C++ dành cho Doanh Nghiệp)*

**Khóa đào tạo lập trình C++ –v2020**

Dưới đây là thông tin chi tiết lịch triển khai khóa đào tạo lập trình C++ dành cho đội ngũ nhân sự của Doanh nghiệp. Chi tiết như sau:

| Day | Date | Content | Duration | HomeWork |
|---|---|---|---|---|
| Day 1 | 6-May-2020 | **1. Git training:**<br>- Benefits of Version Control Systems?<br>- Local, Central and Distributed Version Control Systems?<br>- What is GIT?<br>- Why to use Git?<br>- Installing and learning Git?<br>- Getting Stated with Git?<br>- Recording Changes to the Respository?<br>- Viewing the commit history?<br>- Undo things - Unstaging a staged file - Unmodifying a Modified file?<br>- Tagging in Git?<br>- Git Alias<br>- Basic Branching and Merging<br>- Branching Workflows<br>- Rebasing<br>**2. C++ training:**<br>- Setting up C++ Development Environment<br>- Writing first C++ program(Practice)<br>- void main or main()<br>- C++ Data Types(Practice)<br>- Basic Input/Output<br>- Response on exceeding valid range of data types<br>- Operators in C++(Practice) | 08 hour | Theory and Practice with Git/C++ |

| | | | | |
|---|---|---|---|---|
| | | - Decision Making in C++(Practice)<br>- Execute both if and else simultaneously<br>- How to compile 32-bit program on 64-bit gcc in C and C++<br>- Switch statement in C++(Practice)<br>- Loops (Practice) | | |
| Day 2 | 8-May-2020 | **<u>1. Functions in C/C++</u>**<br>- Default Arguments in C++<br>- C++ function argument and return values<br>- Inline Functions in C++<br>- Return from void functions in C++<br>- std::tuple, std::pair \| Returning multiple values from a function using Tuple and Pair in C++<br>- A C++ Function Call Puzzle<br>- C/C++ Preprocessors<br>- Ceil and Floor functions in C++<br>- Functors in C++<br>- transform() function in C++<br>- Const member functions in C++<br>- atol(), atoll() and atof() functions in C/C++<br>- swap() in C++<br>- wmemmove() function in c++<br>- wcscat() function in C++<br>- wcscmp() function in C++ with Examples<br>- wcscpy() function in C++ with Examples<br>- wcslen() function in C++ with Examples<br>- difftime() function in C++<br>- asctime() function in C++<br>- localtime() function in C++<br>- scalbn() function in C++<br>- isunordered() function in C++<br>- isnormal() in C++<br>- isinf() function in C++<br>- quick_exit() function in C++ with Examples | 08 hour | Theory, Practice and Test with C++ *(Functions, Array and Object Oriented Programming)* |

| | | | | |
|---|---|---|---|---|
| | | - ctime() Function in C/C++<br>- clock() function in C/C++<br>- nearbyint() function in C++<br>- quick_exit() function in C++ with Examples<br>- wcscmp() function in C++ with Examples<br>- wcscpy() function in C++ with Examples<br>- wcslen() function in C++ with Examples<br>**2. Arrays and Strings**<br>- Arrays in C/C++<br>- Array of Strings in C++ (5 Different Ways to Create)<br>- Multidimensional arrays in C/C++<br>- Raw string literal<br>- Converting string to number and vice-versa<br>- Find size of array in C/C++ without using sizeof<br>- How to quickly reverse a string in C++?<br>- Getline() function and character array<br>- Convert string to char array in C++<br>- C++ string class and its applications , Set 2<br>- Print size of array parameter<br>- Stringstream in C++ and its applications<br>- Std::lexicographical_compare() in C++STL<br>- Std::string::at in C++<br>- Std::substr() in C/C++<br>- std::stol() and std::stoll() functions in C++<br>**3. Object Oriented Programming(OOP)**<br>- Object oriented design<br>- Introduction to OOP in C++<br>- Classes and Objects<br>- Access Modifiers | | |
| Day 3 | 11-<br>May- | **1. Object Oriented Programming(OOP)**<br>- Inheritance<br>- Polymorphism<br>- Encapsulation | 08 hour | Theory,<br>Practice and |

| | | | | |
|---|---|---|---|---|
| | 2020 | - Data Abstraction<br>- Structure vs class<br>- Can a C++ class have an object of self type?<br>- Why is the size of an empty class not zero?<br>- Static data members in C++<br>- Some interesting facts about static member functions<br>- Friend class and function<br>- Local Class<br>- Nested Classes<br>- Simulating final class<br>- Virtual Functions and Runtime Polymorphism in C++<br>- Multiple Inheritance in C++<br>- What happens when more restrictive access is given to a derived class method in C++?<br>- Object Slicing in C++<br>- Hiding of all overloaded methods with same name in base class<br>- Inheritance and friendship<br>- Simulating final class in C++<br>**2. Constructor and Destructor**<br>- Constructors<br>- Copy Constructor<br>- Destructors<br>- Does compiler create default constructor<br>- Initialization of data members<br>- Use of explicit keyword<br>- When do we use Initializer List in?<br>- Default Constructors<br>- Private Destructor<br>- Playing with Destructors<br>- Copy elision<br>- C++ default constructor \| Built-in types<br>- When does compiler create a default constructor and copy constructor?<br>- Why copy constructor argument should be const in C++? | | Test with C++ *(OOP, Constructor, Function Overloading, Operator Overloading)* |

| | | | | |
|---|---|---|---|---|
| | | - C++ Internals \| Default Constructors<br>- When are static objects destroyed?<br>- Is it possible to call constructor and destructor explicitly?<br>**3. Function Overloading**<br>- Functions that can't be overloaded<br>- Function overloading and const keyword<br>- Function overloading and return type<br>- Does overloading work with Inheritance?<br>- Can main() be overloaded<br>- Function Overloading and float<br>**4. Operator Overloading**<br>- Copy constructor vs assignment operator<br>- When should we write our<br>own assignment operator?<br>- Operators that cannot be overloaded<br>- Conversion Operators<br>- Is assignment operator inherited?<br>- Default Assignment Operator and References<br>- Overloading array index operator [] | | |
| Day 4 | 13-May-2020 | **1. Virtual Functions**<br>- Virtual Functions and Runtime Polymorphism<br>- Default arguments and virtual function<br>- Virtual functions in derived classes<br>- Can static functions be virtual?<br>- Virtual Destructor<br>- Virtual Constructor<br>- Virtual Copy Constructor<br>- RTTI (Run-time type information)<br>- Can virtual functions be private?<br>- Inline virtual function<br>- Pure Virtual Functions and Abstract Classes<br>- Pure virtual destructor<br>**2. Pointers and References** | 08 hour | Theory, Practice and Test with C++ *(Virtual, Pointer, Exception, Dynamic Memory)* |

| | | | | |
|---|---|---|---|---|
| | | - Pointers in C and C++<br>- What is Array Decay in C++? How can it be prevented?<br>- Opaque Pointer<br>- References<br>- Can references refer to invalid location?<br>- Pass arguments by reference or pointer<br>- Smart Pointers<br>- 'this' pointer<br>- Type of 'this' pointer<br>- "delete this"<br>- auto_ptr, unique_ptr, shared_ptr and weak_ptr<br>- Dangling, Void , Null and Wild Pointers<br>- Passing by pointer Vs Passing by Reference<br>- NaN in C++ – What is it and how to check for it?<br>- nullptr<br>- Pointers vs References in C++<br>**3. Exception Handling**<br>- Exception Handling Basics<br>- Stack Unwinding<br>- Catching base and derived classes as exceptions<br>- Catch block and type conversion<br>- Exception handling and object destruction<br>**4. Dynamic memory allocation**<br>- new and delete operator in C++<br>- malloc() vs new<br>- delete() and free()<br>- Std::get_temporary_buffer in C++ | | |
| Day 5 | 15-May-2020 | **1. Standard Template Library (STL)**<br>- Algorithms:<br>+ Introduction to STL<br>+ Sorting<br>+ Searching<br>- Containers: | 8 hour | Theory, Practice and Test with C++ *(STL,* |

| | | | | |
|---|---|---|---|---|
| | | + Pair (Practice)<br>+ Vector (Practice)<br>+ List<br>+ Dequeue<br>+ Queue (Practice)<br>+ Stack (Practice)<br>+ Set (Practice)<br>+ Map of pairs in STL<br>+ Map vs unordered_map in C++<br>+ Map (Practice)<br>+ Heap using STL C++<br>- Multimap in C++ Standard Template Library (STL)<br>**2. Namespace**<br>- Namespace in C++ \| Set 1 (Introduction)<br>- Set 2 (Extending namespace and Unnamed namespace)<br>- Namespace in C++ \| Set 3 (Accessing, creating header, nesting and aliasing)<br>- Inline namespaces and usage of the "using" directive inside namespaces<br>- Can namespaces be nested?<br>**3. C++ Library**<br>- <random> file – generators and distributions<br>- Array type manipulation<br>- C++ programming and STL facts<br>- Inbuilt library functions for user Input<br>- Rename function in C++<br>- Character Classification: cctype<br>- Snprintf() in C library<br>- Boost::split in C++ library<br>- Modulus of two float or double numbers<br>- Array sum in C++ STL<br>- exit() vs _Exit() in C and C++<br>- std::none_of in C++<br>- isprint() in C++<br>- partition_point in C++ | | *NameSpace, Library)* |

| | | | | | |
|---|---|---|---|---|---|
| | | - Array class<br>- Tuples<br>- Regex (Regular Expression)<br>- Lambda expression in C++<br>- C++ \| Signal Handling<br>- How to restrict dynamic allocation of objects in C++? | | | |
| | | | | | |

Hà Nội, Ngày…..Tháng….Năm 2020

PHÒNG ĐÀO TẠO IMIC TECHNOLOGY