



Nanodegree Program Syllabus

C++ Nanodegree Program Syllabus

Course 1: C++ Foundations

Course 2: Object-Oriented Programming

Course 3: Memory Management

Course 4: Concurrency

Capstone Project: Build Your Own C++ App

Course 1: C++ Foundations

Learn basic C++ syntax, functions, containers, and compiling and linking with multiple files. Use OpenStreetMap and the 2D visualization library IO2D to build a route planner that displays a path between two points on a map.

| | | |
|-----------------------------|---|---|
| Week One | Introduction to the C++ Language | In this week, you will build on your previous programming experience to learn the basics of the C++ language. By the end of this week, you will use vectors, loops, and I/O libraries to parse data from a file and print an ASCII board. You will use this board in the next lesson for a simplified route planning application. |
| Week Two | A* Search | In this week, you will learn about the A* search algorithm. Over a sequence of exercises, you will use your A* search implementation to plan a path through the obstacles in the ASCII board. |
| Mid-Course Project | ASCII A* Search | Provide a grid file with obstacles to your program. Your program will then be able to use A* search to find a path from the start to the end of the board through the obstacles. The program will also be able to print the solution to the screen with nice ASCII formatting. |
| Week Three | Writing Larger Programs and Using OpenStreetMap Data | In this week, you will learn the syntax for C++ language features that you will need for the next steps in the course. This includes an overview of header files, pointers, build tools, and classes. You'll also learn about OpenStreetMap data and look at the IO2D map display code that you will be extending for your project. |
| Week Four | A* with OpenStreetMap Data | Over a sequence of exercises during this week, you will extend the IO2D map display code to use A*, so your program will be able to find a path between two points on the map. |
| Course Final Project | Build an OpenStreetMap Route Planner | This project will combine the work that you have completed over the previous two weeks. When the project is finished, you will be able to select starting and ending areas on a city map, and your program will find a path along the city streets to connect the start and end. |

Course 2: Object-Oriented Programming

Explore Object-Oriented Programming (OOP) in C++ with examples and exercises covering the essentials of OOP like abstraction and inheritance all the way through to advanced topics like polymorphism and templates. In the end, you'll build a Linux system monitor application to demonstrate C++ OOP in action!

| | | |
|-----------------------------|---|---|
| Week One | Introduction to OOP In C++ | In the first lessons of this course, you'll meet your instructors, get some context for what object oriented programming (OOP) is and practice implementing some of the basic features of OOP, like encapsulation and abstraction. |
| Week Two | Access Modifiers and Inheritance | C++ classes have extensive functionality when it comes to what kind of members you can define within a class and how you can prevent or provide access to those members. In addition, like many other languages, one class can inherit from another. In this lesson you'll investigate the intricacies of access modifiers and inheritance to build more complex C++ classes. |
| Week Three | Polymorphism and Templates | With polymorphism, you can write member functions for a class that do different things depending on what parameters you pass to them. Using templates, you can write generic functions that accept many different kinds of input parameter types. With these tools you can add more diverse functionality to your C++ classes and in this lesson you'll do just that. |
| Course Final Project | System Monitor | In this project, you'll get a chance to put C++ OOP into action! You'll write a Linux system monitor with similar functionality to the widely used htop application. This will not only provide you more familiarity the Linux operating system, but also give you insights into how a collection of objects can function together in C++ to form an exciting and complete application! |

Course 3: Memory Management

Discover the complexity of memory management in C++ by diving deep into stack vs. heap, pointers, references, new, delete and much more. By the end, you'll write your very own smart pointer!

| | | |
|-----------------------------|---------------------------------------|---|
| Week One | Stack vs. Heap | Generally speaking memory management in C++ is divided into two main categories by their storage in the host machine, stack and heap. In this lesson, you'll learn to identify the difference between the two and how they are used in practice. |
| Week Two | Pointers and References | Pointers and references are used in C++ to give one variable access to the contents of another. While they share many similarities, their differences are important when it comes to implementation. In this lesson you'll get hands on practice with pointers and references and how best to use them. |
| Week Three | New, Delete, Memset and Malloc | The New, Delete, Memset and Malloc operators allow you to perform what's known as dynamic memory allocation in C++. This gives you the flexibility to allocate memory to objects of variable size, like linked lists or trees. In this lesson, you'll learn how to dynamically allocate and deallocate memory and avoid memory leaks! |
| Week Four | Smart Pointers | Smart pointers are the C++ solution to achieving better memory management in your code. Smart pointers automatically deallocate memory for objects that go out of scope and make it much easier for you as a programmer to avoid memory leaks. In this lesson, you'll get experience working with smart pointers and see how they can make your code more robust. |
| Course Final Project | Smart Pointers | The best way to synthesize all the memory management concepts you've worked with in this course is to build your own smart pointer and compare it to the native implementation. In this project you'll get to do just that! |

Course 4: Concurrency

Concurrent programming runs multiple threads of execution in parallel. Concurrency is an advanced programming technique that, when properly implemented, can dramatically accelerate your C++ programs.

| | | |
|-----------------------------|-------------------------------|--|
| Week One | Parallel Algorithms | Take advantage of parallel algorithms from the standard library. Many important libraries have parallel implementations that will handle parallelization themselves. Relying on these algorithms minimizes your work, and optimizes performance. |
| Week Two | Threads | Threads are parallel lines of execution that run independently but share common memory and state. Executing different tasks in different threads allows the system to distribute work to different processors and run concurrently. |
| Week Three | Thread Synchronization | Executing parallel threads that must communicate with each other is an advanced skill. This must be implemented carefully, to prevent synchronization errors. But successful implementation can significantly improve performance. |
| Course Final Project | Chatbot | Build a multithreaded chatbot that can carry on hundreds of conversations simultaneously! |

Build Your Own C++ Application

Put your C++ skills to use on a project of your own! You'll utilize the core concepts from this Nanodegree program - object-oriented programming, memory management, and concurrency - to build your own application using C++.

| | | |
|-----------------|----------------------|---|
| Week One | Capstone Work | Choose your application. Design the architecture. Build a prototype. |
| Week Two | Capstone Work | Complete your application, utilizing the core skills you have developed: C++ fundamentals, object-oriented programming, memory management, and concurrency. |