UDACITY

NANODEGREE PROGRAM SYLLABUS

C++

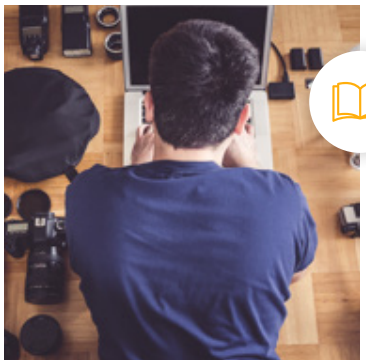
Overview

Learn C++, a high-performance programming language used in the world's most exciting engineering jobs -- from self-driving cars and robotics, to web browsers, media platforms, servers, and even video games.

Get hands-on experience by coding five real-world projects. Learn to build a route planner using OpenStreetMap data, write a process monitor for your computer, and implement your own smart pointers. Finally, showcase all your newfound skills by building a multithreaded traffic simulator and coding your own C++ application.

Prerequisite Knowledge: To optimize your chances of success in the C++ Nanodegree program, we recommend intermediate knowledge of any programming language.

Estimated Time: 4 Months at 10hrs/week



Prerequisites: Intermediate Programming

Flexible Learning: Self-paced, so you can learn on the schedule that works best for you.

Need Help?
udacity.com/advisor
Discuss this program with an enrollment advisor.

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.

Course Project
Build an OpenStreetMap Route Planner

You'll learn about OpenStreetMap data and look at IO2D map display code. 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. 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.

	LEARNING OUTCOMES	
LESSON ONE	**Introduction to the C++ Language**	• Build on your previous programming experience to learn the basics of the C++ language • 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.
LESSON TWO	**A* Search**	• Learn about the A* search algorithm. • Use your A* search implementation to plan a path through the obstacles in the ASCII board. The program will also be able to print the solution to the screen with clean ASCII formatting.
LESSON THREE	**Writing Multifile Programs**	• Learn the syntax for C++ language features • Complete an overview of header files, pointers, build tools, and classes

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!

Course 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!

	LEARNING OUTCOMES	
LESSON ONE	**Introduction to OOP in C++**	• Meet your instructors, get some context for what object oriented programming (OOP) is • Practice implementing some of the basic features of OOP, like encapsulation and abstraction.
LESSON 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.
LESSON THREE	**Polymorphism and Templates**	• Write member functions for a class that do different things depending on what parameters you pass to them. • Using templates, write generic functions that accept many different kinds of input parameter types. With these tools you will add more diverse functionality to your C++ classes

Course 3: Memory Management

Discover the power of memory management in C++ by diving deep into stack vs. heap, pointers, references, new, delete, smart pointers, and much more. By the end, you'll be ready to work on a chatbot using modern C++ memory management techniques!

Course Project
ChatBot

The ChatBot project creates a dialogue where users can ask questions about some aspects of memory management in C++. Your task will be to optimize the project with memory management in mind using modern concepts such as smart pointers and move semantics.

	LEARNING OUTCOMES	
LESSON ONE	**Overview of Memory Types**	• Understand the memory hierarchy in computer systems, which is the basis for efficient memory management • Cover basic concepts such as cache, virtual memory, and the structure of memory addresses. • Demonstrate how the debugger can be used to read data from memory
LESSON TWO	**Variables and Memory**	• In this section, the process memory model is introduced, which contains the two fundamental memory areas, heap and stack, which play an important role in C++. • Review the concepts of call-by-value and call-by-reference to lay the foundations for the memory-efficient passing of parameters.
LESSON THREE	**Dynamic Memory Allocation (The Heap)**	• This section introduces dynamic memory allocation on the heap. Understand the main difference between stack and heap - the latter requires the programmer to take decisions about the correct allocation and deallocation of memory. • Learn the commands malloc and free, as well as new and delete, that are available for allocation of memory. • Review some of the most common problems with manual memory management

	LEARNING OUTCOMES	
LESSON FOUR	**Resource Copying Policies**	• Customize resource copying using the Rule of Three. • Learn the basis for move semantics, lvalue and rvalue • Understand how the mechanism for memory efficient programming is one of the most important innovations in C++ and enables fast and low-cost data transfers between program scopes. • Understand the Rule of Five, which helps develop a thorough memory management strategy in your code.
LESSON FIVE	**Smart Pointers**	• Understand why smart pointers are a valuable tool for C++ programmers and how they help to avoid memory leaks and make it possible to establish a clear and concise resource ownership model. • Compare the three types of smart pointers in C++ • Learn how to transfer ownership from one program part to another using copy and move semantics.

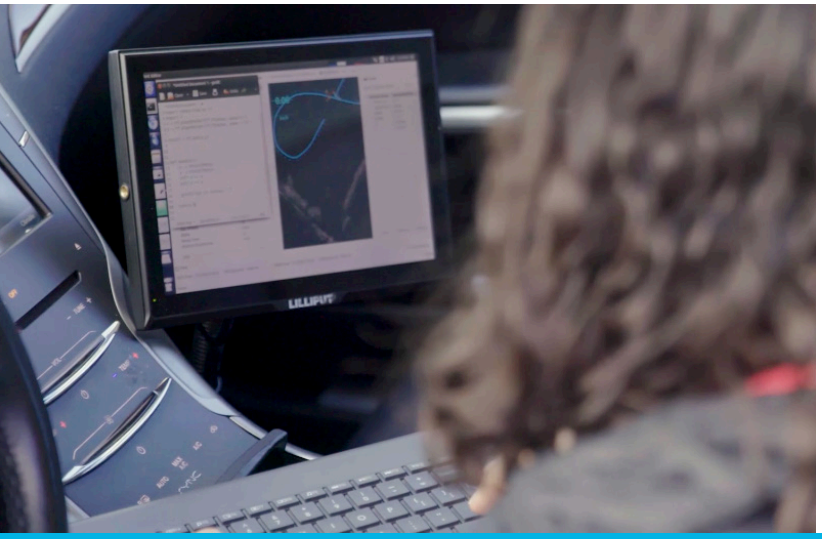
Course 4: Concurrency

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.

Course Project
Concurrent Traffic Simulation

Build a multithreaded traffic simulator using a real urban map. Run each vehicle on a separate thread, and manage intersections to facilitate traffic flow and avoid collisions using state-of-the-art concurrency concepts.

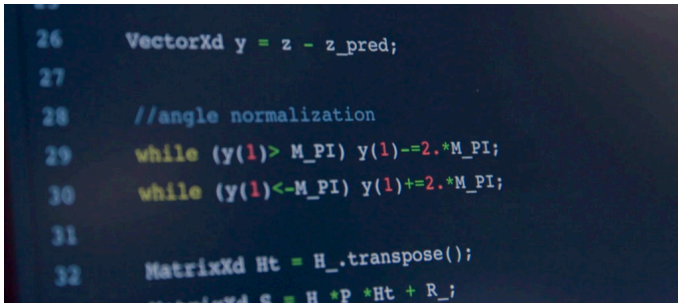
	LEARNING OUTCOMES	
LESSON ONE	**Managing Threads**	• Learn the differences between processes and threads. • Start your own threads in various ways and pass data to them. • Write your own concurrent program running multiple threads at the same time.
LESSON TWO	**Passing Data Between Threads**	• Use promises and futures as a safe communication channel between threads. • Use tasks as an easy alternative to threads. • Understand data races and learn about strategies to avoid them.
LESSON THREE	**Mutexes, Locks, and Condition Variables**	• Use mutexes and locks to safely access shared data from various threads. • Use condition variables as a basic synchronization tool between threads. • Understand and implement a concurrent message queue for flexible inter-thread communication.



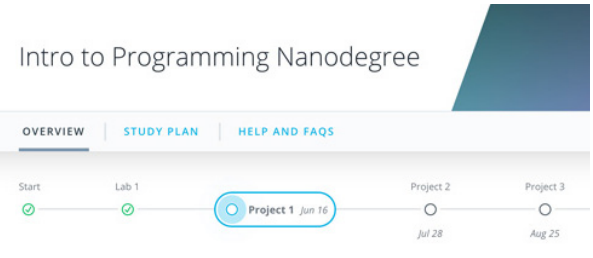
Capstone Project: Build Your Own C++ Application

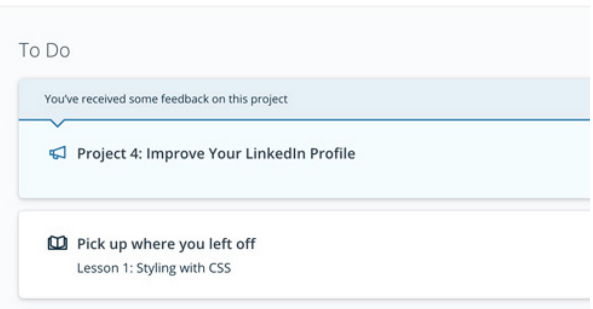
Put your C++ skills to use on a project of your own! You'll utilize the core concepts from this Nanodegree program - object-oriented programming, memory management, and concurrency - to build your own application using C++.

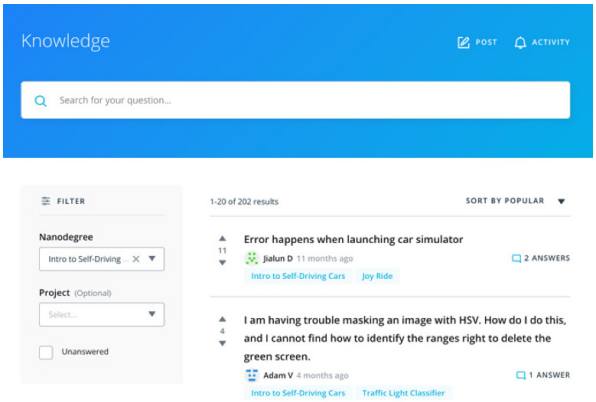
		LEARNING OUTCOMES
PART ONE	**Capstone Work**	• Choose your application. • Design the architecture. • Build a prototype.
PART TWO	**Capstone Work**	• Complete your application, utilizing the core skills you have developed: C++ fundamentals, object-oriented programming, memory management, and concurrency.

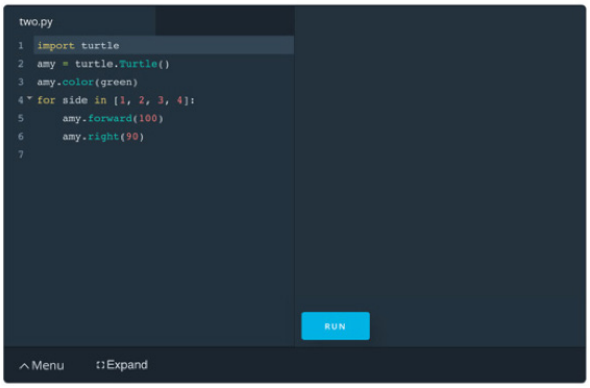


Our Classroom Experience









REAL-WORLD PROJECTS

Build your skills through industry-relevant projects. Get personalized feedback from our network of 900+ project reviewers. Our simple interface makes it easy to submit your projects as often as you need and receive unlimited feedback on your work.

KNOWLEDGE

Find answers to your questions with Knowledge, our proprietary wiki. Search questions asked by other students and discover in real-time how to solve the challenges that you encounter.

STUDENT HUB

Leverage the power of community through a simple, yet powerful chat interface built within the classroom. Use Student Hub to connect with your technical mentor and fellow students in your Nanodegree program.

WORKSPACES

See your code in action. Check the output and quality of your code by running them on workspaces that are a part of our classroom.

QUIZZES

Check your understanding of concepts learned in the program by answering simple and auto-graded quizzes. Easily go back to the lessons to brush up on concepts anytime you get an answer wrong.

CUSTOM STUDY PLANS

Work with a mentor to create a custom study plan to suit your personal needs. Use this plan to keep track of your progress toward your goal.

PROGRESS TRACKER

Stay on track to complete your Nanodegree program with useful milestone reminders.

Learn with the Best

David Silver

HEAD OF SELF-DRIVING CARS
AT UDACITY

David Silver leads the School of Autonomous Systems at Udacity. Before Udacity, David was a research engineer on the autonomous vehicle team at Ford. He has an MBA from Stanford, and a BSE in Computer Science from Princeton.

Stephen Welch

CONTENT DEVELOPER
AT UDACITY

Stephen is a Content Developer at Udacity and has worked on the C++ and Self-Driving Car Engineer Nanodegree programs. He started teaching and coding while completing a Ph.D. in mathematics, and has been passionate about engineering education ever since.

Ermin Kreponic

SOFTWARE ENGINEER
AT ABSTRACT THINKING

Ermin Kreponic is a skilled Java & C++ developer who has taught dozens of online courses in multiple coding languages. Ermin currently works as a cyber-security training architect and is a strong proponent of open-source technologies.

Andreas Haja

PROFESSOR
AT UNIVERSITY OF APPLIED SCIENCES

Andreas Haja is an engineer, educator, and autonomous vehicle enthusiast. Andreas now works as an engineering professor in Germany. Previously, he developed computer vision algorithms and autonomous vehicle prototypes using C++.

All Our Nanodegree Programs Include:

EXPERIENCED PROJECT REVIEWERS

REVIEWER SERVICES

- Personalized feedback & line by line code reviews
- 1600+ Reviewers with a 4.85/5 average rating
- 3 hour average project review turnaround time
- Unlimited submissions and feedback loops
- Practical tips and industry best practices
- Additional suggested resources to improve

TECHNICAL MENTOR SUPPORT

MENTORSHIP SERVICES

- Questions answered quickly by our team of technical mentors
- 1000+ Mentors with a 4.7/5 average rating
- Support for all your technical questions

PERSONAL CAREER SERVICES

CAREER COACHING

- Personal assistance in your job search
- Monthly 1-on-1 calls
- Personalized feedback and career guidance
- Access to Udacity Talent Program used by our network of employers to source candidates
- Advice on negotiating job offers
- Interview preparation
- Resume services
- Github portfolio review
- LinkedIn profile optimization

Frequently Asked Questions

PROGRAM OVERVIEW

WHY SHOULD I ENROLL?

C++ is a compiled, high-performance language. Robots, automobiles, and embedded software all depend on C++ for speed of execution. This program is designed to turn software engineers into C++ developers. You will use C++ to develop object-oriented programs, to manage memory and system resources, and to implement parallel programming.

WHAT JOBS WILL THIS PROGRAM PREPARE ME FOR?

C++ is the industry standard for high-performance computer programming. As such, advanced knowledge of this programming language can open the doors for you to work in a variety of industries, including C++ engineering, robotics software, IoT, mobile communications, video game development, operating systems, networking, AI, embedded systems, and more. Your opportunities and roles might include:

- C++ Software Developer
- Self-Driving Car Engineer
- Robotics Software Engineer
- Embedded Systems Engineer
- Entry-Level Game Programmer

HOW DO I KNOW IF THIS PROGRAM IS RIGHT FOR ME?

This program is right for you if you're an intermediate-level programmer familiar with functions and classes who wants to become a C++ developer or pursue a career in robotics software, IoT, mobile communications, video game development, operating systems, networking, AI, embedded systems, and more.

ENROLLMENT AND ADMISSION

DO I NEED TO APPLY? WHAT ARE THE ADMISSION CRITERIA?

There is no application. This Nanodegree program accepts everyone, regardless of experience and specific background.

WHAT ARE THE PREREQUISITES FOR ENROLLMENT?

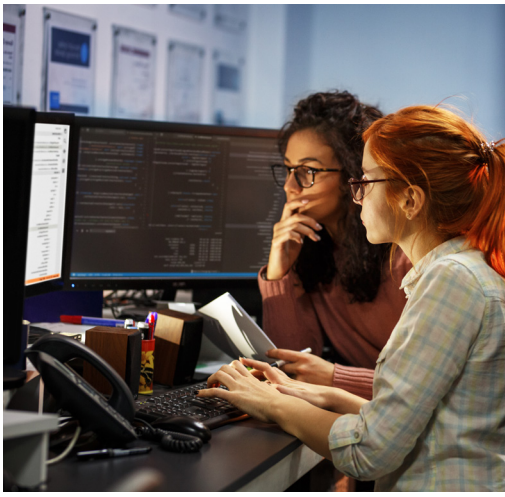
To optimize your chances of success in the C++ Nanodegree program, we recommend intermediate knowledge of any programming language.

IF I DO NOT MEET THE REQUIREMENTS TO ENROLL, WHAT SHOULD I DO?

For students who have little or no coding background, our Introduction to Programming Nanodegree program is an opportunity to learn object-oriented programming in Python. If you are interested in self-driving cars and have no programming experience, the Intro to Self-Driving Cars Nanodegree program will

FAQs Continued

teach you the basics of object-oriented programming in C++, as well as linear algebra and calculus.



TUITION AND TERM OF PROGRAM

HOW IS THIS NANODEGREE PROGRAM STRUCTURED?

The C++ Nanodegree program is comprised of content and curriculum to support five (5) projects. We estimate that students can complete the program in four (4) months, working 10 hours per week.

Each project will be reviewed by the Udacity reviewer network. Feedback will be provided and if you do not pass the project, you will be asked to resubmit the project until it passes.

HOW LONG IS THIS NANODEGREE PROGRAM?

Access to this Nanodegree program runs for the length of time specified in the payment card on the Nanodegree program overview page. If you do not graduate within that time period, you will continue learning with month to month payments. See the **Terms of Use** for other policies around the terms of access to our Nanodegree programs.

I HAVE GRADUATED FROM THE C++ NANODEGREE PROGRAM, WHERE SHOULD I GO FROM HERE?

We highly recommend the Self-Driving Car Engineer, Robotics Software Engineer, and Flying Car and Autonomous Flight Engineer Nanodegree programs. All of these programs use C++, and as a graduate of the C++ Nanodegree program, you'll have the coding skills necessary to succeed in these programs and the opportunity to specialize in specific areas of robotics and autonomous systems.

CAN I SWITCH MY START DATE? CAN I GET A REFUND?

Please see the Udacity Nanodegree program **FAQs** for policies on enrollment in our programs.

SOFTWARE AND HARDWARE

WHAT SOFTWARE AND VERSIONS WILL I NEED IN THIS PROGRAM?

For this Nanodegree program, you will code with C++17. An internet connection is required. All coding can be done in our GPU-enabled Linux Workspace that runs in your browser.