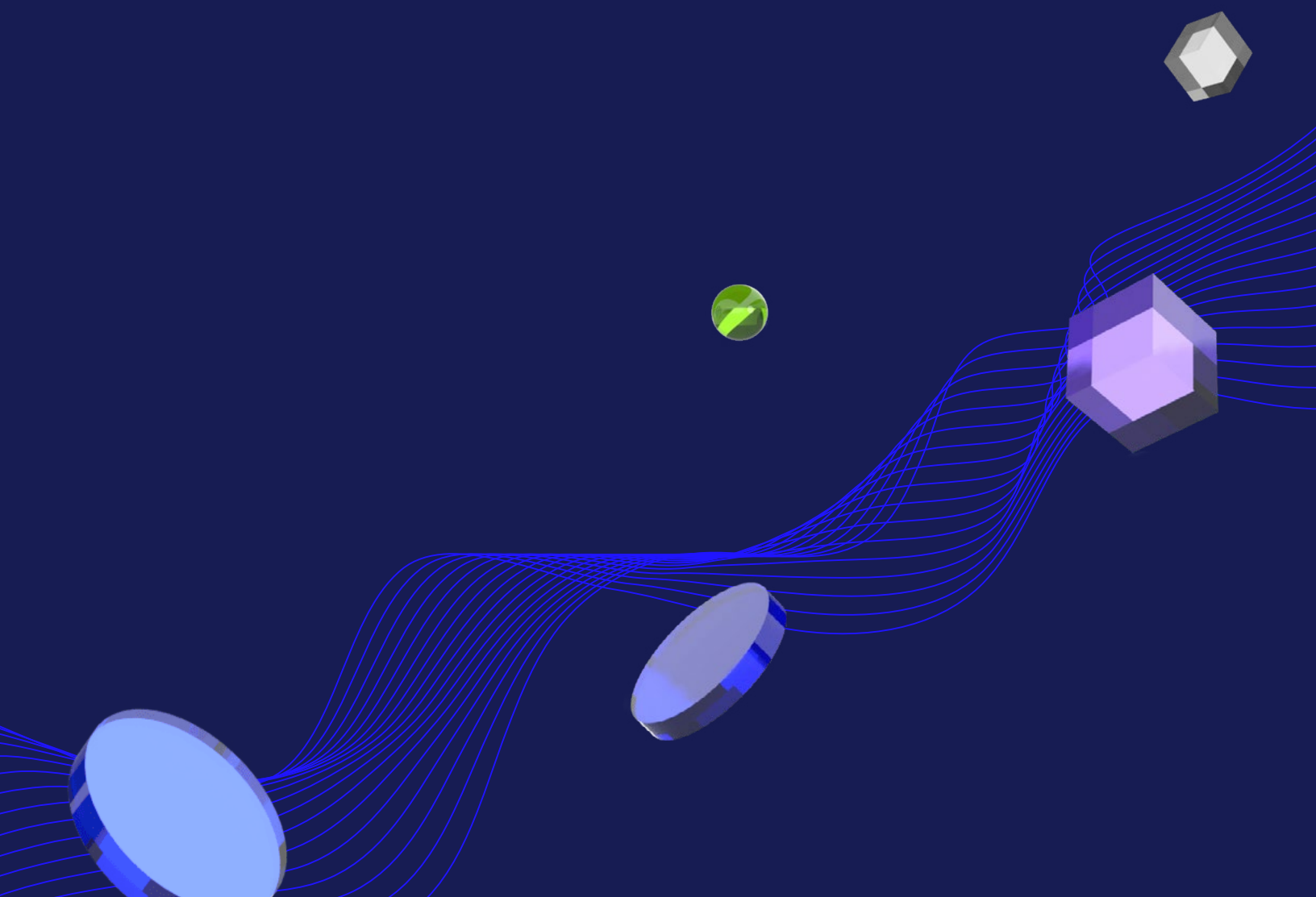
INDIVIDUAL LEARNERS

UDACITY

SCHOOL OF PROGRAMMING & DEVELOPMENT

# C++

## Nanodegree Program Syllabus



# 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 a computer, and implement smart pointers. Finally, showcase all these newfound skills by building a multithreaded traffic simulator and coding an original C++ application.

# Program information

### Estimated Time

4 months at 10hrs/week*

### Skill Level

Intermediate

### Prerequisites

A well-prepared learner should have intermediate knowledge of any programming language.

### Required Hardware/Software

Learners will need an internet connection and our GPU-enabled Linux Workspace that runs in a web browser.

*The length of this program is an estimation of total hours the average student may take to complete all required coursework, including lecture and project time. If you spend about 5-10 hours per week working through the program, you should finish within the time provided. Actual hours may vary.

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

Look a IO2D map display code. Extend the IO2D map display code to use A*, so the program will be able to find a path between two points on the map. When the project is finished, one will be able to select starting and ending areas on a city map, and the program will find a path along the city streets to connect the start and end.

Lesson 1

### Introduction to the C++ Language

- Build on 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. Use this board in the next lesson for a simplified route planning application.

Lesson 2

### A* Search

- Learn about the A* search algorithm.
- Use the 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 3

### 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, build a Linux system monitor application to demonstrate C++ OOP in action.

Course Project

## System Monitor

In this project, learners will get a chance to put C++ OOP into action! They'll write a Linux system monitor with similar functionality to the widely used htop application. This will not only provide learners more familiarity the Linux operating system, but also give them insights into how a collection of objects can function together in C++ to form an exciting and complete application.

Lesson 1

### Introduction to OOP in C++

- Meet the 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 2

### Access Modifiers & Inheritance

- C++ classes have extensive functionality when it comes to what kind of members one can define within a class and how one can prevent or provide access to those members. In addition, like many other languages, one class can inherit from another. In this lesson, learners will investigate the intricacies of access modifiers and inheritance to build more complex C++ classes.

Lesson 3

### Polymorphism & Templates

- Write member functions for a class that do different things depending on what parameters are passed to them.
- Using templates, write generic functions that accept many different kinds of input parameter types. With these tools learners will add more diverse functionality to their 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, learners will be ready to work on a chatbot using modern C++ memory management techniques.

Course Project

### ChatBot

The ChatBot project creates a dialogue where used can ask questions about some aspects of memory management in C++. Optimize the project with memory management in mind using modern concepts such as smart pointers and move semantics.

Lesson 1

## 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 2

## Variables & 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 3

## 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.

Lesson 4

## Resource Copying Policies

- Customize resource copying using the Rule of Three.
- Learn the basis for move semantics, lvalue, and value.
- 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 5

### 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 one's 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.

Lesson 1

### Managing Threads

- Learn the differences between processes and threads.
- Start one's own threads in various ways and pass data to them.
- Write one's own concurrent program running multiple threads at the same time.

---

Lesson 2

### 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 3

### Mutexes, Locks & 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.

Course 5

# Capstone Project

Utilize the core concepts from this Nanodegree program—object-oriented programming, memory management, and concurrency—to build an original application using C++.

Course Project

## Build Your Own C++ Application

- Choose an application.
- Design the architecture.
- Build a prototype.

Complete the application, utilizing the core skills developed: C++ fundamentals, object-oriented programming, memory management, and concurrency.

# Meet your instructors.

## David Silver

**Head of Curriculum**

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

**Instructor**

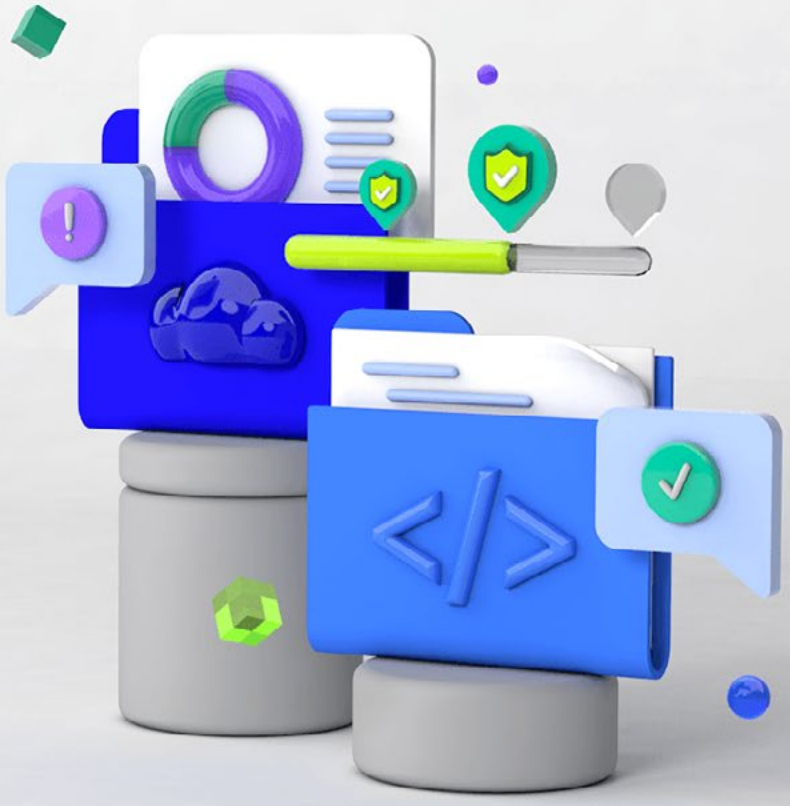
Stephen is a content developer at Udacity and has worked on the C++ and Self-Driving Car Engineer Nanodegree programs. He started teaching and coding while completing a PhD. in mathematics and has been passionate about engineering education ever since.

## Andreas Haja

**Instructor**

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++.

# Udacity's learning experience



### Hands-on Projects

Open-ended, experiential projects are designed to reflect actual workplace challenges. They aren't just multiple choice questions or step-by-step guides, but instead require critical thinking.

### Quizzes

Auto-graded quizzes strengthen comprehension. Learners can return to lessons at any time during the course to refresh concepts.

### Knowledge

Find answers to your questions with Knowledge, our proprietary wiki. Search questions asked by other students, connect with technical mentors, and discover how to solve the challenges that you encounter.

### Custom Study Plans

Create a personalized study plan that fits your individual needs. Utilize this plan to keep track of movement toward your overall goal.

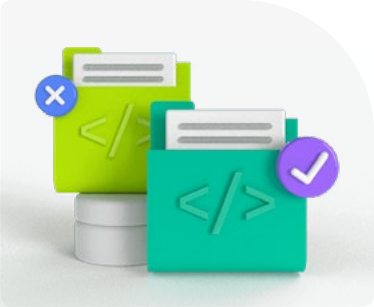
### Workspaces

See your code in action. Check the output and quality of your code by running it on interactive workspaces that are integrated into the platform.

### Progress Tracker

Take advantage of milestone reminders to stay on schedule and complete your program.

# Our proven approach for building job-ready digital skills.



### Experienced Project Reviewers

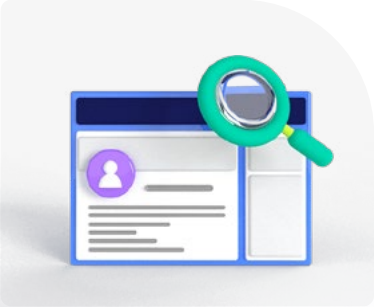
## Verify skills mastery.

- Personalized project feedback and critique includes line-by-line code review from skilled practitioners with an average turnaround time of 1.1 hours.
- Project review cycle creates a feedback loop with multiple opportunities for improvement—until the concept is mastered.
- Project reviewers leverage industry best practices and provide pro tips.

### Technical Mentor Support

## 24/7 support unblocks learning.

- Learning accelerates as skilled mentors identify areas of achievement and potential for growth.
- Unlimited access to mentors means help arrives when it's needed most.
- 2 hr or less average question response time assures that skills development stays on track.



### Personal Career Services

## Empower job-readiness.

- Access to a Github portfolio review that can give you an edge by highlighting your strengths, and demonstrating your value to employers.*
- Get help optimizing your LinkedIn and establishing your personal brand so your profile ranks higher in searches by recruiters and hiring managers.

### Mentor Network

## Highly vetted for effectiveness.

- Mentors must complete a 5-step hiring process to join Udacity's selective network.
- After passing an objective and situational assessment, mentors must demonstrate communication and behavioral fit for a mentorship role.
- Mentors work across more than 30 different industries and often complete a Nanodegree program themselves.

*Applies to select Nanodegree programs only.

UDACITY

Learn more at

**www.udacity.com/online-learning-for-individuals →**

12.22.22 | V1.0