

Introduction to Modern C++ Course Outline

Ryan Baker

January 15, 2025

- 1. Introduction and Setup**
- 2. C++ Programming Basics**
- 3. How C++ Works**
- 4. Introduction to OOP**
- 5. Advanced OOP**
- 6. Templates**
- 7. The C++ Standard Library**
- 8. Safety in C++**
- 9. Compile-Time Programming**

Week 1: Introduction and Setup

1	Course Introduction	2
1.1	Lecture Series Overview	2
2	Introduction to C++	2
2.1	Why C++?	2
2.2	Evolution of C++	2
2.3	C++ vs. Other Languages	2
3	Environment Setup	2
3.1	Tools Required	2
3.1.1	Text Editor	2
3.1.2	Compiler	2
3.2	Installation Guides	2
3.3	"Hello, World!" Example	2
4	Basic Syntax and Structure	2
4.1	<code>int</code> <code>main()</code>	2
4.2	Semicolons, <code>/* comments */</code> , and Whitespace	2
5	Datatypes and Variables	2
5.1	<code>sizeof</code> Operator	2
5.2	Primitive Types	2
5.3	Declaration and Definition	2
5.3.1	Assignment Operator <code>=</code>	2
5.3.2	Brace Initialization <code>{}</code>	2
5.4	Arithmetic Operators <code>+</code> , <code>-</code> , <code>*</code> , <code>/</code> , <code>%</code>	2

Week 2: C++ Programming Basics

1	I/O with iostream	2
1.1	std::cout	2
1.2	std::cin	2
2	Functions	2
2.1	Introduction to Functions	2
2.1.1	return Keyword	2
2.2	Function Overloading	2
2.3	Scope	2
2.3.1	Defining a Scope	2
2.3.2	Types of Scope	2
3	Conditions and Branches	2
3.1	Boolean Statements	2
3.1.1	bool() Casts	2
3.1.2	Comparison Operators ==, !=, <, <=, >, >=	2
3.1.3	Logical Operators !, &&, ——	2
3.2	if Statements	2
3.3	switch Statements	2
3.4	Ternary Operator (? :)	2
4	Loops	2
4.1	while Loops	2
4.1.1	do while Loops	2
4.2	for Loops	2
4.3	Control Flow Statements	2
4.3.1	break Keyword	2
4.3.2	continue Keyword	2

Week 3: How C++ Works

1	The Build Process	2
1.1	The Preprocessor	2
1.1.1	Text Replacement <code>#define</code>	2
1.1.2	Conditional Compilation <code>#if</code> , <code>#ifdef</code>	2
1.1.3	File Inclusion <code>#include</code>	2
1.2	The Compiler	2
1.2.1	Compilation Errors	2
1.3	The Linker	2
1.3.1	Linker Errors	2
2	Memory and Pointers	2
2.1	Introduction to Memory	2
2.2	Pointers	2
2.2.1	NULL Pointers	2
2.2.2	Pointer Arithmetic	2
2.2.3	Pointers to Pointers	2
2.3	References	2
3	Memory Segments	2
3.1	Text Segment	2
3.2	Static Memory	2
3.2.1	<code>static</code> Keyword	2
3.2.2	Initialized vs. Uninitialized Static Data	2
3.3	Heap Segment	2
3.3.1	Operators <code>new</code> and <code>delete</code>	2
3.3.2	Memory Leaks	2
3.4	Stack Segment	2
3.4.1	Stack Pointer	2

Week 4: Introduction to OOP

1	Arrays	2
1.1	Arrays and Pointers	2
1.2	Multidimensional Arrays	2
1.3	Array Initialization	2
1.4	Dynamic Arrays	2
2	Structs	2
2.1	Struct Initialization	2
3	Classes	2
3.1	Constructors and Destructors	2
3.1.1	Initializer Lists	2
3.1.2	Default Initialization	2
3.1.3	Copy Constructors	2
3.2	Access Specifiers	2
3.2.1	<code>private</code> Members	2
3.2.2	<code>protected</code> Members	2
3.2.3	<code>public</code> Members	2
3.2.4	Structs vs. Classes	2
3.3	<code>static</code> Members	2

Week 5: Advanced OOP

1	Principles of OOP	2
1.1	Abstraction	2
1.2	Encapsulation	2
1.3	Inheritance	2
1.3.1	virtual Functions	2
1.3.2	Interfaces	2
1.4	Polymorphism	2
1.5	Composition	2
2	Operator Overloading	2
2.1	Type Casting	2
2.2	friend Functions	2
3	Design Patterns	2
3.1	Creational Design Patterns	2
3.1.1	Singleton	2
3.1.2	Factory	2
3.2	Behavioral Design Patterns	2
3.2.1	Strategy	2
3.3	Structural Design Patterns	2
3.3.1	Adapter	2

Week 6: Templates

1	Introduction to Templates	2
2	Function Templates	2
2.1	Implicit Template Deduction	2
2.2	Template Function Overloading	2
2.3	Function Template Specialization	2
3	Class Templates	2
3.1	Class Template Instantiation	2
3.2	Class Template Specialization	2
4	Non-Type Template Parameters	2
5	Variadic Templates	2

Week 7: The C++ Standard Library

1	Standard Containers	2
2	Iterators	2
3	Ranges and Views	2

Week 8: Safety in C++

1	Undefined Behavior	2
2	Memory Safety with Smart Pointers	2
3	Exception Safety	2

Week 9: Compile-Time Programming

1	Lambdas	2
2	Compile-Time Programming	2
3	Template Metaprogramming	2