

# 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

<b>1</b>	<b>Course Introduction</b>	<b>2</b>
1.1	Lecture Series Overview . . . . .	2
<b>2</b>	<b>Introduction to C++</b>	<b>2</b>
2.1	Why C++? . . . . .	2
2.2	Evolution of C++ . . . . .	2
2.3	C++ vs. Other Languages . . . . .	2
<b>3</b>	<b>Environment Setup</b>	<b>2</b>
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
<b>4</b>	<b>Basic Syntax and Structure</b>	<b>2</b>
4.1	<code>int</code> <code>main()</code> . . . . .	2
4.2	Semicolons, <code>/* comments */</code> , and Whitespace . . . . .	2
<b>5</b>	<b>Datatypes and Variables</b>	<b>2</b>
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

<b>1</b>	<b>I/O with <code>iostream</code></b>	<b>2</b>
1.1	<code>std::cout</code> . . . . .	2
1.2	<code>std::cin</code> . . . . .	2
<b>2</b>	<b>Functions</b>	<b>2</b>
2.1	Introduction to Functions . . . . .	2
2.1.1	<code>return</code> 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
<b>3</b>	<b>Conditions and Branches</b>	<b>2</b>
3.1	Boolean Statements . . . . .	2
3.1.1	<code>bool()</code> Casts . . . . .	2
3.1.2	Comparison Operators <code>==</code> , <code>!=</code> , <code>&lt;</code> , <code>&lt;=</code> , <code>&gt;</code> , <code>&gt;=</code> . . . . .	2
3.1.3	Logical Operators <code>!</code> , <code>&amp;&amp;</code> , <code>——</code> . . . . .	2
3.2	<code>if</code> Statements . . . . .	2
3.3	<code>switch</code> Statements . . . . .	2
3.4	Ternary Operator ( <code>?</code> : ) . . . . .	2
<b>4</b>	<b>Loops</b>	<b>2</b>
4.1	<code>while</code> Loops . . . . .	2
4.1.1	<code>do while</code> Loops . . . . .	2
4.2	<code>for</code> Loops . . . . .	2
4.3	Control Flow Statements . . . . .	2
4.3.1	<code>break</code> Keyword . . . . .	2
4.3.2	<code>continue</code> Keyword . . . . .	2

## Week 3: How C++ Works

<b>1</b>	<b>The Build Process</b>	<b>2</b>
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
<b>2</b>	<b>Memory and Pointers</b>	<b>2</b>
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
<b>3</b>	<b>Memory Segments</b>	<b>2</b>
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

<b>1</b>	<b>Arrays</b>	<b>2</b>
1.1	Arrays and Pointers . . . . .	2
1.2	Multidimensional Arrays . . . . .	2
1.3	Array Initialization . . . . .	2
1.4	Dynamic Arrays . . . . .	2
<b>2</b>	<b>Structs</b>	<b>2</b>
2.1	Struct Initialization . . . . .	2
<b>3</b>	<b>Classes</b>	<b>2</b>
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

<b>1</b>	<b>Principles of OOP</b>	<b>2</b>
1.1	Abstraction . . . . .	2
1.2	Encapsulation . . . . .	2
1.3	Inheritance . . . . .	2
1.3.1	<b>virtual</b> Functions . . . . .	2
1.3.2	Interfaces . . . . .	2
1.4	Polymorphism . . . . .	2
1.5	Composition . . . . .	2
<b>2</b>	<b>Operator Overloading</b>	<b>2</b>
2.1	Type Casting . . . . .	2
2.2	<b>friend</b> Functions . . . . .	2
<b>3</b>	<b>Design Patterns</b>	<b>2</b>
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

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

## **Week 7: The C++ Standard Library**

<b>1</b>	<b>Standard Containers</b>	<b>2</b>
<b>2</b>	<b>Iterators</b>	<b>2</b>
<b>3</b>	<b>Ranges and Views</b>	<b>2</b>



## 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**

<b>1</b>	<b>Lambdas</b>	<b>2</b>
<b>2</b>	<b>Compile-Time Programming</b>	<b>2</b>
<b>3</b>	<b>Template Metaprogramming</b>	<b>2</b>