These are the slides for the Beginning C++ Programming - From Beginner to Beyond on Udemy. They are provided free of charge to all students.

More information about the course: https://lpa.dev/u1bcppp

If you have any questions of queries, please add your feedback in the Q&A section of the course on Udemy.

Best regards,

Tim Buchalka
Learn Programming Academy

Beginning C++ Programming Slides

Main Course Slides.

Welcome and Introduction to the Course

- About me
- My assumptions
- Your background
- The curriculum and Modern C++
- Practice!
- Please ask questions

Why Learn C++?

- Popular
  - Lots of code is still written in C++
  - Programming language popularity indexes
  - Active community, GitHub, stack overflow
- Relevant
  - Windows, Linux, Mac OSX, Photoshop, Illustrator, MySQL, MongoDB, Game engines, more …
  - Amazon, Apple, Microsoft, PayPal, Google, Facebook, MySQL, Oracle, HP, IBM, more…
  - VR, Unreal Engine, Machine learning, Networking & Telecom, more…
- Powerful
  - fast, flexible, scalable, portable
  - Procedural and Object-oriented
- Good career opportunities
  - C++ skills always in demand
  - C++ = Salary++

Modern C++ and the C++ Standard

- Early 1970s
    - C Programming Language
    - Dennis Ritchie
- 1979
    - Bjarne Stroustrup
    - C with Classes
- 1983
    - Name changed to C++
- 1989
    - First commercial release
- 1998
    - C++98 Standard
- 2003
    - C++03 Standard
- **2011**
    - **C++11 Standard**
- **2014**
    - **C++14 Standard**
- **2017**
    - **C++17 Standard**

Modern C++ and the C++ Standard

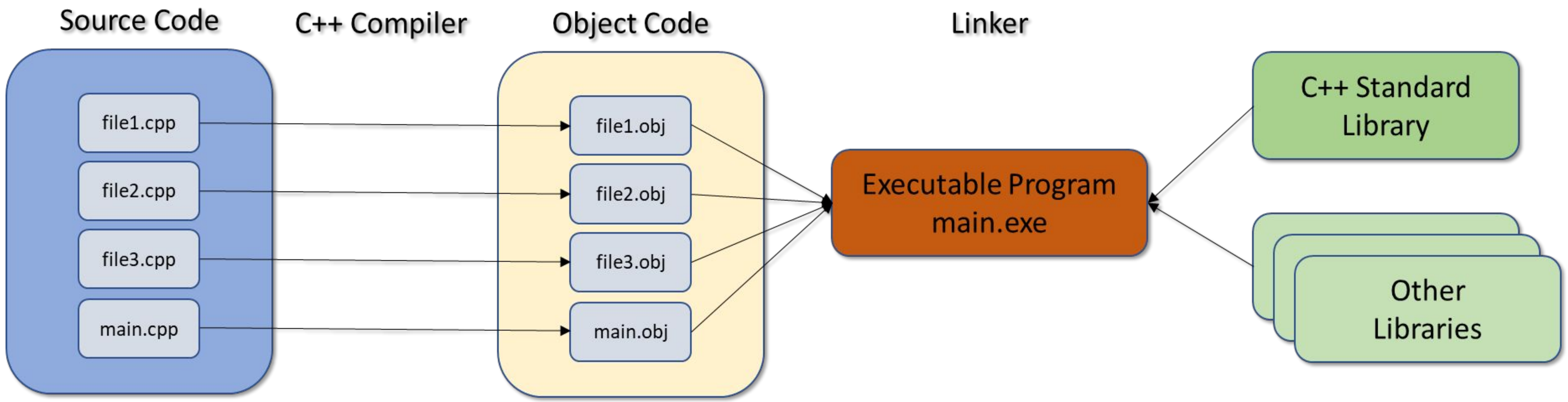
- Classical C++
  - Pre C++11 Standard

- Modern C++
  - C++11
    - Lots of new features
  - C++14
    - Smaller changes
  - C++17
    - Simplification
  - Best practices
  - Core Guidelines

How does it all work?

- You must tell the computer EXACTLY what to do
  - Program – like a recipe
- Programming language
  - source code
  - high-level
  - for humans
- Editor – used to enter program text
  - .cpp and .h files
- Binary or other low-level representation
  - object code
  - for computers
- Compiler – translates from high-level to low-level
- Linker – links together our code with other libraries
  - Creates executable program
- Testing and Debugging – finding and fixing program errors

The C++ Build Process



Integrated Development Environments (IDEs)

- Editor
- Compiler
- Linker
- Debugger
- Keep everything in sync

- CodeLite
- Code::Blocks
- NetBeans
- Eclipse
- CLion
- Dev-C++
- KDevelop
- Visual Studio
- Xcode

Section Overview

- Microsoft Windows, Mac OSX, Ubuntu Linux 18.04
  - C++ Compiler
  - CodeLite Integrated Development Environment (IDE)
  - Configure CodeLite
  - Create a Default CodeLite Project Template

- Using the Command-line

- Using a Web-based compiler

Please check the **Resources** for the videos as I will post updates there as needed.

Using the command-line interface

- A text editor (not a Word Processor)
- A command-prompt or terminal window
- An installed C++ compiler

- No need for an IDE
- Simple, efficient workflow
- Better as you gain experience
- Can be used if you are overwhelmed by IDEs
- Useful if hardware resources are limited

Curriculum Overview

- Getting Started
- Structure of a C++ Program
- Variables and Constants
- Arrays and Vectors
- Strings in C++
- Expressions, Statements and Operators
- Statements and Operators
- Determining Control Flow
- Functions
- Pointers and References
- OOP – Classes and Objects
- Operator Overloading
- Inheritance
- Polymorphism
- Smart Pointers
- The Standard Template Library (STL)
- I/O Streams
- Exception Handling

Curriculum Overview

Challenge Exercises

- At the end of most course sections
- Develop real C++ programs using what we discussed in the section
- Section challenges
  - Description
  - Starting project
  - Completed solution
- Have fun and keep coding!

Curriculum Overview

Quizzes

- At the end of most sections

- Reinforce the concepts learned in each section

- Quiz style
  - Multiple choice
  - Fill in blank
  - Concept oriented vs. code oriented

Section Overview

- CodeLite IDE Quick Tour
- Our first program
  - Building
  - Running
  - Errors
  - Warnings

Compiler Errors

- Programming languages have rules
- Syntax errors – something wrong with the structure

```
std::cout << "Errors << std::endl;

return 0
```

- Semantic errors – something wrong with the meaning

`a + b;` When it doesn't make sense to add `a` and `b`

Compiler Warnings

Do NOT ignore them!

- The compiler has recognized an issue with your code that could lead to a potential problem
- It's only a warning because the compiler is still able to generate correct machine code

```
int miles_driven;
  std::cout << miles_driven;
```

warning: 'miles_driven' is used uninitialized ...

Linker Errors

- The linker is having trouble linking all the object files together to create an executable

- Usually there is a library or object file that is missing

Runtime Errors

- Errors that occur when the program is executing
- Some typical runtime errors
  - Divide by zero
  - File not found
  - Out of memory

Can cause your program to ‘crash’

Exception Handling can help deal with runtime errors

Logic Errors

- Errors or bugs in your code that cause your program to run incorrectly
- Logic errors are mistakes made by the programmer

Suppose we have a program that determines if a person can vote in an election and you must be 18 years or older to vote.

```
if (age > 18) {
  std::cout << "Yes, you can vote!";
}
```

Test your code!!

Section Overview

The Structure of a C++ Program

- Basic Components
- Preprocessor Directives
- The main function
- Namespaces
- Comments
- Basic I/O

The Structure of a C++ Program

Components of a C++ Program

- Keywords
- Identifiers
- Operators
- Punctuation
- Syntax

Keywords

- Have special meaning in C++
- Are reserved by the C++ language

```
#include <iostream>

int main() {

    int favorite_number;

    std::cout << "Enter your favorite number between 1 and 100: ";
    std::cin >> favorite_number;

    std::cout << "Amazing!! That's my favorite number too!" << std::endl;

    std::cout << "No really!!, " << favorite_number << " is my favorite number!" <<std::endl;

    return 0;
}
```

Identifiers

- Programmer-defined names
- Not part of the C++ language
- Used to name variables, functions, etc.

```
#include <iostream>

int main() {

    int favorite_number;

    std::cout << "Enter your favorite number between 1 and 100: ";
    std::cin >> favorite_number;

    std::cout << "Amazing!! That's my favorite number too!" << std::endl;

    std::cout << "No really!!, " << favorite_number << " is my favorite number!" <<std::endl;

    return 0;
}
```

Operators

- Arithmetic operators, assignment, <<, >>
- Are reserved by the C++ language

```
#include <iostream>

int main() {

    int favorite_number;

    std::cout << "Enter your favorite number between 1 and 100: ";
    std::cin >> favorite_number;

    std::cout << "Amazing!! That's my favorite number too!" << std::endl;

    std::cout << "No really!!, " << favorite_number << " is my favorite number!" <<std::endl;

    return 0;
}
```

Punctuation

- Special characters that separate, terminate items

```
#include <iostream>

int main() {

    int favorite_number;

    std::cout << "Enter your favorite number between 1 and 100: ";
    std::cin >> favorite_number;

    std::cout << "Amazing!! That's my favorite number too!" << std::endl;

    std::cout << "No really!!, " << favorite_number << " is my favorite number!" <<std::endl;

    return 0;
}
```

Syntax

- How the programming elements are put together to form a program
- Programming languages have rules

```
#include <iostream>

int main() {

    int favorite_number;

    std::cout << "Enter your favorite number between 1 and 100: ";
    std::cin >> favorite_number;

    std::cout << "Amazing!! That's my favorite number too!" << std::endl;

    std::cout << "No really!!, " << favorite_number << " is my favorite number!" <<std::endl;

    return 0;
}
```

Our Section 4 Challenge Solution

Modified slightly

- Added comments
- Using namespace instead of std::

```
// Preprocessor directive that include the iostream library headers
#include <iostream>

// use the std namespace
using namespace std;

/* Start of the program
 * program execution always begins with main()
 */
int main() {

    int favorite_number;    // declare my favorite number variable

    // Note that I'm no longer using std::
    // Prompt the user to enter their favorite number

    cout << "Enter your favorite number between 1 and 100: ";

    // read the user's input into the variable favorite_number
```

Our Section 4 Challenge Solution

Modified slightly

```
// Preprocessor directive that include the iostream library headers
#include <iostream>

// use the std namespace
using namespace std;

/* Start of the program
 * program execution always begins with main()
 */
int main() {

    int favorite_number;    // declare my favorite number variable

    // Note that I'm no longer using std::
    // Prompt the user to enter their favorite number

    cout << "Enter your favorite number between 1 and 100: ";

    // read the user's input into the variable favorite_number
    cin >> favorite_number;

    // Out put the results to the user
    cout << "Amazing!! That's my favorite number too!" << endl;
    cout << "No really!!, " << favorite_number << " is my favorite number!" << endl;

    return 0;
}
```

Preprocessor Directives

- What is a preprocessor?
- What does it do?
- Directives start with '#'
- Commands to the preprocessor

```
#include <iostream>
#include "myfile.h"

#if
#elif
#else
#endif
```

```
#ifdef
```

Comments

- Ignored by the compiler
- Used to explain your code
- Two styles

```
int favorite_number;  // This will store my favorite number

// The following lines convert Euros to US Dollars

/* This is a comment
   that spans multiple lines.
   All of these lines will be ignored by the compiler
*/

// Why should be have to explain our own code?
```

Functions

- main() is a required function in C++
- Break up your code into units of functionality
- Can optionally receive and return information

```
/* This is a function that expects two integers a and b
   It calculates the sum of a and b and returns it to the caller
   Note that we specify that the function returns an int
*/

int add_numbers(int a, int b)
{
    return a + b;
}

// I can call the function and use the value that is returns

cout << add_numbers(20, 40);
```

Preprocessor Directives

- What is a preprocessor?
- What does it do?
- Directives start with '#'
- Commands to the preprocessor

```
#include <iostream>
#include "myfile.h"

#if
#elif
#else
#endif


#ifdef
#ifndef
#define
#undef

#line
#error
#pragma
```

The main() function

- Every C++ program must have exactly 1 main( ) function
- Starting point of program execution
- return 0 indicates successful program execution
- 2 versions that are both valid

```
int main()
{
   // code

   return 0;
}


program.exe

int main(int argc, char *argv[])
{
   // code

   return 0;
}


program.exe argument1 argument2
```

Namespaces

- Why `std::cout` and not just `cout`?
- What is a naming conflict?
- Names given to parts of code to help reduce naming conflicts
- `std` is the name for the C++ 'standard' namespace
- Third-party frameworks will have their own namespaces
- Scope resolution operator ::
- How can we use these namespaces?

Explicitly using namespaces

```
#include <iostream>

int main()
{
    int favorite_number;

    std::cout << "Enter your favorite number between 1 and 100: ";

    std::cin >> favorite_number;

    std::cout << "Amazing!! That's my favorite number too!" << std::endl;
    std::cout << "No really!!, " << favorite_number
              << " is my favorite number!" << std::endl;

    return 0;
 }
```

The using namespace directive

```
#include <iostream>

using namespace std;     // Use the entire std namespace

int main()
{
    int favorite_number;

    cout << "Enter your favorite number between 1 and 100: ";

    cin >> favorite_number;

    cout << "Amazing!! That's my favorite number too!" << endl;
    cout << "No really!!, " << favorite_number
         << " is my favorite number!" << endl;

    return 0;
 }
```

Qualified using namespace variant

```
#include <iostream>

using std::cout;  // use only what you need
using std::cin;
using std::endl;

int main()
{
    int favorite_number;

    cout << "Enter your favorite number between 1 and 100: ";

    cin >> favorite_number;

    cout << "Amazing!! That's my favorite number too!" << endl;
    cout << "No really!!, " << favorite_number
         << " is my favorite number!" << endl;

    return 0;
 }
```