Programming Language Research Assignment 2

Algonquin College

School of Advanced Technology
Computer Programming

Onur Önel
Mazin Abou-Seido

23F_CST8333_360

Submitted September 30, 2023

A technical report submitted to Algonquin College in partial fulfillment of the requirements for a diploma in Computer Programming

Abstract:

This Assignment delves into the insights gained from exploration of a C++ programming language, focusing on the language's characteristics and my personal learning experience. It also presents a comprehensive evaluation of various learning resources, detailing their time consumption and usefulness. Furthermore, it offers a reflection on the accuracy of time estimations during the first half of the course. Lastly, the document archives all the discussion board posts from the first half, for easy reference and analysis.

Table of Contents

Learning Insights about C++

C++ is a high-speed, object-oriented programming language. While it is often described as a "low-level" language, this highlights its capability to work closely with hardware components and perform system-level programming. I started off programming in Java, switching to C++ was a noticeable change. Unlike Java, which compiles from bytecode and runs on the Java Virtual Machine (JVM), C++ directly compiles into machine code for a specific platform. C++ compilation process includes:

- **Preprocessing:** Before actual compilation, the source code undergoes preprocessing. This step handles directives such as #include and #define, expanding macros and including header files.

- **Compilation:** The preprocessed source code is then compiled into an object file. At this stage, the compiler checks for syntax errors and translates the code into assembly language.

- **Assembly:** The assembler takes the assembly code produced in the compilation step and converts it into machine code, resulting in an object file.

- **Linking:** This is a crucial step in the C++ compilation process. All the object files, possibly from different sources, are linked together along with necessary library files to produce a single executable. The linker resolves references, ensuring that functions or variables referenced from one object file are correctly connected to their definitions in another.

- **Execution:** The final executable can then be run on the target machine or platform.

C++ is an evolution of the C language, included with exciting features to support high-speed applications. In comparison, it holds similarities with C, but also has similarities of higher-level languages like Java. However, unlike interpreted languages such as Python and JavaScript, C++ is compiled

C++ Topics That I Learned

During the course of completing my assignments and delving into my own independent research, I've come across and explored a range of intriguing features that have piqued my interest, including:

- **Standard Library**

  C++ has its own Standard Library, which includes essential classes and functions for tasks like data structures and input/output operations and more.

- **External Libraries**

  To recompense library limitations, C++ developers can use numerous external libraries, such as Boost and Poco, to extend the language's capabilities.

- **Memory Management:**

  Unlike Java's garbage collector C++ employs manual memory management. Developer must allocate and free memory using with keywords *new* and *delete*.

- **Pointers and References:**

  C++ supports both pointers and references, granting developers a high degree of control over memory and object manipulation.

I decided to go with C++ because it's important for me to deeply understand the subjects I dive into. C++ doesn't just teach me how to code; it shows me the connection between programming, computers, and mathematics. By getting a good handle on C++, I believe I'm setting myself up for success in computer science. This knowledge will give me a comprehensive view of both the software side of things and how the underlying hardware operates. It's like building a strong foundation for a house; with a solid base, everything else becomes more stable and reliable.

Weaknesses of the Language

The primary drawback of learning C++ as a programming language is its complexity. It can take a very long time to become proficient in C++ if not familiar with another object-oriented programming language.

The usage of pointers is one of the main characteristics that set C++ apart from other object-oriented programming languages. When working on big projects, pointers might consume a lot of system memory, which is not desirable. Another big drawback of pointers is that pointers can be difficult to understand and use correctly, using pointers wrongly can cause a result of complex issues and unexpectedly behaviours.

Using an object-oriented programming language such as C++ has a number of security risks due to features like public functions, global variables, and pointers, which is another drawback of the language.

While many people consider having control over memory management to be advantageous, allocating memory manually using pointers can be very time-consuming and easily forgotten when working on code. One major issue of C++ is that it requires manual memory management because it lacks a garbage collector feature that would automatically remove unnecessary data.

Lastly, built-in code threads are not supported by C++. Given that most other languages have this feature, which can make process slower and more complex.

References:

1. Learn C++, https://www.learncpp.com/

2. "C++ compilation process," PrepBytes Blog, https://www.prepbytes.com/blog/cpp-programming/cpp-compilation-process/#:~:text=In%20conclusion%2C%20the%20C%2B%2B%20compilation,run%20on%20the%20target%20system.

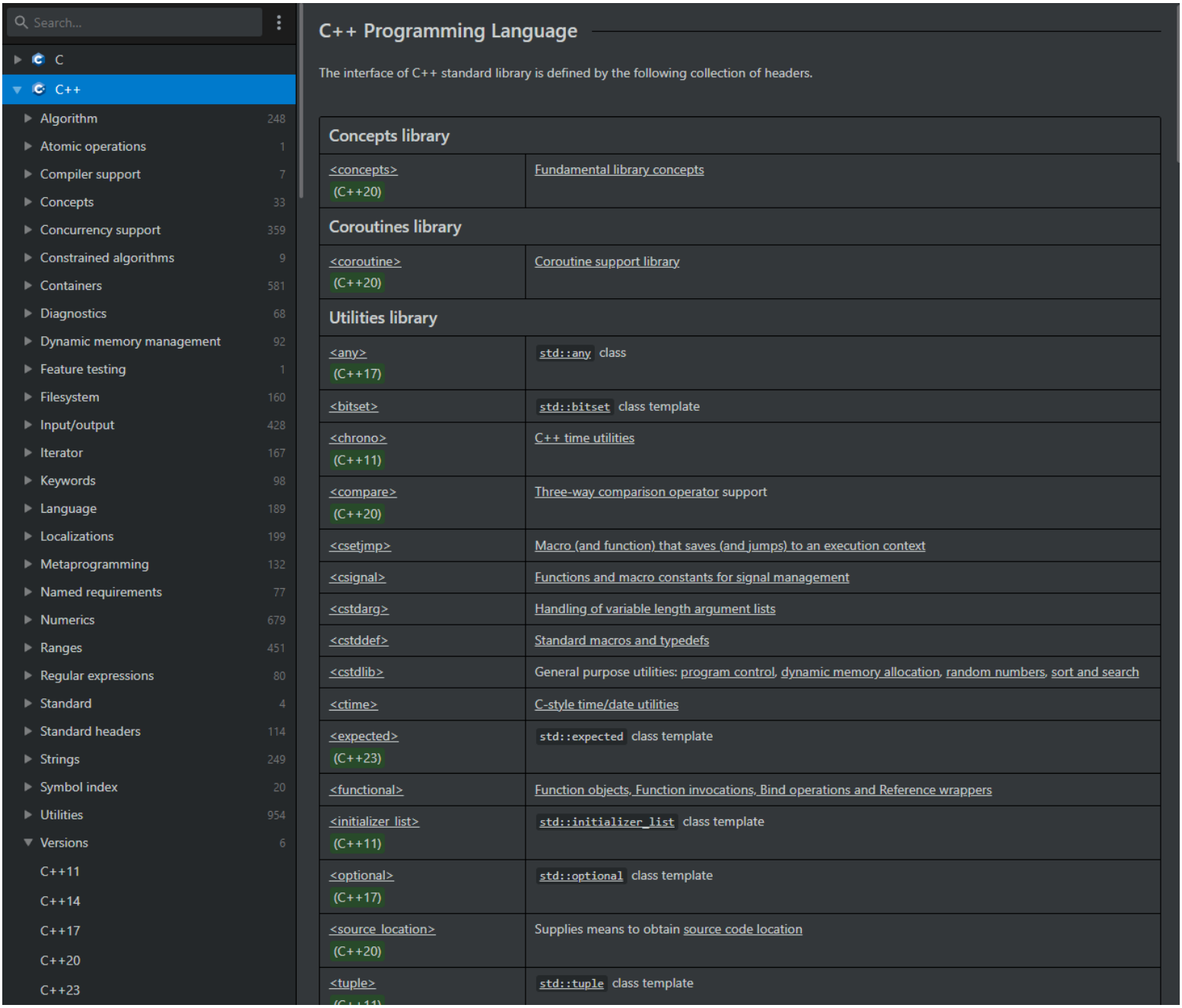
3. "Advantages and disadvantages of C++," KO2 Recruitment, https://www.ko2.co.uk/advantages-disadvantages-of-c-plus-plus/

Evaluating My Learning Resources

"DevDocs" is a user-friendly platform for C++ documentation. It's easy to navigate and brings a lot of insights about C++ topics. The search works fast and the site lets you search at other programming languages as well. However, for detailed C++ information, checking the official documentation might be a better idea.

https://devdocs.io/cpp/

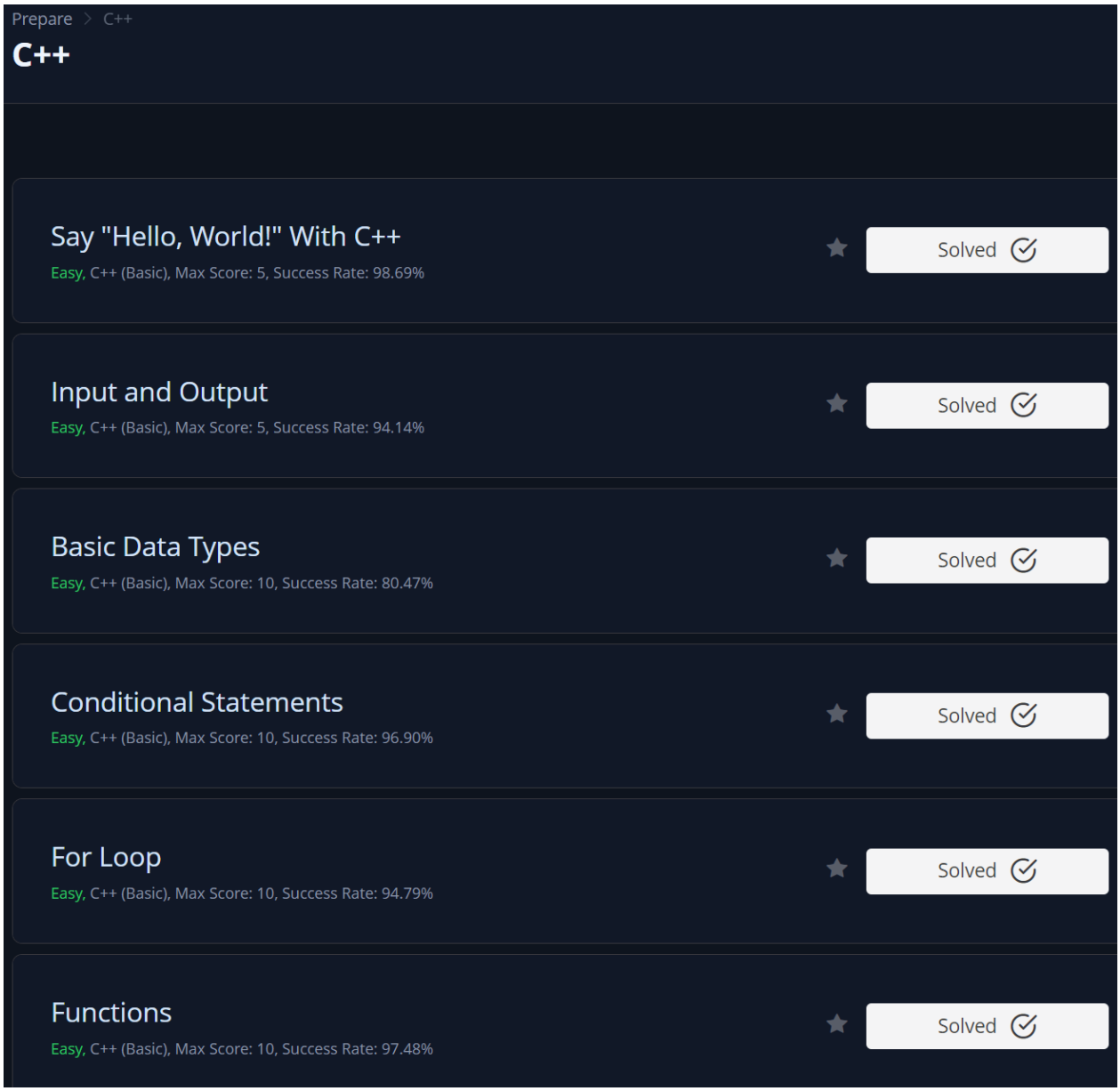
(Documentation)



“HackerRank” is a widely recognized platform for coders who want to improve their skills through challenges and contests across various domains. The site's dashboard is designed to provide a personalized experience, guiding users through practice problems, competitive programming challenges, and even job opportunities. With its intuitive interface, the dashboard highlights users' progress and upcoming contests, allowing for seamless navigation. One of the platform's standout features is its emphasis on real-world problems, bridging the gap between coding theory and its practical application.

https://www.hackerrank.com/dashboard

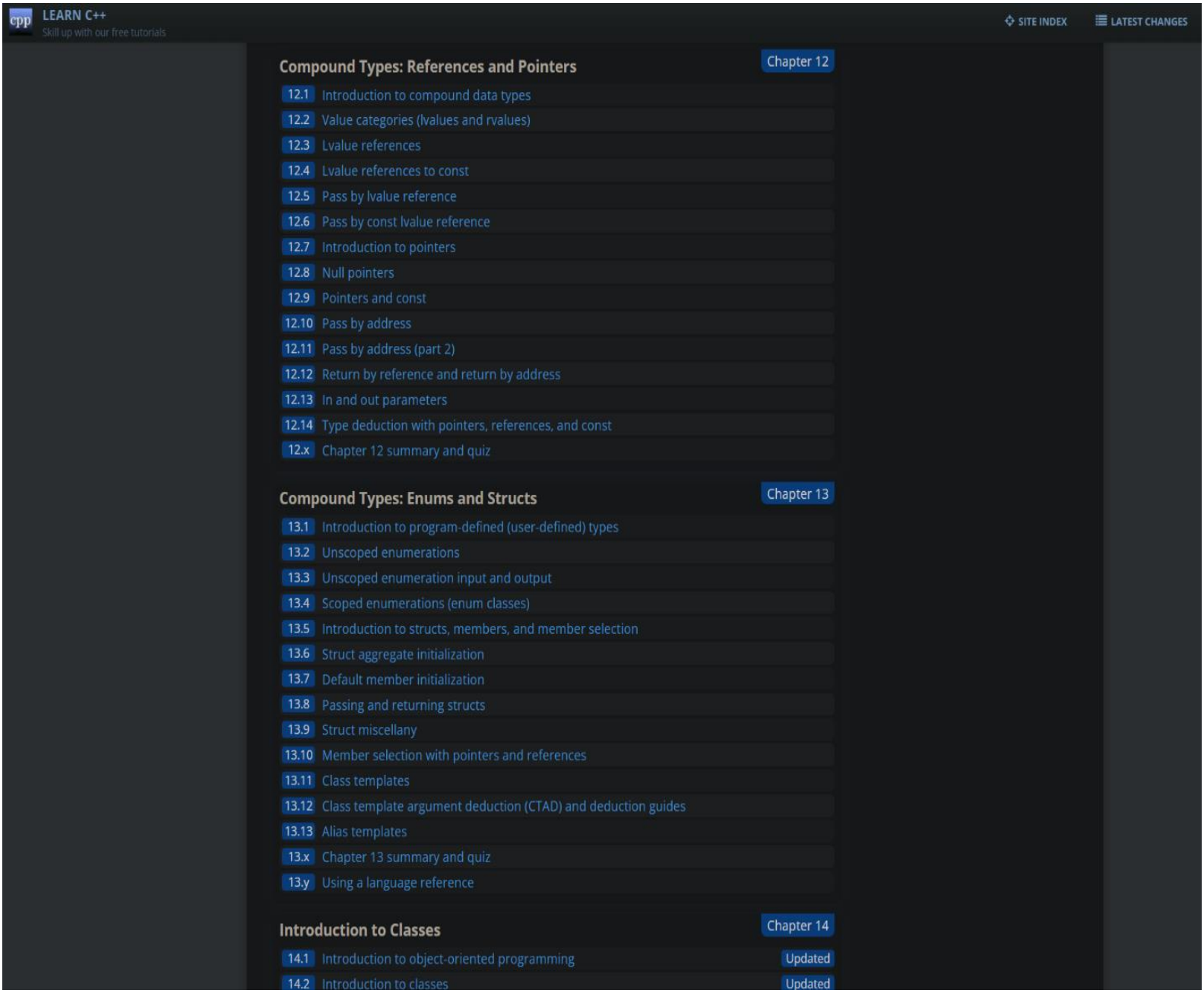
(Practice Tool)



"LearnCpp.com" is a dedicated online resource for master the C++ programming language. This website provides comprehensive tutorials ranging from the basics to more advanced topics, making it suitable for beginners as well as experienced programmers looking to delve deep in C++. The structured layout of LearnCpp.com ensures that users can follow a logical progression in their learning journey. Each tutorial is detailed, with examples and exercises to reinforce understanding.

https://www.learncpp.com/

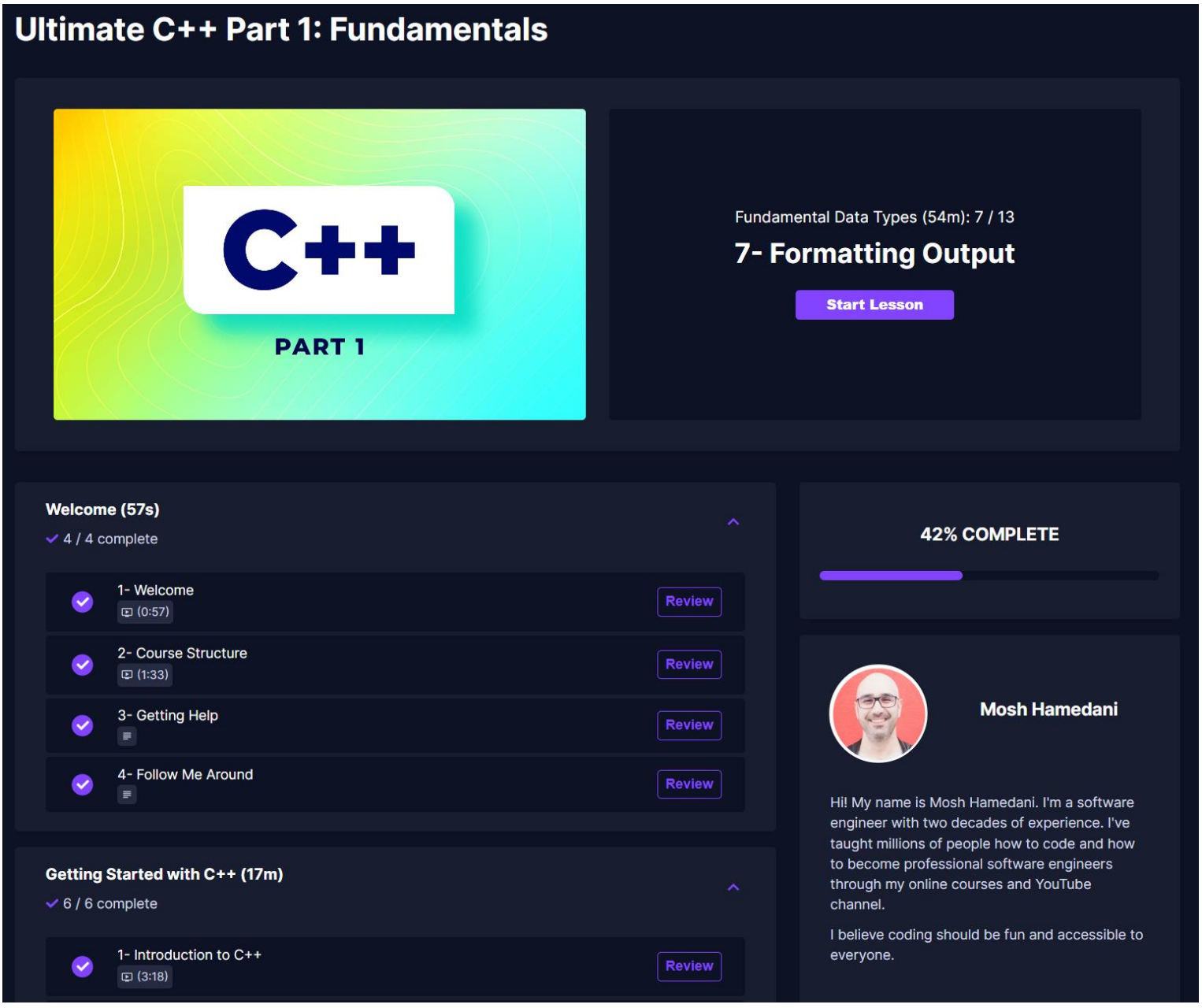
(Tutorials)



"Code with Mosh" is a popular online learning platform published by Mosh Hamedani, a software engineer and educator. The platform is known for offering a range of high-quality courses covering various programming languages, frameworks, and development tools. Mosh's teaching approach is pragmatic, focusing on best practices and real-world application. His courses often cater to both beginners and experienced developers, breaking down complex topics into easily digestible segments.

https://members.codewithmosh.com/courses

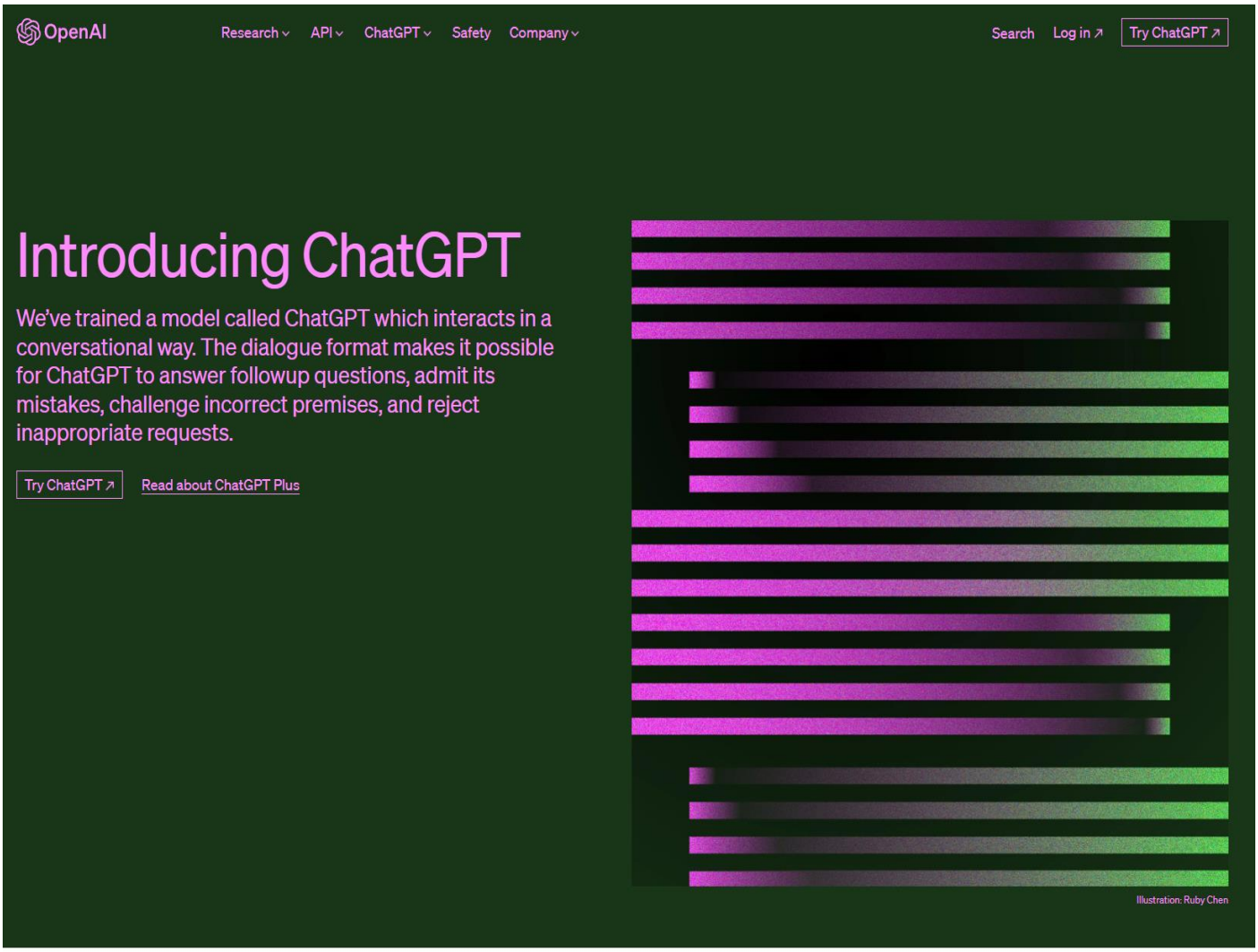
(Practices with Videos)



“ChatGPT”, developed by OpenAI, is an advanced conversational AI model designed to understand and respond to user queries. In the context of learning C++, ChatGPT serves as an invaluable tool. It offers instant clarifications on C++ concepts, assists with coding challenges, provides explanations for error messages, and even suggests best practices. Instead of scrolling through forums or reading lengthy documentation, learners can interact directly with ChatGPT for real-time assistance. This personalized and interactive approach not only makes the learning process more efficient but also fosters a deeper understanding of the language.

https://openai.com/blog/chatgpt

(Artificial Intelligence)



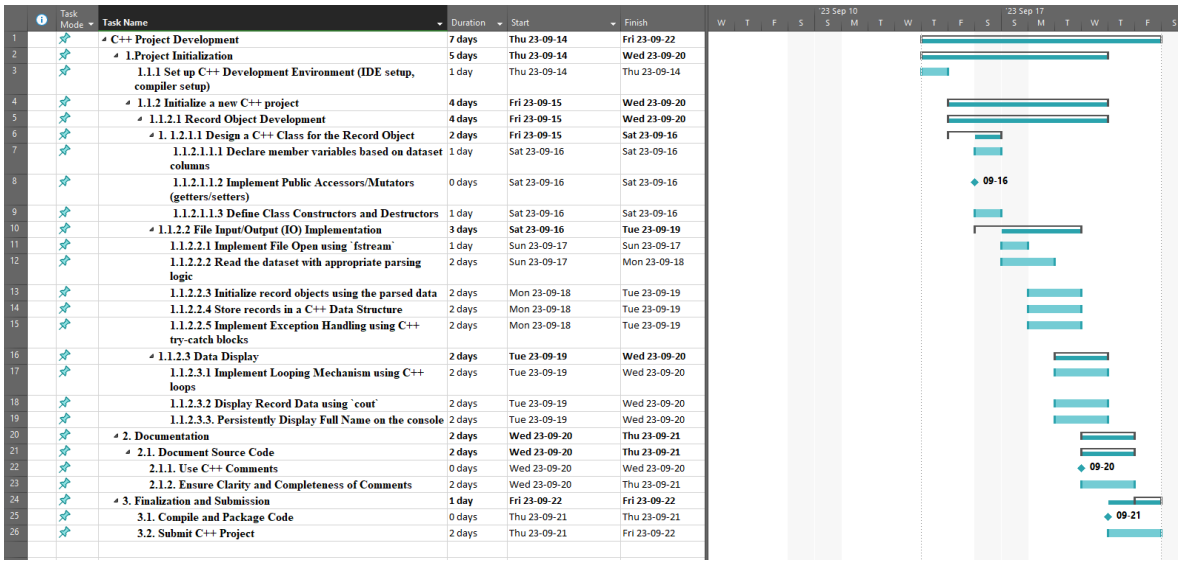
Comparison Table of Resources

Resource	Time Consumption	Usefulness
DevDocs for C++	High	High
HackerRank Dashboard	Low	Medium
LearnCpp	Medium	Medium
Code with Mosh Courses	Medium	Medium
ChatGPT (OpenAI)	Ultra Low	High

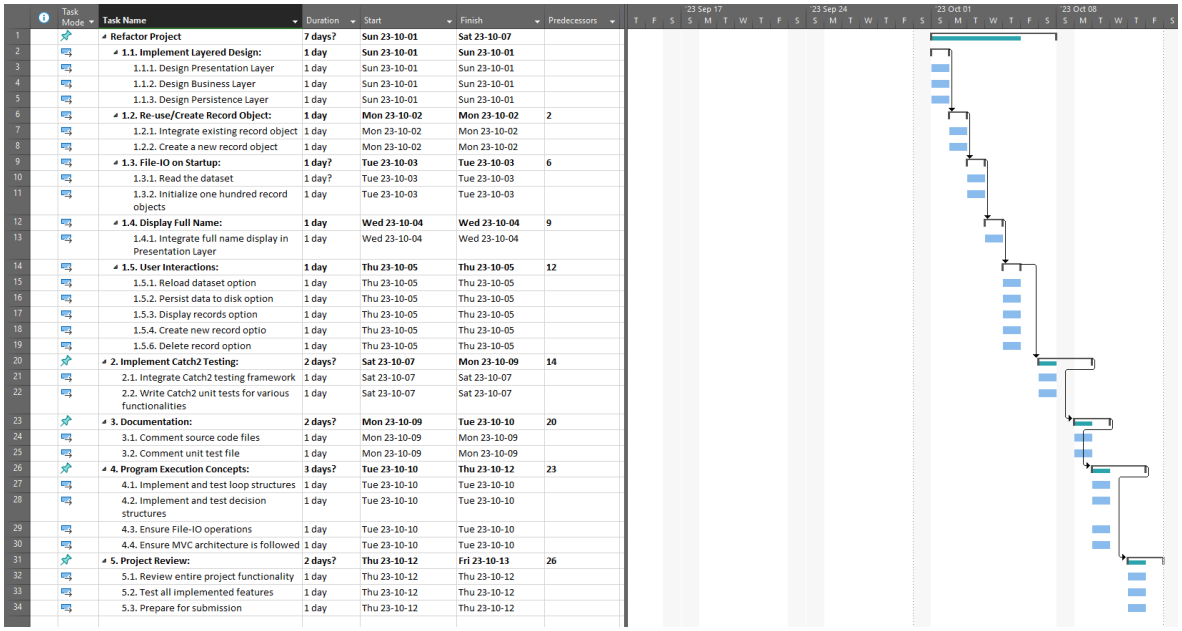
Insights from Work Breakdown Structures and Gantt charts

The WBS allows me to decompose the entire project into manageable steps or tasks, providing a clear hierarchy of what needs to be done. On the other hand, the Gantt chart visualizes the project timeline, showing when each task should start, its duration, and when it's expected to be completed. These tools together ensure that I remain on track and that every aspect of the project is tackled systematically.

With these tools and design decisions at my disposal, I'm more confident about the project's direction and milestones. I anticipate that with this structured approach, the CSV Reader will not only be efficient but also robust and adaptable to future needs.



	Task Mode	Task Name	Duration	Start	Finish
1		C++ Project Development	7 days	Thu 23-09-14	Fri 23-09-22
2		1.Project Initialization	5 days	Thu 23-09-14	Wed 23-09-20
3		1.1.1 Set up C++ Development Environment (IDE setup, compiler setup)	1 day	Thu 23-09-14	Thu 23-09-14
4		1.1.2 Initialize a new C++ project	4 days	Fri 23-09-15	Wed 23-09-20
5		1.1.2.1 Record Object Development	4 days	Fri 23-09-15	Wed 23-09-20
6		1. 1.2.1.1 Design a C++ Class for the Record Object	2 days	Fri 23-09-15	Sat 23-09-16
7		1.1.2.1.1.1 Declare member variables based on dataset columns	1 day	Sat 23-09-16	Sat 23-09-16
8		1.1.2.1.1.2 Implement Public Accessors/Mutators (getters/setters)	0 days	Sat 23-09-16	Sat 23-09-16
9		1.1.2.1.1.3 Define Class Constructors and Destructors	1 day	Sat 23-09-16	Sat 23-09-16
10		1.1.2.2 File Input/Output (IO) Implementation	3 days	Sat 23-09-16	Tue 23-09-19
11		1.1.2.2.1 Implement File Open using `fstream`	1 day	Sun 23-09-17	Sun 23-09-17
12		1.1.2.2.2 Read the dataset with appropriate parsing logic	2 days	Sun 23-09-17	Mon 23-09-18
13		1.1.2.2.3 Initialize record objects using the parsed data	2 days	Mon 23-09-18	Tue 23-09-19
14		1.1.2.2.4 Store records in a C++ Data Structure	2 days	Mon 23-09-18	Tue 23-09-19
15		1.1.2.2.5 Implement Exception Handling using C++ try-catch blocks	2 days	Mon 23-09-18	Tue 23-09-19
16		1.1.2.3 Data Display	2 days	Tue 23-09-19	Wed 23-09-20
17		1.1.2.3.1 Implement Looping Mechanism using C++ loops	2 days	Tue 23-09-19	Wed 23-09-20
18		1.1.2.3.2 Display Record Data using `cout`	2 days	Tue 23-09-19	Wed 23-09-20
19		1.1.2.3.3. Persistently Display Full Name on the console	2 days	Tue 23-09-19	Wed 23-09-20
20		2. Documentation	2 days	Wed 23-09-20	Thu 23-09-21
21		2.1. Document Source Code	2 days	Wed 23-09-20	Thu 23-09-21
22		2.1.1. Use C++ Comments	0 days	Wed 23-09-20	Wed 23-09-20
23		2.1.2. Ensure Clarity and Completeness of Comments	2 days	Wed 23-09-20	Thu 23-09-21
24		3. Finalization and Submission	1 day	Fri 23-09-22	Fri 23-09-22
25		3.1. Compile and Package Code	0 days	Thu 23-09-21	Thu 23-09-21
26		3.2. Submit C++ Project	2 days	Thu 23-09-21	Fri 23-09-22



	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		Refactor Project	7 days?	Sun 23-10-01	Sat 23-10-07	
2		1.1. Implement Layered Design:	1 day	Sun 23-10-01	Sun 23-10-01	
3		1.1.1. Design Presentation Layer	1 day	Sun 23-10-01	Sun 23-10-01	
4		1.1.2. Design Business Layer	1 day	Sun 23-10-01	Sun 23-10-01	
5		1.1.3. Design Persistence Layer	1 day	Sun 23-10-01	Sun 23-10-01	
6		1.2. Re-use/Create Record Object:	1 day	Mon 23-10-02	Mon 23-10-02	2
7		1.2.1. Integrate existing record object	1 day	Mon 23-10-02	Mon 23-10-02	
8		1.2.2. Create a new record object	1 day	Mon 23-10-02	Mon 23-10-02	
9		1.3. File-IO on Startup:	1 day?	Tue 23-10-03	Tue 23-10-03	6
10		1.3.1. Read the dataset	1 day?	Tue 23-10-03	Tue 23-10-03	
11		1.3.2. Initialize one hundred record objects	1 day	Tue 23-10-03	Tue 23-10-03	
12		1.4. Display Full Name:	1 day	Wed 23-10-04	Wed 23-10-04	9
13		1.4.1. Integrate full name display in Presentation Layer	1 day	Wed 23-10-04	Wed 23-10-04	
14		1.5. User Interactions:	1 day	Thu 23-10-05	Thu 23-10-05	12
15		1.5.1. Reload dataset option	1 day	Thu 23-10-05	Thu 23-10-05	
16		1.5.2. Persist data to disk option	1 day	Thu 23-10-05	Thu 23-10-05	
17		1.5.3. Display records option	1 day	Thu 23-10-05	Thu 23-10-05	
18		1.5.4. Create new record optio	1 day	Thu 23-10-05	Thu 23-10-05	
19		1.5.6. Delete record option	1 day	Thu 23-10-05	Thu 23-10-05	
20		2. Implement Catch2 Testing:	2 days?	Sat 23-10-07	Mon 23-10-09	14
21		2.1. Integrate Catch2 testing framework	1 day	Sat 23-10-07	Sat 23-10-07	
22		2.2. Write Catch2 unit tests for various functionalities	1 day	Sat 23-10-07	Sat 23-10-07	
23		3. Documentation:	2 days?	Mon 23-10-09	Tue 23-10-10	20
24		3.1. Comment source code files	1 day	Mon 23-10-09	Mon 23-10-09	
25		3.2. Comment unit test file	1 day	Mon 23-10-09	Mon 23-10-09	
26		4. Program Execution Concepts:	3 days?	Tue 23-10-10	Thu 23-10-12	23
27		4.1. Implement and test loop structures	1 day	Tue 23-10-10	Tue 23-10-10	
28		4.2. Implement and test decision structures	1 day	Tue 23-10-10	Tue 23-10-10	
29		4.3. Ensure File-IO operations	1 day	Tue 23-10-10	Tue 23-10-10	
30		4.4. Ensure MVC architecture is followed	1 day	Tue 23-10-10	Tue 23-10-10	
31		5. Project Review:	2 days?	Thu 23-10-12	Fri 23-10-13	26
32		5.1. Review entire project functionality	1 day	Thu 23-10-12	Thu 23-10-12	
33		5.2. Test all implemented features	1 day	Thu 23-10-12	Thu 23-10-12	
34		5.3. Prepare for submission	1 day	Thu 23-10-12	Thu 23-10-12	

Discussion Board 1

Onur Onel - C++

Onur Önel posted Sep 24, 2023 6:48 PM Subscribed

1. **What did you learn about your programming language?**

- C++ is an extension of the C language that incorporates object-oriented features.
- It is commonly used for developing high-performance applications like operating systems and databases.
- The language provides low-level memory management capabilities.
- Syntax and conventions of C++ have influenced modern languages like Java and C#.
- C++ remains highly valued in today's job market.

2. **What was interesting or fun about what you have learned, and why?**

I found it fascinating that C++ can act as a stepping stone to other programming languages. Learning its syntax and understanding its low-level functionalities can make it easier to pick up other languages. This aspect makes C++ not just a language to learn but an investment in my broader programming career.

3. **What one(s) worked best for you, and which one(s) was (were) less successful for you and why?**

The JetBrains CLion documentation and cppreference.com were most effective for me. They provided thorough, up-to-date insights directly relevant to the IDE I was using and the language version I was focused on (C++20). On the other hand, general resources like the Wikipedia comparison of IDEs were less effective. They provided broad overviews but didn't deeply cater to the specific nuances of C++ or the tools I was using.

4. **What one(s) was (were) the most time-consuming, and what one(s) was (were) the least time-consuming and why?**

The most time-consuming resource was likely the RedMonk Index While informative about the general popularity of programming languages, it required cross-referencing with other resources to get the information I needed for C++. The least time-consuming were the Catch2 GitHub releases and Google Test user's guide, as they were straightforward and focused solely on testing in C++, allowing me to quickly grasp the key concepts.

5. **List your resources, and indicate what ones were the most effective for you**

- JetBrains CLion documentation - Most Effective
- cppreference.com - Very Effective
- Catch2 GitHub releases - Effective for Testing
- Google Test user's guide - Effective for Testing
- RedMonk Index - Informative but time-consuming
- Wikipedia comparison of IDEs - Least Effective

6. **Was your WBS accurate, i.e. was it too high-level, or too detailed?**

My Work Breakdown Structure was moderately detailed but could have included more information on the debugging and testing phases, which turned out to be more time-consuming than anticipated.

7. **How will you improve your process when creating a new WBS in the future?**

In the future, I'll aim for a more detailed WBS that accounts for contingencies and unexpected delays, like debugging or additional research.

8. **How was your time estimation on your Gantt chart?**

My time estimation in the Gantt chart was mostly accurate for the coding part but underestimated for the testing and debugging phases.

9. **How will you improve your time estimation in the future?**

In the future, I'll consider adding buffer time for testing and debugging while estimating time for individual tasks, to create a more realistic Gantt chart.

Discussion Board 2

Practical Project 2 - C++

Onur Önel posted Oct 16, 2023 10:04 PM Subscribed

What did you learn about your programming language?

I delved into Cmake, focusing on understanding dependencies and linking libraries. Additionally, I explored the MVC structure and deepened my knowledge in advanced C++ concepts, especially vectors. Although challenging at first, by dedicating more time and prioritizing it, I made significant strides.

What was interesting or fun about what you have learned, why?

I found it fascinating to contrast C++ with Java. The more hands-on nature of C++ offers a unique perspective, bringing one closer to the foundational workings of computers and intricate mathematical processes. It provided a richer understanding of computational principles.

What resources were most beneficial to you, and which ones were less successful, and why?

I greatly benefited from Catch2's documentation, which assisted in unit test integration. Video tutorials from JetBrains Clion clarified intricacies related to the IDE and compiler setups, while Conan's documentation was instrumental for grasping package management.

Which tasks consumed most of your time, and which ones took the least, and why?

I spent a significant amount of time on Cmakelist and compiler-related tasks. These areas are intricate and demanded close attention and thoroughness.

List your resources and indicate which ones stood out in their effectiveness.

One notable resource was a YouTube video titled "Cmake and Catch2 integration" presented by Clare Macrae. With her extensive experience in software development she covers topics related to setting up Cmake to build projects that utilize Catch2 for unit testing. This involves creating appropriate Cmake files, linking libraries, setting up test targets, and ensuring the build process runs the Catch2 tests appropriately.

Was your WBS precise, meaning was it overly detailed or too generalized?

The Work Breakdown Structure (WBS) I crafted was well-balanced, avoiding extremes of being overly detailed or too generalized.

How do you plan to refine your approach to creating a WBS in the future?

For future WBS creations, I aim to incorporate feedback and elaborate on areas that might necessitate a more in-depth breakdown.

How will you refine your time estimation skills for future tasks?

While my time projections on the Gantt chart weren't always spot-on, I learned valuable lessons. Moving forward, I'll analyze past projects to pinpoint estimation inaccuracies and refine my methods accordingly.