ICT LAB

FINAL PROJECT REPORT



**NUMGAME**

Powered by: WARAV

**Submitted by:**

|  |  |
| --- | --- |
| Muhammad Waleed  24L-6023 | Raveha Munir  24L-6004 |

**Submitted to:**

Muiz Malik

07-DEC-2024

Department of Electrical Engineering

National University of Computer and Emerging Sciences, Lahore

## Introduction

1. **Problem Analysis**

## Design Requirements

1. **Feasibility Analysis**

## Possible Solutions

1. **Preliminary Design**

## Design Description

## Software Simulation

## Experimental Results

1. **Performance Analysis**
2. **Future Scope**

## Social and Cultural Implications

## Conclusion

1. **Reference**

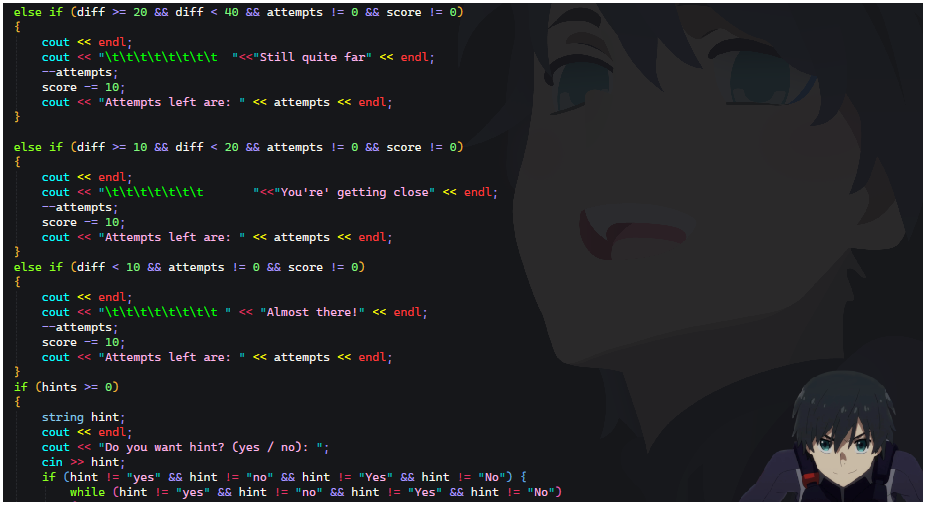
Creating simple interactive games happens to be one of the most effective ways in the field of computer programming for developers, especially beginners, to understand basic programming concepts. This report outlines the development process of a number-choosing game implemented in C++. The player is supposed to guess a number that is randomly generated within a defined range.

This game will enhance the understanding of some very important programming principles such as:

1. **Loops (**while, for, do while**)**
2. **Conditionals (**if, else, switch case, statements**)**
3. **Random number generation (**rand, s rand & use of new library (cstdlib) **)**
4. **Even input validation.**
5. Additionally, this will build **Problem-solving skills**
6. Provide an interface for anybody who may be interested in something that is not only fun but **educational too**.

This game is therefore not just about having a tool that has practical hands-on experience of C++ basic aspects but fun for new developers such as me. This project is a great example of how interactive software can be developed using core programming constructs. Throughout the process of development, extra care was taken to ensure the game plays intuitively and responds quickly,

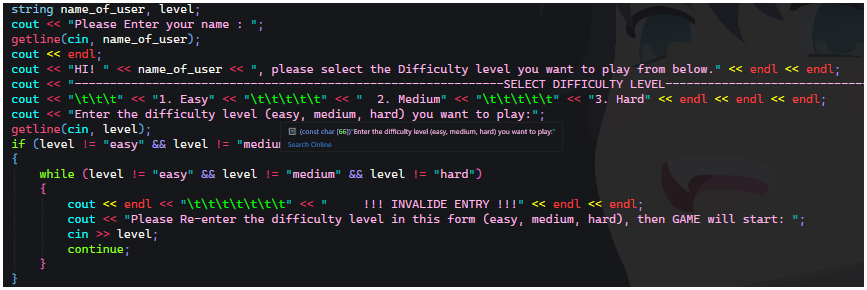
being simple with code structure,i.e. usage of (if, else statements) only as conditions.



A good number-guessing game needs quite a few steps for fluid gameplay as well as robust error handling. Here are the key problems that were encountered in this regard:

1. **Mechanism To Take Difficulty Level Of Game From User:**

The first challenge is to design the system, where the game takes input of user that what kind of difficulty level he or she want to play. In interactive games, taking the difficulty level input from the user is a crucial step to ensure personalized gameplay. The mechanism involves presenting the user with predefined difficulty levels **(e.g., Easy, Medium, Hard)** at the start of the game. This input can be collected using simple methods such as menu selection or text-based input.



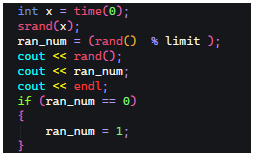
**getline(**cin, level**)** is used here, this is because when function such as string I called in my programing code then then it just takes the only one complete word note the whole line .

For example: if I enter my name such as Muhammad Waleed, then without using getline it just takes Muhammad only, not the complete line so that’s why I used getline command to take the whole line.

1. **Random Number Generation:**

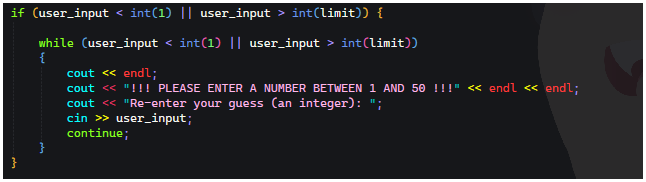
To generate a number a number between a given range is another challenging step. In C++, this has been accomplished using the rand() function in such a way that for each iteration, it will offer a new number to guess for. To make this I have to include a library named:

**#include<cstdlib>** & #**include<ctime>**

The player must enter their guess through the console. Input should be validated to ensure that it is a number and falls within the defined range. Non-numeric input should be handled carefully to avoid crashes or unexpected behavior.

1. **Game Logic:**

The game must check if the guess of the player is correct, too high, or too low. In case of an incorrect guess, the program should provide feedback and ask the player to guess again.



1. **Feedback Mechanism:**

The effective feedback loop must be established in the game so that the player gets guided through the process. It should give the player feedback that the guess is too high or too low and display the number of attempts taken to guess correctly.

A computer screen with text

Description automatically generated

1. **End Conditions:**

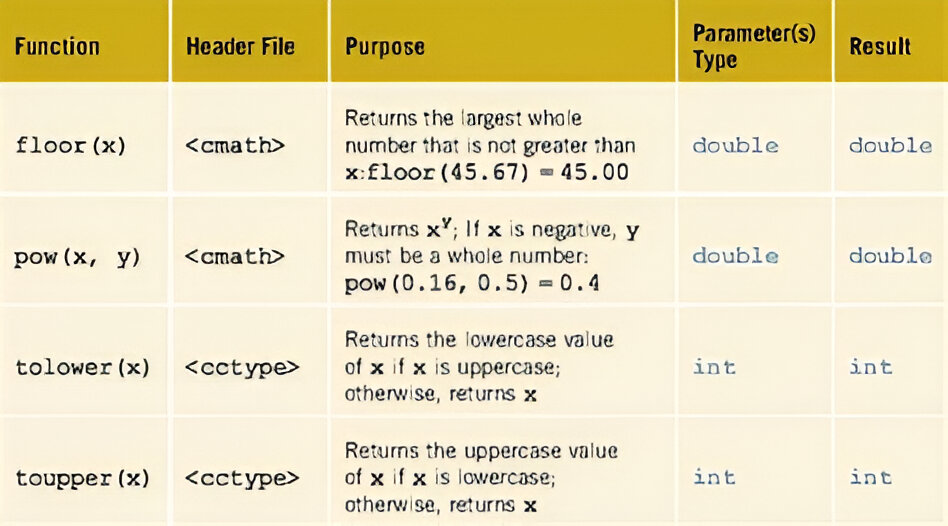
The game should be closed when either the correct number is guessed or exceeded several allowed guesses. When the player fails to guess the number before the set limit, the game should include the right number in the messages.

A computer screen shot

Description automatically generated

For the game to meet user and technical expectations and work efficiently, several key design requirements were defined:

1. **Functional Requirements:**
2. The game produces a random number between some predefined range, such as (1 to 100).
3. The player inputs his or her guess using the keyboard.
4. The game compares the guess against the generated number and provides feedback, such as "**Way Far**", "**Still Away**", or "**Almost There**!!"
5. The game tracks the count of guesses and allows the player to retry for the correct number till he exhausts his attempts.
6. **Non-Functional Requirements:**
7. **Performance:** The game must be able to run without a delay. Input handling and feedback must be immediate, with no perceived lag or system slowdown.
8. **Reliability:** The game should be robust, in that it must be able to handle different kinds of incorrect input, such as non-numeric values or inputs outside the acceptable range, without crashing.
9. **Usability:** The game should be easy to learn and play. The interface could be text-based but should have clarity and conciseness for simple technical knowledge among players.
10. **Technical Feasibility:**

 The game has been developed using C++ that provides all the tools required to complete the tasks. C++ has a very rich standard library such as ***cstdlib*** , ***cmath*** ,with functions like rand() & sqrt(), pow() for generating random numbers and handling input/output via cin and cout. The project uses basic programming constructs like loops, conditionals, and simple input/output operations that make it feasible for a beginner to understand and modify it.

1. **Economic Feasibility:**

C++ is an open-source programming language, and it's free. This game can be created without any special libraries, plugins, or external software. The code can be compiled and then executed on any system that includes a C++ compiler. Thereby, it's cost-effective in this regard as no licenses will need to be purchased or expensive hardware.



1. **Operational feasibility:**

 The game runs effectively in a console application and is built to be simple; it works very well even on computers with less power. The game is accessible by using a keyboard and has a straightforward text-based user interface that will be easily understandable to any user regardless of age and background.

During the planning and development of the game, several possible solutions were considered to enhance gameplay and user interaction:

1. **Single-Player Mode:**

The most basic implementation of the game is the single-player mode, where the player guesses a randomly generated number. This solution is the easiest to implement, keeping the focus on random number generation, input handling, and providing feedback.

1. **Multiplayer Mode:**

A possible future feature might be a multiplayer mode where the two or more players take turns guessing the same number or compete by guessing a number in as few attempts as possible. This could include additional logic for player turns and scoring.

1. **Difficulty Levels:**

It may make the game more engaging to introduce various levels of difficulties such as Easy, Medium, Hard, and so on. In the Easy mode, the guessing numbers could range between 1 and 10; in Hard, between 1 and 1000. The number of guesses might be allowed differently by the different levels of difficulties.

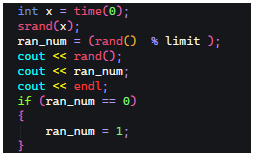
1. **Leaderboard and Achievements**

To make the game more competitive and replay able, a leaderboard system could be added to track the number of attempts it takes for players to guess the number. Achievements or milestones could also be introduced to encourage replay ability

1. **User-Centered Design (UCD)**

To make the game more competitive and replay able, a leaderboard system could be added to track the number of attempts it takes for players to guess the number. Achievements or milestones could also be introduced to encourage replay ability

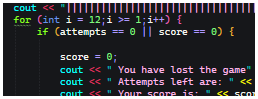
The preliminary design of the number-choosing game follows a clear, modular approach to maintain simplicity and clarity:

1. **Random Number Generation:**

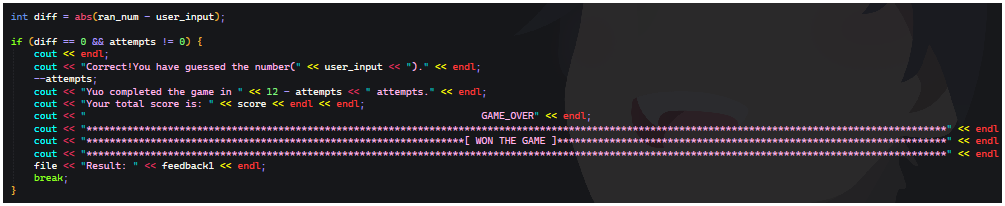
The game begins by generating a random number between

1 and 100 using the rand() function. That number is never revealed to the player but acts as the target for his guesses.

1. **Main Game Loop:**

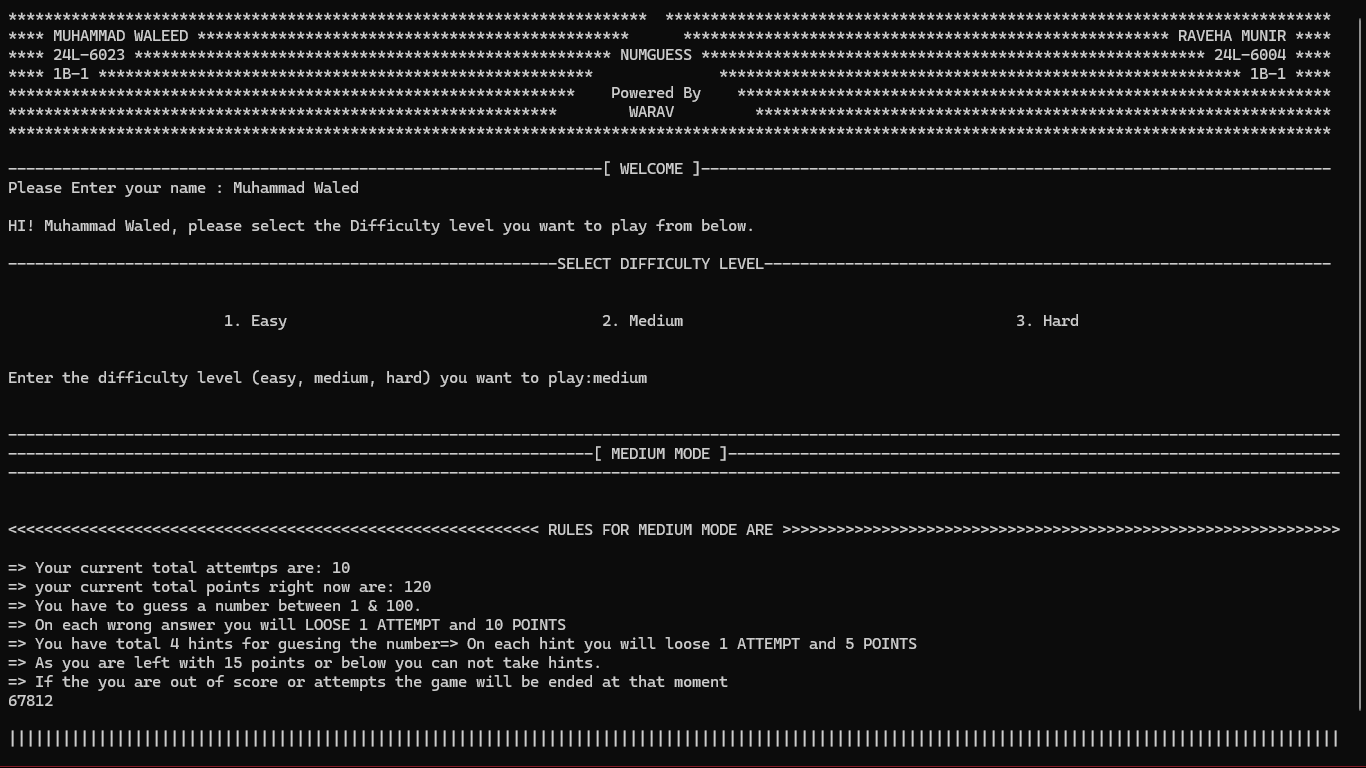
A loop is used to repeatedly prompt the player to enter a guess. After each guess, the program compares it to the generated number and provides feedback. The loop continues until the player guesses correctly or runs out of attempts.

1. **End Condition:**

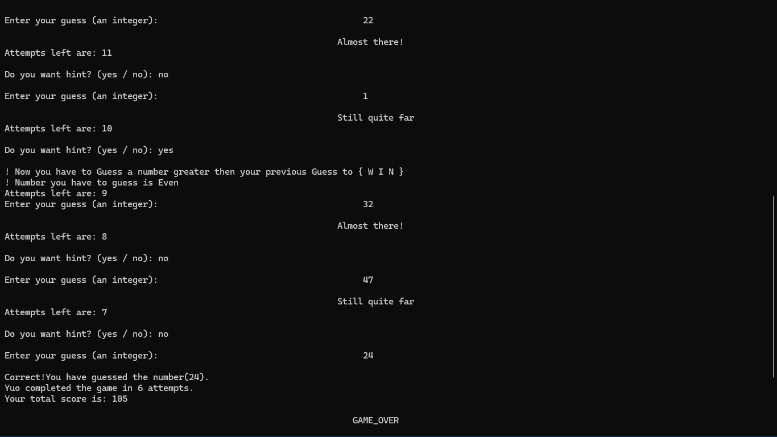
If the player guesses the correct number, the game ends with a congratulatory message. In case the player crosses the maximum number of attempts, then the game ends, and it shows the correct number.

The number-choosing game is based on a simple text-based user interface. The flow of the game is as follows:

1. **Initialization:**

Welcome message along with the reference of creators.

1. **Game Loop:**

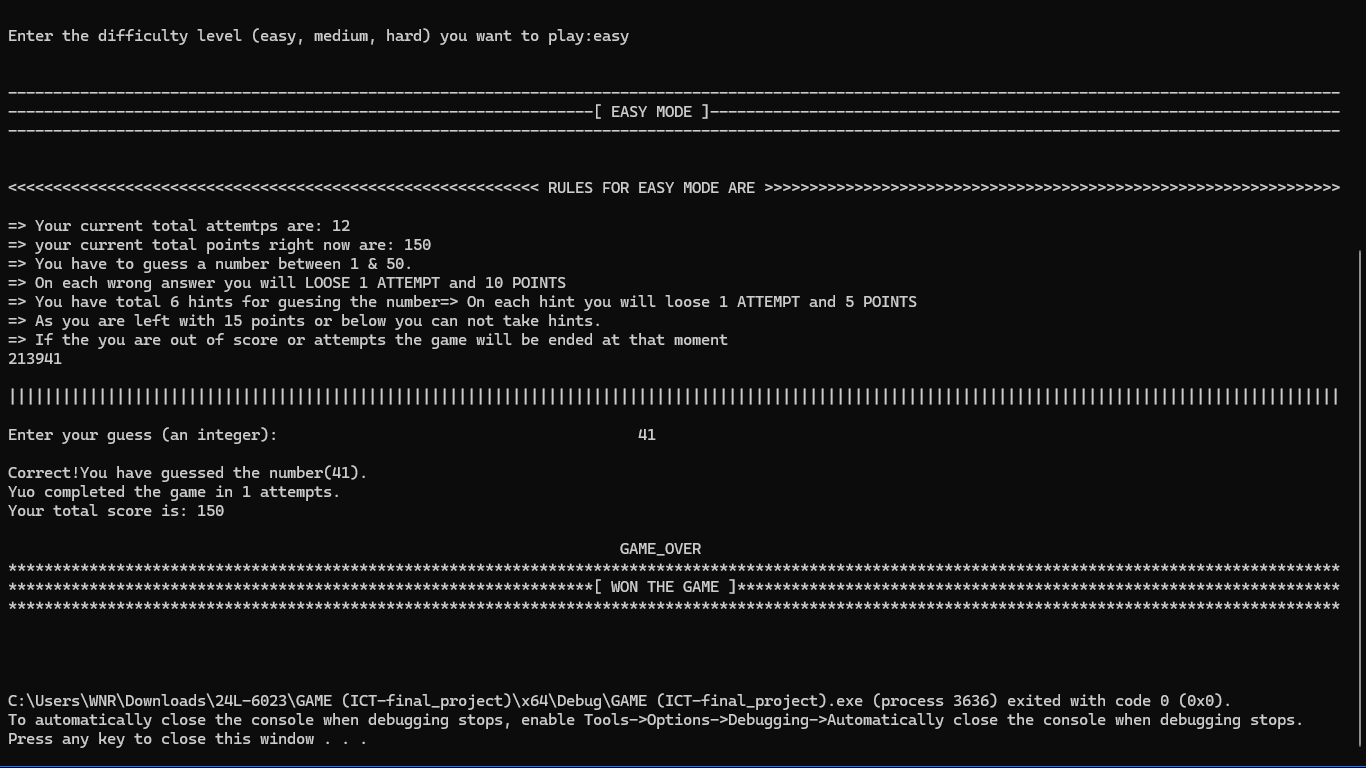


* Ask the player to guess the number.
* If the guess is too high, display "Way Far"
* If the guess is too low, display "Coming closer"
* If the guess is correct, display "Congratulations! You guessed the number!"
* Continue until the player guesses the number or runs out of attempts.

1. **End of Game:**

Show attempts in case of correct guess by player

In case the player is out of attempts, it displays the correct number and also asks the player to play again.



It is simulated using a C++ development environment.

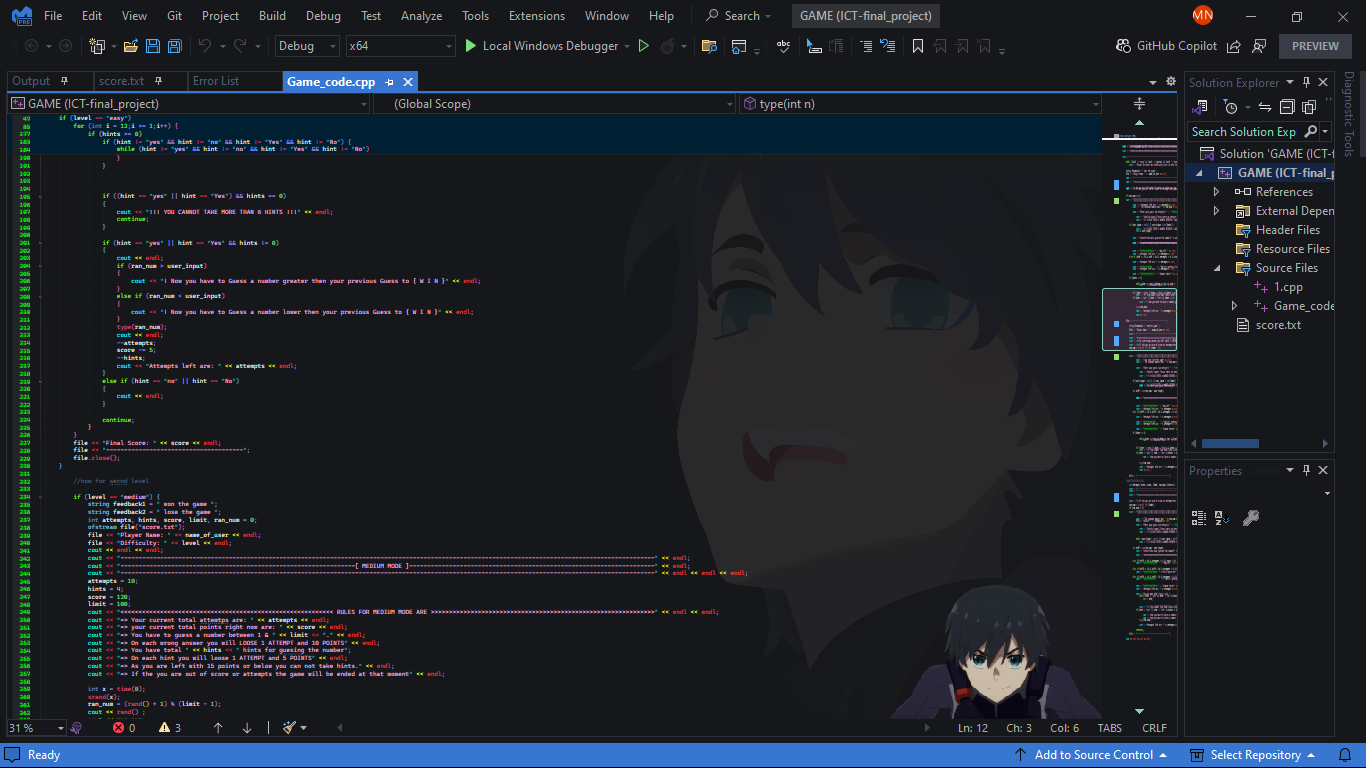
For example: Visual Studio, Code: Blocks.

**This game was tested for the following cases:**

* Test Case 1: Guess by the player within first attempt
* Test Case 2: Non-numeric inputs by the player.
* Test Case 3: The player runs out of attempts.
* Test Case 4: The player makes several attempts to guess the number.

All these tests were successful in making sure that the game behaves as per logic, with

correct generation of random numbers and giving appropriate feedback.



1. The results of the pilot testing phase are that it has been successful in generating all the random numbers within the chosen range and accepting valid numeric input from the players.
2. Input validation successfully prevented crashes due to incorrect data types.
3. Provides accurate feedback ("Too high," "Too low," or "Correct!").
4. Ends correctly when the player guesses the number or exhausts all attempts.

1. **Speed:**

The program runs quickly, with minimal delays in processing user input and generating feedback.

1. **Memory Usage:**

The game has a low memory footprint, as it stores only a few variables such as the secret number, the user’s guess, and the number of attempts.

1. **Usability:**

The game is intuitive in nature with clear prompts and instructions. It is accessible for players of all ages and programming backgrounds.

Future versions of the game can make use of the following features:

**GUI:**

A transition from a text-based interface to the GUI can be made using libraries like SFML or SDL so that it enhances user interaction.

A screenshot of a computer screen

Description automatically generated

**Multiplayer Support:**

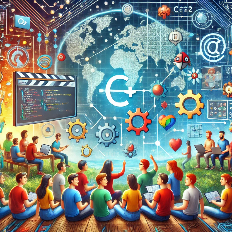
Allow different players to take turns guessing, thereby adding a competitive aspect.

**Difficulty Levels:**

Introduce different scopes for numbers and various attempts for each difficulty.

**High Score Tracking:**

Add a mechanism to store high scores or number of successful games played.

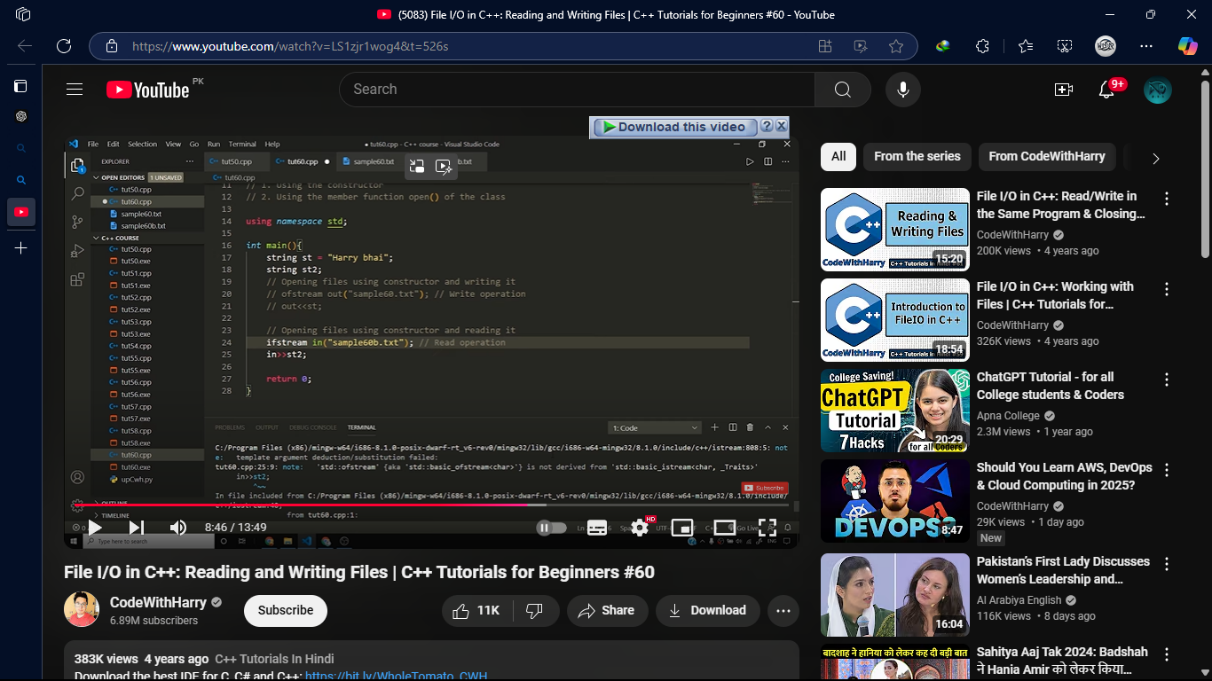
This game has the potential to engage users of all backgrounds. It encourages logical thinking and problem-solving in a fun and interactive way. The simplicity of the game ensures that it is universally accessible, regardless of the user's cultural background or technical experience.

In conclusion, the creation of the number-guessing game in C++ has served as an excellent opportunity to solidify fundamental programming concepts. The key insights gained from this project include:

* **Mastery of Core Programming Concepts**: Developing the game provided hands-on experience with essential programming elements, such as:
  1. **Loops** (while, for, do-while)
  2. **Conditional Statements** (if, else, switch-case)
  3. **Random Number Generation** (using rand, srand, and the cstdlib library)
  4. **Input Validation**
  5. **Enhancing Problem-Solving Abilities**
* **Interactive and Educational Experience**: The game not only provides an engaging and fun experience but also serves as an educational tool for anyone eager to explore programming through interactive software.
* **Practical Application of C++ Fundamentals**: This project was a hands-on way to implement basic C++ concepts, demonstrating how foundational programming techniques can be applied to create an interactive and user-friendly game.
* **Simple and Intuitive Design**: Throughout the development process, special attention was given to creating a game that is easy to understand and responsive, with a straightforward code structure centered around if-else statements for logical decision-making.
* The number-choosing game developed in C++ is a simple yet effective tool for learning the basic concepts of programming. It combines essential programming techniques such as loops, conditionals, and random number generation, offering an interactive and educational experience. With further enhancements, such as multiplayer modes and graphical interfaces, this game could evolve into a more complex and enjoyable experience.

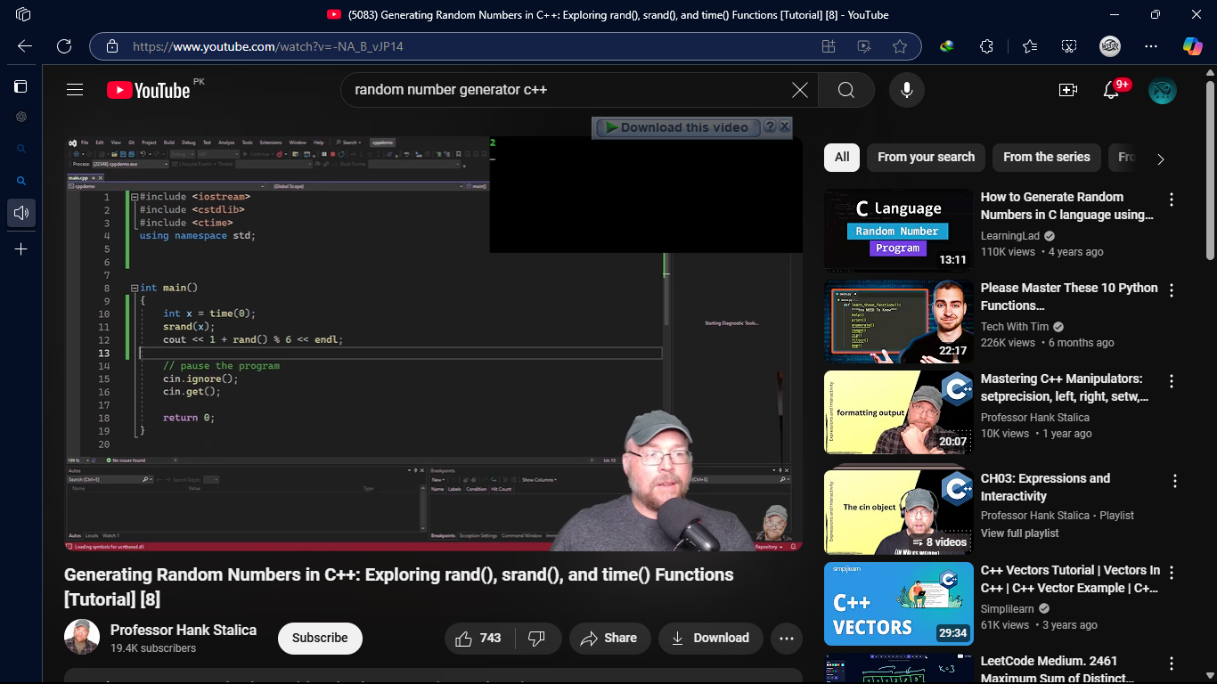
In summary, this project illustrates how fundamental programming skills can be used to create both an enjoyable and educational tool, benefiting new developers and those learning the basics of C++ programming.

* C++ Programming Documentation. (File Handling): <https://www.youtube.com/watch?v=LS1zjr1wog4&t=526s>



* Online tutorials on random number generation in C++.

<https://www.youtube.com/watch?v=-NA_B_vJP14>



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EA1**: Range of resources  **EA2**: Level of interaction  **EA3**: Innovation  **EA4**: Consequences for society and environment  **EA5**: Familiarity | * ***EA1: Range of resources*** -- The design involves the use of diverse resources, such as, money, equipment, information, and technology. * ***EA2: Level of Interaction*** – The design requires resolution of various problems arising from interactions between wide-ranging or conflicting technical or other issues. * ***EA3: Innovation*** – The design addresses sustainability and reduces cost requiring creative use of engineering principles and research-based knowledge in novel ways. * ***EA5: Familiarity –*** The design activity emphasizes the integration of existing knowledge and tools (or familiar solutions) with new or unfamiliar challenges | | | |
| Rubrics | | LLOs | Marks |
| Explains the design process including engagement of resources clearly | EA1 | LLO3 |  |
| Demonstrate the final design clearly with all supporting information | EA1, EA5 | LLO4 |  |
| Demonstrate how innovation has been used in design to resolve conflicting requirements including the impact on society and environment | EA2,  EA3 | LLO4 |  |
| Prepares a report which explain the design process including engagement of resources clearly, and is free from grammatical errors. | EA1 | LLO3 |  |