VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY

UNIVERSITY OF SCIENCE

FACULTY OF INFORMATION TECHNOLOGY

Poker Game

Report Name: Final Project

Course name: Fundamentals of Programming

CSC1002 - 24C03

Student name:
Lam Chi Hao
Vo Nhat Tien
Le Nguyen Anh Tri

Teacher name:
Nguyen Hai Minh
Tran Thi Thao Nhi
Phan Thi Phuong Uyen

07/12/2024

Contents

1 Foreword

We would like to give our most sincere thanks to Mrs. Tran Thi Thao Nhi and Mrs.Phan Thi Phuong Uyen for this opportunity to do the project. Your valuable advice and encouragement during the course of our work mean a great deal to us, and we cannot appreciate enough the time and effort you have devoted to making us successful. We hope that our results satisfy your expectations.

Although the project is not without its limitations, such as one of our team members being unable to contribute due to personal circumstances, we have done our best to complete it. We would appreciate your understanding of our situation and the effort we have put into overcoming these challenges.

We really do hope you will like the project, and it will live up to your vision. Your valued comments and feedback will go a long way in our growth and service improvement in times to come.

We are particularly very lucky to have been in this project for experience and learning. Any shortcomings and mistakes are solely our own liability, and we take this as an opportunity to improve our skill and knowledge further.

Thank you once more for sparing your valuable time and considering our request. It was an honor and a pleasure working under your guidance, and we hope that someday in life, we get to work with you again.

Best regards,

2 Introduction

This project focus on developing a Poker game using C++11 programming language. This was done in a group of two members, which are Le Nguyen Anh Tri and Lam Chi Hao (originally three members but Vo Nhat Tien was unable to contribute due to personal circumstances), for the final project of the course Fundamentals of Programming (CSC10012) at University of Science, Vietnam National University - Ho Chi Minh City.

Here is our group information for this project:

	ID	**Name**	**Title**
1	24127034	Lam Chi Hao	Leader
2	24127567	Le Nguyen Anh Tri	Member

Table 1: Information of Group 02 member.

Group management and communication were pretty simple in our two-man team. Messenger served us for fast discussions, while GitHub was used for versioning and collaboration. This simplicity excluded any formal work management board or a developed communication plan. So, roles were divided quite clearly: one member of the team worked on core game logic, while the second was developing a GUI. Then we will take our time to review code, test, and make merges towards the end that would bring everything together as it needed.

This is our table of contribution for this project, which shows the work we have done:

Problem	**Rubric**	**24127034**	**24127557**	**24127567**
Standard features	-			
Initialization	20	Score	Score	Score
Dealing	20	Score	Score	Score
Game play	20	Score	Score	Score
Player's information	20	Score	Score	Score
Advanced features	-			
Draw poker	30	Score	Score	Score
5-card stud	30	Score	Score	Score
Color / Sound effects	30	Score	Score	Score
Leader board	30	Score	Score	Score
Creations	30 / each	Score	Score	Score
Report	-			
	100	Score	Score	Score
	Total	Score	Score	Score

Table 2: Table of contribution

This project breaks into three significant parts, which are the core game logic, the GUI, and the testing process, which make up the majority of the work. The core game logic was the most challenging part, as it required a deep understanding of the game rules and the ability to implement them in C++. The GUI was the most time-consuming part, as it required a lot of trial and error to get the layout and design right. The testing process was the most tedious part, as it required a lot of manual testing to ensure that the game was bug-free and worked as expected. Each part of the project had its challenges, but we were able to overcome them by working together and supporting each other throughout the process, more details will be discussed in each of the following sections.

To enhance the project a little further, we used the SDL2 library to make a graphical user interface for the Poker game. Such GUI makes interactions with the game much easier and fun. It gives good visual clarity so that one understands the actual happening of the game better. In that way, by integrating SDL2 into our project, we made it more fun, not to mention challenging as well.

2.1 About the Poker Game

2.1.1 Poker Hand Rakings

In Poker, a hand comprises five cards, irrespective of the cards being from a single deck or different decks provided the number is five altogether. Below are the official poker hand rankings, in order from highest to lowest.

1. **Royal Flush**: The best possible poker hand, formed by the Ace, King, Queen, Jack, and 10, all of the same suit, for example, hearts. (Note: This hand is not implemented in our project since it is very rare and not called for in the project specification.)

Figure 1: Royal Flush demonstration.

2. **Straight Flush**: Five sequential cards all of one suit (such as 9, 8, 7, 6, 5, all hearts).

Figure 2: Straight Flush demonstration.

3. **Four of a Kind**: Four cards of one rank (such as four Aces).

Placeholder

Figure 3: Four of a Kind demonstration.

4. **Full House**: Three cards of one rank and two cards of another rank (such as three Kings and two 4s).

Placeholder

Figure 4: Full House demonstration.

5. **Flush**: Any five cards of the same suit, but not in sequential order (such as 2, 4, 6, 9, King, all in diamonds).

Placeholder

Figure 5: Flush demonstration.

6. **Straight**: Five consecutive cards of different suits, like 10 of hearts, Jack of spades, Queen of diamonds, King of clubs, Ace of hearts.

Placeholder

Figure 6: Straight demonstration.

7. **Three of a Kind**: Three cards of the same rank, such as three 7s.

Placeholder

Figure 7: Three of a Kind demonstration.

8. **Two Pair**: Two cards of one rank and two cards of another rank, such as two 5s and two 9s.

Placeholder

Figure 8: Two Pair demonstration.

9. **One Pair**: Two cards of the same rank, such as two 8s.

Placeholder

Figure 9: One Pair demonstration.

10. **High Card**: If no other hand is made, then the highest card played becomes the rank (such as King, 10, 8, 6, 4; this would be a ”King-high” hand).

Placeholder

Figure 10: High Card demonstration.

In all, there are 10 possible Poker hands ranging from the highest, Royal Flush, to the lowest, High Card. However, in this project we will implement only hands ranked from Straight Flush to High Card due to the requirements of our project, since it's too rare to happen and the project requirements are not including this as well.

2.1.2 Poker Game Variants

Based on those information, there are a lot of variants of Poker, and each has its own rules and way of playing. Some popular variants include:

1. **Texas Hold'em**: Each player is dealt two private cards, and then five community cards are turned over on the table; players can use any number of these seven cards in attempting to build the highest possible five-card hand.
2. **Draw Poker**: Players are dealt five cards and can choose to discard and replace some of them (typically 1 to 3 cards at a time). After this, the player with the best hand wins. The gameplay is structured as follows:
   - **Round 1**: All players are dealt 5 cards each. Players must now decide whether to play on by selecting "raise, "call," or "fold".
   - **Round 2**: Players can discard and replace one to three cards once (in some versions, if a player has an Ace, he/she can discard four cards and draw four new ones).
   - **Round 3**: Players decide again to either "raise," "call," or "fold" depending on their now improved hands.
   - **Round 4**: Hands are revealed, and the player with the highest hand wins.
3. **Stud Poker**: Some cards are dealt face up and others face down, with several rounds of betting. The highest five-card hand wins the game.
4. **Simplified Poker**: This is a Poker variant, included in the project requirements, is subject to the rules listed below: Each player is initially dealt 5 cards. The player with the highest-ranking hand wins. In case of a tie, when two players hold the same hand, it is won by the player with the highest-ranking card. If two or more players have the same hand and the same high card, then the pot is divided equally. In this project.

We are going to implement two game variants: Simplified Poker and Draw Poker. Even though the Simplified Poker game fulfilled all the basic requirements set in the project, we felt it was a bit simple and basic. In order to make the project exciting and challenging, we decided to implement the Draw Poker variant. This is an extension of the framework set by the Simplified Poker game, further allowing us to fulfill some of the advanced requirements stipulated in the project.

2.2 Implementation of the Poker Game

In this project, we have deliberately avoided creating separate, isolated codebases for each game mode that we would like to have. Instead, we designed a unified structure, which contains all game modes under one implementation. This will let us take advantage of the object-oriented features of C++ - that is classes introduced in C++11—to achieve modularity and expandability. By doing so, we can easily modify or extend the project with more game modes in the future without rewriting the whole codebase.

2.2.1 Class Design

The game of Poker was implemented by using the C++ programming language with SDL2 library and IDE Visual Studio Code. For this, we used the C++11 class feature to design and implement the game. In fact, this usage of classes is beyond the basic requirements of the project. We did it for the following two main reasons:

1. **Pratical Learning**: This exercise used classes, which is good programming practice and prepares us for the future, more complex projects.
2. **Efficiency**: Classes allow for encapsulation of data and methods, making it easier to extend functionality and organize compared to the traditional struct in C99.

In this project, we used classes in a way similar to how struct would be used in C99, with the added flexibility - thanks to encapsulation - of being able to pack both data and behavior. This allowed us to create better-organized code that is maintainable and more readable.

So we will explain each of the class we've implemented for this project.

1. **Card**: This class is used to represent a single card in a standard 52-card deck. This class is used to define the suit and rank of a card. The suit and rank are represented by two enumerations, `Suits` and `Ranks`, respectively. The enumerate variable is initialized with the values of `-1` to `12` for the ranks and `-1` to `3` for the suits. The class also contains a method for printing the card to the console at the early state of the project. Card Implementation.
   - `string suitToString()`: This method converts the suit of the card to a string representation via a switch statement.
   - `string rankToString()`: This method converts the rank of the card to a string representation via a switch statement.
2. **Deck**: This class is used to represent a Standard 52-card deck. This class is built up using the `Card` class. The main idea is having an array of 52 objects of the type `Card` (Card cards[52]). In this file, we develop the methods for shuffling the deck, dealing a card, and reseting the deck. Deck Implementation.
   - `void setup()`: This method initializes the deck with 52 cards, one of each rank and suit.
   - `void shuffle()`: This method uses the `std::shuffle` function with a Mersenne Twister random number generator (`std::mt19937`) seeded with the current time (`std::time(0)`) to ensure a unique shuffle each time.
   - `void reset()`: This method simply calls the `setup()` method and then the `shuffle()` method to reset the deck. Also set the remaining card back to 52.

3. **Strength**: This class is for represent the strength of a hand in Poker. This class contains methods for calculating the strength of a hand, checking the rank of the hand as we've mentioned above Poker Hand Rankings.

4. **Hand**: This class is used to represent a hand of card in the game. This class is built up using the `Card` class. The main idea is having an array of 5 objects of the type `Card` (`Card cards[5]`). In this file we develop the methods for sorting the hand, checking the rank of the hand, and evaluating the hand. Hand Implementation.

   - `void show()`: A simple for loop that prints out the cards in the hand.
   - `void sortCards()`: For this to work, we configure the standard sort function to sort the cards in the hand by rank using a lambda function. Then assign those card into a `Card sortedCards[5]` array.
   - `void evaluateHand()`: This method calls the `sortCards()` method and the `Strength` class to use its methods to evaluate the hand.

3 List of References

- LaTeX template
- LaTeX on Visual Studio Code tutorial
- SDL-2.30.9 library
- SDL-2.30.9 installation tutorial
- CMake-3.21.3 library