

BLACKJACK

Card game

Project 2

T.md Najib Sahab

CSC-5

Summer 2024

45760

Introduction

Blackjack, also known as 21, is a classic card game enjoyed by many around the world. It combines elements of skill, strategy, and luck, making it a favorite in both casual and casino settings. The goal is simple: beat the dealer by having a hand value closer to 21 without going over. In this version, there is also imaginary money involved. The more you win, the more you earn. Inspired by the real game of Blackjack, I developed a program to simulate the game, allowing players to test their luck and strategy on their computer.

How the Card Game Works

Object of the Game

The goal is to acquire a hand value that is closer to 21 than the dealer's hand without exceeding 21 and to get however much imaginary money you want.

Rules of the Game

Blackjack is typically played between a player and a dealer. The game follows these basic rules:

1. **Card Values:**
 - Number cards (2-10) are worth their face value.
 - Face cards (Jack, Queen, King) are each worth 10 points.
 - Aces can be worth either 1 or 11 points, depending on which value benefits the hand most.
2. **Initial Deal:**
 - Both the player and the dealer are dealt two cards each. The player's cards are both face up, while the dealer has both of his cards hidden.
3. **Player's Turn:**
 - The player can choose to "hit" (draw another card) or "stand" (keep their current hand). The player can continue to hit until they either stand or their hand value exceeds 21 (busts).
4. **Dealer's Turn:**
 - The dealer reveals their hidden card and must hit until their hand value is 17 or higher. If the dealer busts, the player wins.
5. **Winning the Game:**
 - If the player's hand value is closer to 21 than the dealer's, the player wins. If the player busts, the dealer wins. If both the player and the dealer have

the same hand value, it's a tie. The dealer can choose whether they want to stop playing the game with whichever imaginary money they have or can choose to continue playing if they have more than enough money to keep going.

How the Program Works

Initialization

The program begins by initializing the random seed and declaring variables to store the player's money, bet amount, game results, and other game-related data. The player starts by choosing their initial amount of money (with a minimum of 11 and a maximum of 200). The player can then choose how much of their total amount they want to bet for the next round.

Main Game Loop

The game proceeds in rounds. For each round:

- The player places a bet.
- Initial cards are dealt to both the player and the dealer.
- The player's cards and one of the dealer's cards are displayed.
- The player can choose to "hit" (take another card) or "stand" (keep their current hand) until they either bust (exceed a total of 21) or decide to stand.
- If the player stands without busting, the dealer reveals their hidden card and continues to hit until their hand value reaches at least 17.

Dealer's Play

The dealer's turn is automated:

- The dealer keeps drawing cards ("hits") until the total value of their hand is at least 17.
- Aces in the dealer's hand are treated strategically to avoid busting whenever possible.

Determining the Winner

After both the player and the dealer have completed their turns, the program calculates the final hand values:

- If the player busts, the dealer wins.
- If the dealer busts, the player wins.
- If neither busts, the hand closest to 21 without exceeding it wins.
- The player wins the amount they bet if they win the round. If the dealer wins, the player's bet amount is subtracted from their total money. In the case of a tie, the bet is returned to the player.

End of Game

The game continues until the player either chooses to stop playing or runs out of money. The total number of games played, along with game statistics (player wins, dealer wins, ties), are displayed at the end. The game results are also recorded in a file for future reference.

Code Analysis

The program is written in C++ and utilizes standard libraries for various functionalities:

- **Random Number Generation:** Used to simulate the dealing of cards.
- **Input Validation:** Ensures that the player's inputs for money and bet amounts are within valid ranges.
- **Game Logic:** Implements the rules of Blackjack, including handling Aces, player decisions, and dealer behavior.
- **File Output:** Records game results to a file for future reference.
- **Sorting and Searching:** Implements functions for sorting and searching scores using selection sort and binary search.

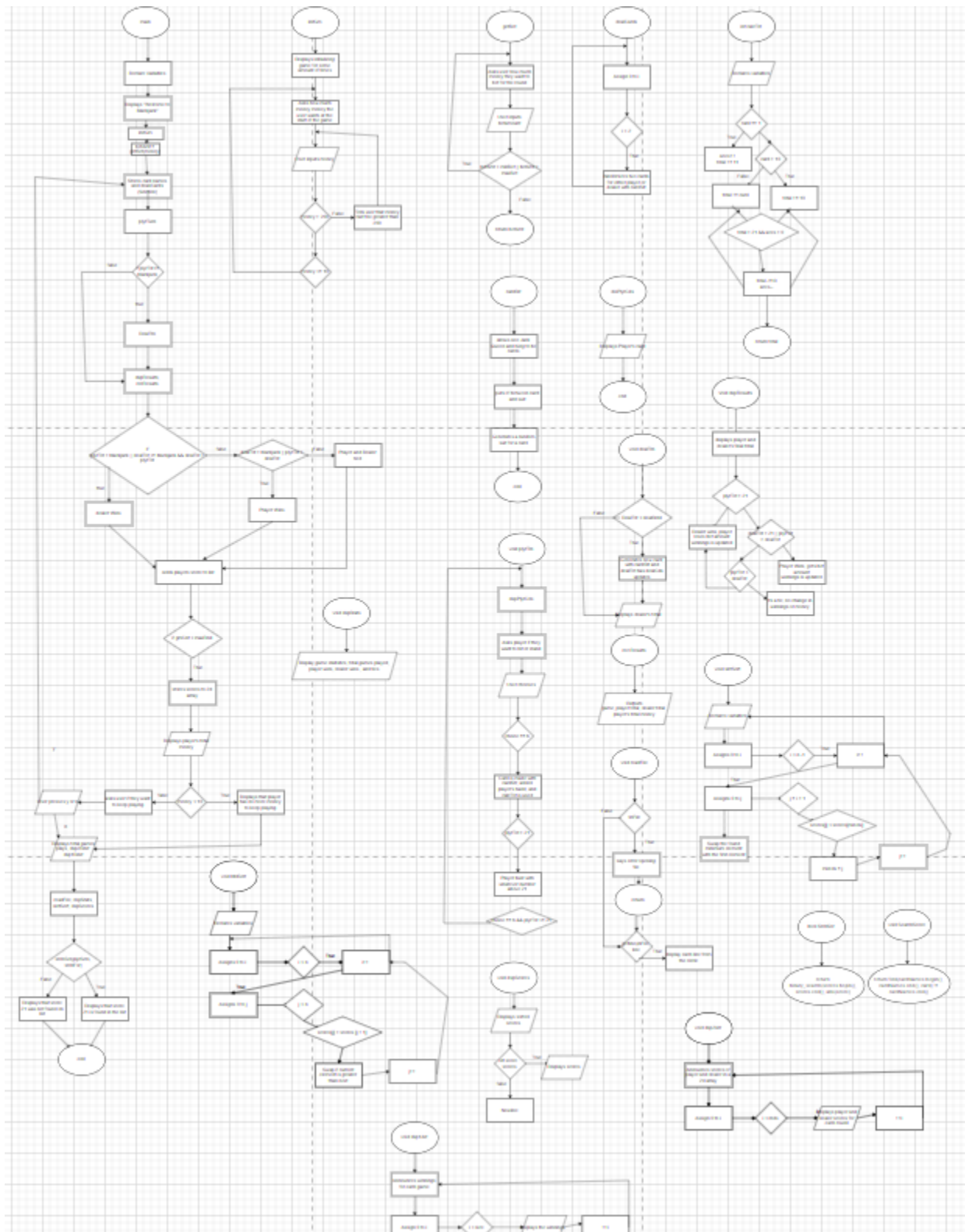
References: book, class lectures, and repository

(pseudocode is modeled within comments of final code or version 4)

Differences between code and real game:

- Usually the game has more than one player. Because the program is not a multiplayer game, there are faster rounds compared to games played in real life.
- Final scores or other displays don't use arrays quite often in real games compared to my program.
- Aces are dealt as 11 in the program, but are adjusted to 1 if the value exceeds 21 in the game while the player can choose the value of an Ace in the real game.
- The code is limited to two options only: stand and hit. In the real Blackjack game, there are options like doubling down, splitting pairs, and taking insurance.

FlowChart (if not clear here, pdf is available with this file and on github)



Conclusion

This Blackjack program provides a simple yet engaging simulation of the classic card game. By combining random elements with strategic decision-making, it offers players an opportunity to experience the excitement of Blackjack in a virtual setting. Whether you're an experienced or a new player, this could be a new and fun experience for new users.

iostream Library

Name	Frequency	Description
static_cast	1	seeds the random number generator
Cout	39	Display output
Cin	4	Display input

cstdlib Library

Name	Frequency	Description
Rand	14	chooses suit, and value of card With dealer's bet amount
Srand	1	random # seed
Exit	3	Exists the program on error

ctime Library

Name	Frequency	Description
Time	1	seeds random number generator

iomanip Library

Name	Frequency	Description
Setprecision	2	formats money
Fixed	2	formats money

string Library

Name	Frequency	Description
string	20	storing card information and declaring variables

cmath Library

Name	Frequency	Description
Abs	20	absolute value

fstream Library

Name	Frequency	Description
ofstream	3	opening, writing in, and closing file
ifstream	3	reading from a file
Outfile	12	Instance for managing output file operations

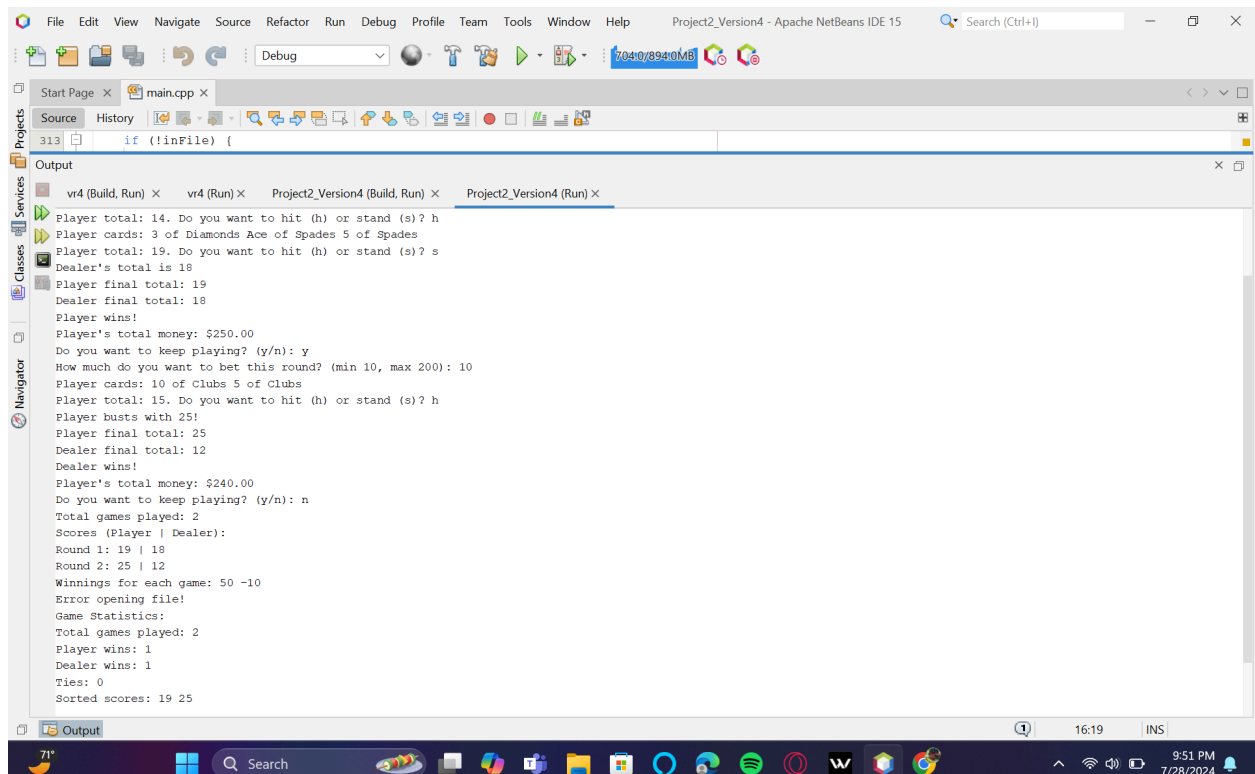
vector Library

Name	Frequency	Description
Vector	38	Storing and managing a dynamic array of values

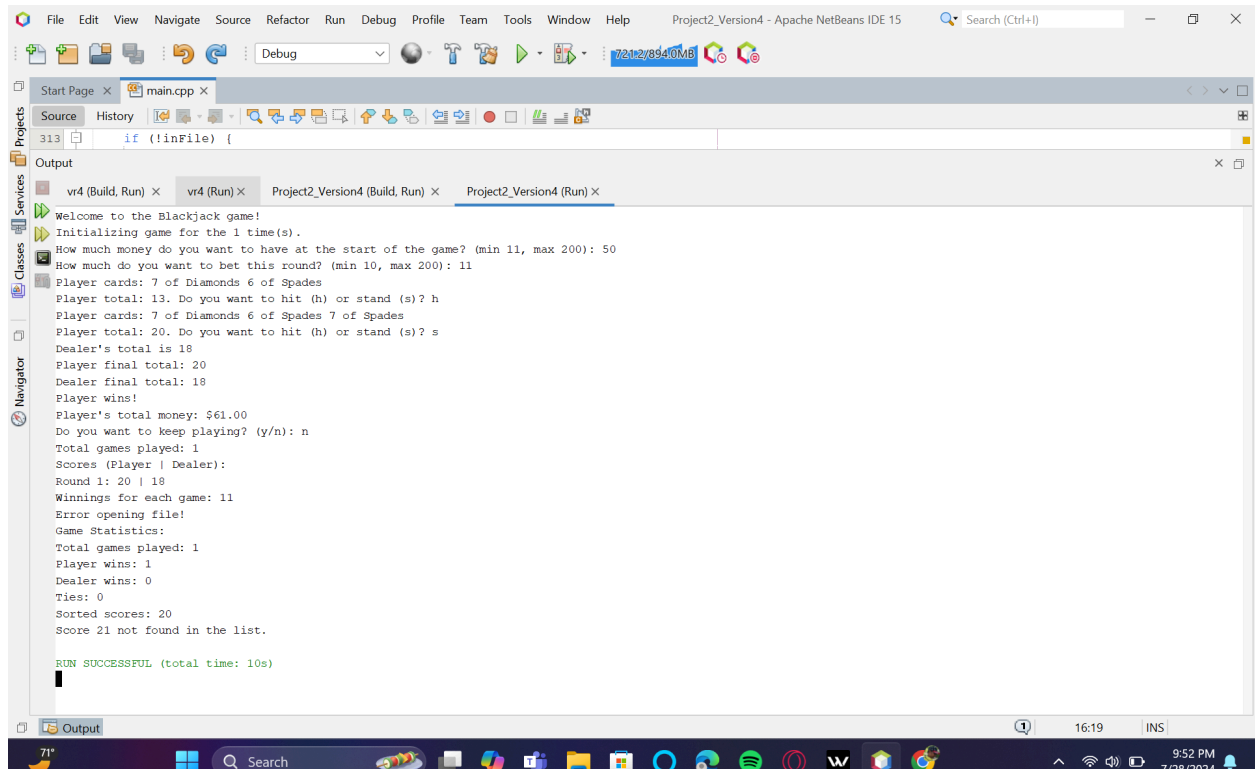
algorithm Library

Name	Frequency	Description
Find	1	searching for an element in a vector
Sort	2	Sorting elements in a vector
Swap	2	swapping elements in a vector
Binary_search	1	searching for an element in a sorted vector

Screenshots of the Code in action:



```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help Project2_Version4 - Apache NetBeans IDE 15 Search (Ctrl+I)
Debug
Start Page x main.cpp x
Source History
313 if (!infile) {
Output
vr4 (Build, Run) x vr4 (Run) x Project2_Version4 (Build, Run) x Project2_Version4 (Run) x
Player total: 14. Do you want to hit (h) or stand (s)? h
Player cards: 3 of Diamonds Ace of Spades 5 of Spades
Player total: 19. Do you want to hit (h) or stand (s)? s
Dealer's total is 18
Player final total: 19
Dealer final total: 18
Player wins!
Player's total money: $250.00
Do you want to keep playing? (y/n): y
How much do you want to bet this round? (min 10, max 200): 10
Player cards: 10 of Clubs 5 of Clubs
Player total: 15. Do you want to hit (h) or stand (s)? h
Player busts with 25!
Player final total: 25
Dealer final total: 12
Dealer wins!
Player's total money: $240.00
Do you want to keep playing? (y/n): n
Total games played: 2
Scores (Player | Dealer):
Round 1: 19 | 18
Round 2: 25 | 12
Winnings for each game: 50 -10
Error opening file!
Game Statistics:
Total games played: 2
Player wins: 1
Dealer wins: 1
Ties: 0
Sorted scores: 19 25
Output 16:19 INS 7/28/2024
```



```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help Project2_Version4 - Apache NetBeans IDE 15 Search (Ctrl+I)
Debug
Start Page x main.cpp x
Source History
313 if (!infile) {
Output
vr4 (Build, Run) x vr4 (Run) x Project2_Version4 (Build, Run) x Project2_Version4 (Run) x
Welcome to the Blackjack game!
Initializing game for the 1 time(s).
How much money do you want to have at the start of the game? (min 11, max 200): 50
How much do you want to bet this round? (min 10, max 200): 11
Player cards: 7 of Diamonds 6 of Spades
Player total: 13. Do you want to hit (h) or stand (s)? h
Player cards: 7 of Diamonds 6 of Spades 7 of Spades
Player total: 20. Do you want to hit (h) or stand (s)? s
Dealer's total is 18
Player final total: 20
Dealer final total: 18
Player wins!
Player's total money: $61.00
Do you want to keep playing? (y/n): n
Total games played: 1
Scores (Player | Dealer):
Round 1: 20 | 18
Winnings for each game: 11
Error opening file!
Game Statistics:
Total games played: 1
Player wins: 1
Dealer wins: 0
Ties: 0
Sorted scores: 20
Score 21 not found in the list.
RUN SUCCESSFUL (total time: 10s)
Output 16:19 INS 7/28/2024
```


Project2_Version4 - Apache NetBeans IDE 15

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Debug

767/278940MB

Start Page x main.cpp x

Source History

313 if (!infile) {

Output

vr4 (Build, Run) x vr4 (Run) x Project2_Version4 (Build, Run) x Project2_Version4 (Run) x

Welcome to the Blackjack game!
Initializing game for the 1 time(s).
How much money do you want to have at the start of the game? (min 11, max 200): 10
How much money do you want to have at the start of the game? (min 11, max 200): 10
How much money do you want to have at the start of the game? (min 11, max 200): 210
Overflow error: Starting money cannot be greater than 200. Starting Price is \$200
How much do you want to bet this round? (min 10, max 200): 11
Player cards: 2 of Spades King of Hearts
Player total: 12. Do you want to hit (h) or stand (s)? s
Dealer's total is 20
Player final total: 12
Dealer final total: 20
Dealer wins!
Player's total money: \$189.00
Do you want to keep playing? (y/n): y
How much do you want to bet this round? (min 10, max 200): 180
Player cards: Jack of Clubs 6 of Clubs
Player total: 16. Do you want to hit (h) or stand (s)? s
Dealer's total is 21
Player final total: 16
Dealer final total: 21
Dealer wins!
Player's total money: \$9.00
You don't have enough money to continue playing. Game over!

RUN SUCCESSFUL (total time: 23s)

Output 16:19 INS

71° Search 9:52 PM 7/28/2024

Project2_Version4 - Apache NetBeans IDE 15

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Debug

677/178940MB

Start Page x main.cpp x

Source History

313 if (!infile) {

Output

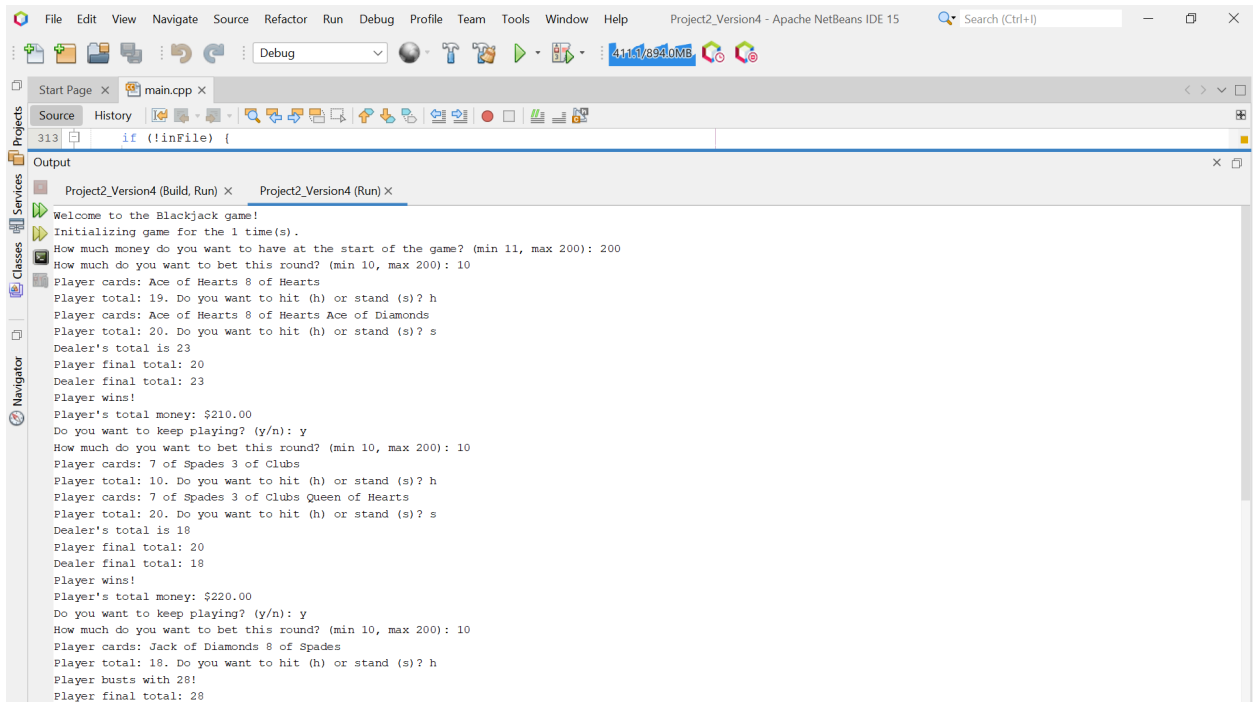
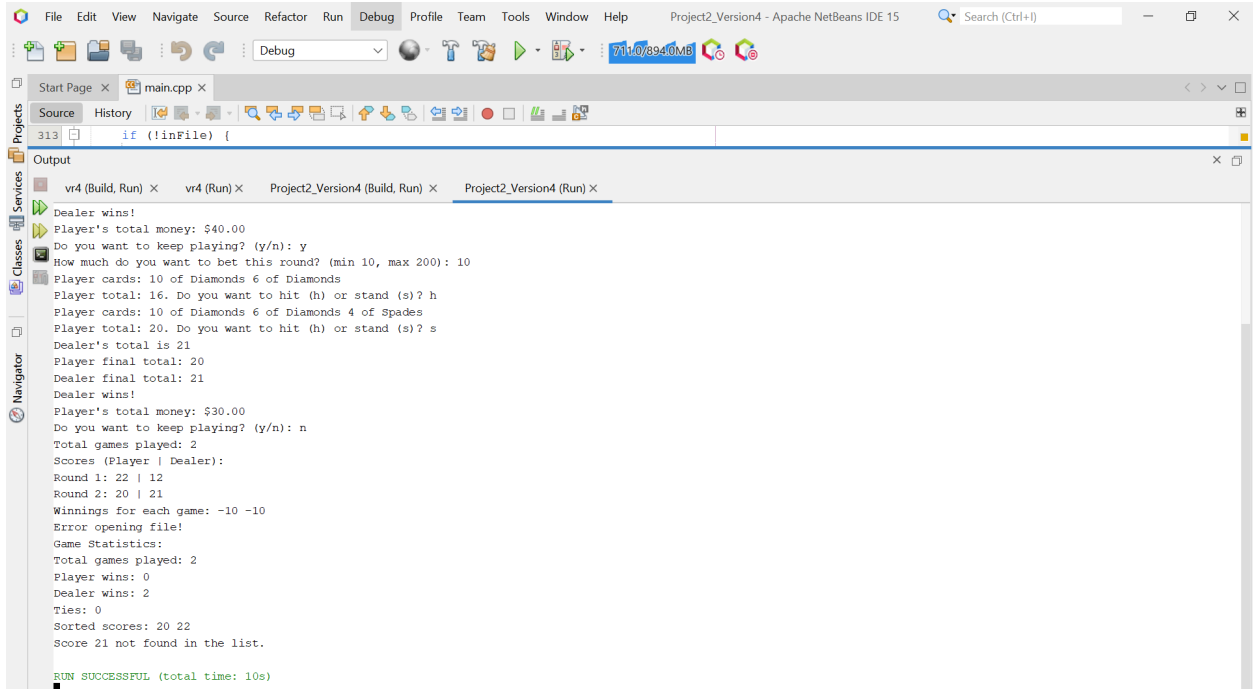
vr4 (Build, Run) x vr4 (Run) x Project2_Version4 (Build, Run) x Project2_Version4 (Run) x

Do you want to keep playing? (y/n): y
How much do you want to bet this round? (min 10, max 200): 30
Player cards: 4 of Hearts 7 of Clubs
Player total: 11. Do you want to hit (h) or stand (s)? h
Player cards: 4 of Hearts 7 of Clubs 9 of Spades
Player total: 20. Do you want to hit (h) or stand (s)? s
Dealer's total is 18
Player final total: 20
Dealer final total: 18
Player wins!
Player's total money: \$240.00
Do you want to keep playing? (y/n): n
Total games played: 4
Scores (Player | Dealer):
Round 1: 20 | 18
Round 2: 23 | 20
Round 3: 20 | 22
Round 4: 20 | 18
Winnings for each game: 10 -10 10 30
Error opening file!
Game Statistics:
Total games played: 4
Player wins: 3
Dealer wins: 1
Ties: 0
Sorted scores: 20 20 20 23
Score 21 not found in the list.

RUN SUCCESSFUL (total time: 32s)

Output 16:19 INS

71° Search 9:54 PM 7/28/2024



```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help Project2_Version4 - Apache NetBeans IDE 15 Search (Ctrl+I)
Debug 7240/894.0MB
Start Page x main.cpp x
Source History
313 if (!infile) {
Output
Project2_Version4 (Build, Run) x Project2_Version4 (Run) x
Do you want to keep playing? (y/n): y
How much do you want to bet this round? (min 10, max 200): 10
Player cards: 4 of Hearts 2 of Clubs
Player total: 6. Do you want to hit (h) or stand (s)? h
Player cards: 4 of Hearts 2 of Clubs Jack of Diamonds
Player total: 16. Do you want to hit (h) or stand (s)? h
Player cards: 4 of Hearts 2 of Clubs Jack of Diamonds 4 of Clubs
Player total: 20. Do you want to hit (h) or stand (s)? s
Dealer's total is 24
Player final total: 20
Dealer final total: 24
Player wins!
Player's total money: $220.00
Do you want to keep playing? (y/n): n
Total games played: 2
Scores (Player | Dealer):
Round 1: 20 | 17
Round 2: 20 | 24
Winnings for each game: 10 10
Error opening file!
Game Statistics:
Total games played: 2
Player wins: 2
Dealer wins: 0
Ties: 0
Sorted scores: 20 20
Score 21 not found in the list.
RUN SUCCESSFUL (total time: 12s)
```

```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help Project2_Version4 - Apache NetBeans IDE 15 Search (Ctrl+I)
Debug 422.17894.0MB
Start Page x main.cpp x
Source History
313 if (!infile) {
Output
Project2_Version4 (Build, Run) x Project2_Version4 (Run) x
Welcome to the Blackjack game!
Initializing game for the 1 time(s).
How much money do you want to have at the start of the game? (min 11, max 200): 200
How much do you want to bet this round? (min 10, max 200): 10
Player cards: Jack of Hearts Jack of Diamonds
Player total: 20. Do you want to hit (h) or stand (s)? s
Dealer's total is 17
Player final total: 20
Dealer final total: 17
Player wins!
Player's total money: $210.00
Do you want to keep playing? (y/n): y
How much do you want to bet this round? (min 10, max 200): 10
Player cards: 4 of Hearts 2 of Clubs
Player total: 6. Do you want to hit (h) or stand (s)? h
Player cards: 4 of Hearts 2 of Clubs Jack of Diamonds
Player total: 16. Do you want to hit (h) or stand (s)? h
Player cards: 4 of Hearts 2 of Clubs Jack of Diamonds 4 of Clubs
Player total: 20. Do you want to hit (h) or stand (s)? s
Dealer's total is 24
Player final total: 20
Dealer final total: 24
Player wins!
Player's total money: $220.00
Do you want to keep playing? (y/n): n
Total games played: 2
Scores (Player | Dealer):
Round 1: 20 | 17
Round 2: 20 | 24
Winnings for each game: 10 10
```

Final code (Version 4)

```
#include <iostream>
#include <iomanip>
```

```

#include <cmath>
#include <cstdlib>
#include <fstream>
#include <string>
#include <ctime>
#include <vector>
#include <algorithm>

//System Libraries
using namespace std;

//User Libraries

//Function Prototypes
void initGm(float &money); // Initialize the game with starting money
int getBet(float money, int minBet = 10, int maxBet = 200); // Get the bet amount for the current
round
void dealCards(vector<int> &plyrCds, vector<int> &dealCds, vector<string> &plyrNms,
vector<string> &dealNms); // Deal initial cards to player and dealer
string cardStr(int card, int suit); // Convert a card number and suit to a string representation
void dspPlyrCds(const vector<string> &cards); // Display the player's cards
int calcTot(const vector<int> &cards); // Calculate the total value of the cards
void plyrTrn(vector<int> &plyrCds, vector<string> &plyrNms, int &plyrTot); // Player's turn to hit
or stand
void dealTrn(vector<int> &dealCds, vector<string> &dealNms, int &dealTot, int dealStnd); //
Dealer's turn to hit or stand
void dspResults(int plyrTot, int dealTot, float &money, int betAmt, int &winnings); // Display the
final results of the game
void recResults(ofstream &outFile, int gmCnt, int plyrTot, int dealTot, float money); // Record the
game results in a file
void dsp1DArr(const int winnings[], int size); // Function prototype for displaying 1D array
void readFile(ifstream &inFile); // Read and display player scores from a file
void dspStats(int totGms, int plyrWns, int dealWns, int ties); // Display game statistics at the end
void bblSort(vector<int> &scores); // Bubble sort to sort scores
void slctSort(vector<int> &scores); // Selection sort to sort scores
void dspScores(const vector<int> &scores); // Display the sorted scores
bool srchScr(const vector<int> &scores, int score); // Search for a specific score in the sorted list
bool srchCrd(const vector<string> &cardNms, const string &card); // Overloaded function to
search for a specific card in the player's hand
void dsp2DArr(const int scores[][2], int rnds); void dsp1DArr(const int winnings[], int size); //
Function prototype for displaying 1 and 2D array

//Execution Begins Here
int main(int argc, char** argv) {

```

```

//Seed the random number generator
srand(static_cast<unsigned int>(time(0))); // Seed the random number generator

//Declare Variables
char again; // Variable to check if player wants to play again
float money; // Variable for player's money
int betAmt; // Variable for bet amount
ofstream outFile("game_results.txt"); if (!outFile) { cout << "Error opening file to write results!"
<< endl; exit(1);} // Exit if file cannot be opened
ifstream inFile("game_scores.txt"); // File to read player scores
int gmCnt = 0; // Count of games played
int plyrWns = 0; // Count of player wins
int dealWns = 0; // Count of dealer wins
int ties = 0; // Count of ties
const int maxRnd = 10; // Maximum number of rounds
vector<int> plyrScrs; // Vector to store player scores
int scores[maxRnd][2] = {}; int winnings[maxRnd] = {}; // 2D array to store player and dealer
scores and 1d to store player winnings

//Welcome message
cout << "Welcome to the Blackjack game!" << endl; // Welcome message

//Initialize Variables
initGm(money); // Initialize the game with starting money

//Map Inputs to Outputs -> Process
//Main game loop
do {
    const int blackjack = 21; // Constant for blackjack value
    const int dealStnd = 17; // Constant for dealer stand value

    betAmt = getBet(money); // Get the bet amount for the current round

    // Vectors to store the cards and card names of the player and dealer
    vector<int> plyrCds;
    vector<int> dealCds;
    vector<string> plyrNms;
    vector<string> dealNms;

    dealCards(plyrCds, dealCds, plyrNms, dealNms); // Deal initial cards to player and dealer

    int plyrTot = calcTot(plyrCds); // Calculate the total value of player's cards
    int dealTot = calcTot(dealCds); // Calculate the total value of dealer's cards

```

```

    plyrTrn(plyrCds, plyrNms, plyrTot); // Player's turn to hit or stand

    if (plyrTot <= blackjack) { // If player does not bust, dealer takes their turn
        dealTrn(dealCds, dealNms, dealTot, dealStnd);
    }
    int roundWinnings = 0;
    dspResults(plyrTot, dealTot, money, betAmt, roundWinnings); // Display the final results of
the game
    winnings[gmCnt] = roundWinnings;
    recResults(outFile, gmCnt, plyrTot, dealTot, money); // Record the game results in a file

    // Update win counts based on game results
    if (plyrTot > blackjack || (dealTot <= blackjack && dealTot > plyrTot)) {
        dealWns++;
    } else if (dealTot > blackjack || plyrTot > dealTot) {
        plyrWns++;
    } else {
        ties++;
    }

    plyrScrs.push_back(plyrTot); // Add player score to the list

    if (gmCnt < maxRnd) { // Store the scores in the 2D array
        scores[gmCnt][0] = plyrTot;
        scores[gmCnt][1] = dealTot;
    }

    cout << "Player's total money: $" << fixed << setprecision(2) << money << endl; // Display
the player's total money

    if (money < 10) { // Check if the player has enough money to continue playing
        cout << "You don't have enough money to continue playing. Game over!" << endl;
        outFile.close(); inFile.close(); exit(0);
        // End the game if the player doesn't have enough money
    }

    cout << "Do you want to keep playing? (y/n): "; // Ask the player if they want to keep
playing
    cin >> again;

    gmCnt++; // Increment the number of games played

    } while (again == 'y' && money >= 10 && gmCnt < maxRnd); // Repeat loop if player wants to
play again and has enough money

```

```
//Display Inputs/Outputs
cout << "Total games played: " << abs(gmCnt) << endl; // Display the total number of games
played
```

```
    dsp2DArr(scores, gmCnt); dsp1DArr(winnings, gmCnt); // Display all scores using the 2D
array and winnings with 1d array
```

```
//Exit the Program - Cleanup
```

```
outFile.close(); // Close the output file
```

```
readFile(inFile); // Read and display player scores from a file
```

```
dspStats(gmCnt, plyrWns, dealWns, ties); // Display game statistics
```

```
slctSort(plyrScrs); // Sort and display player scores using selection sort
```

```
dspScores(plyrScrs);
```

```
int srchFor = 21; // Search for a specific score in the sorted list
```

```
if (srchScr(plyrScrs, srchFor)) {
```

```
    cout << "Score " << abs(srchFor) << " found in the list." << endl;
```

```
} else {
```

```
    cout << "Score " << abs(srchFor) << " not found in the list." << endl;
```

```
}
```

```
return 0;
```

```
}
```

```
// Initialize the game with the starting money
```

```
void initGm(float &money) {
```

```
    static int initCls = 0; // Static variable to count the number of times the function is called
```

```
    initCls++;
```

```
    cout << "Initializing game for the " << abs(initCls) << " time(s)." << endl;
```

```
    do {
```

```
        cout << "How much money do you want to have at the start of the game? (min 11, max
200): ";
```

```
        cin >> money;
```

```
        if (money > 200) {
```

```
            cout << "Overflow error: Starting money cannot be greater than 200. Starting Price is
$200" << endl;
```

```
            money = 200;
```

```
        }
```

```
    } while (money <= 10);
```

```
}
```

```
// Get the bet amount for the current round with a default minimum bet amount
```

```
int getBet(float money, int minBet, int maxBet) {
```

```

    int betAmt;
    do {
        cout << "How much do you want to bet this round? (min " << abs(minBet) << ", max " <<
abs(maxBet) << "): ";
        cin >> betAmt;
    } while (betAmt < minBet || betAmt > maxBet);
    return betAmt;
}

```

```

// Deal the initial cards to the player and dealer
void dealCards(vector<int> &plyrCds, vector<int> &dealCds, vector<string> &plyrNms,
vector<string> &dealNms) {
    for (int i = 0; i < 2; i++) {
        int card = rand() % 13 + 1;
        int suit = rand() % 4;
        plyrCds.push_back(card);
        plyrNms.push_back(cardStr(card, suit));

        card = rand() % 13 + 1;
        suit = rand() % 4;
        dealCds.push_back(card);
        dealNms.push_back(cardStr(card, suit));
    }
}

```

```

// Convert a card number and suit to a string representation
string cardStr(int card, int suit) {
    string crdStr;
    switch (card) {
        case 1: crdStr = "Ace"; break;
        case 11: crdStr = "Jack"; break;
        case 12: crdStr = "Queen"; break;
        case 13: crdStr = "King"; break;
        default: crdStr = to_string(card);
    }
    crdStr += " of ";
    switch (suit) {
        case 0: crdStr += "Clubs"; break;
        case 1: crdStr += "Diamonds"; break;
        case 2: crdStr += "Hearts"; break;
        case 3: crdStr += "Spades"; break;
    }
    return crdStr;
}

```



```

// Display the player's cards
void dspPlyrCds(const vector<string> &cards) {
    cout << "Player cards: ";
    for (const string &card : cards) {
        cout << card << " ";
    }
    cout << endl;
}

```

```

// Calculate the total value of the cards
int calcTot(const vector<int> &cards) {
    int total = 0;
    int aces = 0;
    for (int card : cards) {
        if (card == 1) {
            aces++;
            total += 11;
        } else if (card > 10) {
            total += 10;
        } else {
            total += card;
        }
    }
    while (total > 21 && aces > 0) {
        total -= 10;
        aces--;
    }
    return total;
}

```

```

// Player's turn to hit or stand
void plyrTrn(vector<int> &plyrCds, vector<string> &plyrNms, int &plyrTot) {
    char choice;
    do {
        dspPlyrCds(plyrNms); // Show player's cards
        cout << "Player total: " << abs(plyrTot) << ". Do you want to hit (h) or stand (s)? "; // Ask
        player if they want to hit or stand
        cin >> choice;
        if (choice == 'h') {
            int card = rand() % 13 + 1;
            int suit = rand() % 4;
            plyrCds.push_back(card); // Add new card to player's hand

```

```

        plyrNms.push_back(cardStr(card, suit)); // Convert card to string and add to player's
card names
        plyrTot = calcTot(plyrCds); // Calculate new total of player's cards
        if (plyrTot > 21) {
            cout << "Player busts with " << abs(plyrTot) << "!" << endl; // Player busts if total is
over 21
        }
    }
} while (choice == 'h' && plyrTot <= 21);
}

```

// Dealer's turn to hit until they reach the stand limit

```

void dealTrn(vector<int> &dealCds, vector<string> &dealNms, int &dealTot, int dealStnd) {
    while (dealTot < dealStnd) {
        int card = rand() % 13 + 1;
        int suit = rand() % 4;
        dealCds.push_back(card); // Add new card to dealer's hand
        dealNms.push_back(cardStr(card, suit)); // Convert card to string and add to dealer's card
names
        dealTot = calcTot(dealCds); // Calculate new total of dealer's cards
    }
    cout << "Dealer's total is " << abs(dealTot) << endl;
}

```

// Display the final results of the game

```

void dspResults(int plyrTot, int dealTot, float &money, int betAmt, int &winnings) {
    cout << "Player final total: " << abs(plyrTot) << endl;
    cout << "Dealer final total: " << abs(dealTot) << endl;

    if (plyrTot > 21) {
        cout << "Dealer wins!" << endl; winnings = -betAmt; // Update winnings for this round
        money -= betAmt; // Dealer wins, player loses bet amount
    } else if (dealTot > 21 || plyrTot > dealTot) {
        cout << "Player wins!" << endl; winnings = betAmt; // Update winnings for this round
        money += betAmt; // Player wins, player gains bet amount
    } else if (plyrTot < dealTot) {
        cout << "Dealer wins!" << endl; winnings = -betAmt; // Update winnings for this round
        money -= betAmt; // Dealer wins, player loses bet amount
    } else {
        cout << "It's a tie!" << endl; winnings = 0; // No change in winnings for a tie and game is tie
    }
}

```

// Record the game results in a file

```

void recResults(ofstream &outFile, int gmCnt, int plyrTot, int dealTot, float money) {
    outFile << "Game " << gmCnt + 1 << ": " << endl;
    outFile << "Player total: " << abs(plyrTot) << endl;
    outFile << "Dealer total: " << abs(dealTot) << endl;
    outFile << "Player's total money: $" << fixed << setprecision(2) << abs(money) << endl;
    outFile << "-----" << endl;
}

```

// Read and display player scores from a file

```

void readFile(ifstream &inFile) {
    if (!inFile) {
        cout << "Error opening file!" << endl;
        return;
    }

    string line;
    while (getline(inFile, line)) {
        cout << line << endl; // Display each line from the file
    }
}

```

// Display game statistics at the end

```

void dspStats(int totGms, int plyrWns, int dealWns, int ties) {
    cout << "Game Statistics:" << endl;
    cout << "Total games played: " << abs(totGms) << endl;
    cout << "Player wins: " << abs(plyrWns) << endl;
    cout << "Dealer wins: " << abs(dealWns) << endl;
    cout << "Ties: " << abs(ties) << endl;
}

```

// Bubble sort to sort scores

```

void bblSort(vector<int> &scores) {
    int n = scores.size();
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (scores[j] > scores[j + 1]) {
                swap(scores[j], scores[j + 1]); // Swap if current element is greater than next
            }
        }
    }
}

```

```

    }
}
}

```

// Selection sort to sort scores

```

void slctSort(vector<int> &scores) {
    int n = scores.size();
    for (int i = 0; i < n - 1; i++) {
        int minIdx = i;
        for (int j = i + 1; j < n; j++) {
            if (scores[j] < scores[minIdx]) {
                minIdx = j;
            }
        }
        swap(scores[minIdx], scores[i]); // Swap the found minimum element with the first element
    }
}

```

// Display the sorted scores

```

void dspScores(const vector<int> &scores) {
    cout << "Sorted scores: ";
    for (int score : scores) {
        cout << abs(score) << " ";
    }
    cout << endl;
}

```

// Search for a specific score in the sorted list

```

bool srchScr(const vector<int> &scores, int score) {
    return binary_search(scores.begin(), scores.end(), abs(score));
}

```

// Overloaded function to search for a specific card in the player's hand

```

bool searchScore(const vector<string> &cardNames, const string &card) {
    return find(cardNames.begin(), cardNames.end(), card) != cardNames.end();
}

```

// Display all scores from the 2D array

```

void dsp2DArr(const int scores[][2], int rnds) {
    cout << "Scores (Player | Dealer):" << endl;
    for (int i = 0; i < rnds; ++i) {
        cout << "Round " << i + 1 << ": " << scores[i][0] << " | " << scores[i][1] << endl; // Display
        player and dealer scores for each round
    }
}

```

```
}  
// Displays winning in 1d array  
void dsp1DArr(const int winnings[], int size) { cout << "Winnings for each game: "; for (int i = 0; i  
< size; ++i) { cout << winnings[i] << " "; } cout << endl; }
```