Black Jack

Card Game

Zhang Qingkai CIS 5

Spring 2020

40402

## TABLE OF CONTENTS

[TABLE OF CONTENTS 2](#_Toc12385)

[Introduction 3](#_Toc29995)

[Summary 3](#_Toc15627)

[Description 4](#_Toc2845)

[The Logic of it All 5](#_Toc13954)

[Flowchart 5](#_Toc8053)

[Constructs & Concepts Utilized 3](#_Toc24425)

[iostream Library 3](#_Toc15000)

[cstdlib Library 3](#_Toc30103)

[iomanip Library 3](#_Toc17892)

[string Library 3](#_Toc23634)

[fstream Library 3](#_Toc8568)

[Data Types: 4](#_Toc7524)

[Conditional Statements: 4](#_Toc22909)

[Loops: 4](#_Toc8139)

[Linear Search: 4](#_Toc32388)

[Sorting: 4](#_Toc10526)

[Proof of a Working Product 5](#_Toc494)

[References 7](#_Toc9944)

[Program 7](#_Toc15994)

# Introduction

Title: Blackjack

Blackjack is the American version of a popular global banking game known as Twenty-One, It is a comparing card game between one or more players and a dealer, this project just containing one player and one dealer.

For the first round, the dealer is dealt two cards, normally one up (exposed) and one down (hidden), and two up for player. The value of cards two through ten is their pip value (2 through 10). Face cards (Jack, Queen, and King) are all worth ten. Aces can be worth one or eleven. A hand's value is the sum of the card values. Players are allowed to draw additional cards to improve their hands.

Once all the players have completed their hands, it is the dealer's turn. The dealer hand will not be completed if all players have either busted or received blackjacks. The dealer then reveals the hidden card and must hit until the cards total up to 17 points. At 17 points or higher the dealer must stay. (At most tables the dealer also hits on a "soft" 17, i.e. a hand containing an ace and one or more other cards totaling six.) You are betting that you have a better hand than the dealer. The better hand is the hand where the sum of the card values is closer to 21 without exceeding 21. The detailed outcome of the hand follows:

If the player is dealt an Ace and a ten-value card (called a "blackjack" or "natural"), and the dealer does not, the player wins and usually receives a bonus.

If the player exceeds a sum of 21 ("busts"); the player loses, even if the dealer also exceeds 21.

If the dealer exceeds 21 ("busts") and the player does not; the player wins.

If the player attains a final sum higher than the dealer and does not bust; the player wins.

If both dealer and player receive a blackjack or any other hands with the same sum, called a "push", no one wins.

# Summary

Project size: about 268 lines

The number of variables including 2 arrays, and around 50 int variables

Including 6 Functions, to be specific, 1 for linear search and 1 for sorting using selection sort.

This project includes many concepts that we learned from the chapters in the book, using parts of previous Project as well as the codes from homework that I used to code. Due to the limited time and the coding ability I owned, I cannot develop the project perfectly; it still has many possibilities to be extended for other concepts. For instance, developing it to multiple players or developing banking system.

It took almost two weeks because I tried to do many debugging work for this project to check for the logic of the game, as the poker game has so many potential rules, I found there are some tiny problems when I was debugging , and I fix them one by one. Since I have already tried multiple test case, but there still may be some logic problems remain that professor may notice. Please tell me once find that. Thanks for your sincere help.

To be honest, I’m really satisfied with this project; it is a kind of fulfillment for me since it is the first time for me to develop such a complex mini-project.

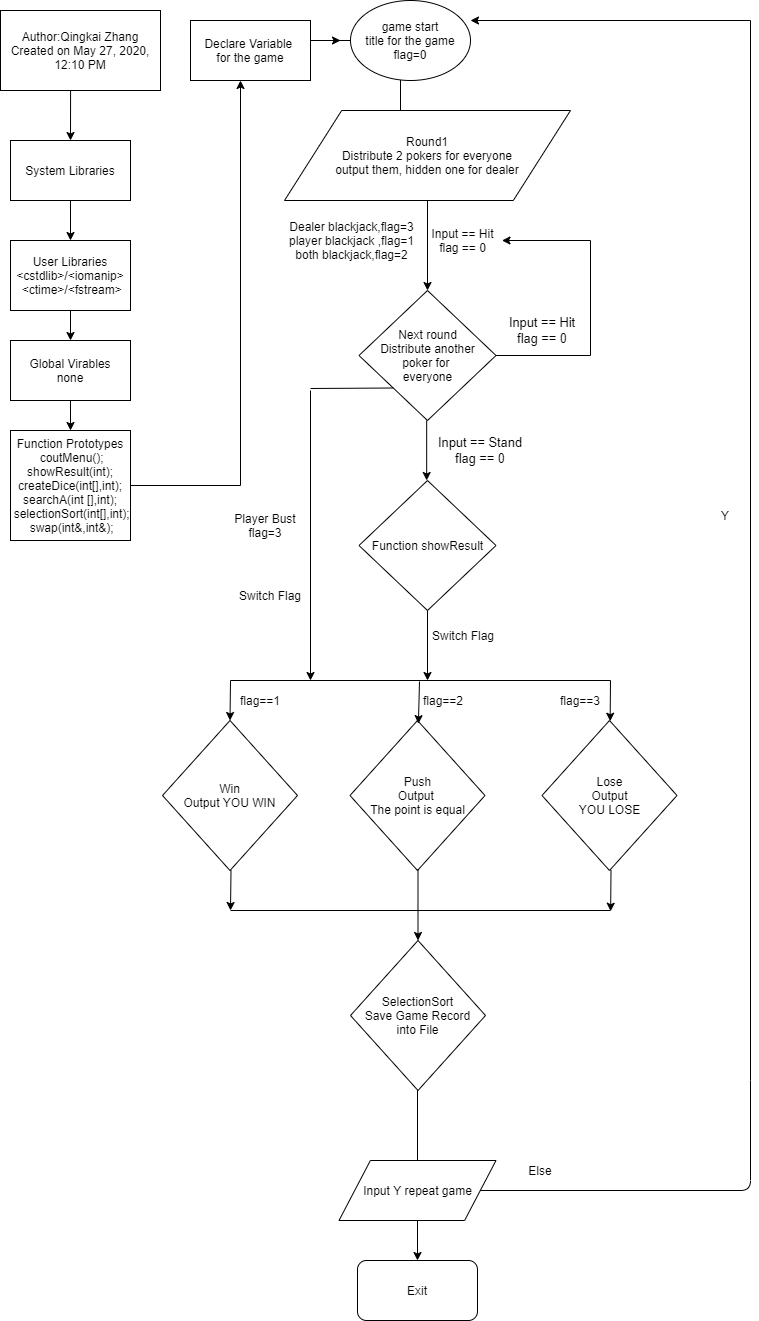
Although this is a very simple game, I think I tried to reflect every concepts we have learned like for loop, if condition, while loop, and searching and sorting fo arrays etc.

# Description

The main point that I programmed this project is the game logic. Although it is a poker game, it needs to check for the result at every round, and the situation is a little bit different from each other. I use flag to refer to the result, and switch flag to output the corresponding results. As a result, there is no need for me to output result every round. And when the game ends, it would record the game result into the file, and sorting the pokers every round for both players and dealers received every round into the sequence from small to the big.

# The Logic of it All

## Flowchart



Cross-List\_From\_Proj1

**Cross Reference from Project 1**

You are to fill-in with where located in code

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Chapter** | **Section** | **Topic** | **Where Line #''s** | **Pts** | **Notes** |
| 2 | 2 | cout | 62 |  |  |
|  | 3 | libraries | 14-18 | 5 | iostream, iomanip, cmath, cstdlib, fstream, string, ctime |
|  | 4 | variables/literals | 35 |  | No variables in global area, failed project! |
|  | 5 | Identifiers |  |  |  |
|  | 6 | Integers | 41 | 1 |  |
|  | 7 | Characters | 45 | 1 |  |
|  | 8 | Strings | 44 | 1 |  |
|  | 9 | Floats No Doubles |  | 1 | Using doubles will fail the project, floats OK! |
|  | 10 | Bools | 35 | 1 |  |
|  | 11 | Sizeof \*\*\*\*\* |  |  |  |
|  | 12 | Variables 7 characters or less | 45 |  | All variables <= 7 characters |
|  | 13 | Scope \*\*\*\*\* No Global Variables |  |  |  |
|  | 14 | Arithmetic operators |  |  |  |
|  | 15 | Comments 20%+ | 49 | 2 | Model as pseudo code |
|  | 16 | Named Constants | 264 |  | All Local, only Conversions/Physics/Math in Global area |
|  | 17 | Programming Style \*\*\*\*\* Emulate |  |  | Emulate style in book/in class repositiory |
|  |  |  |  |  |  |
| 3 | 1 | cin | 101 |  |  |
|  | 2 | Math Expression | 72 |  |  |
|  | 3 | Mixing data types \*\*\*\* |  |  |  |
|  | 4 | Overflow/Underflow \*\*\*\* |  |  |  |
|  | 5 | Type Casting | 181 | 1 |  |
|  | 6 | Multiple assignment \*\*\*\*\* |  |  |  |
|  | 7 | Formatting output | 195 | 1 |  |
|  | 8 | Strings | 130 | 1 |  |
|  | 9 | Math Library | 234 | 1 | All libraries included have to be used |
|  | 10 | Hand tracing \*\*\*\*\*\* |  |  |  |
|  |  |  |  |  |  |
| 4 | 1 | Relational Operators |  |  |  |
|  | 2 | if | 242 | 1 | Independent if |
|  | 4 | If-else | 255 | 1 |  |
|  | 5 | Nesting | 88 | 1 |  |
|  | 6 | If-else-if | 135 | 1 |  |
|  | 7 | Flags \*\*\*\*\* | 148 |  |  |
|  | 8 | Logical operators | 87 | 1 |  |
|  | 11 | Validating user input | 101 | 1 |  |
|  | 13 | Conditional Operator | 102 | 1 |  |
|  | 14 | Switch | 167 | 1 |  |
|  |  |  |  |  |  |
| 5 | 1 | Increment/Decrement | 166 | 1 |  |
|  | 2 | While | 132 | 1 |  |
|  | 5 | Do-while | 98 | 1 |  |
|  | 6 | For loop | 189 | 1 |  |
|  | 11 | Files input/output both | 33 | 2 |  |
|  | 12 | No breaks in loops \*\*\*\*\*\* |  |  | Failed Project if included |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| \*\*\*\*\*\* Not | required to | show | Total | 30 |  |

Cross\_List\_For\_Proj2

**Cross Reference for Project 2**

**You are to fill-in with where located in code**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Chapter** | **Section** | **Topic** | **Where Line #''s** | **Pts** | **Notes** |
| 6 |  | Functions |  |  |  |
|  | 3 | Function Prototypes | 25-30 | 4 | Always use prototypes |
|  | 5 | Pass by Value | 26 | 4 |  |
|  | 8 | return | 243 | 4 | A value from a function |
|  | 9 | returning boolean | 263 | 4 |  |
|  | 10 | Global Variables |  | XXX | Do not use global variables -100 pts |
|  | 11 | static variables | 166 | 4 |  |
|  | 12 | defaulted arguments | 264 | 4 |  |
|  | 13 | pass by reference | 30 | 4 |  |
|  | 14 | overloading | 247 | 5 |  |
|  | 15 | exit() function |  | 4 |  |
| 7 |  | Arrays |  |  |  |
|  | 1 to 6 | Single Dimensioned Arrays | 39 | 3 |  |
|  | 7 | Parallel Arrays | 39-40 | 2 |  |
|  | 8 | Single Dimensioned as Function Arg | **28**  uments | 2 |  |
|  | 9 | 2 Dimensioned Arrays |  | 2 | Emulate style in book/in class repositiory |
|  | 12 | STL Vectors |  | 2 |  |
|  |  | Passing Arrays to and from Function | S 29 | 5 |  |
|  |  | Passing Vectors to and from Functio | Ns | 5 |  |
|  |  |  |  |  |  |
| 8 |  | Searching and Sorting Arrays |  |  |  |
|  | 3 | Bubble Sort |  | 4 |  |
|  | 3 | Selection Sort | 247 | 4 |  |
|  | 1 | Linear or Binary Search | 236 | 4 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| \*\*\*\*\*\* Not r | equired to | show | Total | 70 | Other 30 points from Proj 1 first sheet tab |

# Constructs & Concepts Utilized

## iostream Library

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| cout | 28 | Output Data | Throughout |
| cin | 2 | Input Data | Throughout |

## cstdlib Library

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| srand() | 1 | Random # seed | Line 36 |
| rand() | 1 | Generates rand # | Line 233 |

## iomanip Library

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| setw() | 3 | Format final game stats | Line 190 196 261 |

## string Library

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| string | 1 | Declare var. | Line 44 |

## fstream Library

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| out.open() | 1 | Open file | Line 33 |
| out.close() | 1 | Close file | Line 204 |
| outputFile | 19 | WriteData | Through file |

## Data Types:

|  |  |  |
| --- | --- | --- |
| Data Types | Frequency | Location |
| int | 52 | Through File |
| char | 1 | Line 45 |
| string | 1 | Line 44 |
| ofstream | 1 | Line 32 |

## Conditional Statements:

|  |  |  |
| --- | --- | --- |
| Conditional Statement | Frequency | Starting Location |
| if | 13 | Through the File |
| if/else | 8 | Through the File |
| if/else if | 6 | Through the File |
| switch | 2 | Line 167 221 |

## Loops:

|  |  |  |
| --- | --- | --- |
| Loops | Frequency | Starting Location |
| for | 11 | Through the File |
| while | 2 | Line 37,132 |
| do-while | 1 | Line 163 |

## Linear Search:

|  |  |  |
| --- | --- | --- |
| Function | Frequency | Location |
| searchA | 1 | Line 236 |

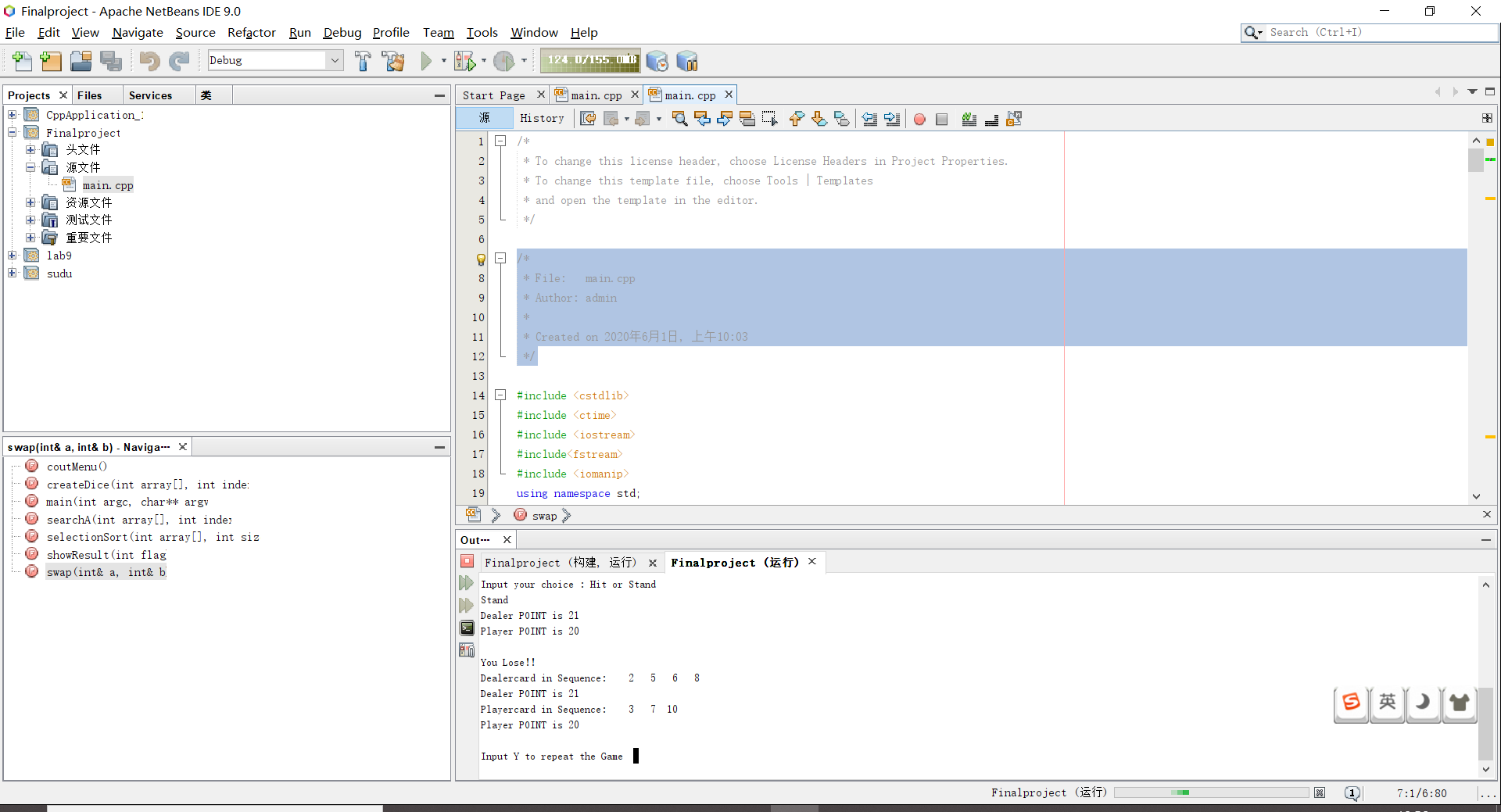
## Sorting:

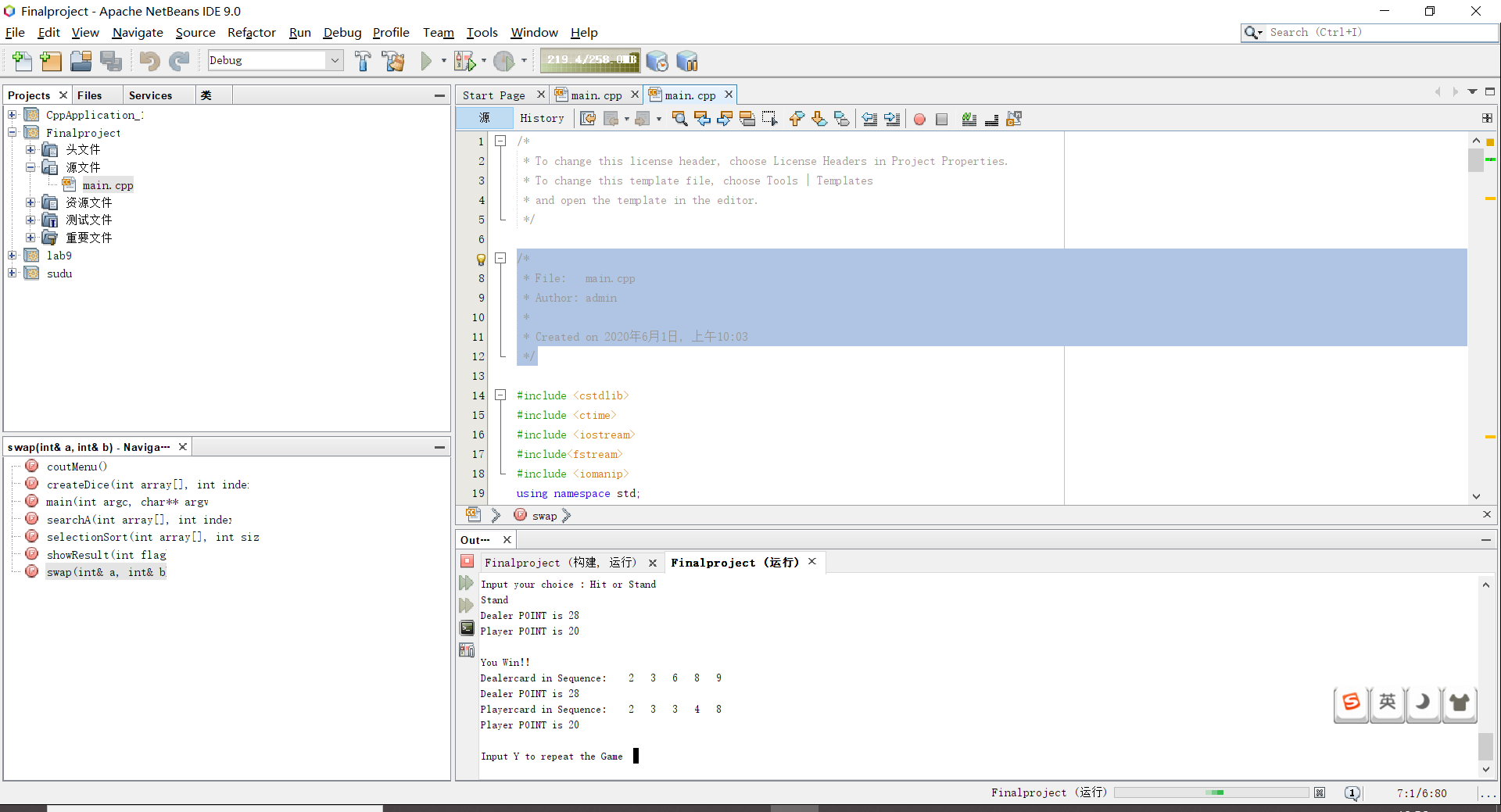
|  |  |  |
| --- | --- | --- |
| Function | Frequency | Location |
| selectionSort | 1 | Line 247 |

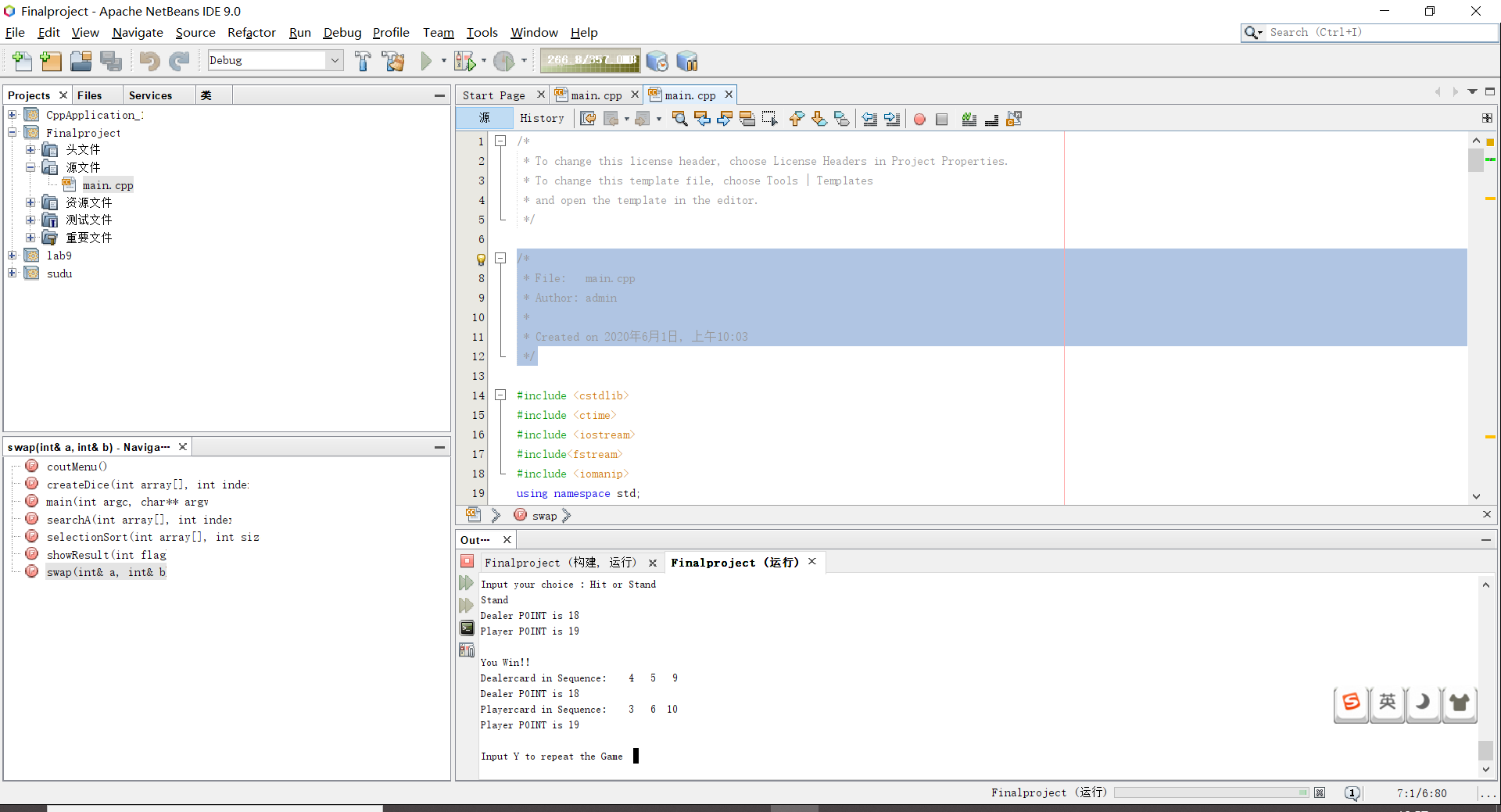
# 

# Proof of a Working Product

Here are some results of my final projects.









# References

1. codeE homework& Lab
2. “Starting Out with C++: From Control Structures through Objects” Gaddis, Tony. 9th Edition. (Textbook)

# Program

/\*

\* File: main.cpp

\* Author: Qingkai

\*

\* Created on 2020年6月1日, 上午10:03

\*/

#include <cstdlib>

#include <ctime>

#include <iostream>

#include<fstream>

#include <iomanip>

using namespace std;

/\*

\*

\*/

//Fuction Prototypes

void coutMenu();

void showResult(int);

void createDice(int[],int);

bool searchA(int [],int);

void selectionSort(int[],int);

void swap(int&,int&);

int main(int argc, char\*\* argv) {

ofstream outputFile;

outputFile.open("GameRecord.txt");

outputFile << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Game Record\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

bool status = true;

int x = 1;

while(status){

srand(unsigned(time(0)));

int Dealercard[10];

int Playercard[10];

int sumD=0;

int sumP=0;

int point;

string input;

char input1;

int flag = 0;

//games status, true for continue, false for finish

status=true;

//set status to true for default

status=true;

//first turn

createDice(Dealercard,1);

createDice(Dealercard,2);

createDice(Playercard,1);

createDice(Playercard,2);

//rolling the dice by getting random number

//round 1

cout<<"Dealer card1 is Dealercard1"<<endl;

cout<<"Dealer card2 is : "<<Dealercard[2]<<endl<<endl;

if(Dealercard[1]==1||Dealercard[2]==1)

sumD=Dealercard[1]+Dealercard[2]+10;

else

sumD=Dealercard[1]+Dealercard[2];

cout<<"Player card1 is "<<Playercard[1]<<endl;

cout<<"Player card2 is "<<Playercard[2]<<endl;

if(Playercard[1]==1||Playercard[2]==1)

sumP= Playercard[1]+ Playercard[2]+10;

else

sumP = Playercard[1]+ Playercard[2];

cout<<"Player POINT is "<<sumP<<endl<<endl;

if(Dealercard[1]==10&&Dealercard[2]==1){

if(Playercard[1]==1&&Playercard[2]==10)

flag=2;//push

else if(Playercard[2]==1&&Playercard[1]==10)

flag=2;//push

else

flag=3;//lose Dealer BlackJack

}

if((Playercard[2]==1&&Playercard[1]==10)||(Playercard[1]==1&&Playercard[2]==10)){

if(Dealercard[1]==1&&Dealercard[2]==10)

flag=2;//push

else if(Dealercard[2]==1&&Dealercard[1]==10)

flag=2;//push

else

flag=1;//Win player BlackJack

}

int i =3;

int j;

do

{

cout<<"Input your choice : Hit or Stand"<<endl;

cin>>input;

if(input=="Hit"){

createDice(Dealercard,i);

createDice(Playercard,i);

if(sumD<11 && Dealercard[i]==1)

sumD+=Dealercard[i]+10;

else{

sumD+=Dealercard[i];

}

if( searchA(Dealercard,i-1) && sumD >21)

sumD-=10;

if(sumP<11 && Playercard[i]==1)

sumP+=Playercard[i]+10;

else

sumP+=Playercard[i];

if( searchA(Playercard,i-1) && sumP >21)

sumP-=10;

cout<<"Dealer card"<<i<<" is : "<<Dealercard[i]<<endl<<endl;

cout<<"Player card"<<i<<" is "<<Playercard[i]<<endl;

cout<<"Player POINT is "<<sumP<<endl<<endl;

i++;

if(sumP>21 )

flag = 3; //lose

}

if(input=="Stand"){

j = i;

while(sumD<=17){//Dealer would keep Hit until the point is over 17

createDice(Dealercard,j);

if(sumD<11 && (searchA(Dealercard,j)))

sumD+=Dealercard[j]+10;

else

sumD+=Dealercard[j];

j++;

if(sumD>21)

flag=1;

}

cout<<"Dealer POINT is "<<sumD<<endl;

cout<<"Player POINT is "<<sumP<<endl<<endl;

if(sumD<=21 && sumP<=21){

if(sumD > sumP){

flag = 3;//lose

}

else if(sumD == sumP){

flag = 2;//push

}

else if(sumD < sumP){

flag = 1;//Win

}

}

else if(sumP>21)

flag = 3;//lose

else if(sumD>21 && sumP<=21)

flag = 1;//win

}

}while(flag==0);

outputFile << "Round "<<x<<" :"<<endl;

x++;

switch(flag){

case 1:

outputFile <<"You Win!!"<<endl;

break;

case 2:

outputFile <<"The Point is Equal!!"<<endl;

break;

case 3:

outputFile <<"You Lose!!"<<endl;

break;

}

showResult(flag);

cout << "Dealercard in Sequence: ";

selectionSort(Dealercard,j-1);

cout<<"Dealer POINT is "<<sumD<<endl;

cout << "Playercard in Sequence: ";

selectionSort(Playercard,i-1);

cout<<"Player POINT is "<<sumP<<endl<<endl;

outputFile<<"Dealer POINT is "<<sumD<<endl;

outputFile << "Dealercard in Sequence: ";

for(int y =1; y <=j-1; y++)

outputFile << setw(4) << Dealercard[y];

outputFile << endl;

outputFile<<"Player POINT is "<<sumP<<endl;

outputFile << "Playercard in Sequence: ";

for(int y =1; y <=i-1; y++)

outputFile << setw(4) << Playercard[y];

outputFile << endl<<endl;

cout<<"Input Y to repeat the Game";

cin>>input1;

cout<<endl;

if(input1!='Y')

status = false;

}

outputFile.close();

return 0;

}

void coutMenu(){

for(int i=0;i<15;i++)

{

cout<<"\*";

}

cout<<" Game Start ";

for(int i=0;i<15;i++)

{

cout<<"\*";

}

cout<<endl<<endl;

}

void showResult(int flag){

switch(flag){

case 1: cout<<"You Win!!"<<endl;

// outputFile <<"You Win!!"<<endl;

break;

case 2: cout<<"The Point is Equal!!"<<endl;

// outputFile <<"The Point is Equal!!"<<endl;

break;

case 3: cout<<"You Lose!!"<<endl;

// outputFile <<"You Lose!!"<<endl;

break;

}

}

void createDice(int array[],int index){

array[index] =(rand()%(10))+1;

}

bool searchA(int array[],int index){

bool status = false;

for(int i = 0;i<index;i++){

if(array[i]==1)

status = true;

}

if(status)

return true;

else

return false;

}

void selectionSort(int array[],int size){

int minIndex,minValue;

for(int start = 1; start <(size); start++){

minIndex = start;

minValue = array[start];

for(int index = start+1;index<=size;index++){

if(array[index]<minValue){

minValue = array[index];

minIndex = index;

}

}

swap(array[minIndex],array[start]);

}

for(int i =1; i <=size; i++)

cout << setw(4) << array[i];

cout << endl;

}

void swap(int &a, int &b){

int temp =a;

a = b;

b= temp;

}