**Project 1**

**<Dice Rolling Game>**

**CIS-17A**

**Name: Yuantian Zhou**

**Date: 10/17/19**

**Introduction**

Title: Dice Rolling Game

You roll two dice. Each die has six faces, which contain one,two, three, four,rive and six spots, respectively. After the dice have come to rest, the sum of the spots on the two upward faces is calculated. If the sum is 7 or 11 on the first throw, you win. If the sum is 2 or 12 on the first throw (called “craps”), you lose. If the sum is 3,4,5,6,7,8,9 or 10 on the first throw, that sum becomes your “point.” To win, you must continue rolling the dice until you “make your point”(roll that same point value). You lose by rolling a 7 before making your point.

**Summary**

Project size: 350+ lines

The number of variables: about 20 The number of function: 27

It took around two weeks.

It’s little challenging since it converts pointers and file stream which are new for me.

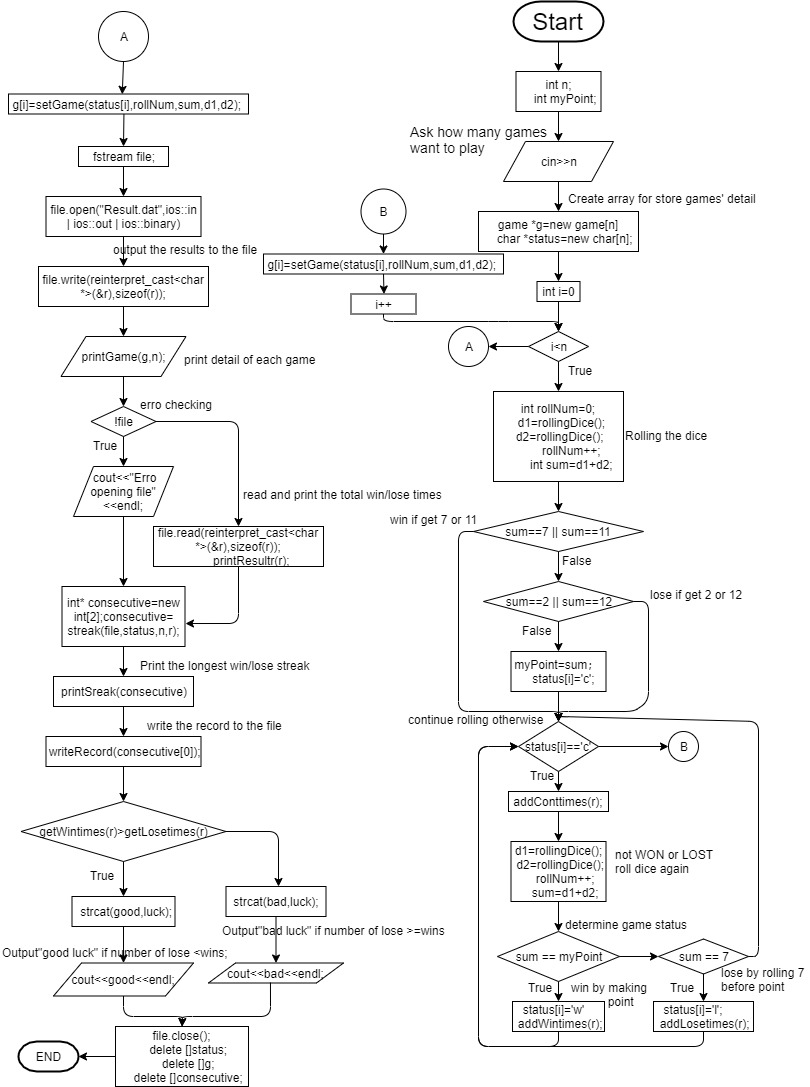
**Description**

**Sample Input/Output**

Input the number of games that you want to play. Then the program will output each game’s detail(ie.Points of each dices, sum of two dices, win/lose, number of times rolling dices.). And output the total win and lose times. Also the longest win.lose streak. And it will tell you if you break the highest streak record. The result and the longest win steak of the games will be output to files. It will congratulate you (“good luck”); If your win times is more than lose times. Otherwise it will call you (“bad luck”).

**FlowChart：**

**Pseudo Code：**



**Major Variables**

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Name | Description | Location |
| int | n | Number of games time | Main() |
|  | d1 | Points of dice 1 | Main() |
|  | d2 | Points of dice 2 | Main() |
|  | sum | Sum of two dice | Main() |
|  | myPoint | Points get from first roll | Main() |
|  | win | Store number of win | rollingResult{} |
|  | lose | Store number of lose | rollingResult{} |
|  | cont | Store number of continues | rollingResult{} |
|  | diceRollingtimes | Store time of rolling dice | game{} |
|  | sum | Sum of two dice | game{} |
|  | Dice1 | Points of dice 1 | game{} |
|  | Dice2 | Points of dice 2 | game{} |
| char[] | good | String for output | Main() |
|  | bad | String for output | Main() |
|  | luck | String for output | Main() |
| char\* | status | Store the status of game | Main() |
|  | narray | Store the status (only contain win/lose) | Streak() |
| game\* | g | Detailes for each game | Main() |
|  |  |  |  |
| rollingResult | r | Result from all games | Main() |
|  |  |  |  |
| fstream | file | File for game results | Main() |
|  |  |  |  |
|  | file | File for longest win streak record | writeRecord() |
| string | gameResult | Store game result | game{} |
|  |  |  |  |
| enum | DiceNum | The point of dices |  |
|  |  |  |  |

**Reference**

1. textbook

**Program**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*

\* File: main.cpp

\* Author: raytim

\*

\* Created on 2019å¹´10æœˆ10æ—¥, ä¸‹å�ˆ2:27

\*/

#include <cstdlib>

#include <iostream>

#include <ctime>

#include <fstream>

#include <cstring>

using namespace std;

//six faces for dice

enum DiceNum {ONE,TWO,THREE,FOUR,FIVE,SIX};

struct rollingResult

{

int win;//number of win times

int lose;//number of lose times

int cont;//number of continues times

};

struct game

{

string gameResult; //win or lose

int diceRolltimes; //number of times of rolling dice

int sum;//sum of two dices

int dice1;

int dice2;

rollingResult r;

};

/\*

\*

\*/

int rollingDice();

void setResult(game& ,char );

void setDiceRollTimes(game& ,int);

void setSum(game& ,int );

void setDice1(game& ,int );

void setDice2(game& ,int );

game setGame(char ,int ,int,int ,int);

void printResult(game );

void printSum(game );

void printDice1(game );

void printDice2(game );

void printDiceRollTime(game );

void printGame(game );

void resetResult(rollingResult&);

void addWintimes(rollingResult& );

void addLosetimes(rollingResult& );

void addConttimes(rollingResult& );

int getWintimes(rollingResult&);

int getLosetimes(rollingResult&);

void printWintimes(rollingResult );

void printLosetimes(rollingResult );

void printConttimes(rollingResult );

void printResultr(rollingResult );

void printGame(game \*,int );

int streak(char\*,char,int);

int\* streak(fstream&,char\*,int,rollingResult&);

void writeRecord(int);

void printStreak(int\*);

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

srand(unsigned(time(0)));

int n;//number of games

int myPoint;

char good[]={"good "};

char bad[]={"bad "};

char luck[]={"luck"};

rollingResult r;

resetResult(r);

cout<<"Input the number of games you want to play"<<endl;

cin>>n;

game \*g=new game[n];

//for store the result of each game

char \*status=new char[n];//w for win, l for lose

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

{

resetResult(g[i].r);

int rollNum=0;//dice rolling times

int d1=rollingDice();

int d2=rollingDice();

rollNum++;

int sum=d1+d2;

//win if get 7 or 11

if(sum==7 || sum==11)

{

status[i]='w';

addWintimes(r);

}

//lose if get 2 or 12

else if(sum==2||sum==12)

{

status[i]='l';

addLosetimes(r);

}

else

{//continue rolling otherwise

myPoint=sum;

status[i]='c';

}

while (status[i]=='c')

{

addConttimes(r);

// not WON or LOST

// roll dice again

d1=rollingDice();

d2=rollingDice();

rollNum++;

sum=d1+d2;

// determine game status

if (sum == myPoint)

{ // win by making point

status[i]='w';

addWintimes(r);

}

else

{

if (sum == 7)

{

// lose by rolling 7 before point

status[i] = 'l';

addLosetimes(r);

}

}

}

g[i]=setGame(status[i],rollNum,sum,d1,d2);

}

fstream file;

file.open("Result.dat",ios::in | ios::out | ios::binary);

//output the results to the file

file.write(reinterpret\_cast<char \*>(&r),sizeof(r));

//print detail of each game

printGame(g,n);

if(!file)//erro checking

{

cout<<"Erro opening file"<<endl;

}

else

{//read and print the total win/lose times

file.read(reinterpret\_cast<char \*>(&r),sizeof(r));

printResultr(r);

}

int\* consecutive=new int[2];//for store the consecutive win/lose

//find the longest win/lose streak

consecutive=streak(file,status,n,r);

printStreak(consecutive);

//write the win streak to file if its longer than history

writeRecord(consecutive[0]);

if(getWintimes(r)>getLosetimes(r))

{//"good luck" for wins more than loses

strcat(good,luck);

cout<<good<<endl;

}

else

{ //otherwise "bad luck"

strcat(bad,luck);

cout<<bad<<endl;

}

file.close();

delete []status;

delete []g;

delete []consecutive;

return 0;

}

int rollingDice()

{

return (rand()%(SIX+1))+1;

}

void setResult(game& g,char result)

{

if(result=='w')

{

g.gameResult="win";

}

else

{

g.gameResult="lose";

}

}

void setDiceRollTimes(game& g,int diceRolltimes)

{

g.diceRolltimes=diceRolltimes;

}

void setSum(game& g,int sum)

{

g.sum=sum;

}

void setDice1(game& g,int dice1)

{

g.dice1=dice1;

}

void setDice2(game& g,int dice2)

{

g.dice2=dice2;

}

game setGame(char result,int diceRolltimes,int sum,int dice1,int dice2)

{

game g;

setResult(g,result);

setDiceRollTimes(g,diceRolltimes);

setSum(g,sum);

setDice1(g,dice1);

setDice2(g,dice2);

return g;

}

void printResult(game g)

{

cout<<"The game result is: "<<g.gameResult<<endl;

}

void printSum(game g)

{

cout<<"The sum of two dice is: "<<g.sum<<endl;

}

void printDice1(game g)

{

cout<<"The dice 1 is: "<<g.dice1<<endl;

}

void printDice2(game g)

{

cout<<"The dice 2 is: "<<g.dice2<<endl;

}

void printDiceRollTime(game g)

{

cout<<"The dice rolling times are: "<<g.diceRolltimes<<endl;

}

void printGame(game g)

{

printResult(g);

printDice1(g);

printDice2(g);

printSum(g);

printDiceRollTime(g);

}

int\* streak(fstream& file,char\* array,int size,rollingResult&r)

{

file.seekg(0,ios::beg);

file.read(reinterpret\_cast<char\*>(&r),sizeof(r));

int nsize=r.win+r.lose;

char\*narray=new char[nsize];

int j=0;

//copy to a new array that only contains win and lose

cout<<endl;

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

{

if(array[i]=='w'||array[i]=='l')

{

narray[j]=array[i];

j++;

}

}

int \*consecu=new int[2];//to store the number of consecutive win/lose

consecu[0]=streak(narray,'w',nsize);//consecutive wins

consecu[1]=streak(narray,'l',nsize);//consecutive loses

delete []narray;

return consecu;

}

int streak(char \*array,char a,int size)

{

int cons=0;//current consecutive win/lose

int max=0;//max consecutive win/lose

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

{//count for consecutive win

if(array[i]==a)

{

while(array[i]==array[i+1])

{

cons++;

i++;

}

if(cons>max)

{

max=cons;

}

cons=0;

}

}

return max+1;

}

void addWintimes(rollingResult& r)

{

r.win++;

}

void addLosetimes(rollingResult& r)

{

r.lose++;

}

void addConttimes(rollingResult& r)

{

r.cont++;

}

int getWintimes(rollingResult& r)

{

return r.win;

}

int getLosetimes(rollingResult& r)

{

return r.lose;

}

void resetResult(rollingResult& r)

{

r.win=0;

r.lose=0;

r.cont=0;

}

void printWintimes(rollingResult r)

{

cout<<"Number of win times are: "<<r.win<<endl;

}

void printLosetimes(rollingResult r)

{

cout<<"Number of lose times are: "<<r.lose<<endl;

}

void printConttimes(rollingResult r)

{

cout<<"Number of continues times are: "<<r.cont<<endl;

}

void printResultr(rollingResult r)

{

printWintimes(r);

printLosetimes(r);

}

void printGame(game \*g,int n)

{

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

{

cout<<"The round "<<i+1<<": "<<endl;

printGame(g[i]);

cout<<endl;

}

}

//Recored the winnig streak to the file if it's longer than the history

void writeRecord(int winStreak)

{

fstream file("LongestWinStreak.txt",ios::in);

int oldRecord;

int oldMax=0;

while(file>>oldRecord)

{//find the higest record of history

if(oldRecord>oldMax)

{

oldMax=oldRecord;

}

}

file.close();

file.open("LongestWinStreak.txt",ios::out | ios\_base::app);

if(winStreak>oldMax)

{//write the new record at the end of the file

file.seekp(0,ios::end);

file<<"\r\n"<<winStreak;

cout<<"Great! You break the record of longest win streak"<<endl;

}

file.close();

}

void printStreak(int \*consecutive)

{

cout<<"The longest winnig streak is: "<<consecutive[0]<<endl;

cout<<"The longest loseing streak is: "<<consecutive[1]<<endl;

}