**Project 1**

**<Dice Rolling Game>**

**CIS-17A**

**Name: Yuantian Zhou**

**Date: 11/23/19**

**Introduction**

Title: Dice Rolling Game

You roll four dice. Each dice 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 four upward faces is calculated. If the sum is 14 or 23 on the first throw, you win. If the sum is 4 or 24 on the first throw (called “craps”), you lose. If the sum is other than 14,23,4,24 in 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 14 before making your point.

**Summary**

Project size: 1060+ lines

The number of variables: about 25 The number of function: 70

It took around two weeks.

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

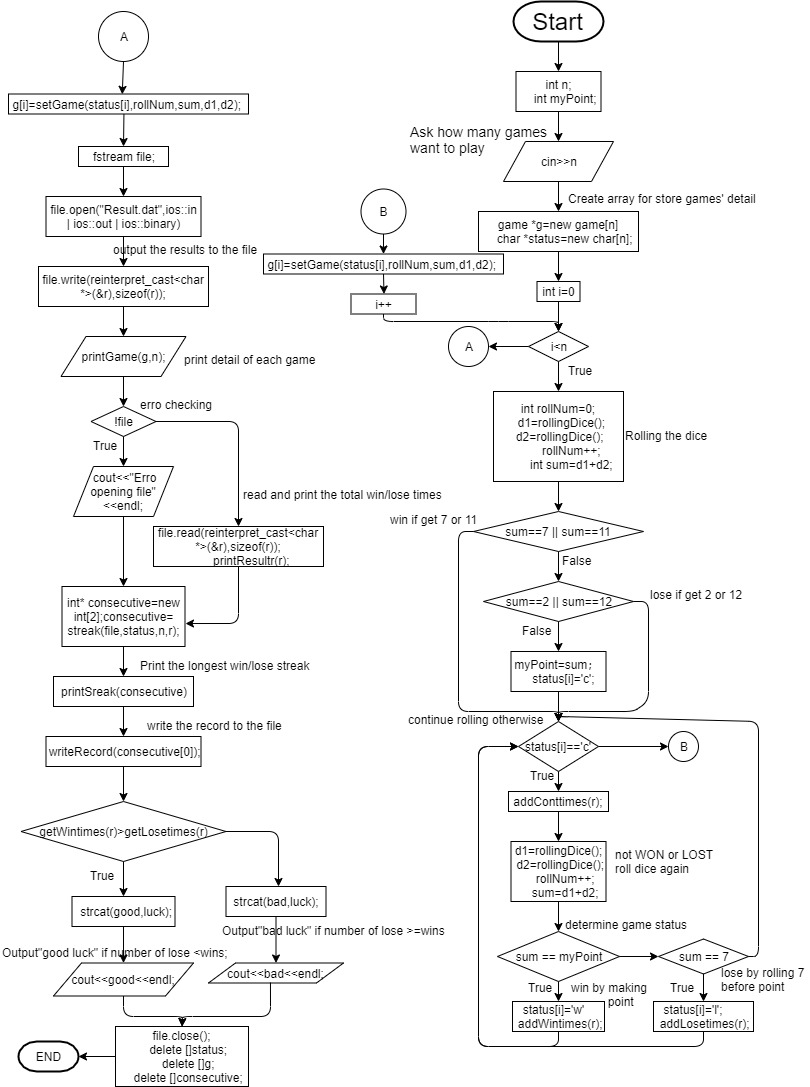
**Description**

**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 four 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() |
|  | d3 | Points of dice 3 | Main() |
|  | d4 | Points of dice 4 | 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{} |
|  | gameCount | Counter of the game | Game{} |
|  | Dice1 | Points of dice 1 | Game{} |
|  | Dice2 | Points of dice 2 | Game{} |
|  | Dice3 | Points of dice 3 | ThreeDiceGane{} |
|  | Dice4 | Points of dice 4 | FourDiceGame{} |
| 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) | Main() |
| 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 | Main() |
| string | gameResult | Store game result | Game{} |
|  |  |  |  |
| enum | DiceNum | The point of dices |  |
| vector | resultList | List of game results | Main() |
| RollingResult | r | Store the rolling result | Game{} |
|  |  |  |  |

**Reference**

1. textbook

**Program**

**main.cpp:**

/\*

\* 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 2019 2:27

\*/

#include <cstdlib>

#include <iostream>

#include <ctime>

#include <fstream>

#include <cstring>

#include <vector>

using namespace std;

#include "RollingResult.h"

#include "Game.h"

#include "ThreeDiceGame.h"

#include "FourDiceGame.h"

//six faces for dice

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

/\*

\*

\*/

int rollingDice();

template <class T>

Game setGame(char ,T ,T,T ,T);

template <class T>

ThreeDiceGame setGame(char ,T ,T,T ,T,T);

template <class T>

FourDiceGame setGame(char ,T ,T,T ,T,T,T);

void printGame(Game \*,int);

void printGame(ThreeDiceGame \*,int);

void printGame(FourDiceGame \*,int);

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

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

int twoDiceSum(int,int);

int threeDiceSum(int,int,int);

int fourDiceSum(int,int,int,int);

void writeRecord(int);

void printStreak(int\*);

void checkDice(int);

void checkInput(int);

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

srand(unsigned(time(0)));

int n;//number of games

//points get from the first time of rolling

int myPoint;

char good[]={"good "};

char bad[]={"bad "};

char luck[]={"luck"};

RollingResult r;

r.resetResult();

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

cin>>n;

//vector of char for store a list of all games result

vector <string>resultList;

try

{

//check if is valid input

checkInput(n);

}

catch(string exception)

{

//output the exception if is invalid

cout<<exception<<endl;

return 0;

}

FourDiceGame \*g=new FourDiceGame[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++)

{

//set the dice rolling times

int rollNum=0;

//roll the dices

int d1=rollingDice();

int d2=rollingDice();

int d3=rollingDice();

int d4=rollingDice();

rollNum++;

int sum=fourDiceSum(d1,d2,d3,d4);

//win if get 14 or 23

if(sum==14 || sum==23)

{

//add 'win' to the result list

resultList.push\_back("win");

status[i]='w';

try

{

r.addWinTimes();

}

catch(string exception)

{

cout<<exception<<endl;

return 0;

}

}

//lose if get 4 or 24

else if(sum==4||sum==24)

{

//add 'lose' to the result list

resultList.push\_back("lose");

status[i]='l';

try

{

r.addLoseTimes();

}

catch(string exception)

{

cout<<exception<<endl;

return 0;

}

}

else

{

//continue rolling otherwise

myPoint=sum;

status[i]='c';

}

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

{

try

{

r.addContTimes();

}

catch(string exception)

{

cout<<exception<<endl;

return 0;

}

// not WON or LOST

// roll dice again

d1=rollingDice();

d2=rollingDice();

d3=rollingDice();

d4=rollingDice();

rollNum++;

sum=d1+d2+d3+d4;

// determine game status

if (sum == myPoint)

{

// win by making point

//add 'win' to the result list

resultList.push\_back("win");

status[i]='w';

try

{

r.addWinTimes();

}

catch(string exception)

{

cout<<exception<<endl;

return 0;

}

}

else

{

if (sum == 14)

{

// lose by rolling 14 before point

status[i] = 'l';

//add 'lose' to the result list

resultList.push\_back("lose");

try

{

r.addLoseTimes();

}

catch(string exception)

{

cout<<exception<<endl;

return 0;

}

}

}

}

g[i]=setGame(status[i],rollNum,sum,d1,d2,d3,d4);

}

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

try

{

printGame(g,n);

}

catch(string exception)

{

cout<<exception<<endl;

return 0;

}

try

{

if(!file)//erro checking

{

throw "erro opening file";

}

else

{

//read and print the total win/lose times

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

r.printResult();

}

}

catch(string exception)

{

cout<<exception<<endl;

}

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(r.getWinTimes()>r.getLoseTimes())

{

//"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;

}

template <class T>

Game setGame(char result,T diceRolltimes,T sum,T dice1,T dice2)

{

Game g;

g.setResult(result);

g.setDiceRollingTimes(diceRolltimes);

g.setSum(sum);

g.setDice1(dice1);

g.setDice2(dice2);

return g;

}

template <class T>

ThreeDiceGame setGame(char result,T diceRolltimes,T sum,T dice1,T dice2,T dice3)

{

ThreeDiceGame g;

g.setResult(result);

g.setDiceRollingTimes(diceRolltimes);

g.setSum(sum);

g.setDice1(dice1);

g.setDice2(dice2);

g.setDice3(dice3);

return g;

}

template <class T>

FourDiceGame setGame(char result,T diceRolltimes,T sum,T dice1,T dice2,T dice3,T dice4)

{

FourDiceGame g;

g.setResult(result);

g.setDiceRollingTimes(diceRolltimes);

g.setSum(sum);

g.setDice1(dice1);

g.setDice2(dice2);

g.setDice3(dice3);

g.setDice4(dice4);

return 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.getWinTimes()+r.getLoseTimes();

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;

}

void printGame(Game \*g,int n)

{

if(n<0)

{

throw "erro: invalid number of games";

}

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

{

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

g[i].printGame();

cout<<endl;

}

}

void printGame(ThreeDiceGame \*g,int n)

{

if(n<0)

{

throw "erro: invalid number of games";

}

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

{

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

g[i].printGame();

cout<<endl;

}

}

void printGame(FourDiceGame \*g,int n)

{

if(n<0)

{

throw "erro: invalid number of games";

}

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

{

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

cout<<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);

winStreak++;

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)

{

if(consecutive[0]>0)

{

cout<<"The longest winnig streak is: "<<consecutive[0]+1<<endl;

}

if(consecutive[1]>0)

{

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

}

if(consecutive[0]==0)

{

cout<<"No winnig streak "<<endl;

}

if(consecutive[1]==0)

{

cout<<"No loseing streak "<<endl;

}

}

ostream &operator<<(ostream&strm,const RollingResult&right)

{

strm<<"Number of win times are: "<<right.win<<endl;

strm<<"Number of lose times are: "<<right.lose<<endl;

strm<<"Number of cont times are: "<<right.cont<<endl;

return strm;

}

istream &operator>>(istream&strm,RollingResult&right)

{

cout<<"input the win times:"<<endl;

strm>>right.win;

cout<<"input the lose times:"<<endl;

strm>>right.lose;

cout<<"input the continue times:"<<endl;

strm>>right.cont;

return strm;

}

ostream &operator<<(ostream&strm,const Game&right)

{

strm<<"The game result is: "<<right.gameResult<<endl;

strm<<"The dice1 is: "<<right.dice1<<endl;

strm<<"The dice2 is: "<<right.dice2<<endl;

strm<<"The sum of two dice is: "<<right.sum<<endl;

strm<<"The dice rolling times are: "<<right.diceRolltimes<<endl;

return strm;

}

istream &operator>>(istream&strm,Game&right)

{

cout<<"input the dice 1: "<<endl;

strm>>right.dice1;

cout<<"input the dice 2: "<<endl;

strm>>right.dice2;

return strm;

}

ostream &operator<<(ostream&strm,const ThreeDiceGame&right)

{

strm<<"The game result is: "<<right.gameResult<<endl;

strm<<"The dice 1 is: "<<right.dice1<<endl;

strm<<"The dice 2 is: "<<right.dice2<<endl;

strm<<"The dice 3 is: "<<right.dice3<<endl;

strm<<"The sum of three dice is: "<<right.sum<<endl;

strm<<"The dice rolling times are: "<<right.diceRolltimes<<endl;

return strm;

}

istream &operator>>(istream&strm,ThreeDiceGame&right)

{

int dice1,dice2,dice3;

cout<<"input the dice 1: "<<endl;

strm>>dice1;

cout<<"input the dice 2: "<<endl;

strm>>dice2;

cout<<"input the dice 3: "<<endl;

strm>>dice3;

right.setDice1(dice1);

right.setDice2(dice2);

right.setDice3(dice3);

return strm;

}

ostream &operator<<(ostream&strm,const FourDiceGame&right)

{

strm<<"The game result is: "<<right.gameResult<<endl;

strm<<"The dice 1 is: "<<right.dice1<<endl;

strm<<"The dice 2 is: "<<right.dice2<<endl;

strm<<"The dice 3 is: "<<right.dice3<<endl;

strm<<"The dice 4 is: "<<right.dice4<<endl;

strm<<"The sum of four dice is: "<<right.sum<<endl;

strm<<"The dice rolling times are: "<<right.diceRolltimes<<endl;

return strm;

}

istream &operator>>(istream&strm,FourDiceGame&right)

{

int dice1,dice2;

cout<<"input the dice 1: "<<endl;

strm>>dice1;

cout<<"input the dice 2: "<<endl;

strm>>dice2;

cout<<"input the dice 3: "<<endl;

strm>>right.dice3;

cout<<"input the dice 4: "<<endl;

strm>>right.dice4;

right.setDice1(dice1);

right.setDice2(dice2);

return strm;

}

void checkInput(int input)

{

if(input<=0)

{

throw "invalid input";

}

}

void checkDice(int dice)

{

if(dice<=0)

{

throw "invalid dice";

}

}

int twoDiceSum(int dice1,int dice2)

{

try

{

checkDice(dice1);

checkDice(dice2);

}

catch(string exception)

{

cout<<exception<<endl;

}

return dice1+dice2;

}

int threeDiceSum(int dice1,int dice2,int dice3)

{

try

{

checkDice(dice1);

checkDice(dice2);

checkDice(dice3);

}

catch(string exception)

{

cout<<exception<<endl;

}

return dice1+dice2+dice3;

}

int fourDiceSum(int dice1,int dice2,int dice3,int dice4)

{

try

{

checkDice(dice1);

checkDice(dice2);

checkDice(dice3);

checkDice(dice4);

}

catch(string exception)

{

cout<<exception<<endl;

}

return dice1+dice2+dice3+dice4;

}

**RollingResult.h:**

/\*

\* 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.

\*/

#ifndef ROLLINGRESULT\_H

#define ROLLINGRESULT\_H

#include <iostream>

using namespace std;

class RollingResult

{

private:

int win;//number of win times

int lose;//number of lose times

int cont;//number of continues times

public:

RollingResult();

virtual ~RollingResult();

template <class T>

RollingResult(T, T, T);

template <class T>

void setWinTimes(T);

template <class T>

void setLoseTimes(T);

template <class T>

void setContTimes(T);

int getWinTimes();

int getLoseTimes();

int getContTimes();

void resetResult();

void addWinTimes();

void addLoseTimes();

void addContTimes();

void printWinTimes();

void printLoseTimes();

void printContTimes();

void printResult();

const RollingResult operator=(const RollingResult&);

friend ostream &operator<<(ostream&strm, const RollingResult&);

friend istream &operator>>(istream&strm, RollingResult&);

};

template <class T>

RollingResult::RollingResult(T win, T lose, T cont)

{

this->win = win;

this->lose = lose;

this->cont = cont;

}

#endif /\* ROLLINGRESULT\_H \*/

**RollingResult.cpp:**

/\*

\* 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.

\*/

#include "RollingResult.h"

RollingResult::RollingResult()

{

win = 0;

lose = 0;

cont = 0;

}

RollingResult::~RollingResult()

{

}

const RollingResult RollingResult::operator=(const RollingResult&right)

{

if (this != &right)

{

win = right.win;

lose = right.lose;

cont = right.cont;

}

return \*this;

}

void RollingResult::addWinTimes()

{

if (win < 0)

{

throw "error: win less than 0";

}

else

{

win++;

}

}

void RollingResult::addLoseTimes()

{

if (lose < 0)

{

throw "error: lose less than 0";

}

else

{

lose++;

}

}

void RollingResult::addContTimes()

{

if (cont < 0)

{

throw "error: cont less than 0";

}

else

{

cont++;

}

}

int RollingResult::getWinTimes()

{

return win;

}

int RollingResult::getLoseTimes()

{

return lose;

}

void RollingResult::resetResult()

{

win = 0;

lose = 0;

cont = 0;

}

void RollingResult::printWinTimes()

{

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

}

void RollingResult::printLoseTimes()

{

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

}

void RollingResult::printContTimes()

{

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

}

void RollingResult::printResult()

{

printWinTimes();

printLoseTimes();

}

**Game.h:**

/\*

\* 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: Game.h

\* Author: raytim

\*

\* Created on 2019年11月22日, 下午11:31

\*/

#ifndef GAME\_H

#define GAME\_H

#include "RollingResult.h"

#include <iostream>

using namespace std;

class Game

{

protected:

static int gameCount; //count the number of 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;

public:

Game();

template <class T>

Game(string, T, T, T, T, RollingResult);

void setResult(char);

template <class T>

void setDiceRollingTimes(T diceRolltimes)

{

this->diceRolltimes = diceRolltimes;

}

template <class T>

void setSum(T sum)

{

this->sum = sum;

}

template <class T>

void setDice1(T dice1)

{

this->dice1 = dice1;

}

template <class T>

void setDice2(T dice2)

{

this->dice2 = dice2;

}

template <class T>

void setGame(char, T, T, T, T);

char getResult();

RollingResult getRollingResult();

int getDiceRollTimes();

int getSum();

int getDice1();

int getDice2();

void printResult();

void printSum();

void printDice1();

void printDice2();

void printGame();

void printDiceRollTimes();

int getGameCount();

Game(const Game&right)

{

diceRolltimes = right.diceRolltimes;

sum = right.sum;

dice1 = right.dice1;

dice2 = right.dice2;

gameResult = right.gameResult;

r = right.r;

}

friend ostream &operator<<(ostream&strm, const Game&);

friend istream &operator>>(istream&strm, Game&);

};

template <class T>

Game::Game(string result, T diceRolltimes, T sum, T dice1, T dice2, RollingResult r)

{

this->diceRolltimes = diceRolltimes;

this->sum = sum;

this->dice1 = dice1;

this->dice2 = dice2;

this->r = r;

}

#endif /\* GAME\_H \*/

**Game.cpp:**

/\*

\* 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.

\*/

#include "Game.h"

int Game::gameCount = 0;

Game::Game()

{

gameCount++;

dice1 = 0;

dice2 = 0;

diceRolltimes = 0;

sum = 0;

r.resetResult();

}

void Game::setResult(char result)

{

if (result == 'w')

{

gameResult = "win";

}

else

{

gameResult = "lose";

}

}

void Game::printResult()

{

cout << "The game result is: " << gameResult.c\_str() << endl;

}

void Game::printSum()

{

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

}

void Game::printDice1()

{

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

}

void Game::printDice2()

{

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

}

void Game::printDiceRollTimes()

{

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

}

void Game::printGame()

{

printResult();

printDice1();

printDice2();

printSum();

printDiceRollTimes();

}

int Game::getGameCount()

{

return gameCount;

}

**ThreeDiceGame.h:**

/\*

\* 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: ThreeDiceGame.h

\* Author: raytim

\*

\* Created on 2019年11月22日, 下午11:31

\*/

#ifndef THREEDICEGAME\_H

#define THREEDICEGAME\_H

#include "Game.h"

class ThreeDiceGame :public Game

{

protected:

int dice3;//one dice more than two

public:

ThreeDiceGame();

template <class T>

void setDice3(T);

int getDice3();

void printDice3();

void printGame();

void printSum();

template <class T>

ThreeDiceGame setGame(char, T, T, T, T, T);

const ThreeDiceGame operator=(const ThreeDiceGame&);

friend ostream &operator<<(ostream&strm, const ThreeDiceGame&);

friend istream &operator>>(istream&strm, ThreeDiceGame&);

};

template <class T>

void ThreeDiceGame::setDice3(T dice3)

{

this->dice3 = dice3;

}

template <class T>

ThreeDiceGame ThreeDiceGame::setGame(char result, T diceRollTimes, T sum, T dice1, T dice2, T dice3)

{

ThreeDiceGame g;

g.setResult(result);

g.setDiceRollingTimes(diceRollTimes);

g.setSum(sum);

g.setDice1(dice1);

g.setDice2(dice2);

g.setDice3(dice3);

return g;

}

#endif /\* THREEDICEGAME\_H \*/

**ThreeDiceGame.cpp:**

/\*

\* 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.

\*/

#include "ThreeDiceGame.h"

ThreeDiceGame::ThreeDiceGame():Game()

{

dice3=0;

}

const ThreeDiceGame ThreeDiceGame::operator =(const ThreeDiceGame&right)

{

if (this != &right)

{

diceRolltimes = right.diceRolltimes;

sum = right.sum;

dice1 = right.dice1;

dice2 = right.dice2;

dice3 = right.dice3;

gameResult = right.gameResult;

r = right.r;

}

return \*this;

}

int ThreeDiceGame::getDice3()

{

return dice3;

}

void ThreeDiceGame::printDice3()

{

cout << "The dice 3 is: " << dice3 << endl;

}

void ThreeDiceGame::printSum()

{

cout << "The sum of three dice is: " << sum << endl;

}

void ThreeDiceGame::printGame()

{

printResult();

printDice1();

printDice2();

printDice3();

printSum();

printDiceRollTimes();

}

**FourDiceGame.h:**

/\*

\* 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: FourDiceGame.h

\* Author: raytim

\*

\* Created on 2019年11月22日, 下午11:31

\*/

#ifndef FOURDICEGAME\_H

#define FOURDICEGAME\_H

#include "ThreeDiceGame.h"

class FourDiceGame :public ThreeDiceGame

{

private:

int dice4;//one dice more than three

public:

FourDiceGame();

template <class T>

void setDice4(T);

int getDice4();

void printDice4();

void printGame();

void printSum();

template <class T>

FourDiceGame setGame(char, T, T, T, T, T, T);

const FourDiceGame operator=(const FourDiceGame&);

friend ostream &operator<<(ostream&strm, const FourDiceGame&);

friend istream &operator>>(istream&strm, FourDiceGame&);

};

template <class T>

void FourDiceGame::setDice4(T dice4)

{

this->dice4 = dice4;

}

template <class T>

FourDiceGame FourDiceGame::setGame(char result, T diceRollTimes, T sum, T dice1, T dice2, T dice3, T dice4)

{

FourDiceGame g;

g.setResult(result);

g.setDiceRollingTimes(diceRollTimes);

g.setSum(sum);

g.setDice1(dice1);

g.setDice2(dice2);

g.setDice3(dice3);

g.setDice4(dice4);

return g;

}

#endif /\* FOURDICEGAME\_H \*/

**FourDiceGame.cpp:**

/\*

\* 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.

\*/

#include "FourDiceGame.h"

FourDiceGame::FourDiceGame() : ThreeDiceGame()

{

dice4 = 0;

}

const FourDiceGame FourDiceGame::operator =(const FourDiceGame&right)

{

if (this != &right)

{

diceRolltimes = right.diceRolltimes;

sum = right.sum;

dice1 = right.dice1;

dice2 = right.dice2;

dice3 = right.dice3;

dice4 = right.dice4;

gameResult = right.gameResult;

r = right.r;

}

return \*this;

}

int FourDiceGame::getDice4()

{

return dice4;

}

void FourDiceGame::printDice4()

{

cout << "The dice 4 is: " << dice4 << endl;

}

void FourDiceGame::printGame()

{

printResult();

printDice1();

printDice2();

printDice3();

printDice4();

printSum();

printDiceRollTimes();

}

void FourDiceGame::printSum()

{

cout << "The sum of four dice is: " << sum << endl;

}