Project 1

The Card Matching Game

CSC-48948

Name: Matthew Yang

Date 10/29/17

Introduction

Title: Matching Card Game

This is a card matching game.

The cards are set before you, and the user needs to input two cards one at a time in order to reveal what number is underneath those two cards. The goal of the game is to pick two cards that have the same number underneath them.

When all cards are matched successfully, the game is over.

This is a family-fun memory game that many play to see how fast they can finish the game in the least amount of turns.

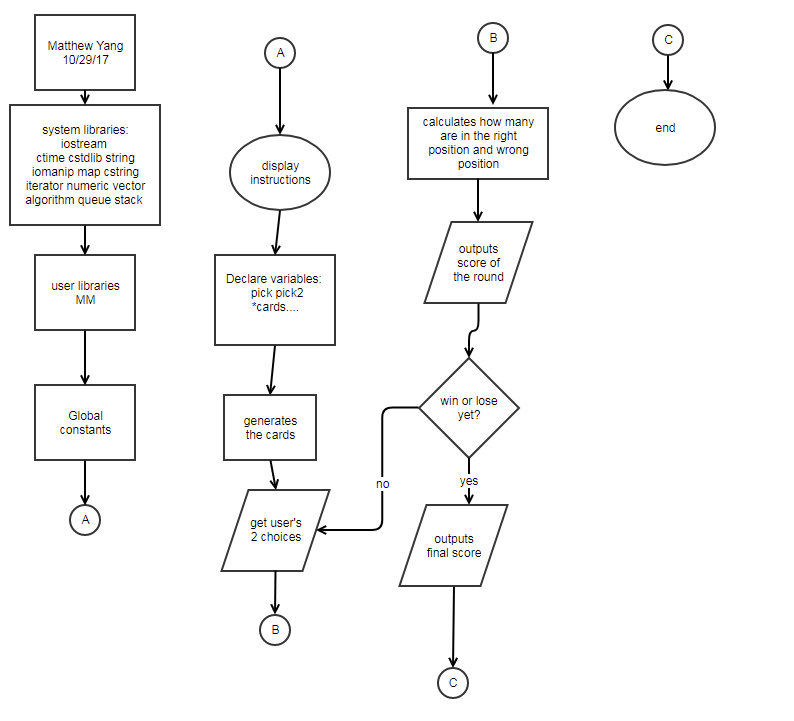
Summary

Project size: 794 lines

Variables: \*Cards holds all of the cards pick and pick2 hold the picks of the user. Choice holds the users choice of easy hard and medium mode.

This project contains most concepts that are required. I do like this project because I actually learned a lot of fun things in the C++ STL. I really did like doing this project although I am kind of bogged down in work from other classes so I could have done more. I look forward to using the STL more, and hope we cover more in class.

Flowchart



Major variables

Type Variable Name Description Location

|  |  |  |  |
| --- | --- | --- | --- |
| Int \* | freqofC | Frequency of cards | In the shuffle class function of Deck |
|  | Cards | Value of the cards | In the shuffle class function of Deck |
|  | Turns | How many turns | In Intro() at the end |
|  | Cheater | Cheater mode yes/no | In the Intro class function of Cards - Lines 450-454 |
| string | Name | Gets user’s name | In the Intro class function of Cards |
| int | pick, pick2 | Users pick of cards | Used in pick(); |
|  | choice | Chooses what mode to play, easy hard medium. | In the Option Main function lines 149-175 |

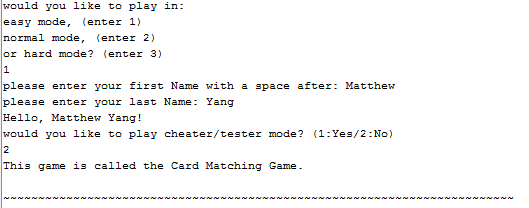
Constructs

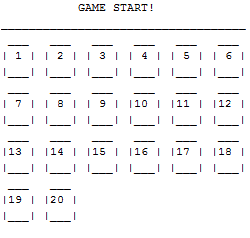
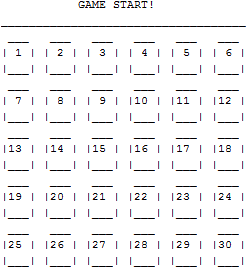
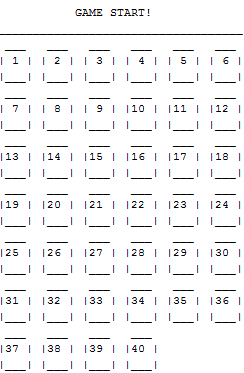
Name Keywords

|  |  |
| --- | --- |
| maps | * Used to map the name of the user |
| queues | * Used to find eliminated cards |
| stacks | * Used to gather the picks of the user |
| iterators | * Used to access containers |
| algorithms | * used to gather the total points accumulate(); |

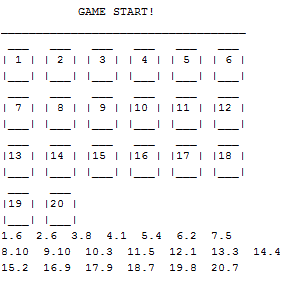
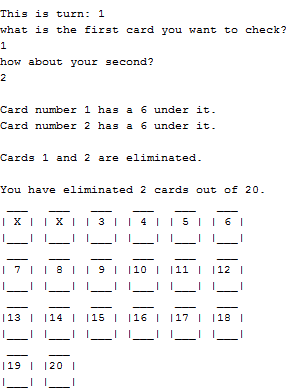
Samples

This is the introduction of my program. You are asked to choose what game difficulty you want to play,(easy-20 cards, med-30 cards, or hard 40 cards). Then you input a name and then choose whether or not to play cheater mode.



These show you easy mode, novice mode, and hard mode in their respective order.

1.This shows you cheater mode. This mode shows you the card number, followed by a period and then the card value.

2. this shows you what happens when you eliminate a card.

Pseudocode

*gets choice of difficulty*

*gives cheater code confirmation to class*

*instantiates card to play with*

*gets name and cheater options*

*gives the introduction*

*outputs the intro and rules to the game into a binary file*

*creates deck of cards*

*operator overload counts turns*

*displays cards*

*displays turns*

*receives input of card picks from user*

*error check/exception here*

*outputs exception message*

*finishes the program if all cards are eliminated*

*gives you the score of your game*

*deletes character array for intro*

Program

class Deck{

protected:

int \*freqofC;

int \*cards;

int MAX;

public:

Deck(int max){

MAX=max;

}

void Shuffle();

~Deck(){

delete [] freqofC;

delete [] cards;

}

};

class Cards:public Deck{

private:

string Name;

int Turns;

int Cheater; //cheater mode

int pointsT[100]={0};

public:

Cards(int max):Deck(max){

Turns=0;

}

void Pick();

void Display();

bool ElimCheck(bool);

void Score();

virtual void Intro();

void Turn();

void operator++(int){

this->Turns++;

}

};

template <class T>

T option(T);

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

int choice2=0; //used to make the choice transfer to the class function

bool isDone = false; //determines when game is done

string intro; //intro holder

char \*intrArr; //binary file for intro

fstream introPr; //declares prologue file

char line[1200]; //binary places into this file

int choice=option(choice2); //gets choice of difficulty

Cards card(choice); //instantiates card to play with

card.Intro(); //get name and cheater options

//gives the introduction

intro= "This game is called the Card Matching Game. \n\n"

"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"

"~~~~~~~~~~~~\n\n"

"The goal of the game is to eliminate all the cards"

" you see below.\n"

" All of which have one number underneath \n"

"them from 1-13. These cards contain two of each number from 1-13.\n"

"You eliminate these cards by first picking two of the 26 cards on\n"

"the screen, one at a time. Then, I will flip them over, and tell\n"

"you what number is underneath them. If these two cards match, "

"then\n"

"they are eliminated from the game, this will be indicated out loud,\n"

"and I will draw an X on the card. If these cards do not match,"

" then\n"

"they will simply be placed back as they were before I flipped "

"them.\n"

"The game is over when all cards are eliminated,(when all cards "

"have\n"

"X's on top of them).\n\n"

"NOTE: To get the full experience of this game, do not scroll up to\n"

"see what I said a card was on a previous turn, "

"this game is based on memory.\n\n"

"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"

"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n\n"

" GAME START!\n"

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

cout<<intro;

// creates deck of cards

card.Shuffle();

do{

card++; //operator overload counts turns

card.Display(); //displays cards

card.Turn(); //displays turns

card.Pick(); //receives input of card picks from user

try{ //error check/exception here

isDone=card.ElimCheck(isDone);

}

catch(string error){

cout <<error; //outputs exception message

}

}while(!isDone); //finishes the program if all cards are eliminated

card.Score(); //gives you the score of your game

delete [] intrArr; //deletes character array for intro

return 0;

}

template <class T>

T option(T choice){

cout << "would you like to play in: "<<endl;

cout << "easy mode, (enter 1)"<<endl;

cout << "normal mode, (enter 2)"<<endl;

cout << "or hard mode? (enter 3)"<<endl;

cin >>choice;

while(choice>3 || choice<1){

cout <<"please enter a value from 1 - 3"<<endl<<endl;

cin >> choice;

}

if (choice==1){choice=20;}

if (choice==2){choice=30;}

if (choice==3){choice=40;}

return choice;

}

void Cards::Intro(){

map <int, string> names;

map <int, string> :: iterator itr;

string fName, lName;

cout <<"please enter your first Name: ";

cin>>fName;

cout <<"please enter your last Name: ";

cin>>lName;

names.insert(pair<int, string>(1,fName));

names.insert(pair<int, string>(2,lName));

itr = names.begin();

Name=fName+' '+lName;

cout << "Hello, "<<itr->second<<"!"<<endl;

cout <<"would you like to play cheater/tester mode? (1:Yes/2:No)"<<endl;

cin >> Cheater;

while(Cheater!=1 && Cheater!=2){

cout << "please enter a valid input from 1-2"<<endl;

cin>>Cheater;

}

}

void Deck::Shuffle(){

unsigned seed = time(0); //creates randomness

srand(seed);

cards = new int [MAX];

freqofC = new int [MAX/2];

for(int i=0; i<MAX/2; ++i) {

freqofC[i]=0;

}

int num=0;

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

int num = rand() % (MAX/2) + 1;

++freqofC[num-1];

if (freqofC[num-1]>2) { //makes sure that no card has the

//same value more than twice.

int numP=num;

do{

--freqofC[num-1];

num = rand() % (MAX/2) + 1;

++freqofC[num-1];

if (num==numP){continue;}

}while(freqofC[num-1]>2);

}

cards[i] = num;

}

}

void Cards::Display(){

string cad1P2= "| 1 |",cad2P2= "| 2 |",

cad3P2= "| 3 |",cad4P2= "| 4 |",

cad5P2= "| 5 |",cad6P2= "| 6 |",

cad7P2= "| 7 |",cad8P2= "| 8 |",

cad9P2= "| 9 |",cad10P2="|10 |",

cad11P2="|11 |",cad12P2="|12 |",

cad13P2="|13 |",cad14P2="|14 |",

cad15P2="|15 |",cad16P2="|16 |",

cad17P2="|17 |",cad18P2="|18 |",

cad19P2="|19 |",cad20P2="|20 |",

cad21P2="|21 |",cad22P2="|22 |",

cad23P2="|23 |",cad24P2="|24 |",

cad25P2="|25 |",cad26P2="|26 |",

cad27P2="|27 |",cad28P2="|28 |",

cad29P2="|29 |",cad30P2="|30 |",

cad31P2="|31 |",cad32P2="|32 |",

cad33P2="|33 |",cad34P2="|34 |",

cad35P2="|35 |",cad36P2="|36 |",

cad37P2="|37 |",cad38P2="|38 |",

cad39P2="|39 |",cad40P2="|40 |",

cadP1="\_\_\_", cadP3= "|\_\_\_|",

X="| X |";

//cards being turned into X's if they are removed from the game

if (cards[0]==0){cad1P2=X;} if (cards[1]==0){cad2P2=X;}

if (cards[2]==0){cad3P2=X;} if (cards[3]==0){cad4P2=X;}

if (cards[4]==0){cad5P2=X;} if (cards[5]==0){cad6P2=X;}

if (cards[6]==0){cad7P2=X;} if (cards[7]==0){cad8P2=X;}

if (cards[8]==0){cad9P2=X;} if (cards[9]==0){cad10P2=X;}

if (cards[10]==0){cad11P2=X;} if (cards[11]==0){cad12P2=X;}

if (cards[12]==0){cad13P2=X;} if (cards[13]==0){cad14P2=X;}

if (cards[14]==0){cad15P2=X;} if (cards[15]==0){cad16P2=X;}

if (cards[16]==0){cad17P2=X;} if (cards[17]==0){cad18P2=X;}

if (cards[18]==0){cad19P2=X;} if (cards[19]==0){cad20P2=X;}

//for normal mode

if (MAX>=30){

if (cards[20]==0){cad21P2=X;} if (cards[21]==0){cad22P2=X;}

if (cards[22]==0){cad23P2=X;} if (cards[23]==0){cad24P2=X;}

if (cards[24]==0){cad25P2=X;} if (cards[25]==0){cad26P2=X;}

if (cards[26]==0){cad27P2=X;} if (cards[27]==0){cad28P2=X;}

if (cards[28]==0){cad29P2=X;} if (cards[29]==0){cad30P2=X;}

}

//for hard mode

if (MAX>=40){

if (cards[30]==0){cad31P2=X;} if (cards[31]==0){cad32P2=X;}

if (cards[32]==0){cad33P2=X;} if (cards[33]==0){cad34P2=X;}

if (cards[34]==0){cad35P2=X;} if (cards[35]==0){cad36P2=X;}

if (cards[36]==0){cad37P2=X;} if (cards[37]==0){cad38P2=X;}

if (cards[38]==0){cad39P2=X;} if (cards[39]==0){cad40P2=X;}

}

//cards being displayed

//easy mode

if (MAX==20){

cout << " "<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)

<<cadP1<<setw(6)<<cadP1<<endl;

cout <<cad1P2<<" "<<cad2P2<<" "<<cad3P2<<" "<<cad4P2<<" "<<cad5P2<<" "

<<cad6P2<<endl;

cout <<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3

<<endl;

cout << " "<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)

<<cadP1<<setw(6)<<cadP1<<endl;

cout <<cad7P2<<" "<<cad8P2<<" "<<cad9P2<<" "<<cad10P2<<" "<<cad11P2<<" "<<

cad12P2<<endl;

cout <<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<

endl;

cout << " "<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)

<<cadP1<<setw(6)<<cadP1<<endl;

cout <<cad13P2<<" "<<cad14P2<<" "<<cad15P2<<" "<<cad16P2<<" "<<cad17P2<<" "

<<cad18P2<<endl;

cout <<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3

<<endl;

cout << " "<<cadP1<<setw(6)<<cadP1<<endl;

cout <<cad19P2<<" "<<cad20P2<<endl;

cout <<cadP3<<" "<<cadP3<<endl;

}

//normal mode

if (MAX==30){

cout << " "<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)

<<cadP1<<setw(6)<<cadP1<<endl;

cout <<cad1P2<<" "<<cad2P2<<" "<<cad3P2<<" "<<cad4P2<<" "<<cad5P2<<" "

<<cad6P2<<endl;

cout <<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3

<<endl;

cout << " "<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)

<<cadP1<<setw(6)<<cadP1<<endl;

cout <<cad7P2<<" "<<cad8P2<<" "<<cad9P2<<" "<<cad10P2<<" "<<cad11P2<<" "<<

cad12P2<<endl;

cout <<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<

endl;

cout << " "<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)

<<cadP1<<setw(6)<<cadP1<<endl;

cout <<cad13P2<<" "<<cad14P2<<" "<<cad15P2<<" "<<cad16P2<<" "<<cad17P2<<" "

<<cad18P2<<endl;

cout <<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3

<<endl;

cout << " "<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)

<<cadP1<<setw(6)<<cadP1<<endl;

cout <<cad19P2<<" "<<cad20P2<<" "<<cad21P2<<" "<<cad22P2<<" "<<cad23P2<<" "

<<cad24P2<<endl;

cout <<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3

<<endl;

cout << " "<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)

<<cadP1<<setw(6)<<cadP1<<endl;

cout <<cad25P2<<" "<<cad26P2<<" "<<cad27P2<<" "<<cad28P2<<" "<<cad29P2<<" "

<<cad30P2<<endl;

cout <<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3

<<endl;

}

//hard mode

if (MAX==40){

cout << " "<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)

<<cadP1<<setw(6)<<cadP1<<endl;

cout <<cad1P2<<" "<<cad2P2<<" "<<cad3P2<<" "<<cad4P2<<" "<<cad5P2<<" "

<<cad6P2<<endl;

cout <<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3

<<endl;

cout << " "<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)

<<cadP1<<setw(6)<<cadP1<<endl;

cout <<cad7P2<<" "<<cad8P2<<" "<<cad9P2<<" "<<cad10P2<<" "<<cad11P2<<" "<<

cad12P2<<endl;

cout <<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<

endl;

cout << " "<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)

<<cadP1<<setw(6)<<cadP1<<endl;

cout <<cad13P2<<" "<<cad14P2<<" "<<cad15P2<<" "<<cad16P2<<" "<<cad17P2<<" "

<<cad18P2<<endl;

cout <<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3

<<endl;

cout << " "<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)

<<cadP1<<setw(6)<<cadP1<<endl;

cout <<cad19P2<<" "<<cad20P2<<" "<<cad21P2<<" "<<cad22P2<<" "<<cad23P2<<" "

<<cad24P2<<endl;

cout <<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3

<<endl;

cout << " "<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)

<<cadP1<<setw(6)<<cadP1<<endl;

cout <<cad25P2<<" "<<cad26P2<<" "<<cad27P2<<" "<<cad28P2<<" "<<cad29P2<<" "

<<cad30P2<<endl;

cout <<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3

<<endl;

cout << " "<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)

<<cadP1<<setw(6)<<cadP1<<endl;

cout <<cad31P2<<" "<<cad32P2<<" "<<cad33P2<<" "<<cad34P2<<" "<<cad35P2<<" "

<<cad36P2<<endl;

cout <<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3

<<endl;

cout << " "<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<endl;

cout <<cad37P2<<" "<<cad38P2<<" "<<cad39P2<<" "<<cad40P2<<endl;

cout <<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<endl;

}

//cheater mode

if (Cheater==1){

for(int i=0 ,j=1;i<MAX;i++,j++){

cout <<i+1<<"."<<cards[i]<<" ";

if(j==7){

j=0;

cout<<endl;

}

}

}

cout <<endl<<endl;

}

void Cards::Pick(){

queue <int> elims;

int pointC=0, points=0;

stack <int> picks;

int pick, pick2;

cout <<"what is the first card you want to check?"<<endl;

cin >>pick;

picks.push(pick);

while(pick<1 || pick>MAX) {

picks.pop();

cout<<"you must pick a card that exists on the screen above,"

"and that does not have an X on it.";

cin>>pick;

picks.push(pick);

}

while(cards[pick-1]==0){

picks.pop();

cout << "You have already picked that card. Please pick a card"

" that is on the board, and that does not have an X on it.";

cin>>pick;

picks.push(pick);

}

cout <<"how about your second?"<<endl;

cin >>pick2;

picks.push(pick2);

while (pick2<1 || pick2>MAX) {

picks.pop();

cout << "you must pick a card "

"that exists on the screen above, and that does not have an X "

"on it."<<endl;

cin>>pick2;

picks.push(pick2);

}

while (picks.top()==pick){

picks.pop();

cout << "you must pick a number that is different than the first "

<<endl<<"card you have picked. Pick another card for your"

" second card."<<endl;

cin>>pick2;

picks.push(pick2);

}

while(cards[pick2-1]==0){

picks.pop();

cout << "You have already picked that card. Please pick a card"

" that is on the board, and that does not have an X on it.";

cin>>pick2;

picks.push(pick2);

}

//outputs the results of the cards

cout << endl << "Card number "<< picks.top()<<" has a "<<

cards[picks.top()-1]<<" under it." <<endl;

picks.pop();

cout <<"Card number "<< picks.top()<<" has a "<<

cards[picks.top()-1]<<" under it." <<endl<<endl;

//if the cards have the same number underneath them, they are eliminated

if (cards[pick2-1]==cards[pick-1]){

elims.push(pick); //queue

elims.push(pick2);

cout <<"Cards "<<elims.front();

elims.pop();

cout << " and " <<elims.front() <<

" are eliminated." << endl<<endl;

elims.pop();

cards[pick2-1]=0; //sets card pick to zero

cards[pick-1]=0;

if (pointC==0){pointsT[Turns]+=5;}

if (pointC>0 && pointC<3){pointsT[Turns]+=2;}

else{pointsT[Turns]++;}

}

else{

pointC++;

}

}

bool Cards::ElimCheck(bool isDone){

int cnt=0;

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

if( cards[i]==0 ) ++cnt;

}

if(cnt==MAX) isDone = true;

// displays the number of terminated cards

cout << "You have eliminated " << cnt << " cards out of "<<MAX<<

"." << endl;

if(cnt<0 || cnt>MAX){

string error;

error ="an error has occurred, exiting program.";

throw error;

}

return isDone;

}

void Cards::Turn(){

cout <<"This is turn: "<<Turns<<endl;

}

void Cards::Score(){

int n = sizeof(pointsT)/sizeof(pointsT[0]);

int perfect=MAX/2; //the perfect game

vector<int> p(pointsT, pointsT+n);

cout << "You have Won! you have completed the game in " <<Turns<<

" turns.";

cout << endl;

cout<<"You have scored: ";

cout<<(accumulate(p.begin(), p.end(), 0));

cout<<" points!"<<endl;

//gives special message if you had a perfect game

if(perfect==Turns){

cout << "WOW! A perfect game! You're amazing!!";

}

//tells you how well you did on the game if you did not have

//a perfect game

if (Turns>perfect && Turns<=perfect+5){

cout <<"That was a pretty awesome game!";

}

if (Turns>perfect+5 && Turns<=perfect+15){

cout << "That was a pretty good game, although you can do better.";

}

if (Turns>perfect+15 && Turns<=perfect+25){

cout << "a decent game. Try again!";

}

if (Turns>perfect+25){

cout << "That was a not so good game. replay?";

}

cout << endl;

cout << "Thanks for playing, "<<Name;

}