War/I Declare War

Card Game

Laurie Guimont

CSC 17A

Spring 2017

42636

TABLE OF CONTENTS

Introduction 3

How the Card game works 3

Object of the game 3

Rules of the Game 3

how to wage war 3

My approach to the game 4

Translating game play rules to programming language 4

Similarities to the card game 4

Differences from the card game 5

The logic of it all 5

Flowchart 5

Constructs & Concepts Utilized 20

iostream library 20

cstdlib library 20

ctime library 20

iomanip library 20

string library 21

cmath library 21

cctype library 21

cstring library 21

fstream library 21

data types 22

conditional statements 22

loops 22

function prototypes 23

structure 23

Proof of a working product 24

References 27

Program 27

**Introduction**

*“WAR! huh! Yeah,*

*What is it good for? Absolutely…”*

*–Edwin Starr*

…any good pass time with friends and family (the card game, that is).

War, also referred to by some as I Declare War, is a popular 2-Player standard card game. The game is played by people of all ages, and can serve as a useful way of helping young children learn how to count. All the players are required to do is display the cards they are dealt and compare them. Although the game does not require much logic and may be considered simplistic to some, the game is still extremely competitive if the right cards are dealt in the right order, or in other words, in an order that helps players beat their opponent. This dynamic of having cards randomly arranged in a way that benefits either player appealed to me and inspired me to write a program around the game.

**How the Card Game Works**

**Object of the Game**

To accumulate all 52 cards.

**Rules of the Game**

War is typically a two person game. The game is very simple:

1. Shuffle and deal the cards evenly between the two players. Therefore, each player should have 26 cards. Jokers are not used in this game.
2. Players should then turn over the top cards in their pile at the same time. Whoever has the higher value card wins both cards. The ranks of cards are as follows:

All number cards are valued according to their number.

Of the face cards the Ace is the highest overall card, followed by the King, then Queen, and the Jack is the lowest ranked face card. Face cards beat number cards.

1. Keep playing until one of the players has collected all of the cards in the deck.

**How to Wage War**

If the players turn over cards that have the same card value, war is waged! At this point, both players must place 2 to 4 cards faced down, then turn over the proceeding card. Whichever player has the higher war card gets all the cards put down, including the cards faced down and the cards that initiated the war.

Note: The number of cards placed faced down before overturning one is based upon player preference. I have seen games played where only 2 cards were placed faced down, but have also seen games where 4 cards were placed down. The latter is interesting because while placing your cards down, the players count and say aloud, "1, 2, 3, 4" then proceed with "I declare war" while overturning the fifth card at the same time the word "war" is said.

Note: More than one war can be declared in a round. If players throw the same war card down, another round of “faced down” cards must be placed along with another war card. This process should be repeated until one of the players has a higher war card.

**My Approach to the Game**

**Translating Game Play Rules to Programming Language**

While thinking about how I was going to program this game, a couple questions arose:

* “Since the card game has four suites, meaning four of each card, how do I tell the computer that I want to limit the number of times a random number is chosen?”
* “Should I have the computer ‘deal’ 26 cards to the user/player, and then have the player chose from their ‘hand’?”
* “How will a player win or lose the game?”

After a couple of hours of planning my program and toiling with the above questions, I realized that I didn’t know enough to completely program the game the way that I wanted to. Therefore, I had to come up with a way to cope with these problems using the constructs and concepts that I already knew.

**Similarities to the Card Game**

My War program follows the same rules of play as the card game:

* The user and the computer “throw down” a card, then it is determined who has the higher card
* If the same card is thrown down, both user and computer place cards faced down, then reveal the war card.
  + This is repeated if the war cards are the same.

**Differences from the Card Game**

The main difference from the card game is the score. I decided that in order to determine the winner of each round in the game, I would numerically value each card and add up points.

* If you win, you gain the value of the card you put down as well as the value of the card the computer put down.
* If you lose, you lose the value of the card you put down. The same goes for a computer loss.

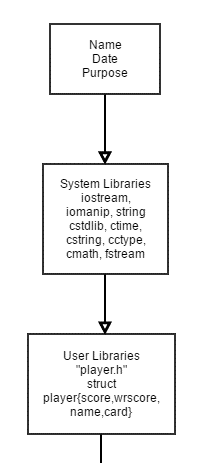
Since the face cards don’t have a numeric value in the regular card game, I assigned a value to each according to their rank in the game. Therefore, the Jack is valued at 11 points and the Ace is valued at 14 points, with the Queen at 12 points, and the King at 13.

In response to my concern about limiting the number of times a card is chosen, I decided to not worry about that and treat the game as if two player were not being dealt cards, but instead were drawing them and putting them back in the pile, all while keeping score. It’s like “War, with Replacements.”

Finally, a player wins or loses based on their score at the time they decide to finish the game. If the player is tired of playing after a while, they can exit the game and the computer will tell them their final score. If the score is higher than the computer’s, they have one. If not, they have lost.

**The Logic of it All**

**Flowchart**

**Since my flowchart is extremely long, I will break it up into smaller pieces and accompany it with pseudocode here. To view my complete flowchart, please visit: <https://www.gliffy.com/go/html5/11933282>

*Write opening comments*

*Bring in 9 system libraries*

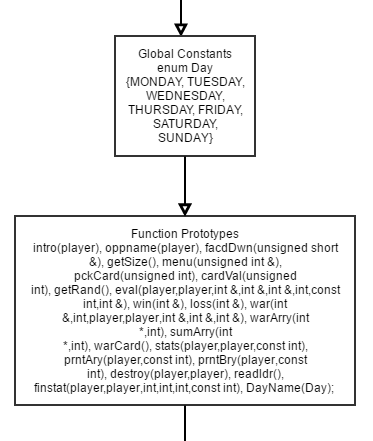
*Bring in 1 user library containing a structure*

*Structure contents:*

*Int pointer called score*

*2 integers called card and wscore*

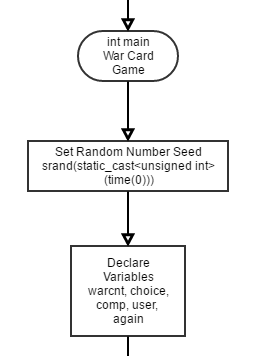
*Char pointer called name*

*Declare enumerated data type Day as a*

*global constant and name the 7*

*days of the week*

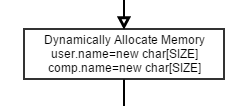
*Bring in all 23 function prototypes*

**

*Enter main, then immediately*

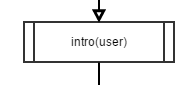
*set random number seed*

*Declare some variables*

**

*Dynamically allocate memory for*

*each player’s name*

**

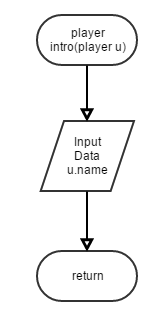
*Call intro function and pass user, which is a structure*

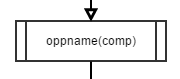
*Enter intro function and let user input name*

*Store name in the name member function of the player*

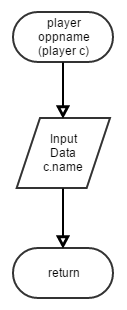
*structure*

*Return to the structure to main*

**

**

*Call oppname function and pass comp, which is a structure*

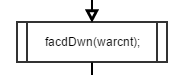
**

*Enter oppname function and let user input opponent name*

*Store name in the name member function of the player*

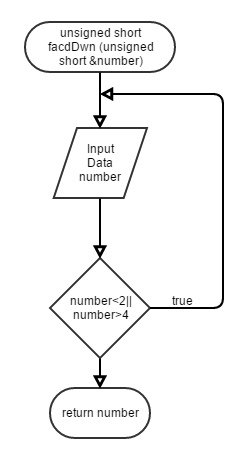
*Structure*

*Return the structure to main*

**

*Call facdDwn function and pass*

*warcnt in*

**

*Enter facdDwn function and prompt*

*user to enter a number to be used*

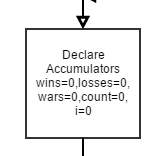
*for warcnt variable*

*Number must be 2,3, or 4*

*Validate the input with while loop*

*Return the number back to main*

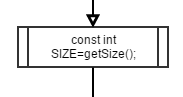
*Pass by reference*

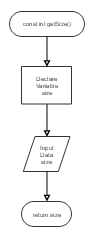
**

*Return from facdDwn and enter a do/while loop*

*the begins by declaring accumulators*

*Call getSize function to set the size of new dynamic*

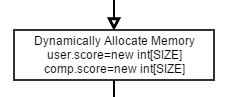
*arrays*

**

*Enter getSize and ask user to enter the number*

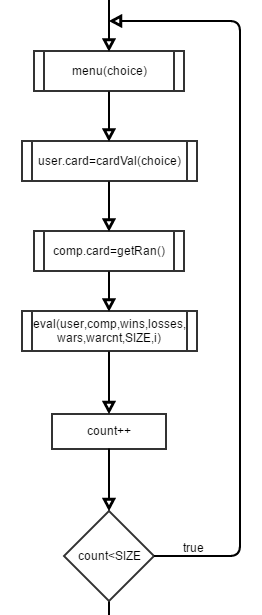
*of hands they would like to play*

*Return size to main*

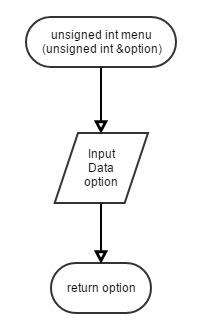
**

*Dynamically allocate more memory for score*

*member in the player structure*

*Begin a nested do/while loop by calling menu function*

*and pass in choice variable*

**

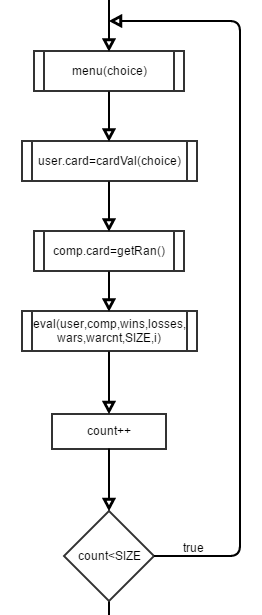
*Enter menu function*

*Prompt to input card type they want to play*

*Either number or face card can be chosen*

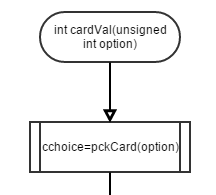
*Return option back to main*

*Pass by reference*

**

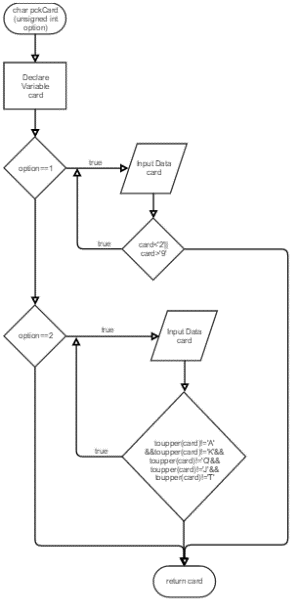
*Return from menu function and call cardVal function*

*while passing in choice to get user’s card*

**

*Enter cardVal function and immediate call pckCard*

*function to set cchoice variable*

*If “number card”(1) or “face card”*

*(2) is selected, input card choice*

*If (1), card options are from 2-9*

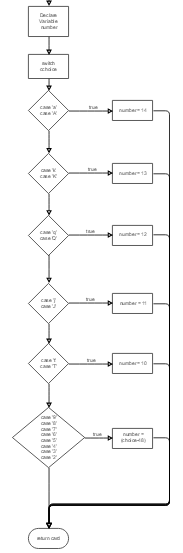
*If (2) card options are t,j,q,k, or a*

*Verify that all data is valid with while loops*

*Return card if (1) or (2)*

*Pass by reference to cardVal*

*Else, simply return to cardVal function*

*Return card choice to cardVal and perform switch *

*to evaluate card value*

*Number will be assigned based on card chosen*

*Face card values are:*

*Ace: 14*

*King: 13*

*Queen: 12*

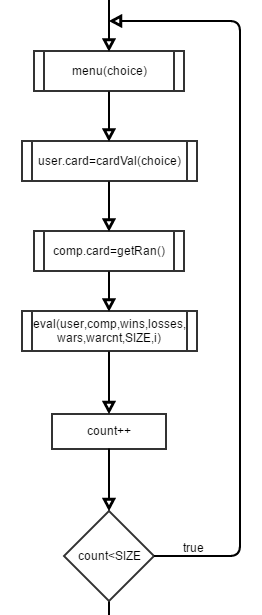
*Jack: 11*

*Ten: 10*

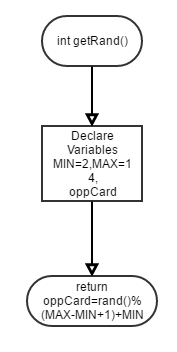
*All other number cards are their own value*

*Return number back to main*

*Card member function is set to number for user*

*Return from cardVal and call getRand function*

*to determine opponent’s card*

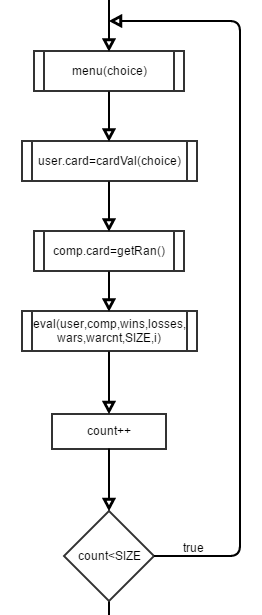
**

*Enter getRand and computer chooses a random number*

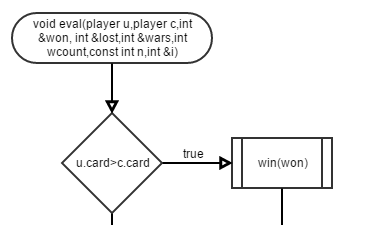
*between 2 and 14 and*

*Return oppCard to main*

*Card member function is set to number for comp*

**

*Call eval function to compare the two cards chosen*

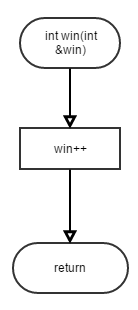
*Enter eval function and compare both cards*

*chosen*

*If user’s card is bigger than comp’s card, user wins*

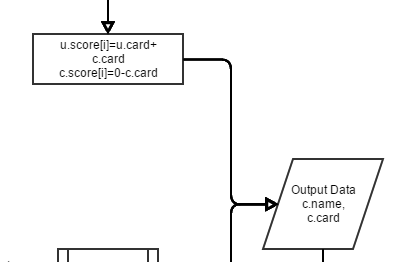
*round and win function is called*

*won variable passed by reference*

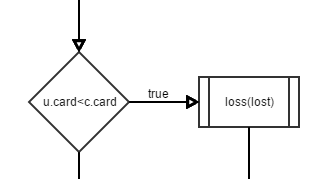
*Enter win, increment number of wins and output*

*message*

*Return to eval function*

*Return from win function and calculate both scores*

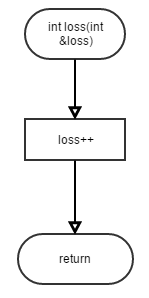
*Output opponent name and losing card*

**

*Else if user’s card is smaller, computer wins*

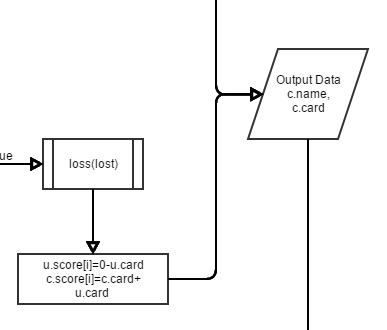
*round and loss function is called*

*lost variable passed by reference*

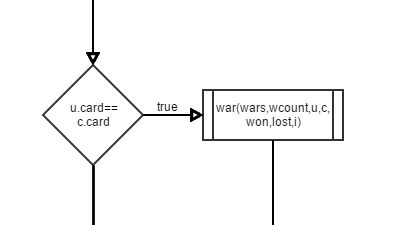
*Enter loss, increment number of losses and output*

*message*

*Return to eval funation*

*Return from loss function and calculate both scores*

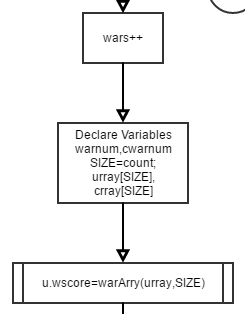
*Output opponent name and winning card*

**

*Else if both numbers are equal*

*User has entered war*

*War function is called*

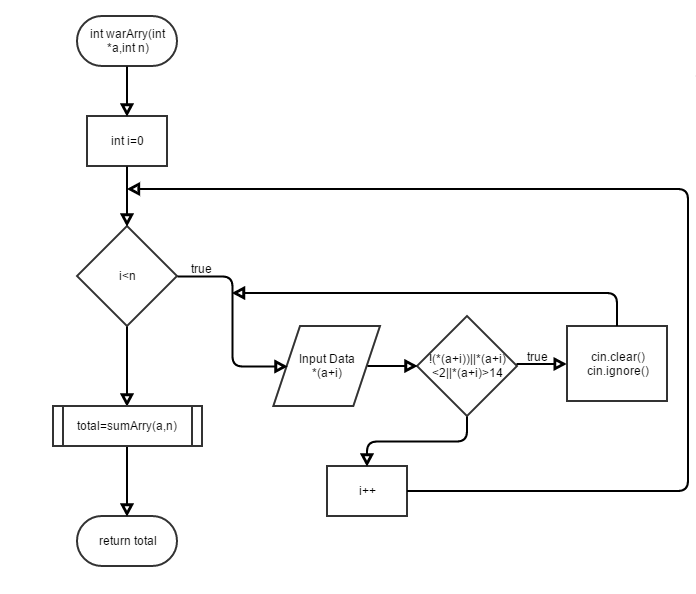
**

*Enter war function*

*Declare more variables for arrays to be used*

*User’s wscore is equal to the value returned by*

*warArry function*

**

*Enter warArry function*

*Prompt user to enter face down cards according*

*to initial input at start of the game using*

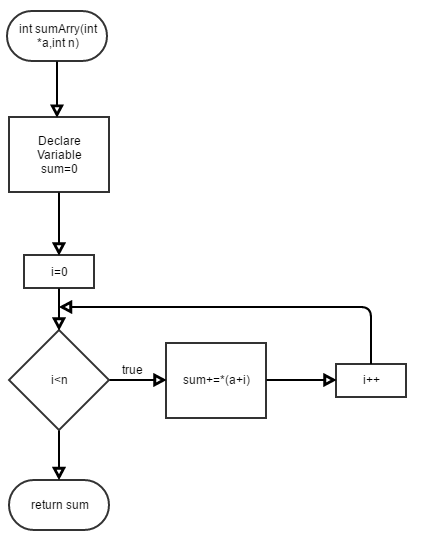
*for loop*

*Validate with while loop*

*total equals the value returned from sumArry*

*function*

*Call sumArry function and pass pointer with size*

**

*Enter sumArry and set sum accumulator equal to*

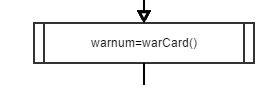
*zero*

*Add the pointer values from warArry and pass*

*sum back by value*

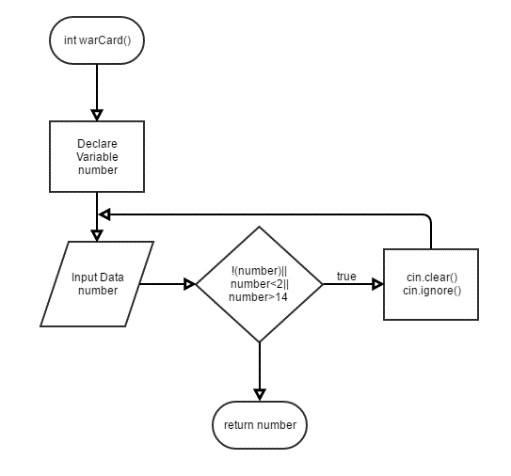
*Return to warArry*

*Return total to main from warArry*

**

*Return from warArry to war function*

*Call warCard function to get warnum*

**

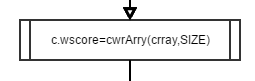
*Enter warCard function and declare number*

*Prompt user to enter number for war card*

*Validate with while loop*

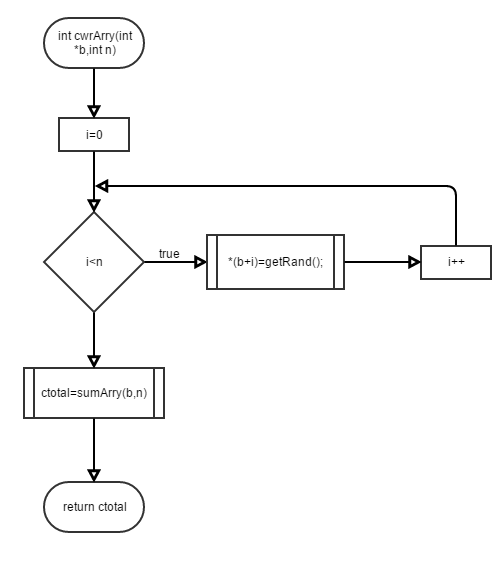
*Return number to war function*

*Default parameters used*

**

*Return from warCard function*

*Call cwrArry function to get comp’s war score*

**

*Enter cwrArry function*

*Computer now chooses faced down cards*

*(loops the same number of times as*

*indicated in facdDwn)*

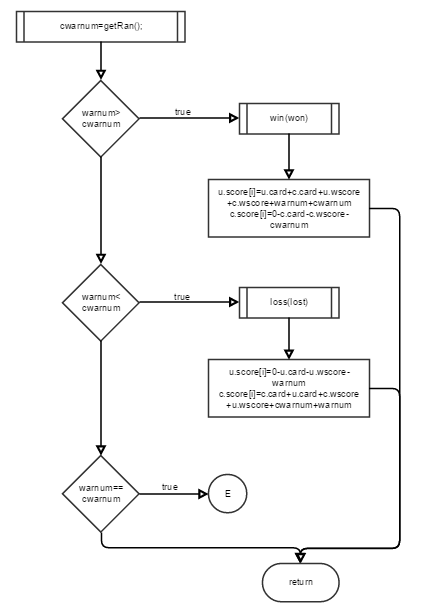
*getRand is called again*

*Value of these cards are summed after a*

*call to sumArry is made once again*

*Return ctotal to war function*

*Pass by value*

**

*Return from cwrArry*

*Call getRand function once again to get*

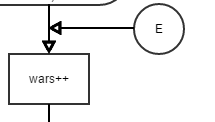
*cwarnum*

*Return fromgetRand function*

*War Card comparison is made*

*whoever has the higher number, wins*

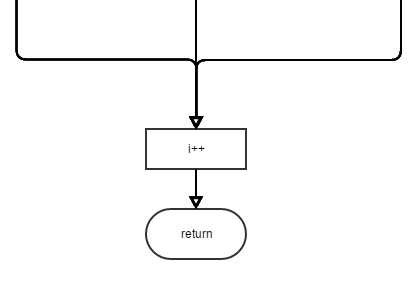
*Loser loses points*

*If warnum equals cwarnum*

*repeat entire war procedure*

*Loop this step until someone has won the war*

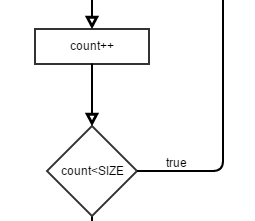
*Return to eval function*

**

*Increment to next position in score array to*

*Keep track of the score for each round*

*Return to main*

**

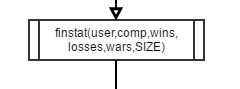
*Increment to keep track of the number of*

*hands the user wants to play*

*Continue to loop the above process for the*

*amount of hands the user indicated in the*

*beginning*

*Once the inner do/while loop is completed*

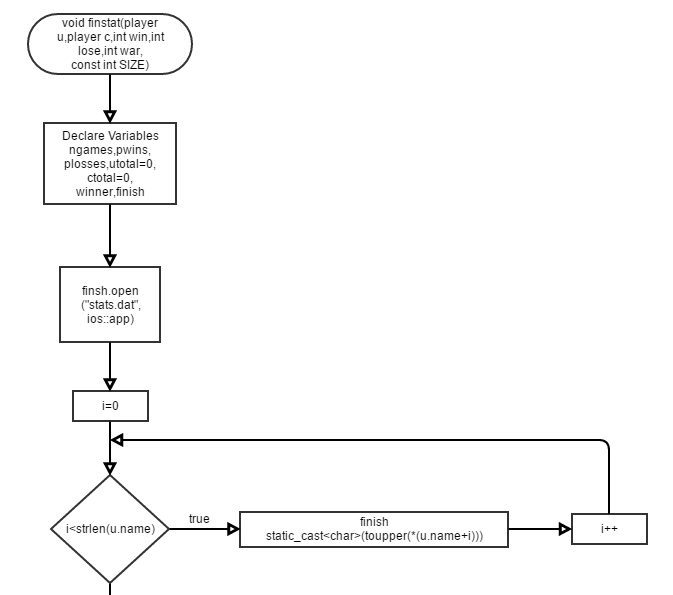
*call the finstat function*

*Enter finstat function to write game*

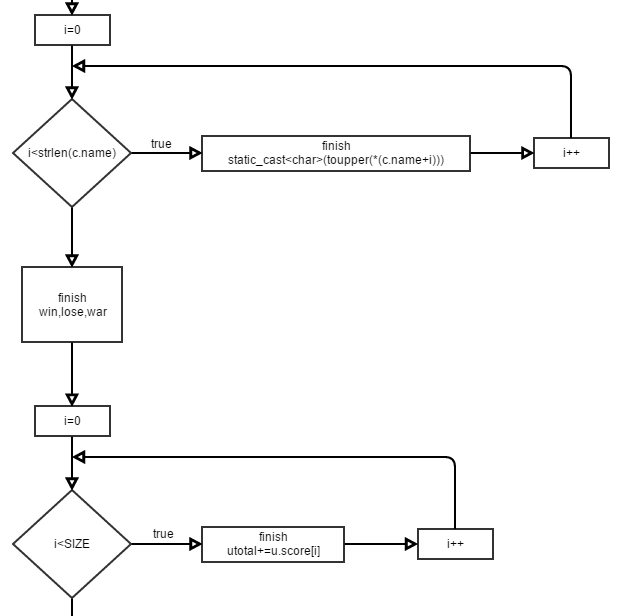
*results to a file*

*Declare variables and the output file in*

*append mode*

*Write names, number of wins, losses,*

*and wars to the file*

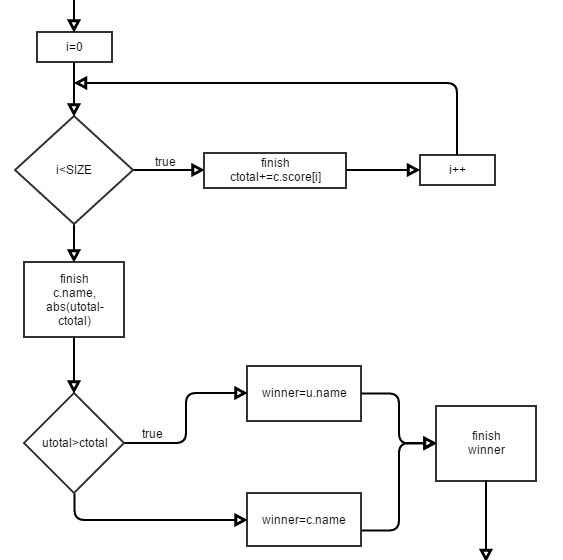
**

*Calculate and total both the user’s*

*total score and the comp’s*

*total score and write to*

*file*

**

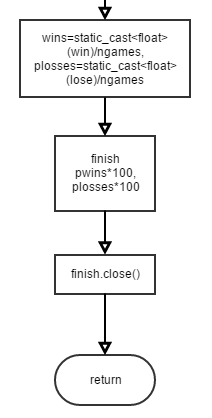
*Determine the winner by comparing*

*the totals of both players*

*whoever is the winner, write*

*that information to the file*

*Calculate the point difference also*

**

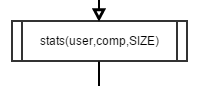
*Calculate the user’s percentage of wins and*

*losses from the hands and output to*

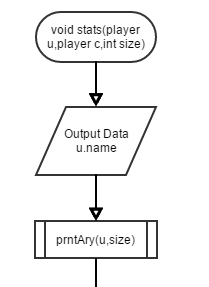
*the same file*

*Close the file*

*Return to main*

**

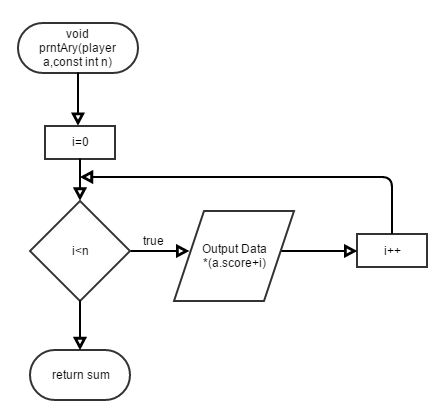
*Call stats function to show in-game stats*

**

*Enter stats function*

*Output the user’s name*

*Call prntAry function to see user’s scores*

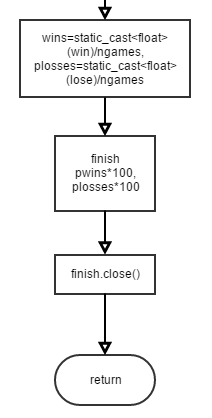
**

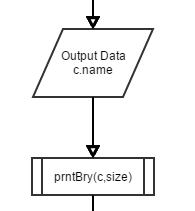
*Enter prntAry function*

*Use a for loop to print the pointer values*

*To the screen*

*Return to stats function*

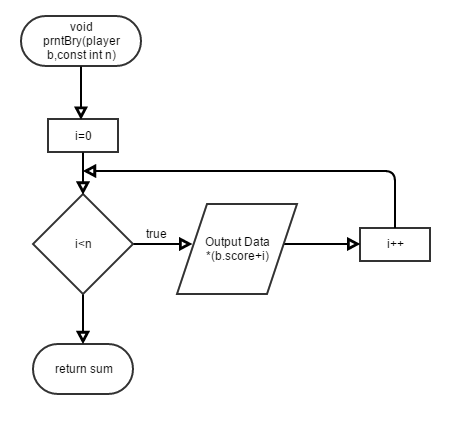
**

**

*Output opponent’s name*

*Call prntBry function to print the*

*Opponent’s scores*

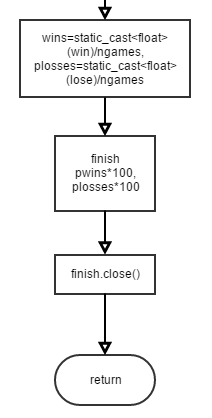
**

*Enter prntBry function*

*Use a for loop to print the pointer values*

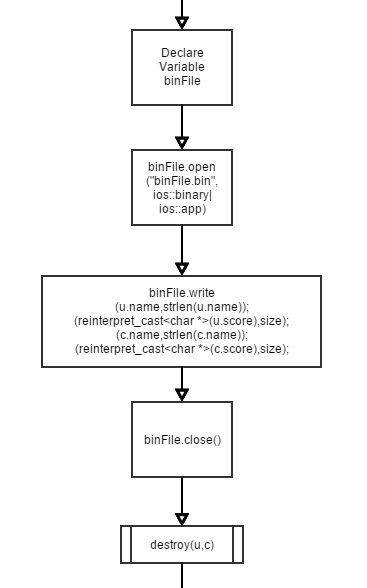
*To the screen*

*Return to stats function*

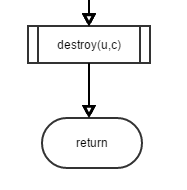
**

*Declare a binary file and open it in append*

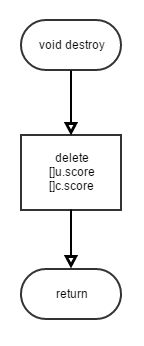
*mode*

*Write both players’ name and scores to the file*

*Close the file*

**

*Call the destroy function*

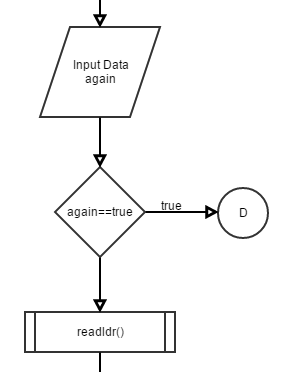
**

*Enter the destroy function*

*Delete the dynamic arrays containing the scores of*

*Both players to prevent a memory leak*

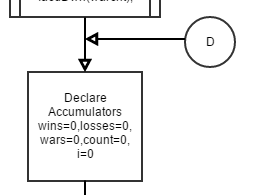
*Return to stats function*

*Return to main*

*Ask the user if they would like to play another*

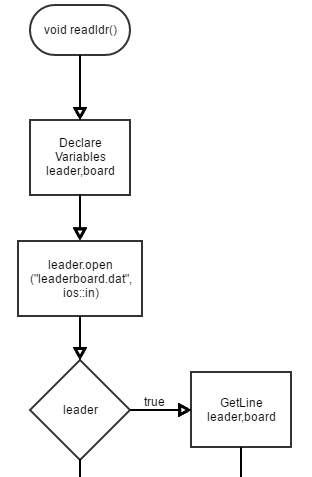
*Game*

*If yes, then loop to the beginning of the outer*

* do/while loop*

*Else, thank user for playing*

*Call readldr file to display leaderboard*

**

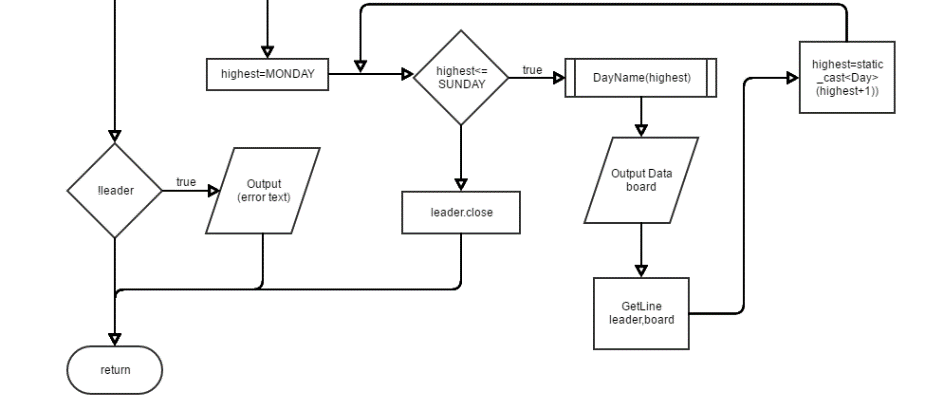
*Enter readldr function*

*Declare some variables*

*Open the leaderboard.dat file*

*If the file exists, read in the lines of*

*the file in a for loop*

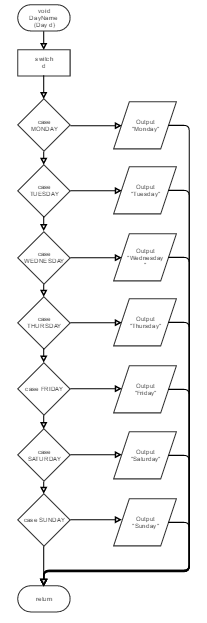
**

*Loop through the lines of the file*

*using the enumerated data*

*type Day*

*Call DayName function*

**

*Enter DayName function*

*Output the actual name of each day*

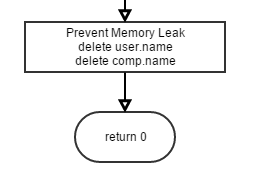
*Using a switch statement*

*Return to readldr function*

*Once the program finishes reading in the file*

*Return to main*

*If program cannot open file, output error and return*

**

*Return from readldr*

*Prevent memory leak by deleting dynamic*

*name arrays*

*Return 0; the program is complete*

**Constructs & Concepts Utilized**

**iostream Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| static\_cast | 6 | Statically cast as different variable | Line 53,116,119, 144,145,180 |
| cout | 54 | Output Data | Throughout |
| cin | 13 | Input Data | Throughout |
| cin.getline | 2 | Reads in char array | Line 475,484 |
| cin.ignore() | 2 | Prevented input problems | Line 241,279 |
| cin.clear() | 2 | Stopped infinite loop | Line 240,278 |
| delete | 4 | Prevent memory leaks | Line 96,97,195,  196 |

**cstdlib Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| srand() | 1 | Random # seed | Line 53 |
| rand() | 1 | Generates rand # | Line 377 |

**ctime Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| time | 1 | Set current time | Line 53 |

**iomanip Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| fixed | 1 | Format final game stats | Line 146 |
| setprecision() | 1 | Format final game stats | Line 146 |
| showpoint | 1 | Format final game stats | Line 146 |
| setw() | 7 | Format final game stats  and leaderboard | Line 183,120,121,  122,128,131,132 |

**string Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| string | 2 | Declare var. | Line 108,170 |
| getline() | 2 | Reads string data | Line 178,184 |

**cmath Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| abs() | 1 | Neg. Score Alert  Point Difference | Line 132 |

**cctype Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| toupper() | 7 | Manipulates Character Case | Line 116,119,  428,429 |

**cstring Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| strlen() | 4 | Determines length of char array | Line 115,118,  221,223 |

**fstream Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| finish.open() | 1 | Open file | Line 112 |
| finish.close() | 1 | Close file | Line 152 |
| leader.open() | 1 | Open file | Line 173 |
| leader.close() | 1 | Close file | Line 188 |
| binFile.open() | 1 | Open file | Line 220 |
| binFile.close() | 1 | Close file | Line 227 |
| binFile.write() | 4 | Write to binary file | Line 221-224 |
| finish | 18 | Write to file | Line 113,114,116, 117,119,123,128, 131,132,137,141, 146-149 |
| leader | 3 | Read in file | Line 177,178,184 |
| fstream | 3 | Declare var. | Line 109,169,218 |
| reinterpret\_cast | 2 | type cast for bin file | Line 222,224 |

**Data Types:**

|  |  |  |
| --- | --- | --- |
| Data Types | Frequency | Location |
| int | 100 | throughout |
| unsigned int | 10 | Line 31-33,53,57,  380,409,437 |
| char | 11 | Line 32,51,62,63,116,  119,222,224,382,  409,411 |
| string | 2 | Already mentioned |
| float | 3 | Line 106,144,145 |
| fstream | 3 | Already mentioned |
| bool | 1 | Line 59 |

**Conditional Statements:**

|  |  |  |
| --- | --- | --- |
| Conditional Statement | Frequency | Starting Location |
| if/else | 2 | Line 132,177 |
| if/else if | 4 | Line 299,323,353,413 |
| switch | 2 | Line 156,386 |

**Loops:**

|  |  |  |
| --- | --- | --- |
| Loops | Frequency | Starting Location |
| for | 10 | Line 115,118,126,129,  179,200,205,251,  258,272 |
| while | 6 | Line 239,277,312,  418,428,465 |
| do-while | 2 | Line 68,75 |

**NOTE: Global Constant ONLY used for declaration of enumerated data type!**

**enum Day {MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY};**

**Function Prototypes:**

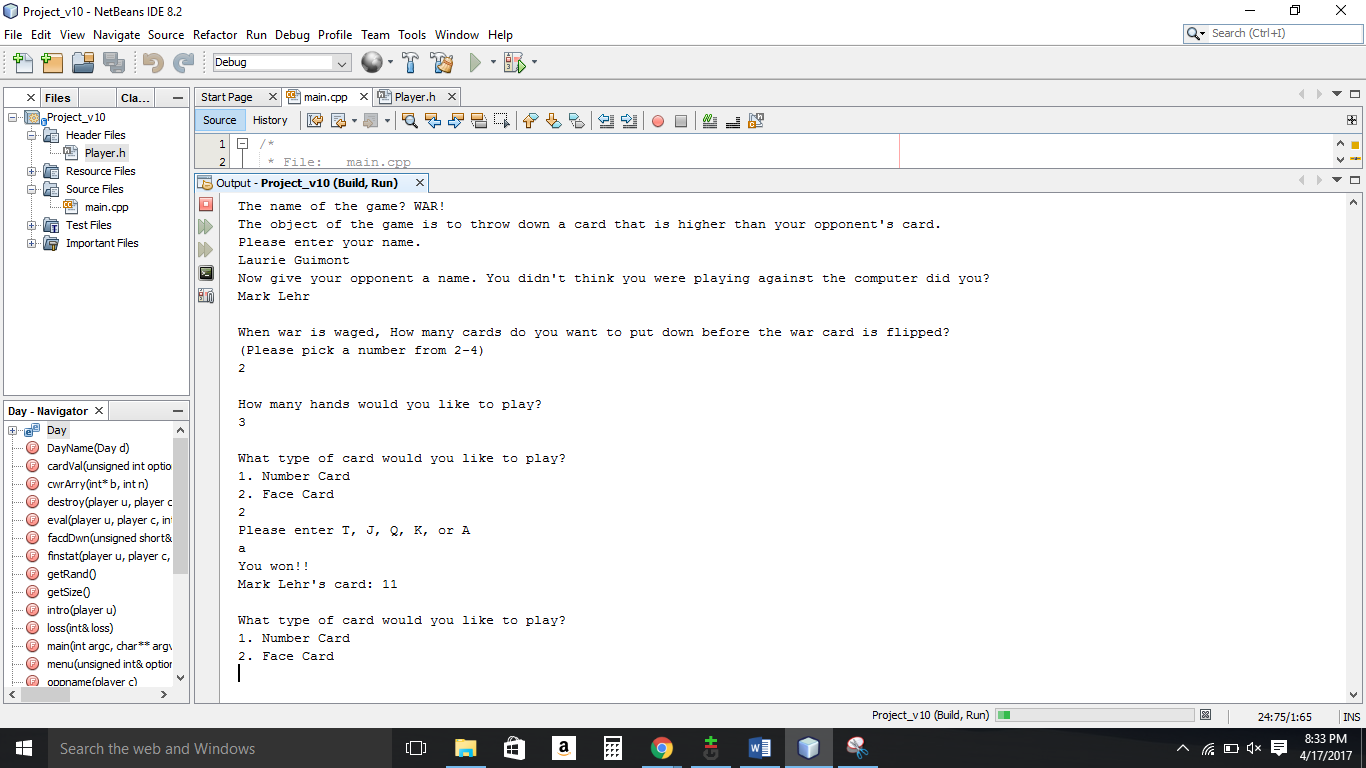
|  |  |  |  |
| --- | --- | --- | --- |
| Type | Name | Parameter Types | Features |
| player | intro | (player) | Structure Pass/Return |
| player | oppname | (player) | Structure Pass/Return |
| unsigned short | facdDwn | (unsigned short &) | Pass by Reference |
| const int | getSize | () | Default Parameters |
| unsigned int | menu | (unsigned int &) | Pass by Reference |
| char | pckCard | (unsigned int) | Pass By Value |
| int | cardVal | (unsigned int) | Pass By Value |
| int | getRand | () | Default Parameters |
| void | eval | (player,player,int &,int &,int &,int,const int, int &) | Pass by Reference, Pass by Value, Structure Pass |
| int | win | (int &) | Pass by Reference |
| int | loss | (int &) | Pass by Reference |
| void | war | (int &,int,player,player,int &,int &, int &) | Pass by Reference, Pass by Value, Structure Pass |
| int | warArry | (int \*,int) | Pointer pass, Value |
| int | cwrArry | (int \*,int) | Pointer pass, Value |
| int | sumArry | (int \*,int) | Pointer pass, Value |
| int | warCard | () | Default Parameters |
| void | stats | (player,player,cons tint) | Structure Pass, Pass by Value |
| void | prntAry | (player,const int) | Structure Pass, Value |
| void | prntBry | (player,cons tint) | Structure Pass, Value |
| void | destroy | (player,player) | Structure Pass |
| void | readldr | () | Default Parameters |
| void | finstat | (player,player,int,int,int,int,cons tint) | Structure Pass, Pass by Value |
| void | DayName | (Day) | Pass enums |

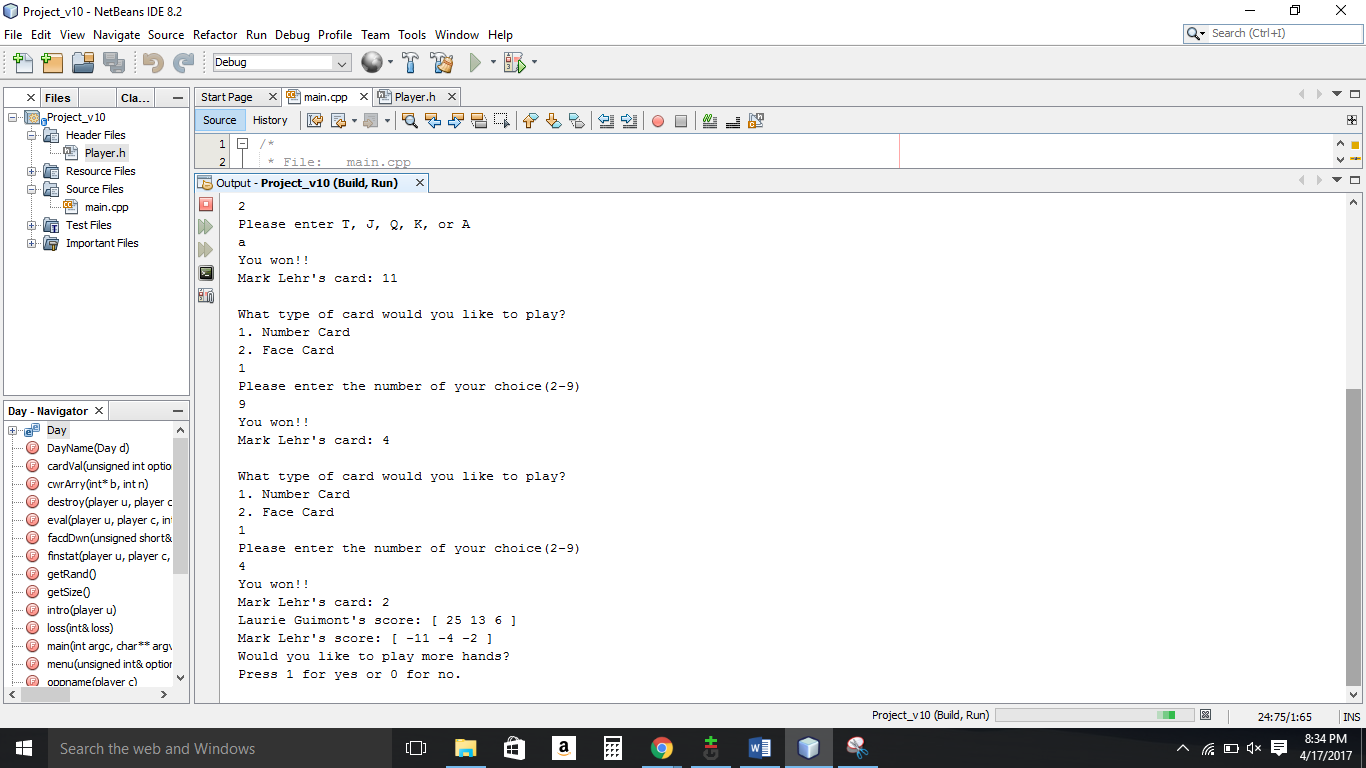
**Structure:**

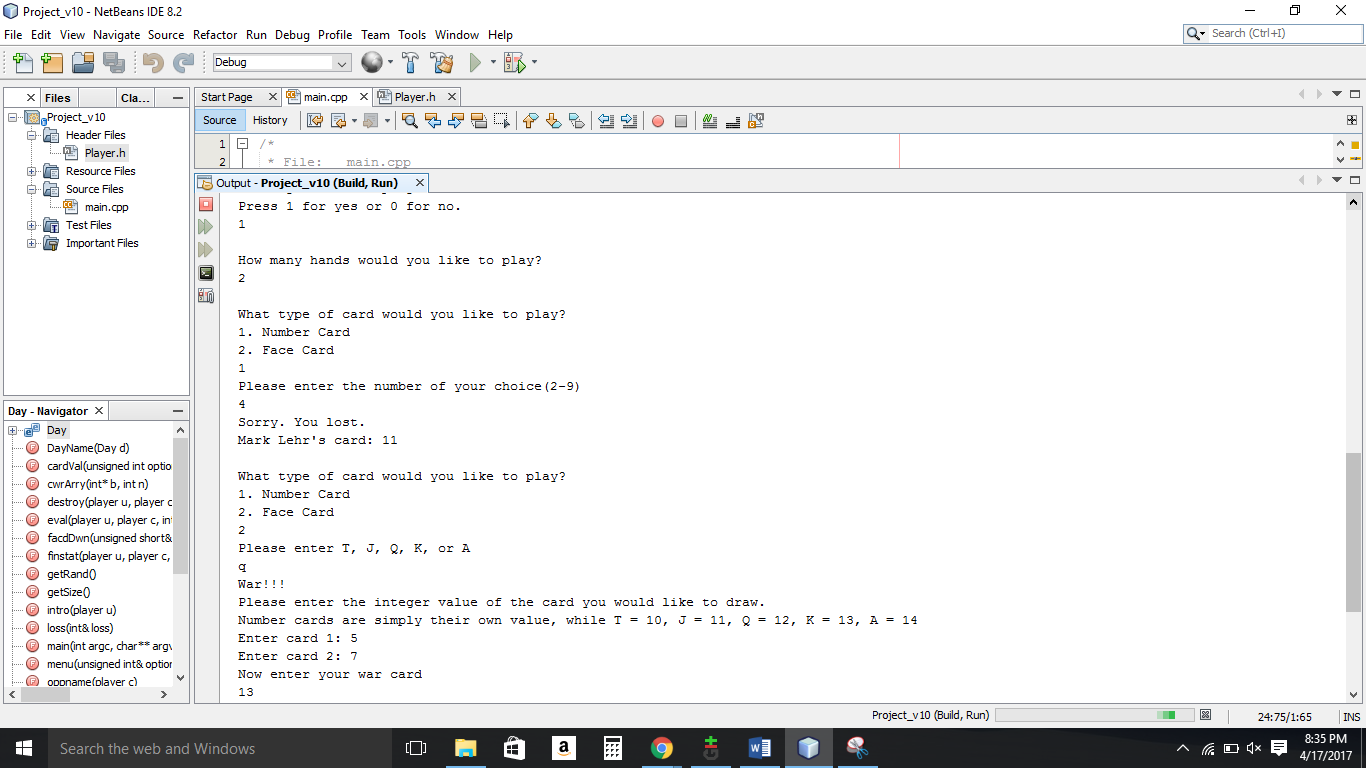
|  |  |  |  |
| --- | --- | --- | --- |
| User Library | Structure Name | Contents | Declaration Location |
| “Player.h” | player | int \*score, int wscore  char \*name, int card | Line 21 |

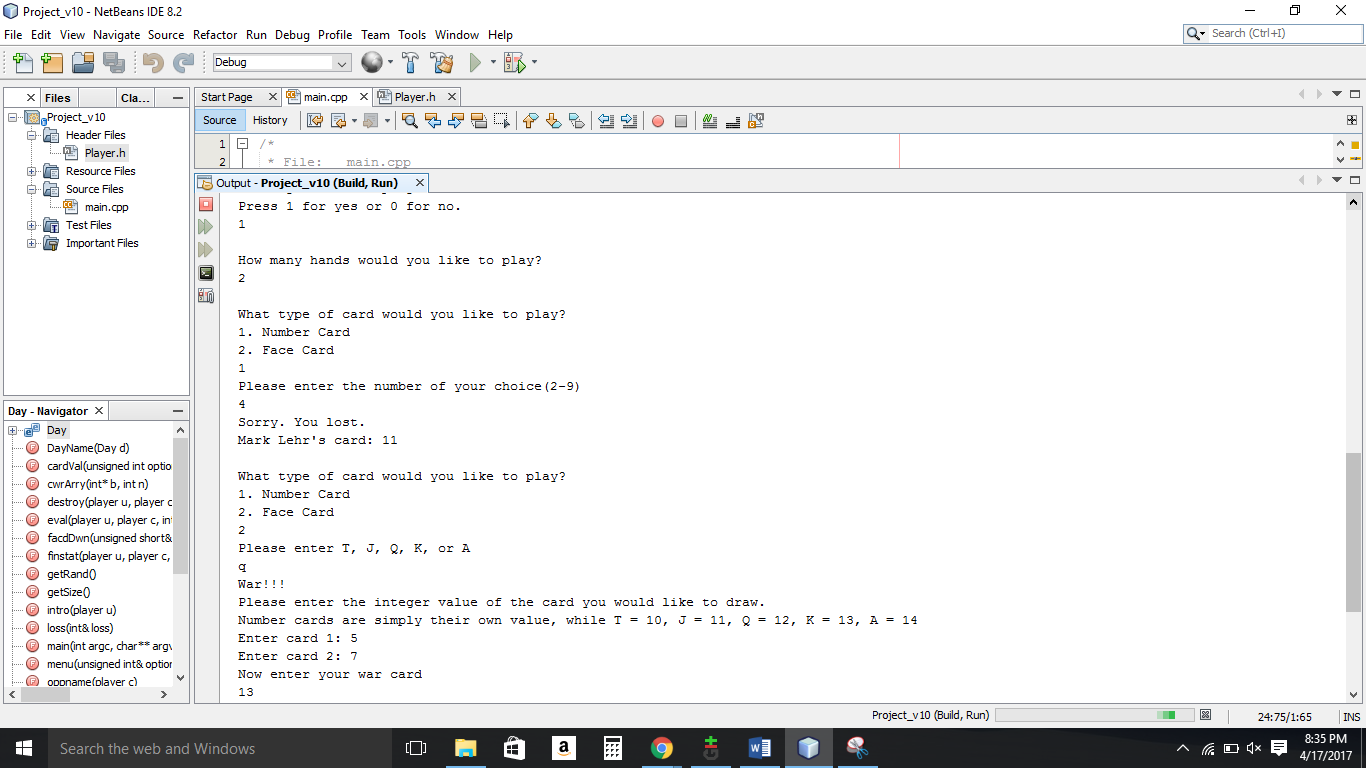
**Proof of a Working Product**

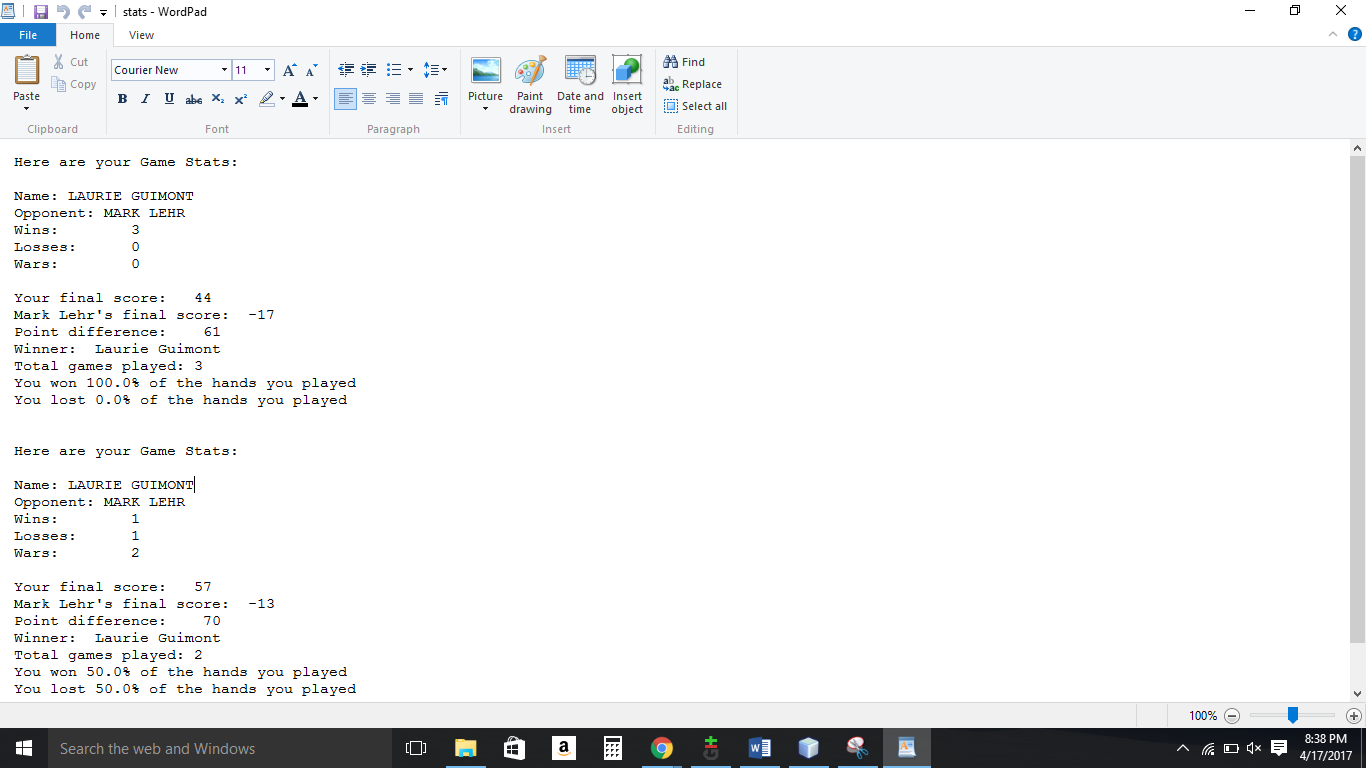
In the event, that my program does not work once it reaches Dr. Lehr, I have provided some screenshots that prove that the program did work at one time on the next few pages.

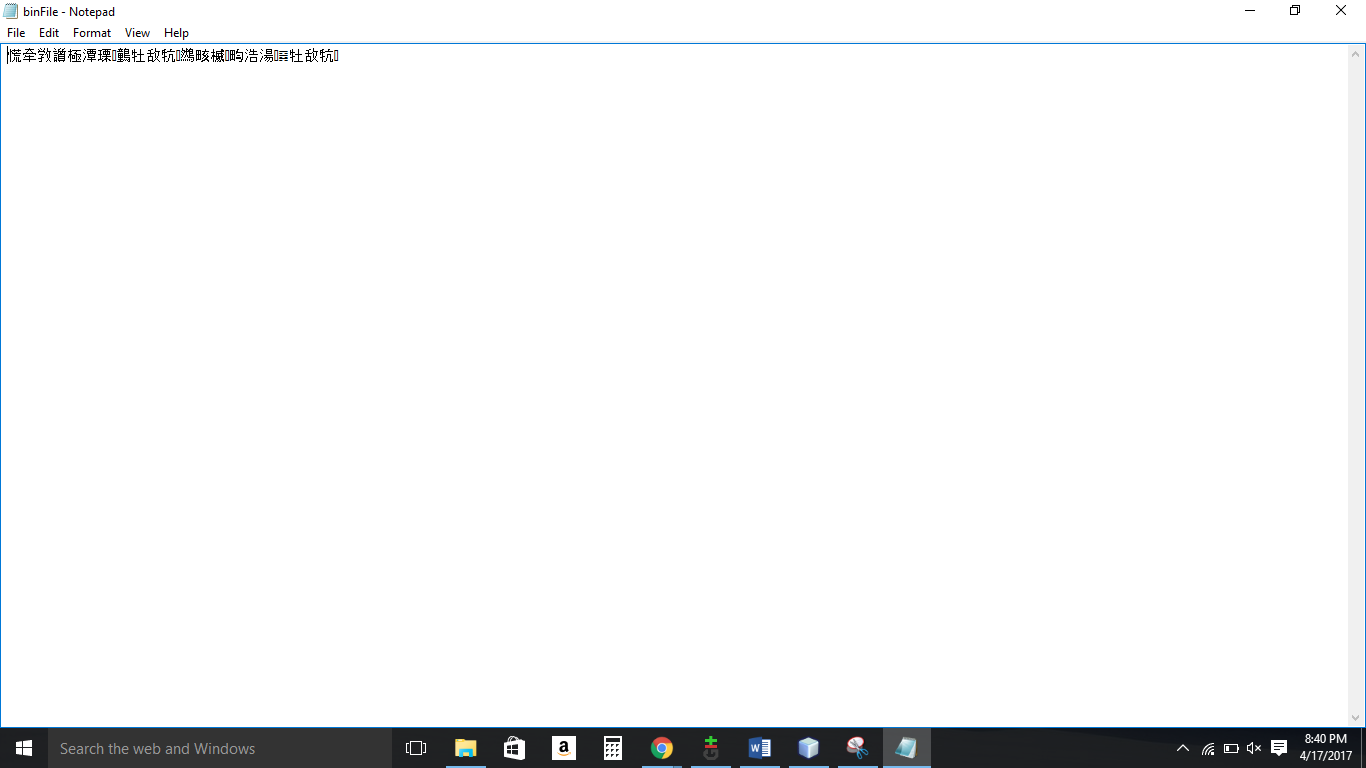












**References**

1. Dr. Lehr’s Lectures & Lab

2. “Starting Out with C++: From Control Structures through Objects” Gaddis,

Tony. 8th Edition. (Textbook)

3. [www.cplusplus.com](http://www.cplusplus.com) (only for the use of cin.clear();)

**Program**

/\*

\* File: main.cpp

\* Author: Laurie Guimont

\* Created on April 14, 2017, 10:30 PM

\* Purpose: War Card Game

\*/

#ifndef PLAYER\_H

#define PLAYER\_H

struct player{

int \*score;

int wscore;

char \*name;

int card;

};

#endif /\* PLAYER\_H \*/

/\*

\* File: main.cpp

\* Author: Laurie Guimont

\* Created on April 14, 2017, 10:30 PM

\* Purpose: War Card Game

\*/

//System Libraries

#include <iostream> //Input/Output Stream Library

#include <iomanip> //Formatting Library

#include <ctime> //Unique Seed Value Library

#include <cstdlib> //Random Value Library

#include <string> //String Library

#include <fstream> //File I/O Library

#include <cmath> //Math Library

#include <cctype> //Character Test Library

#include <cstring> //Character String Library

using namespace std;

//User Libraries

#include "Player.h"

//Global Constants

enum Day {MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY};

//Function Prototypes

player intro(player);

player oppname(player);

unsigned short facdDwn(unsigned short &);

const int getSize();

unsigned int menu(unsigned int &);

char pckCard(unsigned int);

int cardVal(unsigned int);

int getRand();

void eval(player,player,int &,int &,int &,int,const int,int &);

int win(int &);

int loss(int &);

void war(int &,int,player,player,int &,int &,int &);

int warArry(int \*,int);

int sumArry(int \*,int);

int warCard();

void stats(player,player,const int);

void prntAry(player,const int);

void prntBry(player,const int);

void destroy(player,player);

void readldr();

void finstat(player,player,int,int,int,const int);

void DayName(Day);

//Execution Begins Here!

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

//Set the Random Number Seed

srand(static\_cast<unsigned int>(time(0)));

//Declare Variables

unsigned short warcnt;

unsigned int choice; //The menu option that user chooses

player user,comp;

bool again;

//Get Name of Opponent & Establish Number of "Faced Down" Cards

user.name=new char[100];

comp.name=new char[100];

intro(user);

oppname(comp);

facdDwn(warcnt);

do{

int wins=0,losses=0,wars=0;

int count=0,i=0;

const int SIZE=getSize();

user.score=new int[SIZE];

comp.score=new int [SIZE];

do{

menu(choice);

user.card=cardVal(choice);

comp.card=getRand();

eval(user,comp,wins,losses,wars,warcnt,SIZE,i);

count++;

}while(count<SIZE);

//Finishing Stats

finstat(user,comp,wins,losses,wars,SIZE); //Output to File

stats(user,comp,SIZE); //Output to Screen

cout<<"Would you like to play more hands?"<<endl;

cout<<"Press 1 for yes or 0 for no."<<endl;

cin>>again;

}while(again==true);

//End Game & Show Leaderboard

cout<<"Thank you for playing!"<<endl<<endl;

readldr();

//Prevent Memory Leak

delete []user.name;

delete []comp.name;

//Exit Stage Right

return 0;

}

void finstat(player u,player c,int win,int lose,int war,const int SIZE){

//Declare Variables

int ngames;

float pwins,plosses;

int utotal=0,ctotal=0;

string winner;

fstream finish;

//Open & Write to file

finish.open("stats.dat",ios::app);

finish<<"Here are your Game Stats:"<<endl<<endl;

finish<<"Name: ";

for(int i=0;i<strlen(u.name);i++)

finish<<static\_cast<char>(toupper(\*(u.name+i)));

finish<<endl<<"Opponent: ";

for(int i=0;i<strlen(c.name);i++)

finish<<static\_cast<char>(toupper(\*(c.name+i)));

finish<<endl<<"Wins: "<<setw(4)<<win<<endl;

finish<<"Losses: "<<setw(4)<<lose<<endl;

finish<<"Wars: "<<setw(4)<<war<<endl;

finish<<endl;

//Calculate Total Scores

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

utotal+=u.score[i];

finish<<"Your final score: "<<setw(4)<<utotal<<endl;

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

ctotal+=c.score[i];

finish<<c.name<<"'s final score: "<<setw(4)<<ctotal<<endl;

finish<<"Point difference: "<<setw(4)<<abs(utotal-ctotal)<<endl;

//Determine Winner of Game

if(utotal>ctotal) winner=u.name;

else winner=c.name;

finish<<"Winner: "<<winner<<endl;

//Calculate Number of Games

ngames=win+lose;

finish<<"Total games played: "<<ngames<<endl;

//Calculate & Output Percentage of Wins & Losses

pwins=static\_cast<float>(win)/ngames;

plosses=static\_cast<float>(lose)/ngames;

finish<<fixed<<setprecision(1)<<showpoint;

finish<<"You won "<<pwins\*100<<"% of the hands you played"<<endl;

finish<<"You lost "<<plosses\*100<<"% of the hands you played"<<endl;

finish<<endl<<endl;

//Close file

finish.close();

}

void DayName(Day d){

switch(d){

case MONDAY: cout<<"Monday "; break;

case TUESDAY: cout<<"Tuesday "; break;

case WEDNESDAY:cout<<"Wednesday "; break;

case THURSDAY: cout<<"Thursday "; break;

case FRIDAY: cout<<"Friday "; break;

case SATURDAY: cout<<"Saturday "; break;

case SUNDAY: cout<<"Sunday "; break;

}

}

void readldr(){

//Declare the file

fstream leader;

string board;

//Open the file

leader.open("leaderboard.dat",ios::in);

//Send the array to the file

cout<<"Leaderboard:"<<endl;

if(leader){

getline(leader, board);

for(Day highest=MONDAY;highest<=SUNDAY;

highest=static\_cast<Day>(highest+1)){

DayName(highest);

cout<<":";

cout<<setw(6)<<board<<endl;

getline(leader, board);

}

//Close the file

leader.close();

}

else

cout<<"Error. Cannot open file"<<endl;

}

void destroy(player u,player c){

delete []u.score;

delete []c.score;

}

void prntBry(player b,const int n){

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

cout<<\*(b.score+i)<<" ";

}

void prntAry(player a,const int n){

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

cout<<\*(a.score+i)<<" ";

}

void stats(player u,player c,int size){

cout<<u.name<<"'s score: [ ";

prntAry(u,size);

cout<<"]"<<endl;

cout<<c.name<<"'s score: [ ";

prntBry(c,size);

cout<<"]"<<endl;

//Output to Binary File

fstream binFile;

binFile.open("binFile.bin",ios::binary|ios::app);

binFile.write(u.name,strlen(u.name));

binFile.write(reinterpret\_cast<char \*>(u.score),size);

binFile.write(c.name,strlen(c.name));

binFile.write(reinterpret\_cast<char \*>(c.score),size);

//Close Binary File

binFile.close();

//Prevent Memory Leak!

destroy(u,c);

}

int warCard(){

int number;

cout<<"Now enter your war card"<<endl;

cin>>number;

//Input Validation

while(!(number)||number<2||number>14){

cin.clear();

cin.ignore();

cout<<"Invalid input. Please type in an integer";

cout<<" between 2 and 14."<<endl;

cin>>number;

}

return number;

}

int sumArry(int \*a,int n){

int sum=0;

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

sum+=\*(a+i);

}

return sum;

}

int cwrArry(int \*b,int n){

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

\*(b+i)=getRand();

}

int ctotal=sumArry(b,n);

return ctotal;

}

int warArry(int \*a,int n){

cout<<"War!!!"<<endl;

cout<<"Please enter the integer value of the card you ";

cout<<"would like to draw."<<endl;

cout<<"Number cards are simply their own value, while T = 10,"

" J = 11, Q = 12, K = 13, A = 14"<<endl;

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

cout<<"Enter card "<<i+1<<": ";

cin>>\*(a+i);

//Validate

while(!(\*(a+i))||\*(a+i)<2||\*(a+i)>14){

cin.clear();

cin.ignore();

cout<<"Invalid input. Please type in an integer between 2 and 14.";

cin>>\*(a+i);

}

}

int total=sumArry(a,n);

return total;

}

void war(int &wars,int count,player u,player c,int &won,int &lost,int &i){

wars++;

int warnum,cwarnum;

//Declare War Array

const int SIZE=count;

int urray[SIZE],crray[SIZE];

u.wscore=warArry(urray,SIZE);

warnum=warCard();

c.wscore=cwrArry(crray,SIZE);

cwarnum=getRand();

if(warnum>cwarnum){

win(won);

u.score[i]=u.card+c.card+u.wscore+c.wscore+warnum+cwarnum;

c.score[i]=0-c.card-c.wscore-cwarnum;

return;

}

else if(warnum<cwarnum){

loss(lost);

u.score[i]=0-u.card-u.wscore-warnum;

c.score[i]=c.card+u.card+c.wscore+u.wscore+cwarnum+warnum;

return;

}

else{

while(warnum==cwarnum){ //Must War Again!

wars++;

int warnum,cwarnum;

//Declare War Array

const int SIZE=count;

int urray[SIZE],crray[SIZE];

u.wscore=warArry(urray,SIZE);

warnum=warCard();

c.wscore=cwrArry(crray,SIZE);

cwarnum=getRand();

if(warnum>cwarnum){

win(won);

u.score[i]=u.card+c.card+u.wscore+c.wscore+warnum+cwarnum;

c.score[i]=0-c.card-c.wscore-cwarnum;

return;

}

else if(warnum<cwarnum){

loss(lost);

u.score[i]=0-u.card-u.wscore-warnum;

c.score[i]=c.card+u.card+c.wscore+u.wscore+cwarnum+warnum;

return;

}

}

}

}

int loss(int &loss){

loss++;

cout<<"Sorry. You lost."<<endl;

return loss;

}

int win(int &win){

win++;

cout<<"You won!!"<<endl;

return win;

}

void eval(player u,player c,int &won,int &lost,int &wars,int wcount,

const int n,int &i){

if(u.card>c.card){

win(won);

u.score[i]=u.card+c.card;

c.score[i]=0-c.card;

cout<<c.name<<"'s card: "<<c.card<<endl;

}

else if(u.card<c.card){

loss(lost);

u.score[i]=0-u.card;

c.score[i]=c.card+u.card;

cout<<c.name<<"'s card: "<<c.card<<endl;

}

else{

war(wars,wcount,u,c,won,lost,i);

}

i++;

}

int getRand(){

//Declare Variables

const int MIN=2;

const int MAX=14;

int oppCard;

return oppCard=rand()%(MAX-MIN+1)+MIN;

}

int cardVal(unsigned int option){

//Get Card Choice

char cchoice=pckCard(option);

int number;

//Evaluate Card Choice

switch(cchoice){

case 'a':

case 'A':number=14;break;

case 'k':

case 'K':number=13;break;

case 'q':

case 'Q':number=12;break;

case 'j':

case 'J':number=11;break;

case 't':

case 'T':number=10;break;

case '9':

case '8':

case '7':

case '6':

case '5':

case '4':

case '3':

case '2':number=(cchoice-48);break;

}

return number;

}

char pckCard(unsigned int option){

//Declare Variable

char card;

if(option==1){

cout<<"Please enter the number of your choice(2-9)"<<endl;

cin>>card;

//Validate

while(card<'2'||card>'9'){

cout<<"Invalid entry! Please enter (2-9)"<<endl;

cin>>card;

}

}

else if(option==2){

cout<<"Please enter T, J, Q, K, or A"<<endl;

cin>>card;

//Validate

while(toupper(card)!='A'&&toupper(card)!='K'&&toupper(card)!='Q'&&

toupper(card)!='J'&&toupper(card)!='T'){

cout<<"Invalid entry! Please enter one of the choices above"<<endl;

cin>>card;

}

}

return card;

}

unsigned int menu(unsigned int &option){

cout<<endl;

cout<<"What type of card would you like to play?"<<endl;

cout<<"1. Number Card"<<endl;

cout<<"2. Face Card"<<endl;

cin>>option;

return option;

}

const int getSize(){

//Get size of score array

int size;

cout<<endl;

cout<<"How many hands would you like to play?"<<endl;

cin>>size;

return size;

}

unsigned short facdDwn(unsigned short &number){

cout<<endl;

cout<<"When war is waged, How many cards do you want to put down before ";

cout<<"the war card is flipped?"<<endl;

cout<<"(Please pick a number from 2-4)"<<endl;

cin>>number;

//Validation

while(number<2||number>4){

cout<<"Error. Please enter 2, 3 or 4"<<endl;

cin>>number;

}

return number;

}

player oppname(player c){

cout<<"Now give your opponent a name. You didn't think you were playing ";

cout<<"against the computer did you?"<<endl;

cin.getline(c.name,100);

return c;

}

player intro(player u){

cout<<"The name of the game? WAR!"<<endl;

cout<<"The object of the game is to throw down a card that is higher ";

cout<<"than your opponent's card."<<endl;

cout<<"Please enter your name."<<endl;

cin.getline(u.name,100);

return u;

}