War/I Declare War
Card Game

Laurie Guimont

CSC 5

Summer 2016

45276

TABLE OF CONTENTS

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.

3. 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."

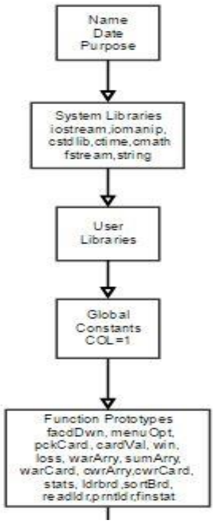
Finally, a player wins or loses based on their score at the time they decide to finish the game. If the player is tired of playing after a while, they can exit the game and the computer will tell them their final score. If the score is higher than the computer's, they have one. If not, they have lost.

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: http://www.gliffy.com/go/publish/10930307

```
Put in opening comments

Bring in 7 system libraries

Declare global constant (to be used

        For 2-Dimensional array ONLY)

Bring in all 17 function prototypes
```

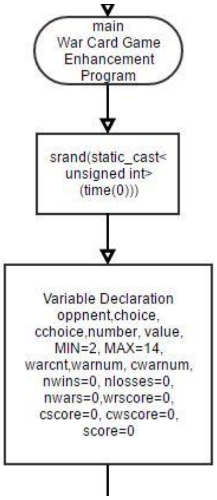


Enter main, then immediately

set random number seed

Declare all variables, initiate some

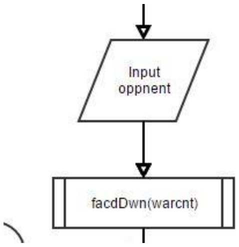
now and some later.



Input opponent name

Call facdDwn function and pass

warcnt in



Enter facdDwn function and prompt

user to enter a number to be used

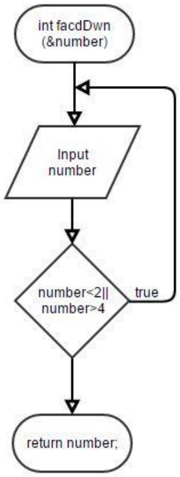
for warcnt variable

Number must be 2,3, or 4

Validate the input with while loop

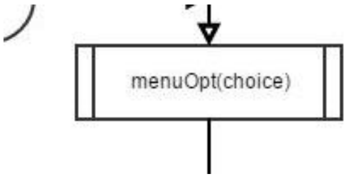
Return the number back to main

Pass by reference



Return from facdDwn and call

menuOpt function now



Enter menuOpt

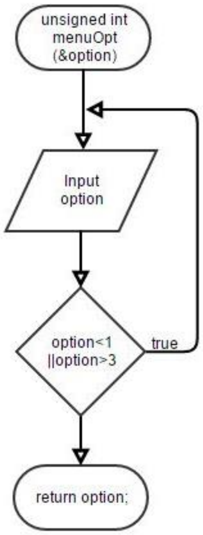
Prompt to input valid menu choice.

Option must be 1,2, or 3

Verify valid data with while loop again

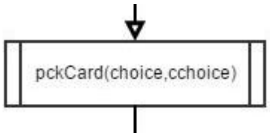
Return option back to main

Pass by reference



Return from menuOpt and call

pckCard function



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

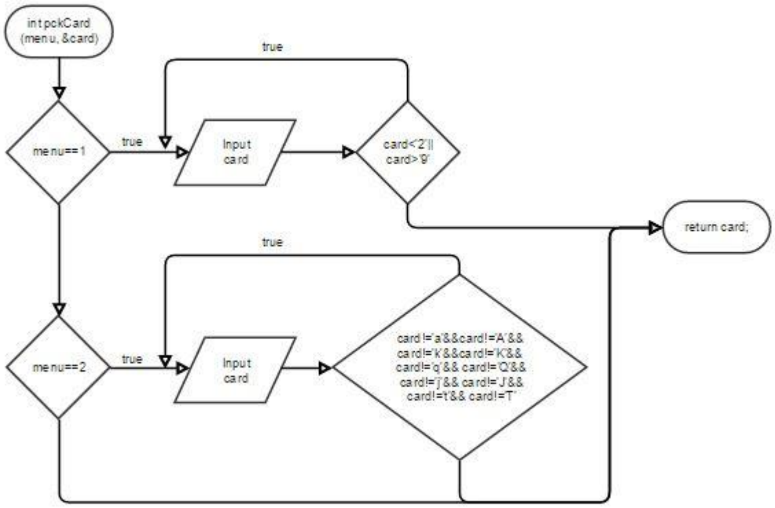
Verify that all data is valid with while

loops

Return card if (1) or (2)

Pass by reference

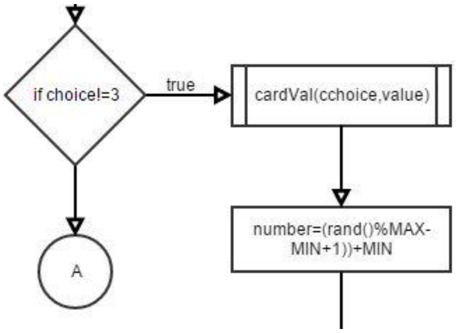
Else, simply return


int pckCard (menu, &card)
menu==1
true
Input card
true
card<'2'|| card>'9'
return card;
true
menu==2
true
Input card
card!='a'&&card!='A'&& card!='k'&&card!='K'&& card!='q'&& card!='Q'&& card!='j'&& card!='J'&& card!='t'&& card!='T'

As long as “End Menu” (3) is not chosen,

call cardVal function

Else game will end.



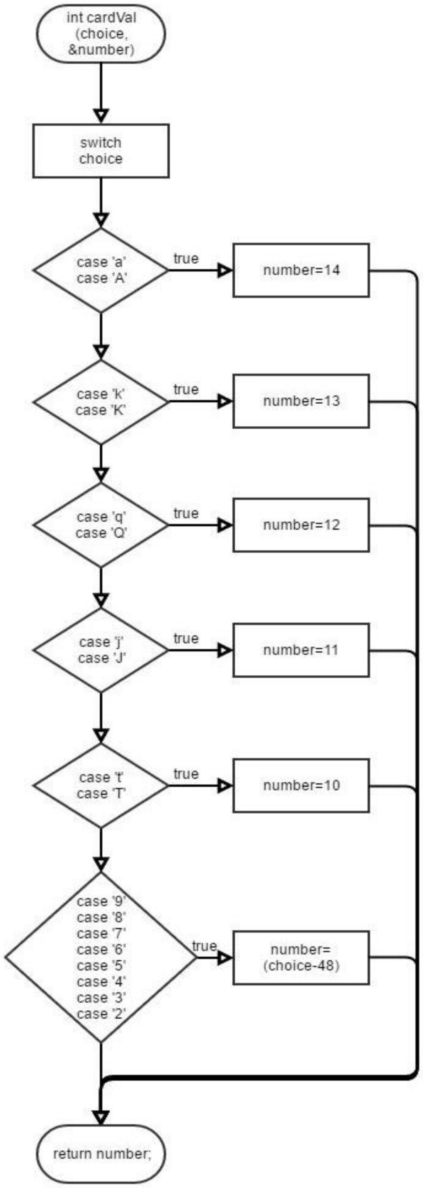
Enter cardVal and perform switch

to evaluate card value

Number will be assigned based on card chosen

Return number back to main

Number passed by reference



Return from cardVal

Computer chooses a random number

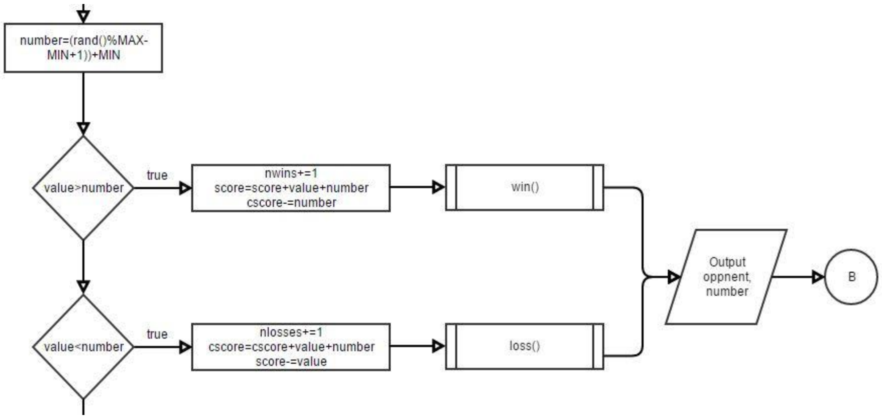
and compares number to

input value

If value is bigger than random number

User wins round and

Score is calculated



Call win function and output message

Default parameters used

Return to main

If value is smaller, computer wins.

Computer wins round

Score is calculated

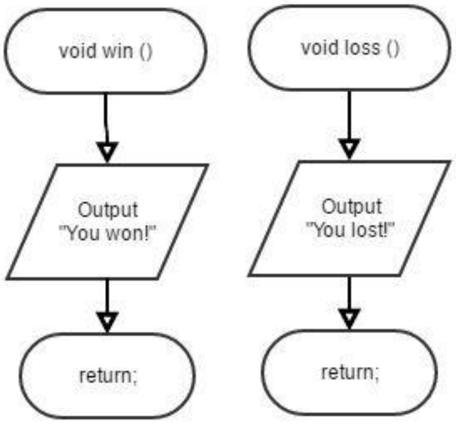
Call loss function and output message

Default parameters used

Return to main

Output opponent name and number

Game results will be displayed



Else if both numbers are equal

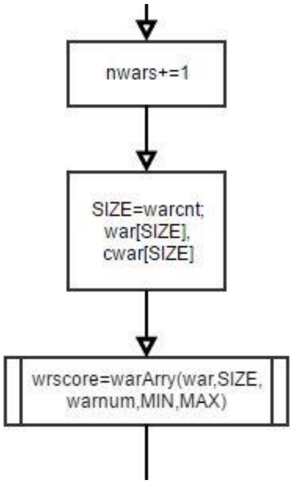
User has entered war

Declare more variables for arrays to be

Used

wrscore is equal to the value returned by

warArry function



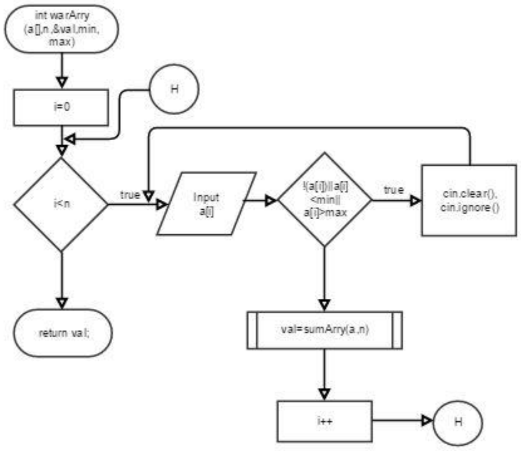
Enter warArry function

Prompt user to enter face down cards according to initial input at start of the game using for loop

Validate with while loop

val equals the value returned from sumArray function

Call function


int warArry (a[],n,&val,min, max)
i=0
H
i<n
true
Input a[i]
!(a[i])||a[i] <min|| a[i]>max
true
cin.clear(), cin.ignore()
return val;
val=sumArry(a,n)
i++
H

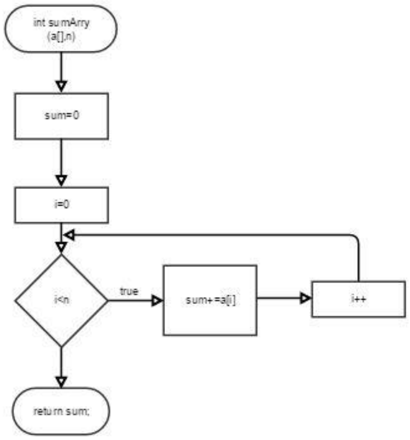
Enter sumArry and set sum accumulator equal to zero

Add the values from warArry and pass sum back by value

Return to warArry

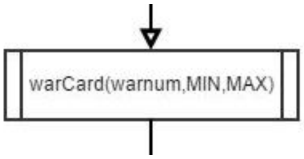
Return val to main from warArry

Pass by value



Return from warArry

Call warCard function



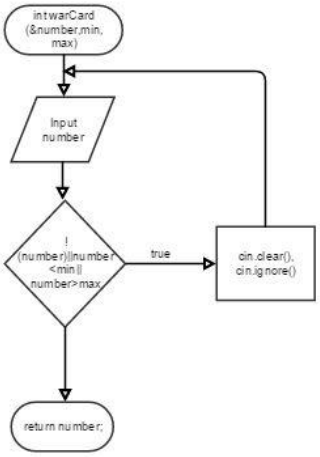
Enter warCard function

Prompt user to enter number for war card

Validate with while loop

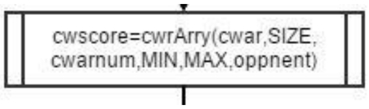
Return number to main

Pass by reference


intwarCard (&number,min, max)
Input number
!(number)||number <min|| number>max
true
cin.clear(), cin.ignore()
return number;

Return from warCard

Call cwscore function



Enter cwrArry function

Output opponent name

Computer now chooses faced down cards

(same number as indicated in

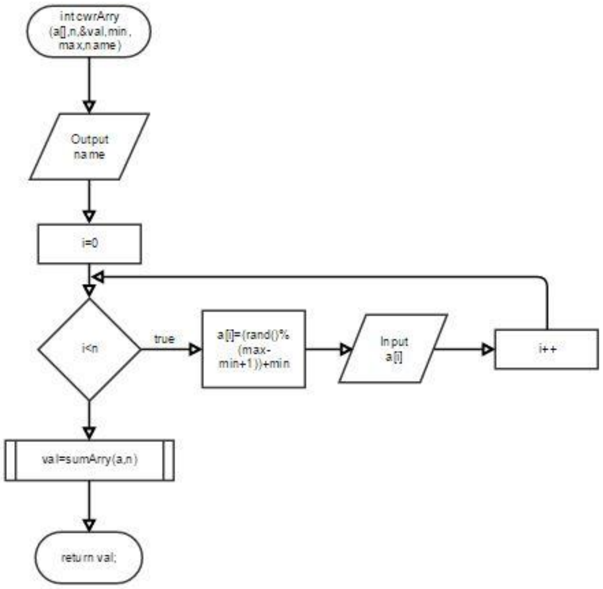
facdDwn)

Value of these cards are summed after a

call to sumArry is made once again

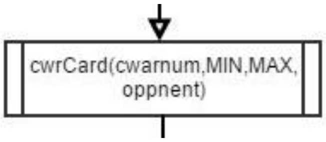
Return val to main

Pass by value



Return from cwrArry

Call cwrCard function



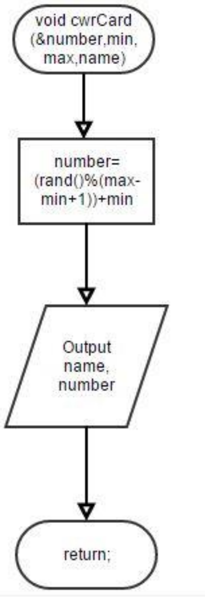
Enter cwrCard

Computer randomly chooses war card

Output opponent name and the random

number

Return to main



Return from cwrCard

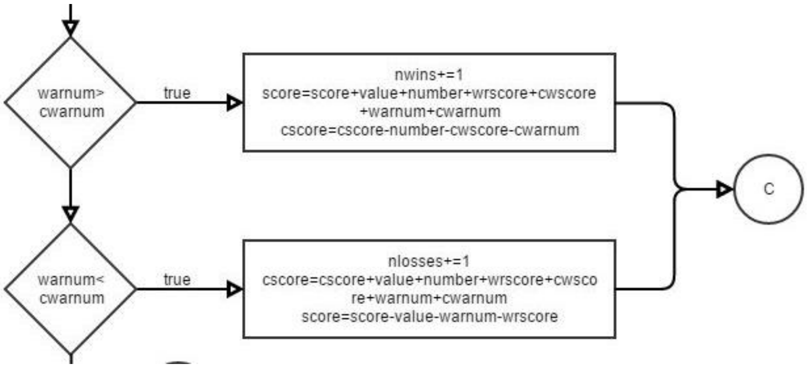
War Card comparison is made

whoever has the higher

number, wins

Loser loses points

Game results will be displayed

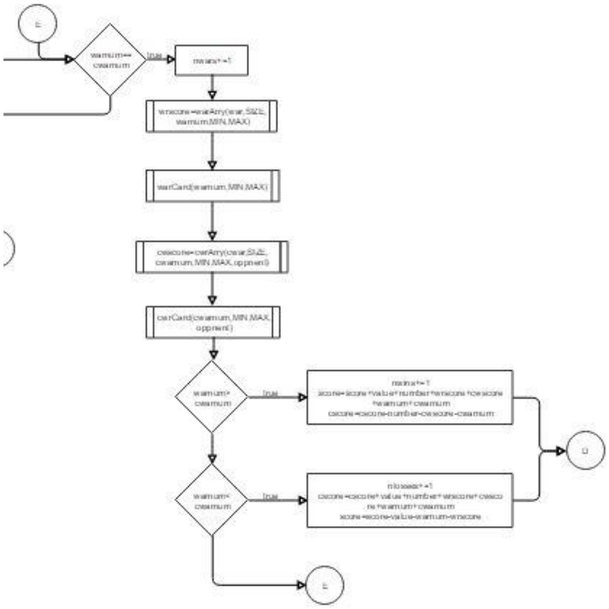

warnum>
cwarnum
true
nwins+=1
score=score+value+number+wrscore+cwscore
+warnum+cwarnum
cscore=cscore-number-cwscore-cwarnum
C
warnum<
cwarnum
true
nlosses+=1
cscore=cscore+value+number+wrscore+cwsco
re+warnum+cwarnum
score=score-value-warnum-wrscore

If warnum equals cwarnum

repeat entire war procedure

Loop this step until someone has won

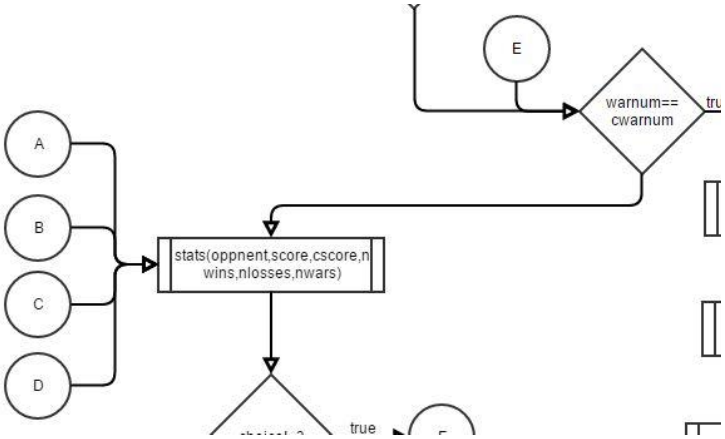
the war



Game stats are kept during gameplay.

After each round of win, loss or war,

call stats function



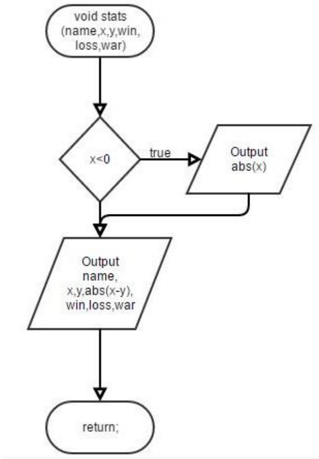
Enter stats function

If score (represented by x) is less than 0,

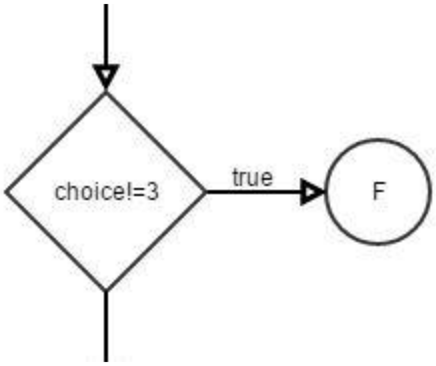
output message that indicates how many more points until score is positive (or 0)

Output number of wins, losses, wars, and point difference between user and opponent

Return to main

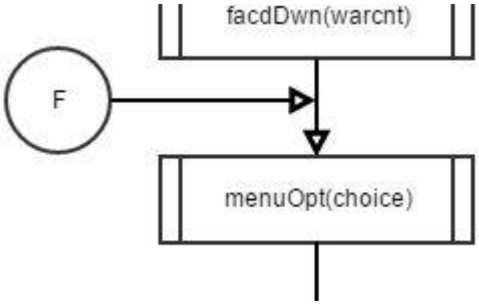

void stats (name,x,y,win, loss,war)
x<0
true
Output abs(x)
Output name, x,y,abs(x-y), win,loss,war
return;

While "End Program"(3) is not selected,



Do process over again starting by

calling menuOpt once again.

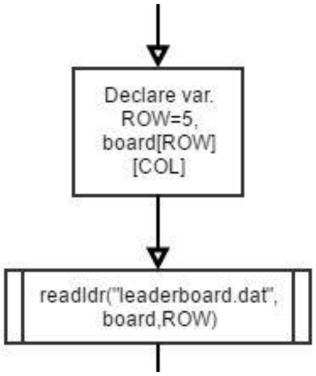


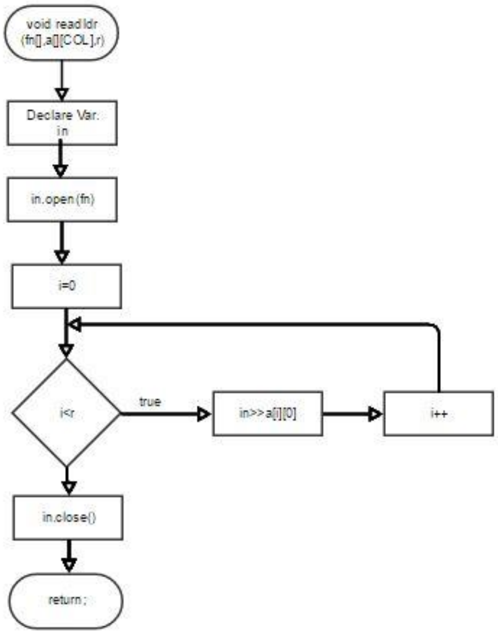
Else, thank user for playing and read in a

Leaderboard file to display.

Declare more variables for this 2D array

Call readldr function


Declare var. ROW=5, board[ROW] [COL]
readldr("leaderboard.dat", board,ROW)



Enter readldr function

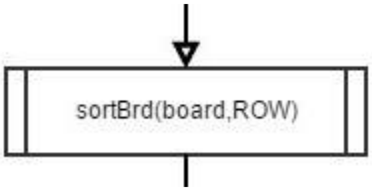
Declare Variable in

Open the file

Read in the file with a for loop

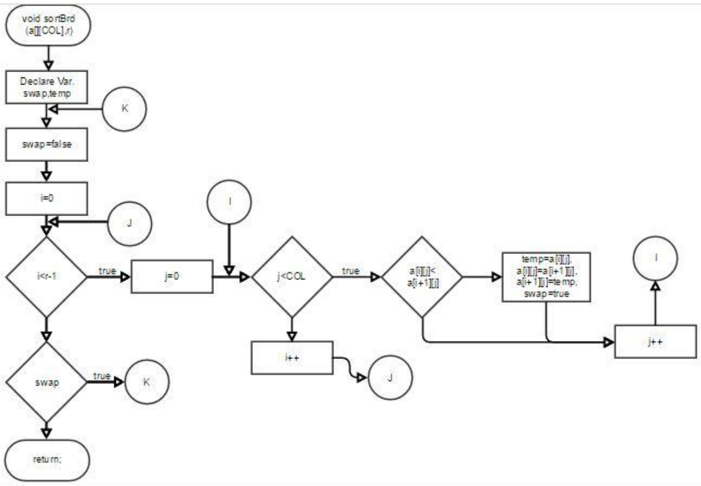
Close the file

Return to main



Return from readldr function

Call sortBrd function since the file has not been sorted



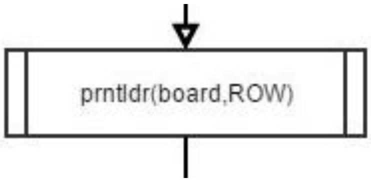
Enter sortBrd function

Declare variables to run sort procedure

Perform sort using bubble sort method using nested for loops

Finishing swaping/sorting

Return to main



Return from sortBrd

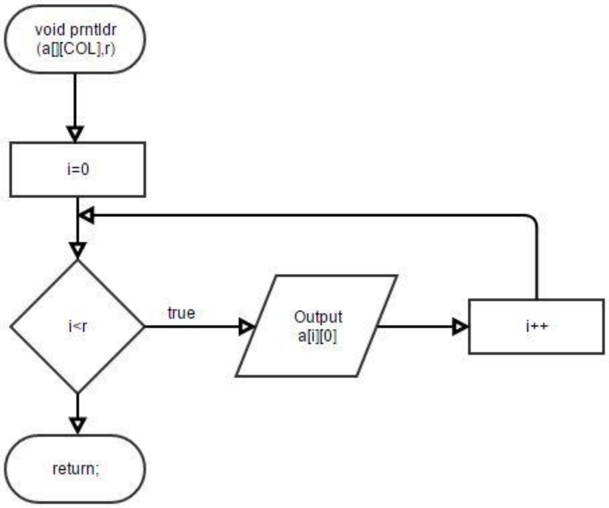
Call prntldr function to display board

Enter prntldr function

Print sorted integer values to the screen

using for loop

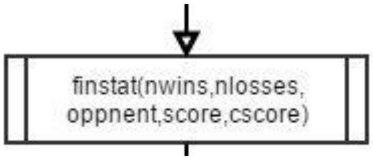
Return to main



Return from prntldr function

Call finstat function to write

finishing stats to an output file.



Enter finstat function

Declare variables

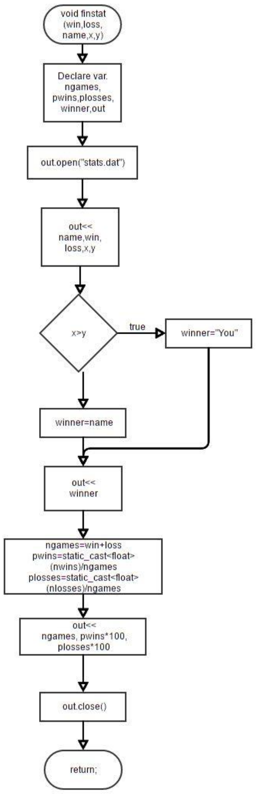
Open the file

Write wins, losses, opponent name, winner,

percentage of wins & losses to file

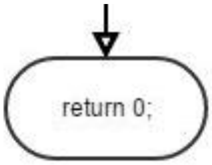
Close the file

Return to main



Return from finstat

Return 0; the program is complete



Constructs & Concepts Utilized

iostream Library

Name	Frequency	Description	Location
static_cast	3	Statically cast as different variable	Line 44,203,204
cout	54	Output Data	Throughout
cin	13	Input Data	Throughout
getline()	1	Reads string data	Line 62
cin.ignore()	2	Prevented input problems	Line 292,335
cin.clear()	2	Stopped infinite loop	Line 291,334

cstdlib Library

Name	Frequency	Description	Location
srand()	1	Random # seed	Line 44
rand()	3	Generates rand #	Line 79,279,312

ctime Library

Name	Frequency	Description	Location
time	1	Set current time	Line 44

iomanip Library

Name	Frequency	Description	Location
fixed	1	Format final game stats	Line 207
setprecision()	1	Format final game stats	Line 207
showpoint	1	Format final game stats	Line 207
setw()	11	Format final game stats	Line 185,186,188,189 254,261-263,271-273

string Library

Name	Frequency	Description	Location
string	10	Declare var./parameters	Line 33,34,35,39 46,174,178,259, 277,308

getline()	already mentioned	already mentioned	already mentioned

<u>cmath Library</u>

Name	Frequency	Description	Location
abs()	2	Neg. Score Alert Point Difference	Line 263,267

<u>fstream Library</u>

Name	Frequency	Description	Location
out.open()	1	Open file	Line 182
out.close()	1	Close file	Line 212
in.open()	1	Open file	Line 242
in.close()	1	Close file	Line 248
out	12	Write to file	Line 183-189, 196,200,207-209
in	1	Read in file	Line 245
ofstream	1	Declare var.	Line 179
ifstream	1	Declare var.	Line 240

Data Types:

Data Types	Frequency	Location
int	96	throughout
unsigned int	13	Line 25,39,44,47,55, 174,176,407
char	7	Line 26,27,36,48 238,356,380
string	10	Already mentioned
float	4	Line 177,203,204
ofstream	1	Line 179
ifstream	1	Line 240
bool	1	Line 218

Conditional Statements:

Conditional Statement	Frequency	Starting Location
if	3	Line 74,225,265
if/else	1	Line 192
if/else if	4	Line 80,110,131,381

switch	1	Line 357

Loops:

Loops	Frequency	Starting Location
for	4	Line 223,224,244,253, 302,311,328,
while	7	Line 123,290,333, 386,396,416,430
do-while	2	Line 68,221

Function Prototypes:

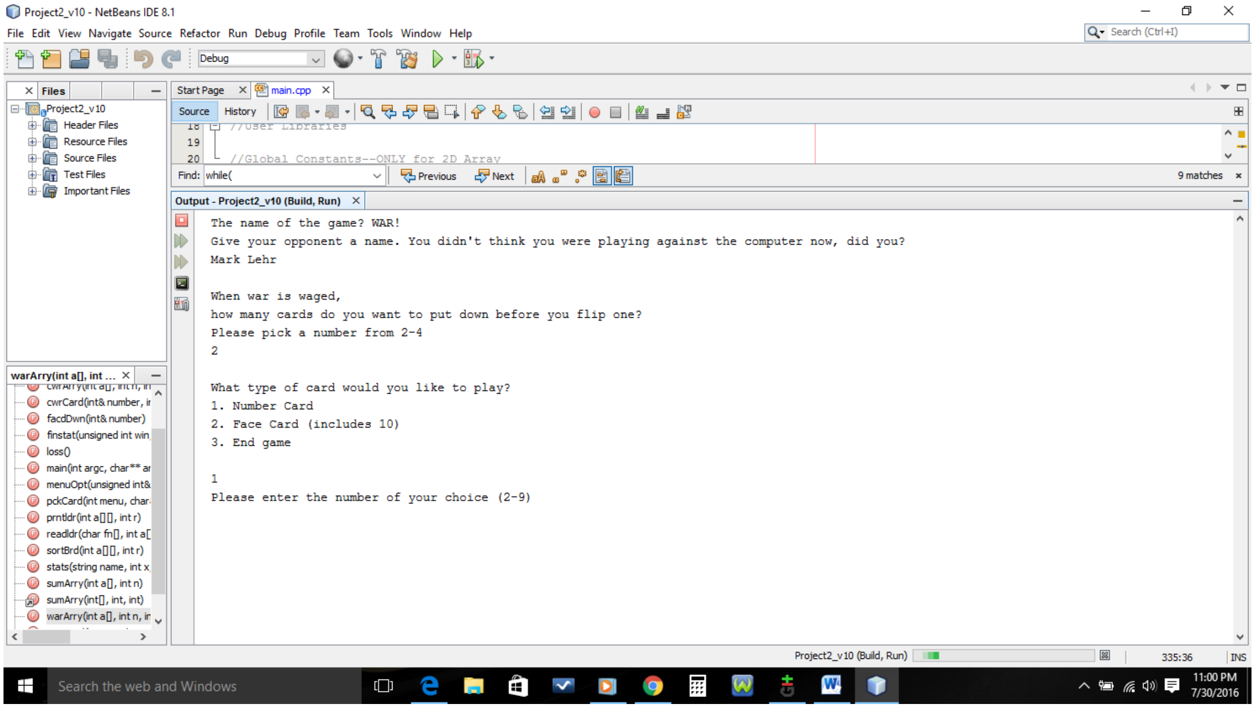
Type	Name	Parameter Types	Features
int	facdDwn	(int &)	Reference
unsigned int	menuOpt	(unsigned int &)	Reference
int	pckCard	(int, char &)	Value, Reference
int	cardVal	(char,int &)	Value, Reference
void	win	()	Default Parameters
void	loss	()	Default Parameters
int	warArry	(int [],int,int &,int,int)	1D Array, Reference, Value
int	sumArry	(int[],int,int)	1D Array, Value
int	warCard	(int &,int,int,string)	Reference, Value
int	cwrArry	(int [],int,int &,int,int,string)	1D Array, Value, Reference
void	cwrCard	(int &,int,int,string)	Reference
void	stats	(string,int,int,int,int,int)	Value
void	readldr	(char [],int,[][COL],int)	2D Array
void	sortBrd	(int [][COL],int)	2D Array
void	prntldr	(int[][COL],int)	2D Array
void	finstat	(unsigned int,unsigned int,string,int,int)	Value

**NOTE: Only for the purpose of the 2-Dimensional Arrays did I use a Global Variable!

const int COL=1

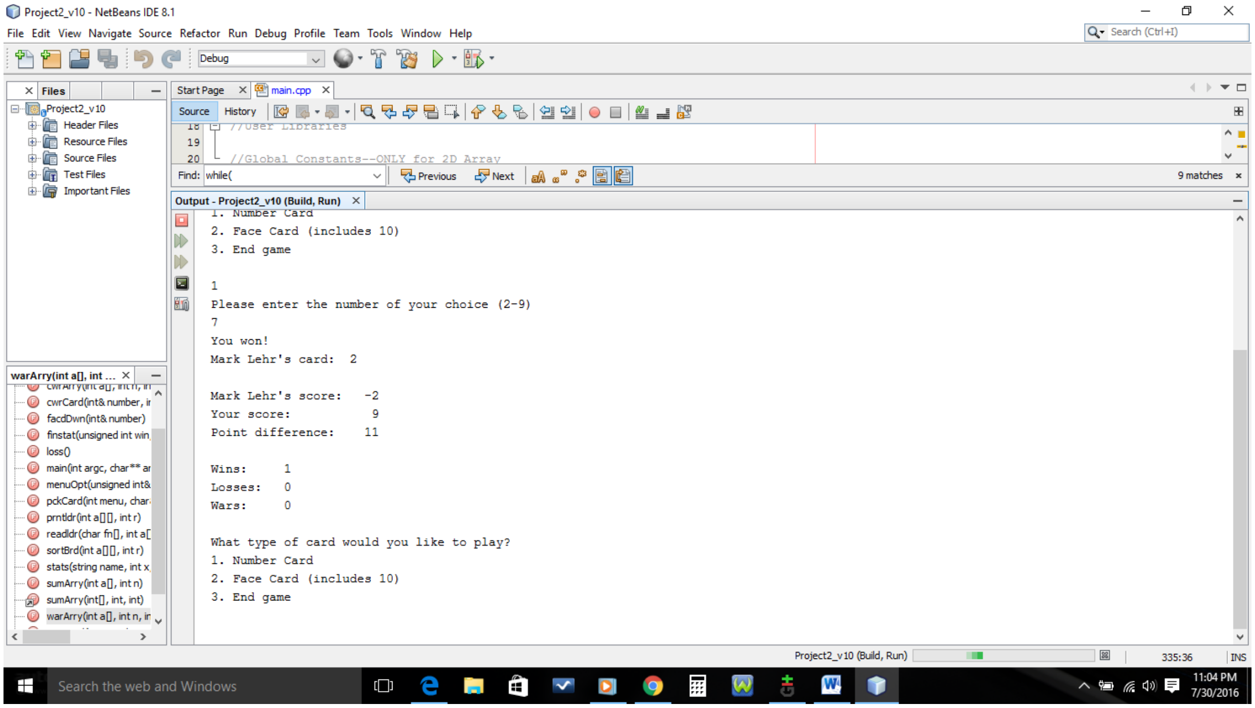
Proof of a Working Product

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.



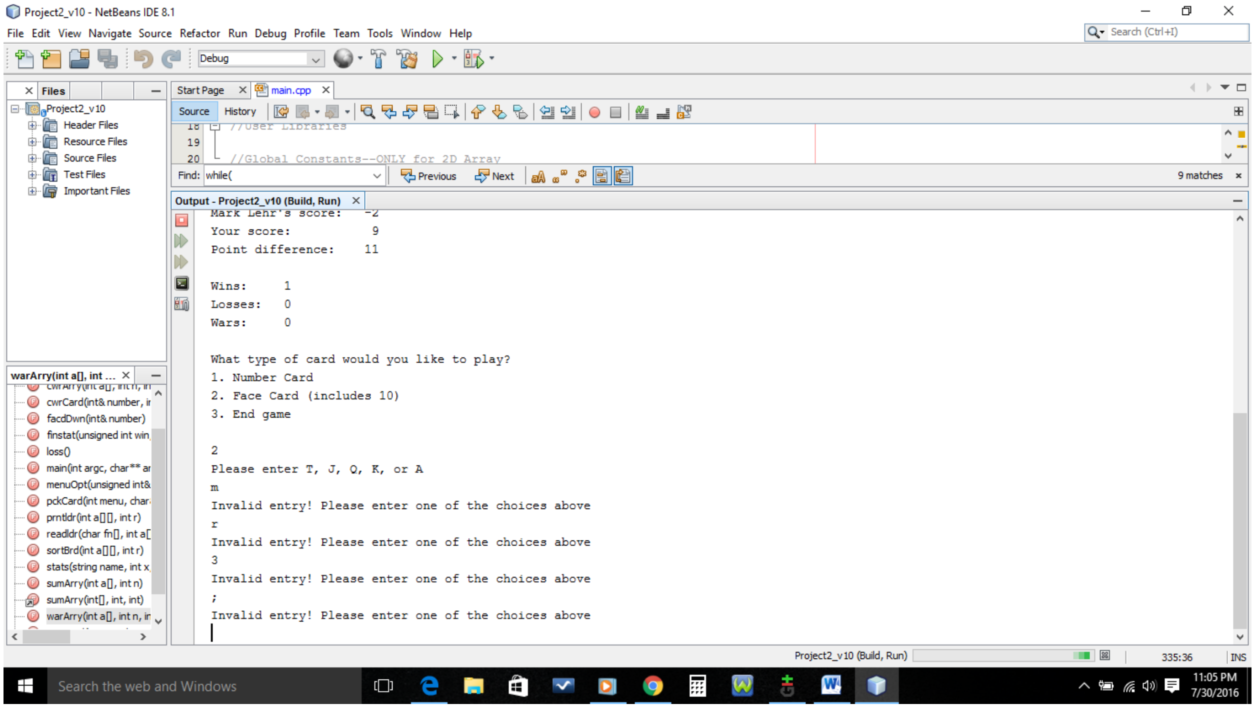
```
//User Libraries

//Global Constants--ONLY for 2D Array
```

```
The name of the game? WAR!
Give your opponent a name. You didn't think you were playing against the computer now, did you?
Mark Lehr

When war is waged,
how many cards do you want to put down before you flip one?
Please pick a number from 2-4
2

What type of card would you like to play?
1. Number Card
2. Face Card (includes 10)
3. End game

1
Please enter the number of your choice (2-9)
```



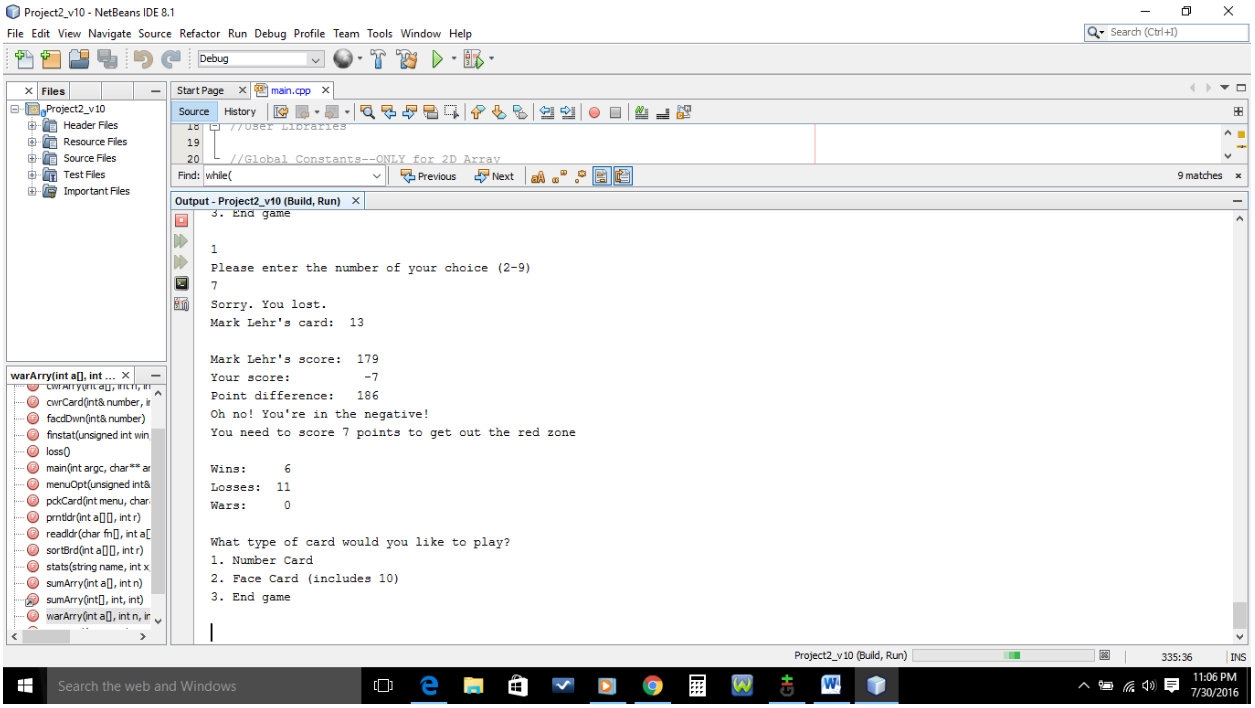
```
//User Libraries

//Global Constants--ONLY for 2D Array
```

```
1. Number Card
2. Face Card (includes 10)
3. End game

1
Please enter the number of your choice (2-9)
7
You won!
Mark Lehr's card:  2

Mark Lehr's score:   -2
Your score:           9
Point difference:    11

Wins:     1
Losses:   0
Wars:     0

What type of card would you like to play?
1. Number Card
2. Face Card (includes 10)
3. End game
```



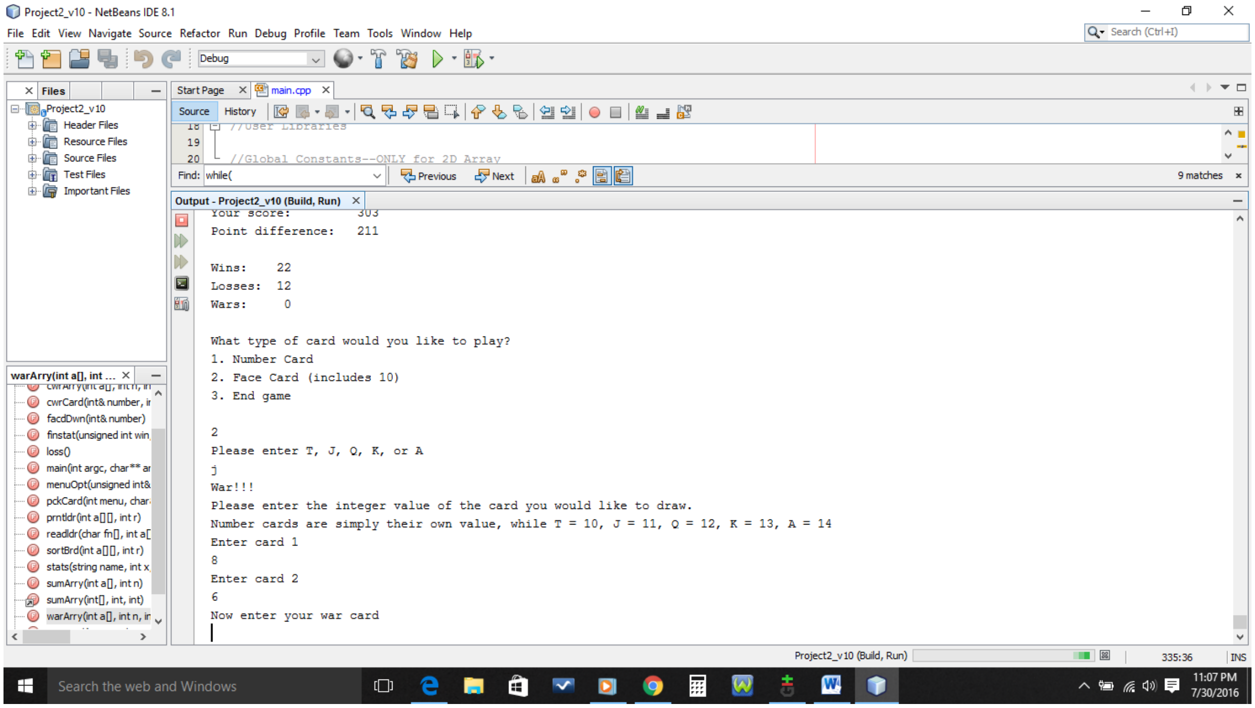
```
//User Libraries

//Global Constants--ONLY for 2D Array
```

```
Mark Lehr's score:   -2
Your score:           9
Point difference:    11

Wins:      1
Losses:    0
Wars:      0

What type of card would you like to play?
1. Number Card
2. Face Card (includes 10)
3. End game

2
Please enter T, J, Q, K, or A
m
Invalid entry! Please enter one of the choices above
r
Invalid entry! Please enter one of the choices above
3
Invalid entry! Please enter one of the choices above
;
Invalid entry! Please enter one of the choices above
```

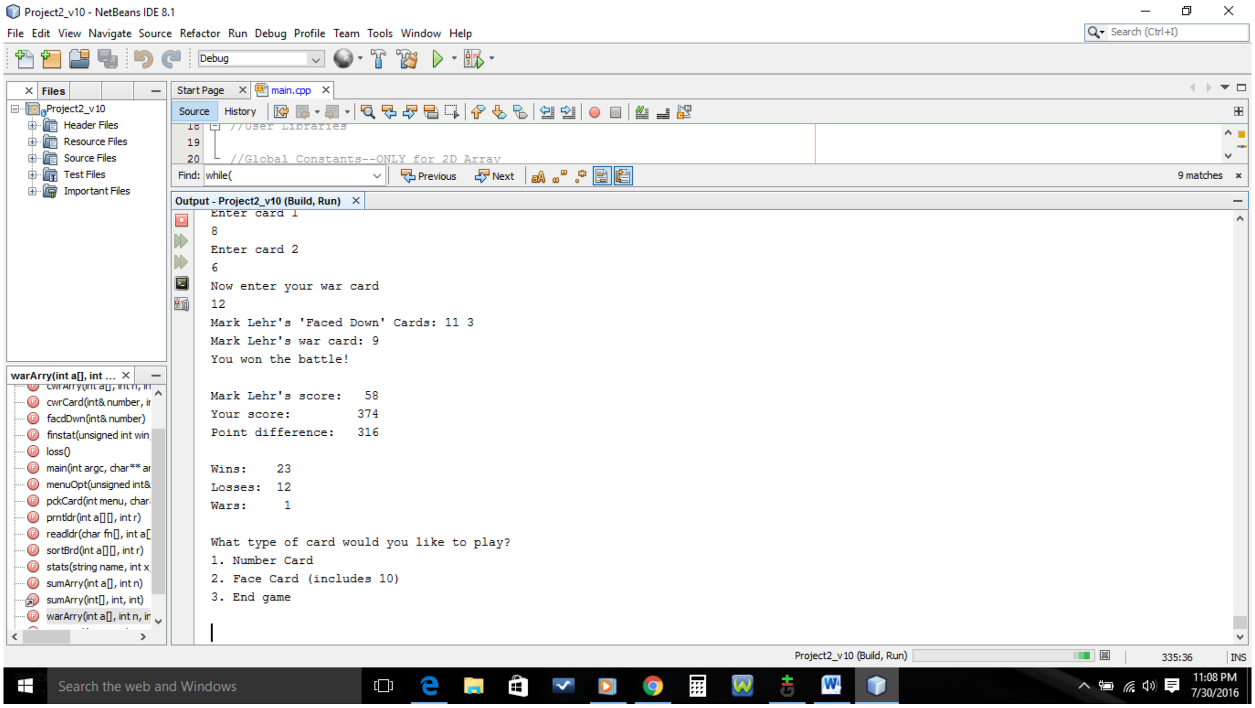


```
//User Libraries

//Global Constants--ONLY for 2D Array
```

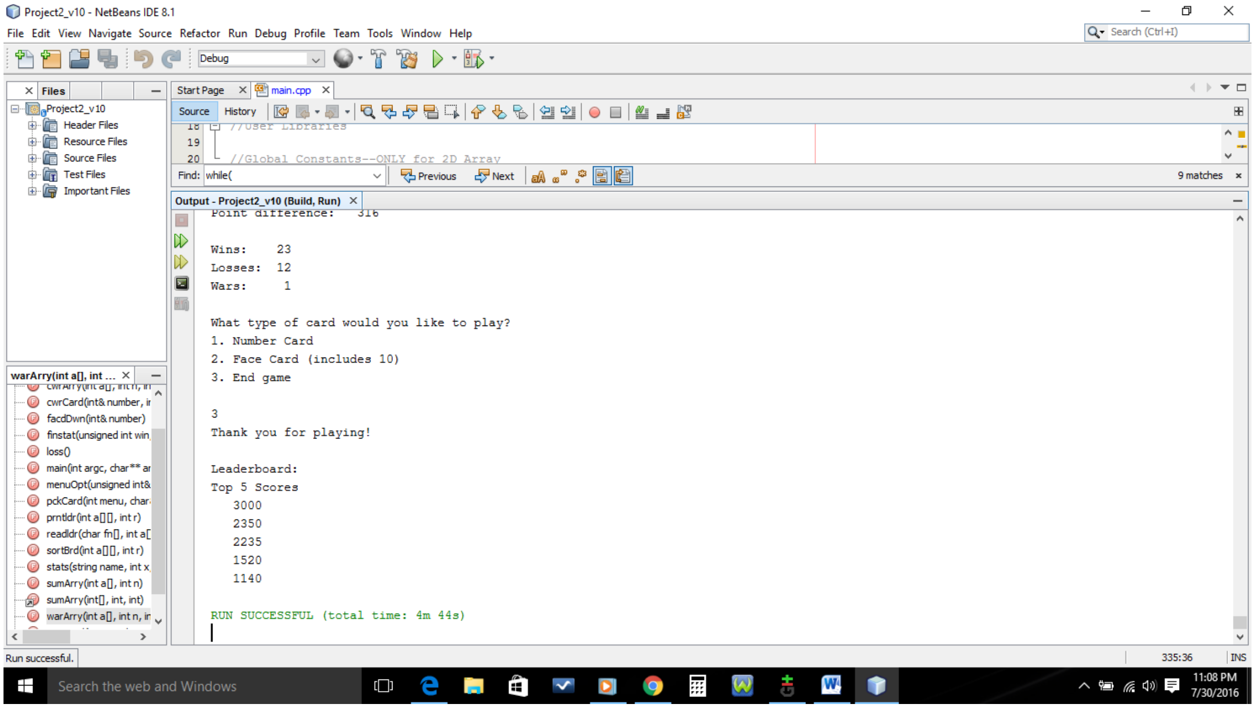
```
3. End game

1
Please enter the number of your choice (2-9)
7
Sorry. You lost.
Mark Lehr's card:  13

Mark Lehr's score:  179
Your score:          -7
Point difference:   186
Oh no! You're in the negative!
You need to score 7 points to get out the red zone

Wins:      6
Losses:   11
Wars:      0

What type of card would you like to play?
1. Number Card
2. Face Card (includes 10)
3. End game
```


Project2_v10 - NetBeans IDE 8.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Output - Project2_v10 (Build, Run)

```
Your score:          303
Point difference:    211


Wins:     22
Losses:   12
Wars:      0


What type of card would you like to play?
1. Number Card
2. Face Card (includes 10)
3. End game


2
Please enter T, J, Q, K, or A
j
War!!!
Please enter the integer value of the card you would like to draw.
Number cards are simply their own value, while T = 10, J = 11, Q = 12, K = 13, A = 14
Enter card 1
8
Enter card 2
6
Now enter your war card
```

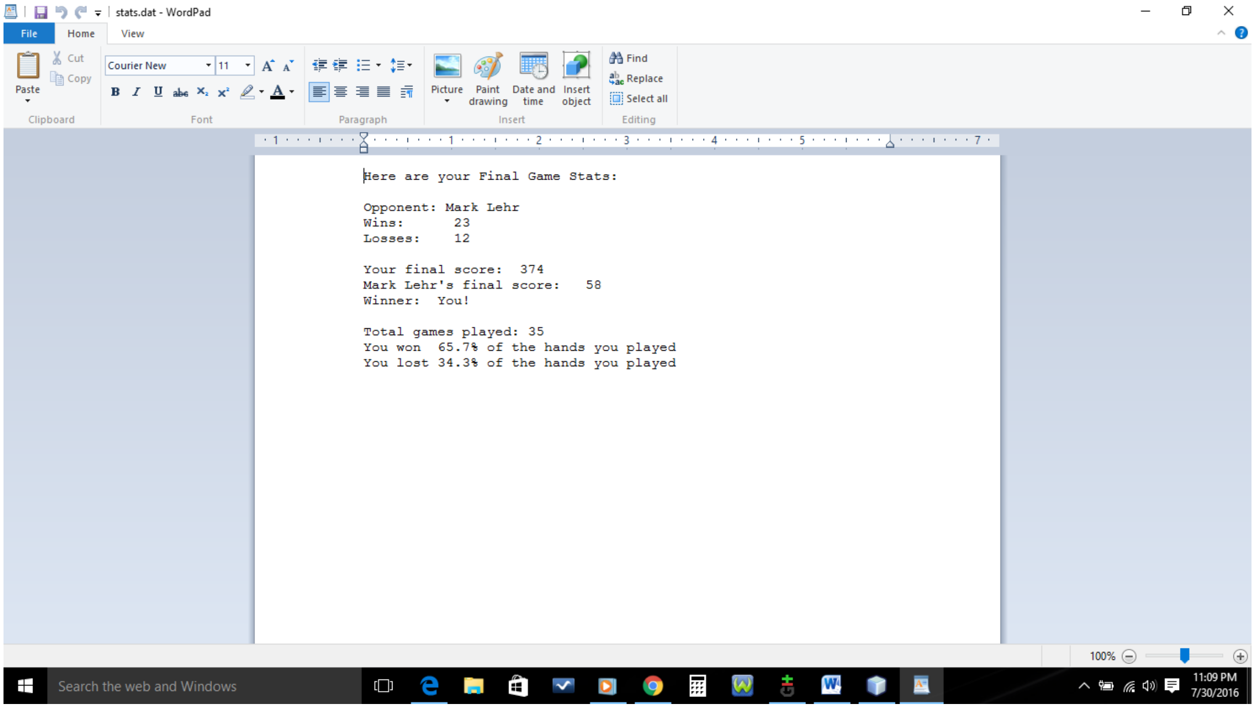

Project2_v10 - NetBeans IDE 8.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Output - Project2_v10 (Build, Run)

```
Enter card 1
8
Enter card 2
6
Now enter your war card
12
Mark Lehr's 'Faced Down' Cards: 11 3
Mark Lehr's war card: 9
You won the battle!


Mark Lehr's score:   58
Your score:          374
Point difference:    316


Wins:     23
Losses:   12
Wars:      1


What type of card would you like to play?
1. Number Card
2. Face Card (includes 10)
3. End game
```



```
//User Libraries

//Global Constants--ONLY for 2D Array
```

```
Point difference:   316

Wins:    23
Losses:  12
Wars:     1

What type of card would you like to play?
1. Number Card
2. Face Card (includes 10)
3. End game

3
Thank you for playing!

Leaderboard:
Top 5 Scores
   3000
   2350
   2235
   1520
   1140

RUN SUCCESSFUL (total time: 4m 44s)
```



```
Here are your Final Game Stats:

Opponent: Mark Lehr
Wins:       23
Losses:     12

Your final score:  374
Mark Lehr's final score:   58
Winner:  You!

Total games played: 35
You won  65.7% of the hands you played
You lost 34.3% of the hands you played
```

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 (only for the use of cin.clear();)

Program

```
/*
 * File:   main.cpp
 * Author: Laurie Guimont
 * Created on July 30, 2016, 1:06 PM
 * Purpose: War Card Game Enhancement
 */

//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
#include <cmath>    //Math Library
using namespace std;

//User Libraries

//Global Constants--ONLY for 2D Array
const int COL=1;

//Function Prototypes
int facdDwn(int &);
unsigned int menuOpt(unsigned int &);
int pckCard(int, char &);
int cardVal(char,int &);
void win();
void loss();
int warArry(int [],int,int &,int,int);
int sumArry(int [],int,int);
int warCard(int &,int,int);
int cwrArry(int [],int,int &,int,int,string);
void cwrCard(int &,int,int,string);
void stats(string,int,int,int,int,int);
void readldr(char [],int [][COL],int);
void sortBrd(int [][COL],int);
void prntldr(int [][COL],int);
void finstat(unsigned int,unsigned int,string,int,int);
```

```
//Execution Begins Here!
int main(int argc, char** argv) {
    //Set the Random Number Seed
    srand(static_cast<unsigned int>(time(0)));
    //Declare variables, no doubles
    string oppnent;      //Who you will be playing
    unsigned int choice; //User menu option
    char cchoice;        //User input representing card they want to play
    int number;          //Random number chosen set to present time
    int value;           //Value of each card
    const int MIN=2;     //Minimum value to choose from
    const int MAX=14;    //Maximum value to choose from
    int warcnt;          //Number of faced down cards before flipping in war
    int warnum,cwarnum;  //Card choice during war
    unsigned int nwins=0,nlosses=0,nwars=0;
    int score=0,wrscore=0,cscore=0,cwscore=0;

    //Open File & Enter Primary Input Data
    cout<<"The name of the game? WAR!"<<endl;
    cout<<"Give your opponent a name. You didn't think you were ";
    cout<<"playing against the computer now, did you?"<<endl;
    getline(cin,oppnent);

    //Establish Number of "Faced Down" Cards for the Game
    facdDwn(warcnt);

    //Process and Output the Data in the Loop
    do{
        //Get Menu & Select Card
        menuOpt(choice);
        pckCard(choice,cchoice);

        //Process the card choice
        if(choice!=3){
            //Call Function & Return Value
            cardVal(cchoice,value);

            //Determine win, loss, or war
            number = (rand() % (MAX - MIN + 1)) + MIN;
            if(value>number){
                nwins+=1;
                score=score+value+number;
                cscore-=number;
                win();
                cout<<oppnent<<"'s card:  "<<number<<endl;
            }
            else if(value<number){
                nlosses+=1;
                score-=value;
                cscore=cscore+value+number;
                loss();
                cout<<oppnent<<"'s card:  "<<number<<endl;
            }
            else{
                nwars+=1;
                //Declare Array Variables
                const int SIZE=warcnt;
                int war[SIZE];
                int cwar[SIZE];

                //Player Process
                wrscore=warArry(war,SIZE,warnum,MIN,MAX);
                warCard(warnum,MIN,MAX);

                //Comp Process
                cwscore=cwrArry(cwar,SIZE,cwarnum,MIN,MAX,oppnent);
                cwrCard(cwarnum,MIN,MAX,oppnent);

                //Compare Cards
                if(warnum>cwarnum){
                    nwins+=1;
                    score=score+value+number+wrscore+cwscore+warnum+cwarnum;
                    cscore=cscore-number-cwscore-cwarnum;
                    cout<<"You won the battle!"<<endl;
                }
                else if (warnum<cwarnum){
                    nlosses+=1;
                    score=score-value-warnum-wrscore;
                    cscore=cscore+value+number+cwscore+wrscore+cwarnum+warnum;
                    cout<<"You lost this battle."<<endl;
                }
                else{
                    while(warnum==cwarnum){  //Must War Again!
                        nwars+=1;
                        wrscore=warArry(war,SIZE,warnum,MIN,MAX);
                        warCard(warnum,MIN,MAX);

                        cwscore=cwrArry(cwar,SIZE,cwarnum,MIN,MAX,oppnent);
                        cwrCard(cwarnum,MIN,MAX,oppnent);

                        if(warnum>cwarnum){
                            nwins+=1;
                            score=score+value+number+wrscore+cwscore+warnum+
                                    cwarnum;
                            cscore=cscore-number-cwscore-cwarnum;
                            cout<<"You won the battle!"<<endl;
                        }
                        else if (warnum<cwarnum){
                            nlosses+=1;
                            score=score-value-warnum-wrscore;
                            cscore=cscore+value+number+cwscore+wrscore+
                                    cwarnum+warnum;
                            cout<<"You lost this battle."<<endl;
                        }
                    }
                }
            }
            //Game Stats
            stats(oppnent,score,cscore,nwins,nlosses,nwars);
        }
```

```
    }
    while(choice!=3);

    //End Game
    cout<<"Thank you for playing!"<<endl<<endl;

    //Show Sorted Leaderboard
    const int ROW=5;
    int board[ROW][COL];

    cout<<"Leaderboard:"<<endl;
    cout<<"Top 5 Scores"<<endl;
    readldr("leaderboard.dat",board,ROW);
    sortBrd(board,ROW);
    prntldr(board,ROW);

    //Finishing Stats - Output to a File
    finstat(nwins,nlosses,oppnent,score,cscore);

    //Exit Stage Right!
    return 0;
}

void finstat(unsigned int win,unsigned int loss,string name,int x,int y){
    //Declare Variables
    unsigned int ngames;
    float pwins,plosses;
    string winner;
    ofstream out;

    //Open & Write to file
    out.open("stats.dat");
    out<<"Here are your Final Game Stats:"<<endl<<endl;
    out<<"Opponent: "<<name<<endl;
    out<<"Wins:    "<<setw(4)<<win<<endl;
    out<<"Losses:  "<<setw(4)<<loss<<endl;
    out<<endl;
    out<<"Your final score: "<<setw(4)<<x<<endl;
    out<<name<<"'s final score: "<<setw(4)<<y<<endl;

    //Determine Winner of Game
    if(x>y)
        winner="You!\n";
    else
        winner=name;
    out<<"Winner:  "<<winner<<endl;

    //Calculate Number of Games
    ngames=win+loss;
    out<<"Total games played: "<<ngames<<endl;

    //Calculate Percentage of Wins and Losses
    pwins=static_cast<float>(win)/ngames;
    plosses=static_cast<float>(loss)/ngames;
    //Output Percentage
    out<<fixed<<setprecision(1)<<showpoint;
    out<<"You won  "<<pwins*100<<"% of the hands you played"<<endl;
    out<<"You lost "<<plosses*100<<"% of the hands you played"<<endl;

    //Close the file
    out.close();
    return;
}

void sortBrd(int a[][COL],int r){
    //Declare Variables
    bool swap;
    int temp;
    //Sort
    do{
        swap=false;
        for(int i=0;i<r-1;i++){
            for(int j=0;j<COL;j++){
                if(a[i][j]<a[i+1][j]){
                    temp=a[i][j];
                    a[i][j]=a[i+1][j];
                    a[i+1][j]=temp;
                    swap=true;
                }
            }
        }
    }
    while(swap);
    return;
}

void readldr(char fn[],int a[][COL],int r){
    //Declare the file
    ifstream in;
    //Open the file
    in.open(fn);
    //Send the array to the file
    for(int i=0;i<r;i++){
        in>>a[i][0];
    }
    //Close the file
    in.close();
    return;
}

void prntldr(int a[][COL],int r){
    for(int i=0;i<r;i++){
        cout<<setw(7)<<a[i][0]<<endl;
    }
    return;
}

void stats(string name,int x,int y,int win,int loss,int war){
    cout<<endl;
    cout<<name<<"'s score: "<<setw(4)<<y<<endl;
    cout<<"Your score:        "<<setw(4)<<x<<endl;
    cout<<"Point difference:  "<<setw(4)<<abs(x-y)<<endl;

    if(x<0){
        cout<<"Oh no! You're in the negative!"<<endl;
        cout<<"You need to score "<<abs(x)<<" points to get out ";
        cout<<"the red zone"<<endl;
    }
    cout<<endl;
    cout<<"Wins:   "<<setw(3)<<win<<endl;
    cout<<"Losses: "<<setw(3)<<loss<<endl;
    cout<<"Wars:   "<<setw(3)<<war<<endl;
    return;
}

void cwrCard(int &number,int min,int max,string name){
    //Opponent's War Card
    number = (rand() % (max - min + 1)) + min;
    cout<<name<<"'s war card: "<<number<<endl;
    return;
}

int warCard(int &number,int min, int max){
    //Player's War Card
    cout<<"Now enter your war card"<<endl;
    cin>>number;

    //Input Validation
    while(!(number)||number<min||number>max){
        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 a[],int n,int &val,int min,int max,string name){
    //Opponent's "Faced Down" Cards
    cout<<name<<"'s 'Faced Down' Cards: ";
    for(int i=0;i<n;i++){
        a[i] = (rand() % (max - min + 1)) + min;
        cout<<a[i]<<" ";
    }
    cout<<endl;
    val=sumArry(a,n);
    return val;
}

int warArry(int a[],int n,int &val,int min, int max){
    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;

    //Player's "Faced Down" Cards
    for(int i=0;i<n;i++){
        cout<<"Enter card "<<i+1<<endl;
        cin>>a[i];

        //Input Validation
        while(!(a[i])||a[i]<min||a[i]>max){
            cin.clear();
            cin.ignore();
            cout<<"Invalid input. Please type in an integer";
            cout<<" between 2 and 14."<<endl;
            cin>>a[i];
        }
        //Add Elements in Array
        val=sumArry(a,n);

    }return val;
}

void loss(){
    cout<<"Sorry. You lost."<<endl;
    return;
}

void win(){
    cout<<"You won!"<<endl;
    return;
}

int cardVal(char choice,int &number){
    switch(choice){
        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=(choice-48);break;
    }
    return number;
}

int pckCard(int menu, char &card){
    if(menu==1){
        cout<<"Please enter the number of your choice (2-9)"<<endl;
        cin>>card;

        //Input Validation
        while(card<'2'||card>'9'){
            cout<<"Invalid entry! Please enter (2-9)"<<endl;
            cin>>card;
        }
    }
    else if(menu==2){
        cout<<"Please enter T, J, Q, K, or A"<<endl;
        cin>>card;

        //Input Validation
        while(card!='a'&&card!='A'&&card!='k'&&card!='K'&&
                card!='q'&&card!='Q'&&card!='j'&&card!='J'&&
                card!='t'&&card!='T'){
            cout<<"Invalid entry! Please enter one of the choices "
                    "above"<<endl;
            cin>>card;
        }
    }
    return card;
}

unsigned int menuOpt(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 (includes 10)"<<endl;
    cout<<"3. End game"<<endl<<endl;
    cin>>option;

    //Input Validation
    while(option<1||option>3){
        cout<<"Invalid entry! Please enter an option from the menu"<<endl;
        cin>>option;
    }
    return option;
}

int facdDwn(int &number){
    cout<<endl<<"When war is waged,"<<endl;
    cout<<"how many cards do you want to put down before you flip one?"<<endl;
    cout<<"Please pick a number from 2-4"<<endl;
    cin>>number;

    //Input Validation
    while(number<2||number>4){
        cout<<"Error. Please enter 2,3, or 4"<<endl;
        cin>>number;
    }
    return number;
}
```