War/I Declare War

Card Game

Laurie Guimont

CSC 5

Summer 2016

45276

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 9

iostream library 9

cstdlib library 10

ctime library 10

iomanip library 10

string library 10

cmath library 10

fstream library 10

data types 11

conditional statements 11

loops 11

Proof of a working product 11

References 15

Program 15

**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 did some research and found that most (if not all) of the questions I was asking myself had a common answer: arrays. Well, I hadn’t learned arrays at that point. 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.

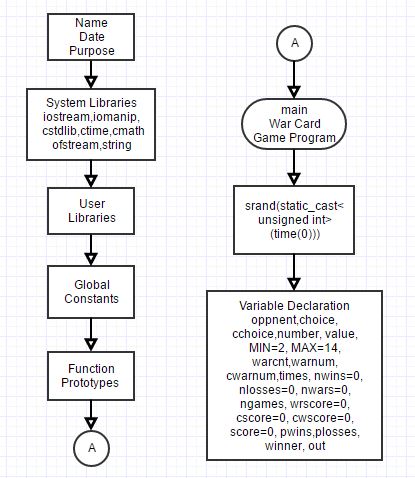
Since the face cards don’t have a numeric value in the regular card game, I assigned a value to each according to their rank in the game. Therefore, the Jack is valued at 11 points and the Ace is valued at 14 points, with the Queen at 12 points, and the King at 13.

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/10902783>

*Put in opening comments*

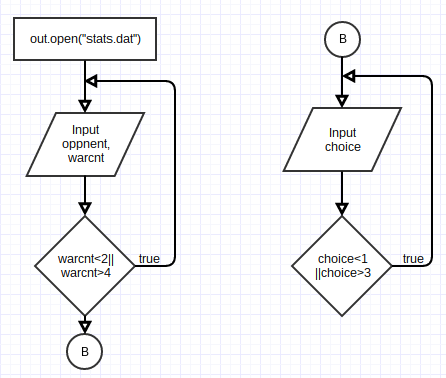
*Bring in system libraries*

*Enter main, then immediately*

*set random number seed*

*Declare all variables, initiate some*

*now and some later.*

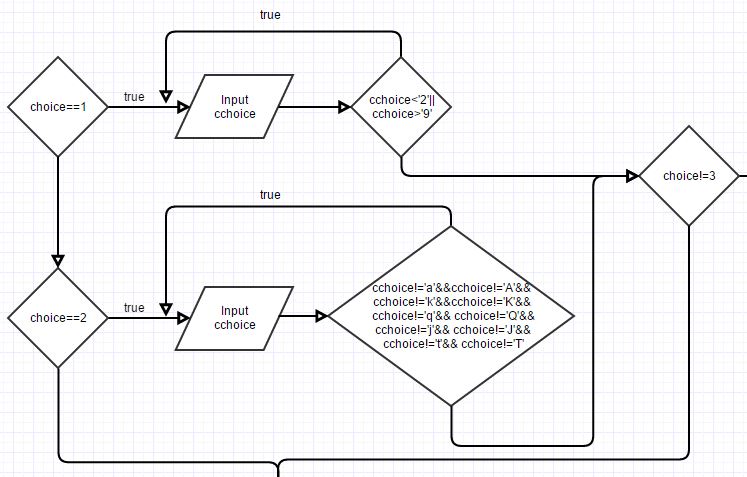
*Open file that we will write data to*

*Input opponent name and number of faced down cards*

*Loop this step until valid data is inputted.*

*Input valid menu choice.*

*Verify valid data with while loop again*

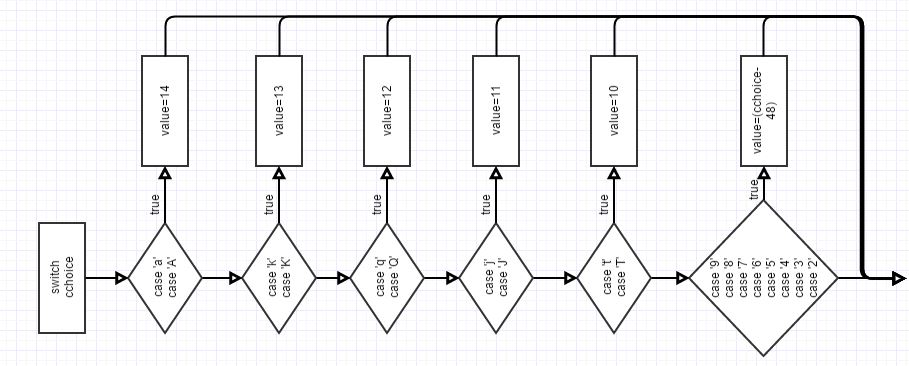
*If “number card” or “face card”,*

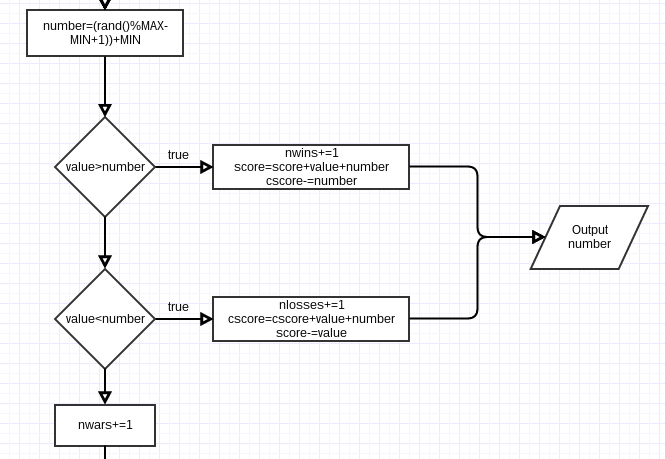
*input card choice*

*Verify that all data is valid.*

*As long as “End Menu” is not chosen,*

*computer processes input by applying*

*assigned value in switch statement*

*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*

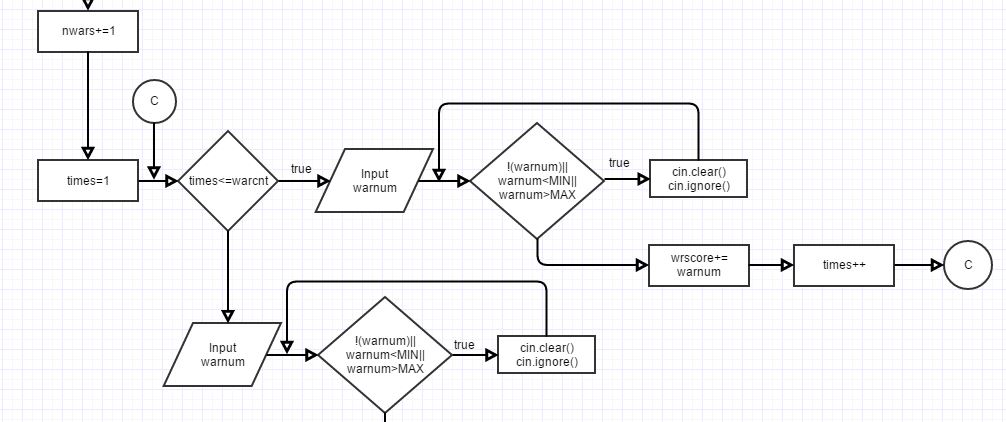
*If value is smaller, computer wins.*

*Computer wins round*

*Score is calculated*

*If both numbers are equal*

*User has entered war*

*Prompt user to enter face down cards*

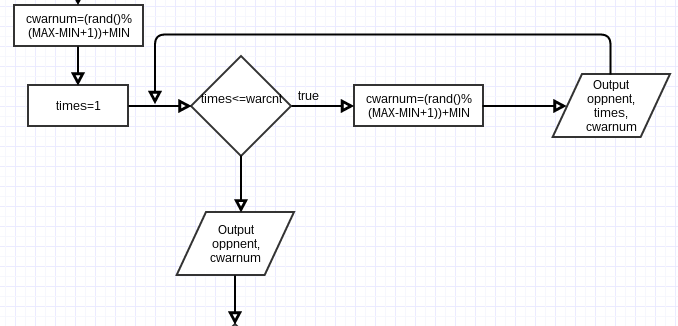
*According to initial input at start*

*of the game*

*Validate*

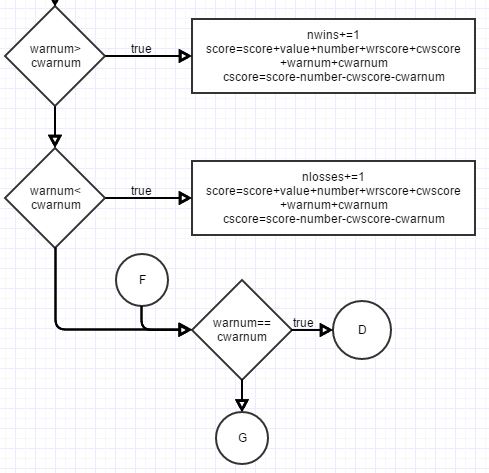
*Prompt user to enter war card*

*Validate*

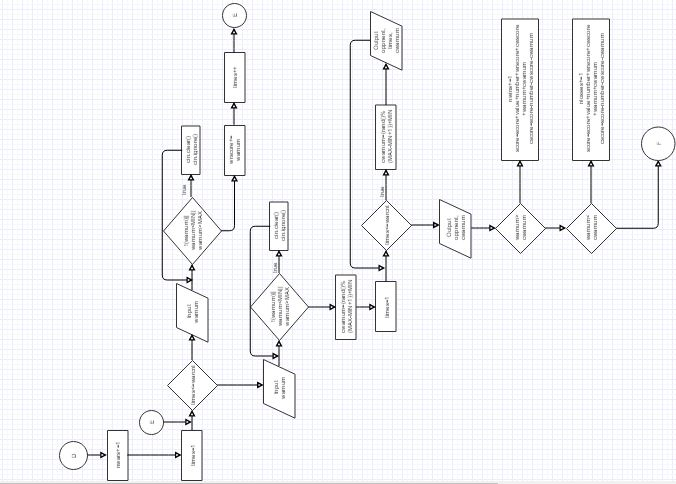
**

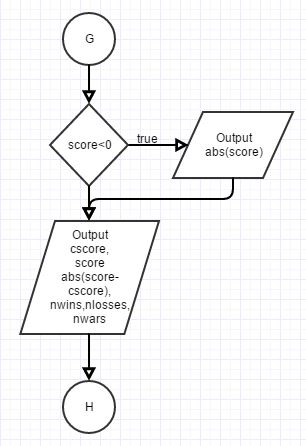
*Computer now chooses faced down cards*

*Computer chooses war card*

*War Card comparison is made*

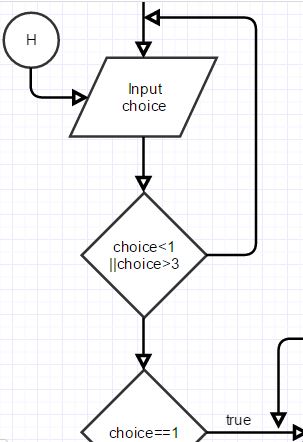
*whoever has the higher number, win.*

*If equal, repeat war.*

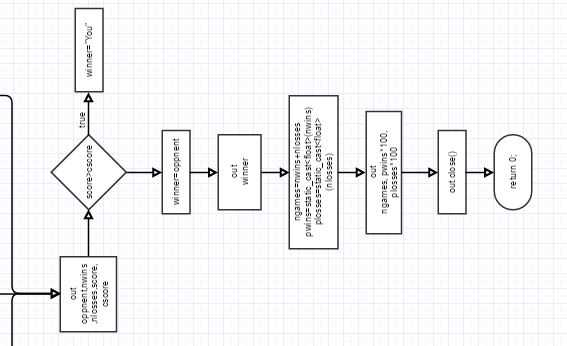
**

*Game stats are kept during gameplay.*

*Display after each round*

*Return to initial choice and repeat entire process*

*as long as “End Program”(3) is not selected.*

*If 3 is selected, thank user for playing and write*

*finishing stats to the output file.*

*Calculate some statistics then close the file*

*Return 0; the program is complete*

**Constructs & Concepts Utilized**

**iostream Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| static\_cast | 3 | Statically cast as different variable | Line 27,308,309 |
| cout | 58 | Output Data | Throughout |
| cin | 25 | Input Data | Throughout |
| getline() | 1 | Reads string data | Line 50 |
| cin.ignore() | 4 | Prevented input problems | Line 156,172,214,230 |
| cin.clear() | 4 | Stopped infinite loop | Line 155,171,213,229 |

**cstdlib Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| srand() | 1 | Random # seed | Line 27 |
| rand() | 5 | Generates rand # | Line 125,179,181,237,239 |

**ctime Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| time | 1 | Set current time | Line 27 |

**iomanip Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| fixed | 1 | Format final game stats | Line 311 |
| setprecision() | 1 | Format final game stats | Line 311 |
| showpoint | 1 | Format final game stats | Line 311 |
| setw() | 10 | Format final game stats | Line 268-270,278-280,290,291,293,  294 |

**string Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| string | 2 | Declare var. | Line 29,42 |
| getline() | already mentioned | already mentioned | already mentioned |

**cmath Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| abs() | 2 | Neg. Score Alert  Point Difference | Line 270,274 |

**fstream Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| out.open() | 1 | Open file | Line 46 |
| out.close() | 1 | Close file | Line 316 |
| out | 12 | Write to file | Line 288-294,301,305,311-313 |
| ofstream | 1 | Declare var. | Line 43 |

**Data Types:**

|  |  |  |
| --- | --- | --- |
| Data Types | Frequency | Location |
| int | 11 | Line 32,33,36,37,40 |
| unsigned int | 7 | Line 27,30,38,39 |
| char | 1 | Line 31 |
| string | 2 | Line 29,42 |
| float | 2 | Line 41 |
| ofstream | 1 | Line 43 |

**Conditional Statements:**

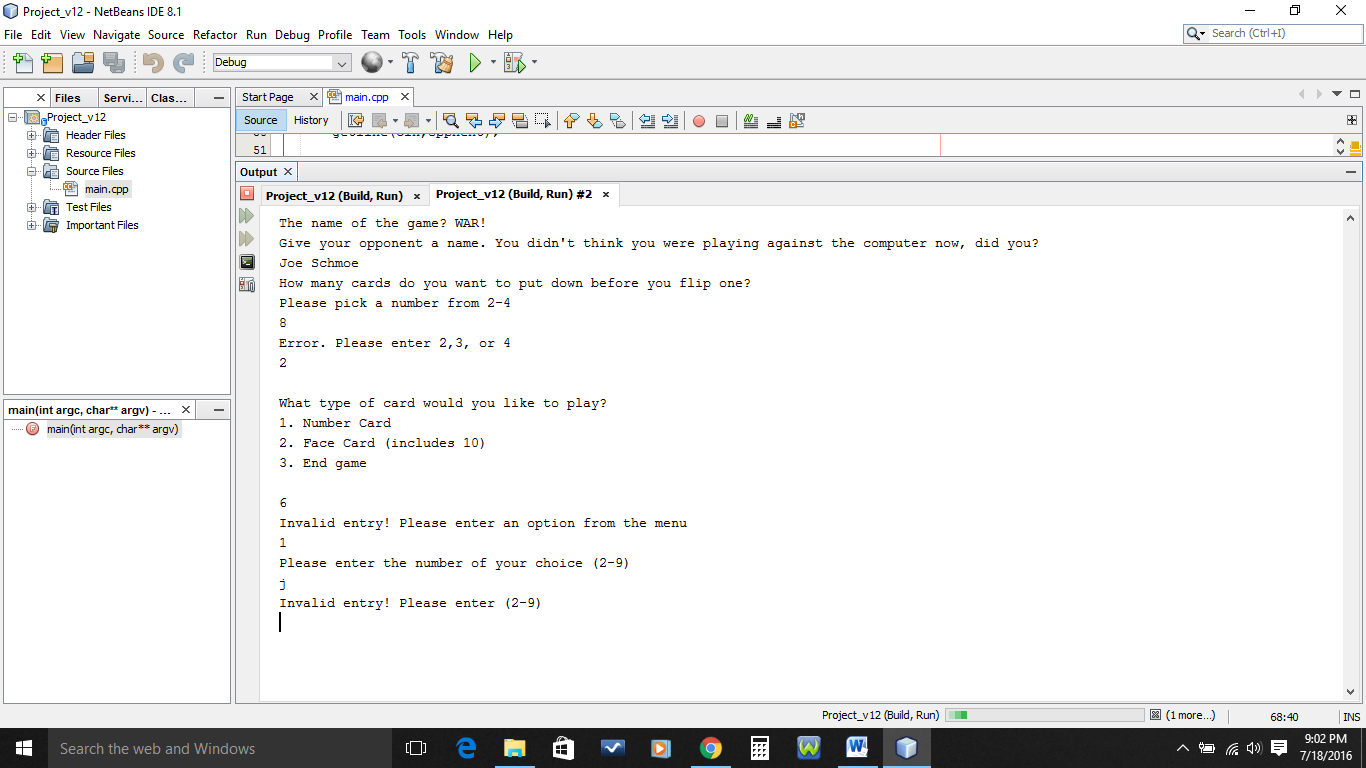
|  |  |  |
| --- | --- | --- |
| Conditional Statement | Frequency | Starting Location |
| if | 2 | Line 102,272 |
| if/else | 1 | Line 297 |
| if/else if | 4 | Line 77,126,189,248 |
| switch | 1 | Line 103 |

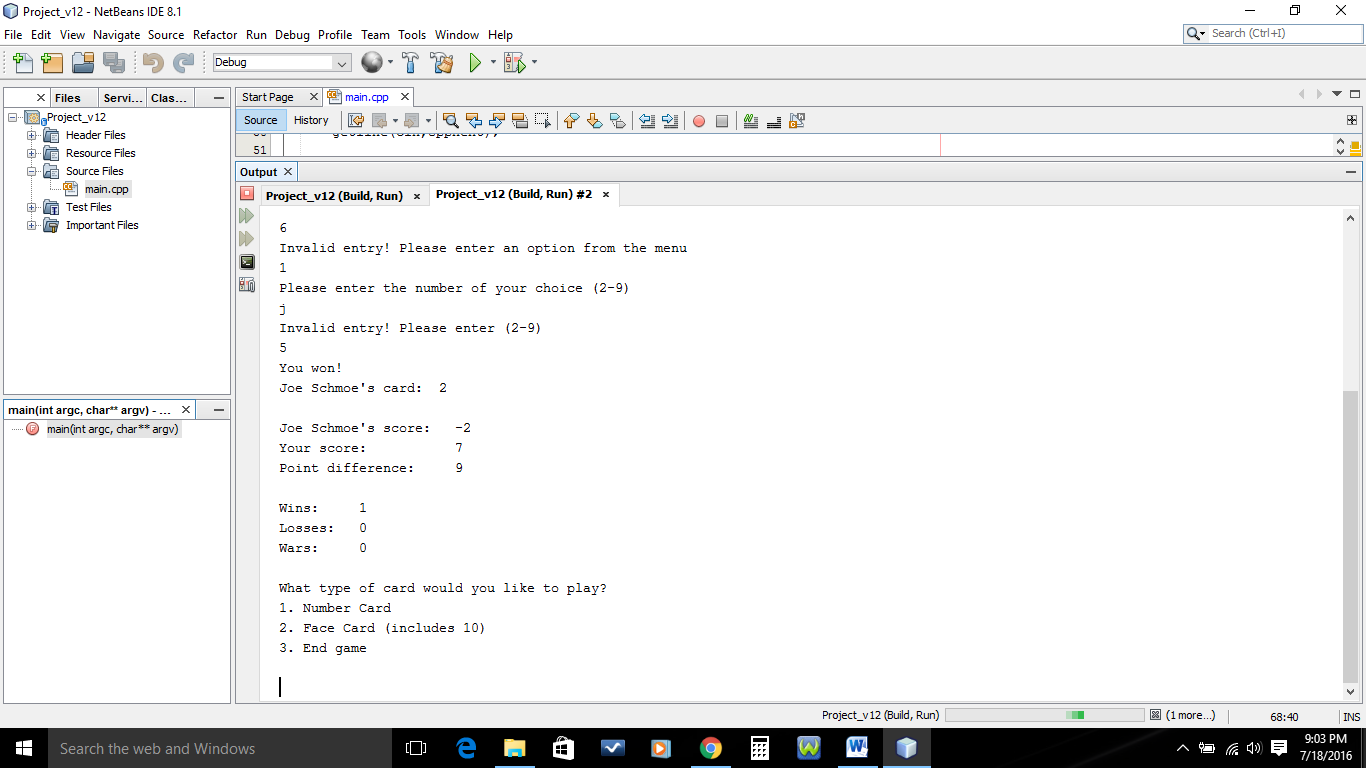
**Loops:**

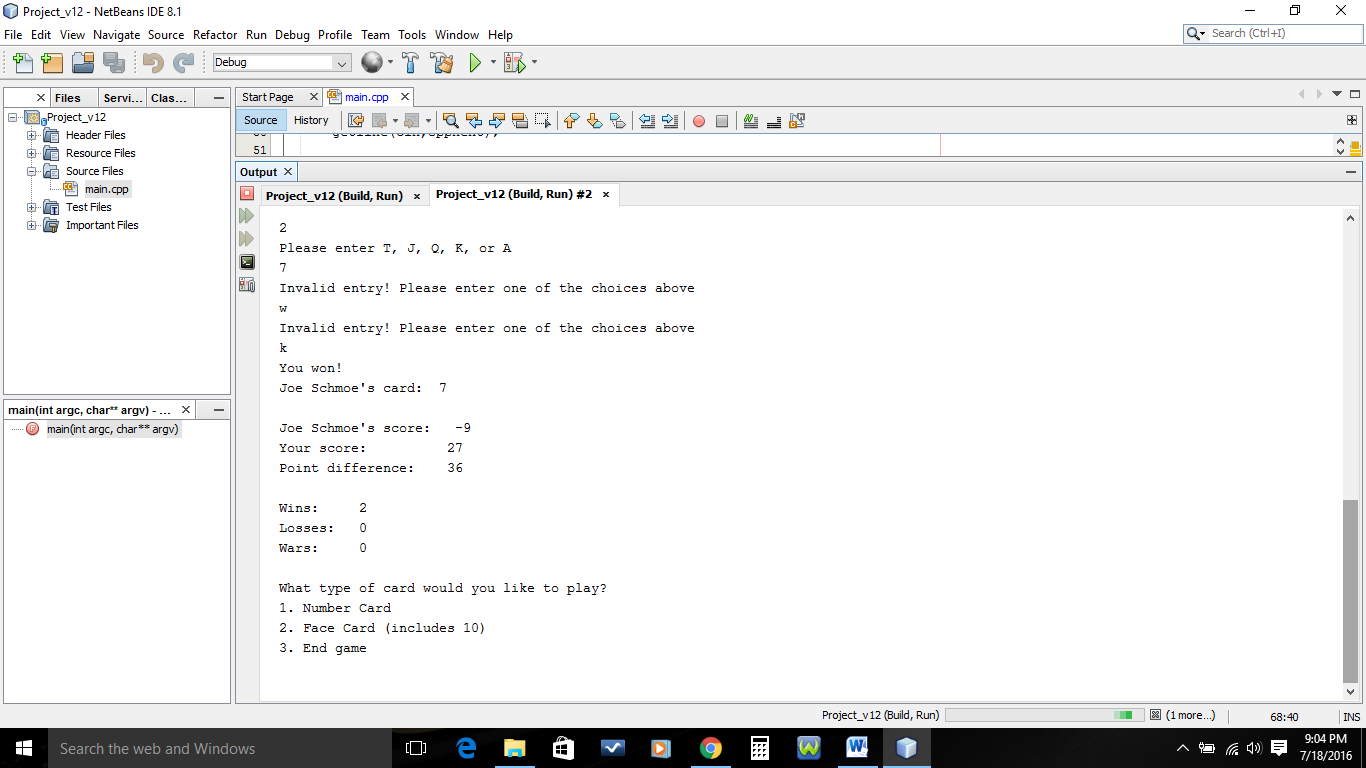
|  |  |  |
| --- | --- | --- |
| Loops | Frequency | Starting Location |
| for | 4 | Line 149,180,207,238 |
| while | 9 | Line 57,72,82,92,154,170,  202,212,228 |
| do-while | 1 | Line 63 |

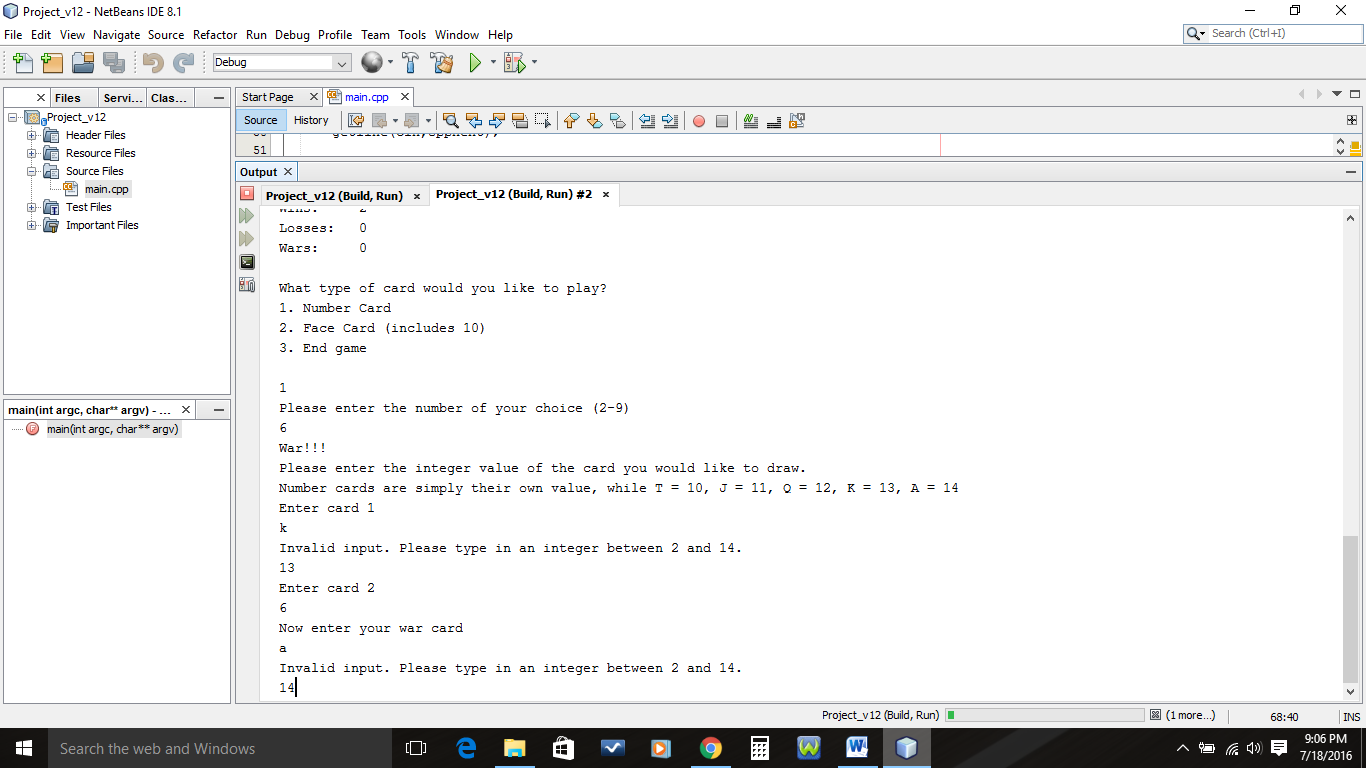
**Proof of a Working Product**

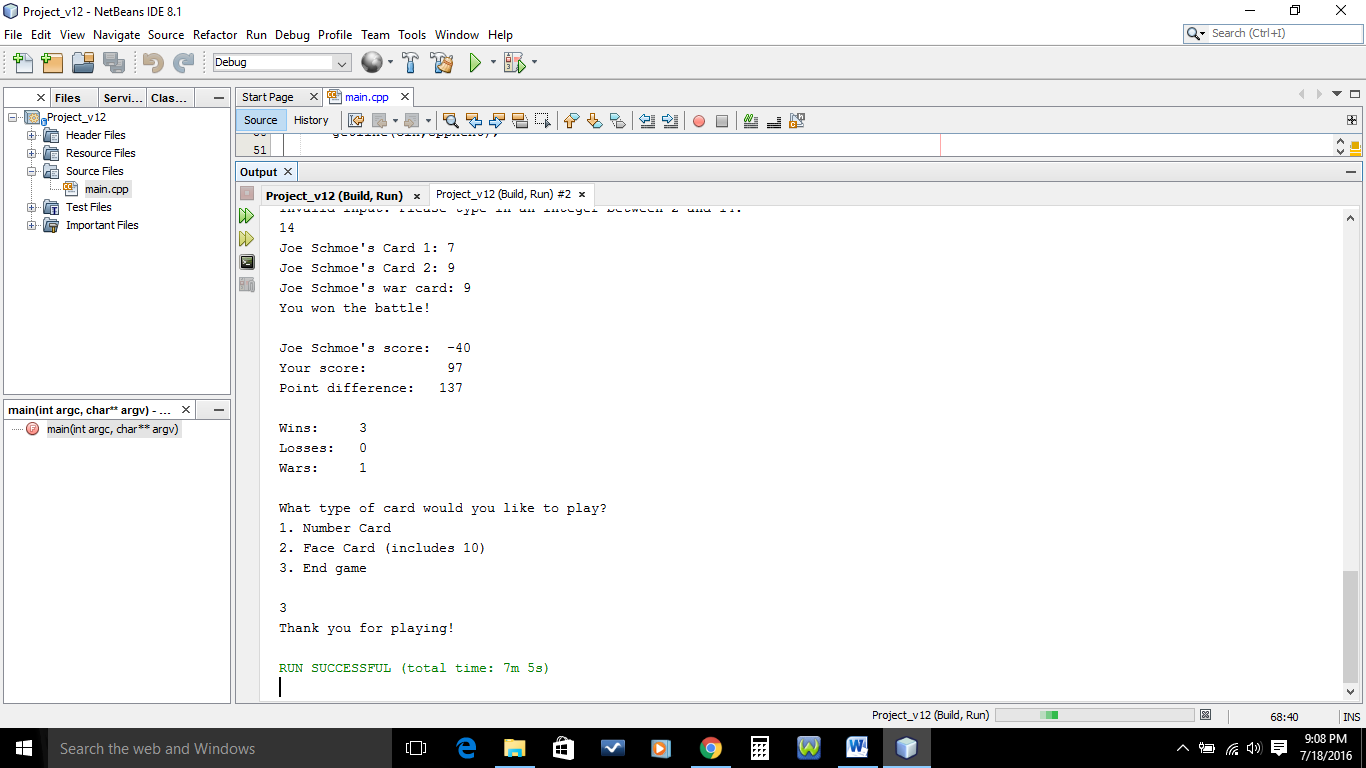
In the event, that my program does not work once it reaches Dr. Lehr, I have provided some screenshots that prove that the program did work at one time on the next few pages.

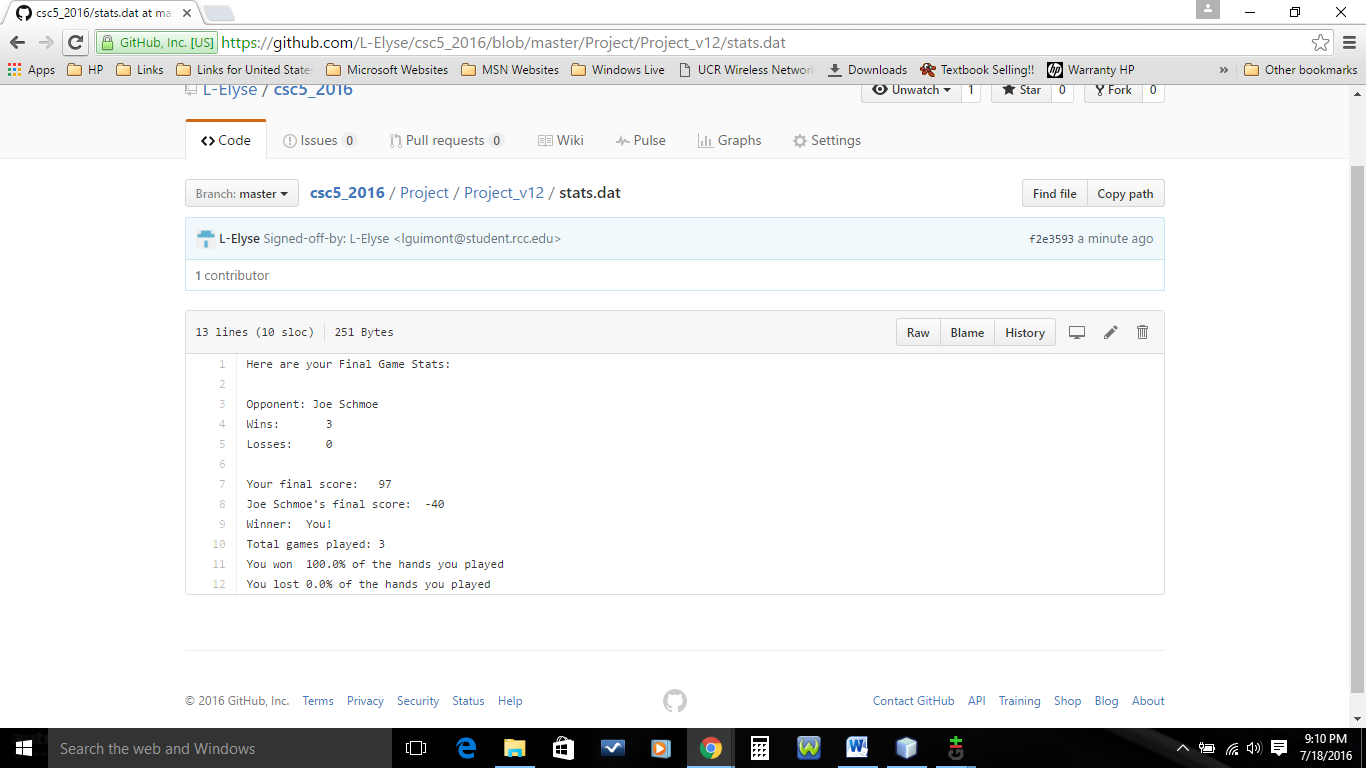












**References**

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**