**Project 1 V2**

**<Tic Tac Toe>**

**CSC 5 - 40108**

**Ghislain Muberwa**

**Date: 01/29/2017**

**Introduction**

Title: Tic Tac Toe

This is a simple game called tic tac toe. It is played with two players on a 3 x 3 board where one player marks ‘X’ and the other player ‘O’ on the board and the purpose is to match three of the same mark in a line in any direction possible. If there is no winner, the game ends in a tie.

**Summary**

Project size: about 300 lines

The numbers of variables: about 29

The number of methods: 7

Able to place most of the process involved in each game from main() method to their own seperate method. The whole game, Tic Tac Toe, is placed in a do-while loop where in the end, the player chooses to play the game again.

Each square is a char variable where it starts as a number and are slowly being replaced by either an ‘X’ or ‘O’.

All inputs are verified to be correct or not.

There is a possibility of either one of the player or neither to win.

After answering no to continue to play the game, there is an output of all the wins and ties of all the matches between the two players, a percentage of the win and tie.

The results from the win and tie is saved into a file and then it is printed again to confirm that the content is in the file the way it is was originally.

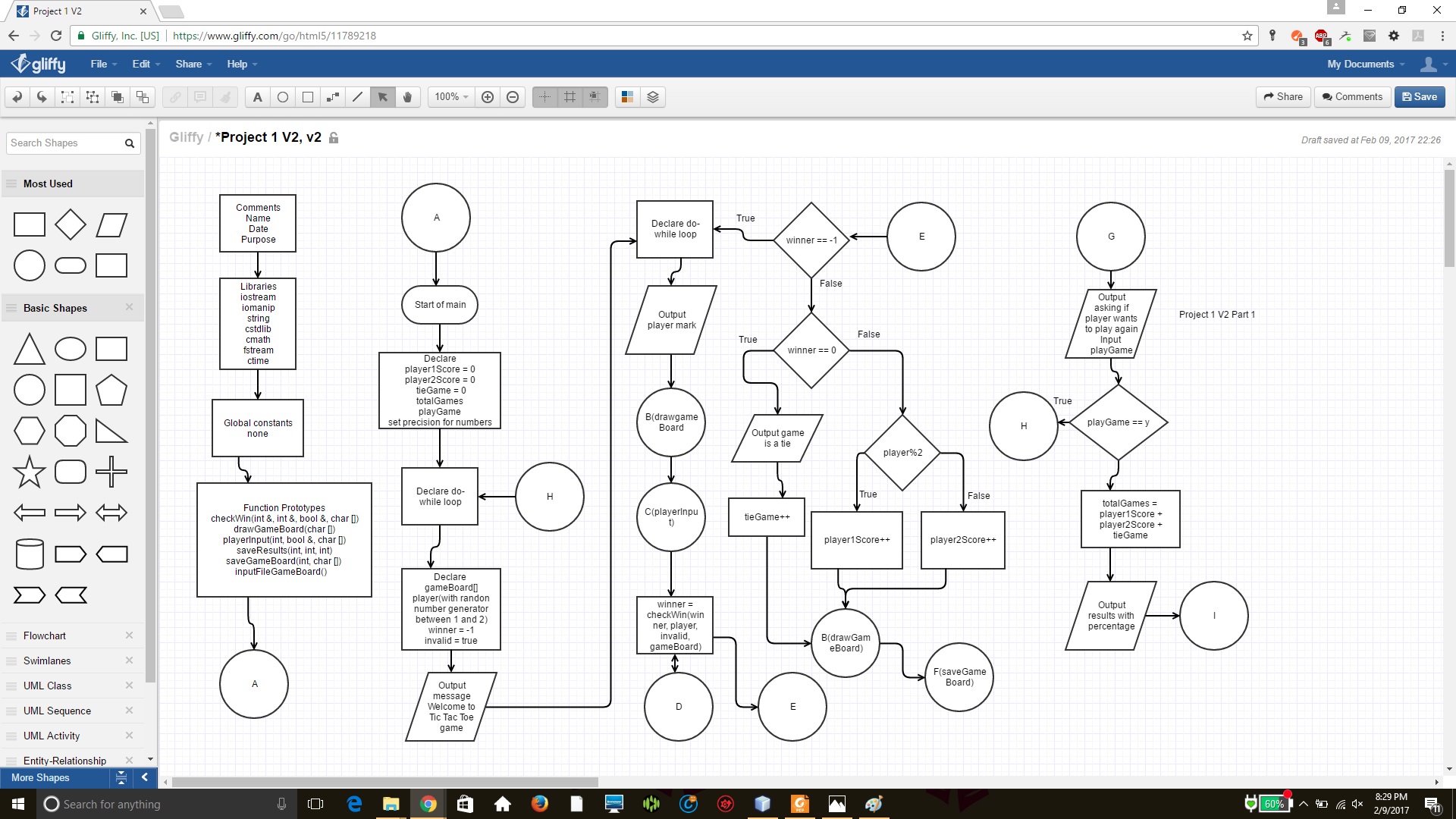
Game board is also saved after each game and displayed at the end of game.

**Description**

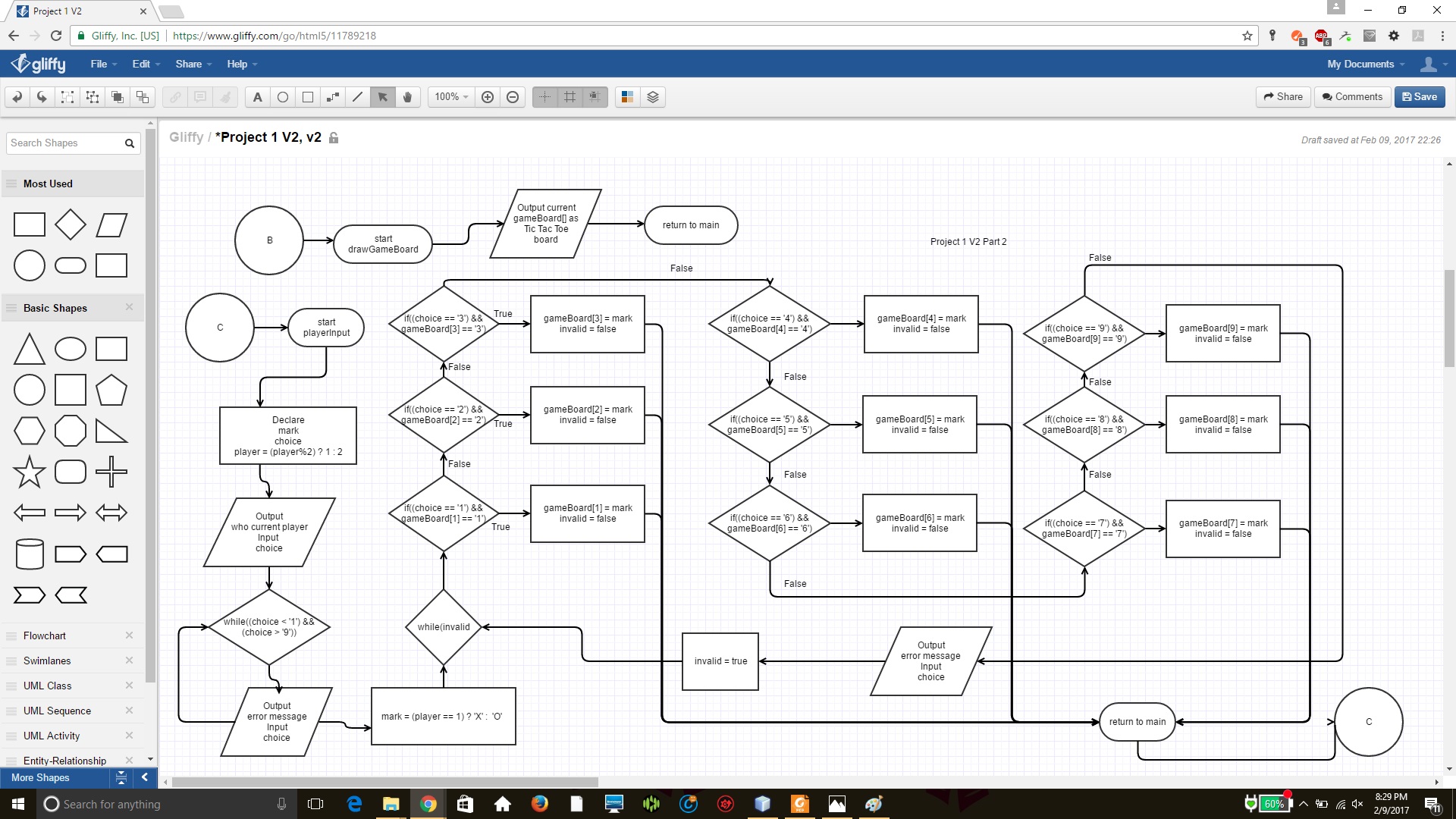
The program is a Tic Tac Toe game that draws a board and update the output based on user input. The game goes until a winner connects three of their mark in a line or no winner is selected. There is also the option of playing again until they do not want to.

**Flowchart**

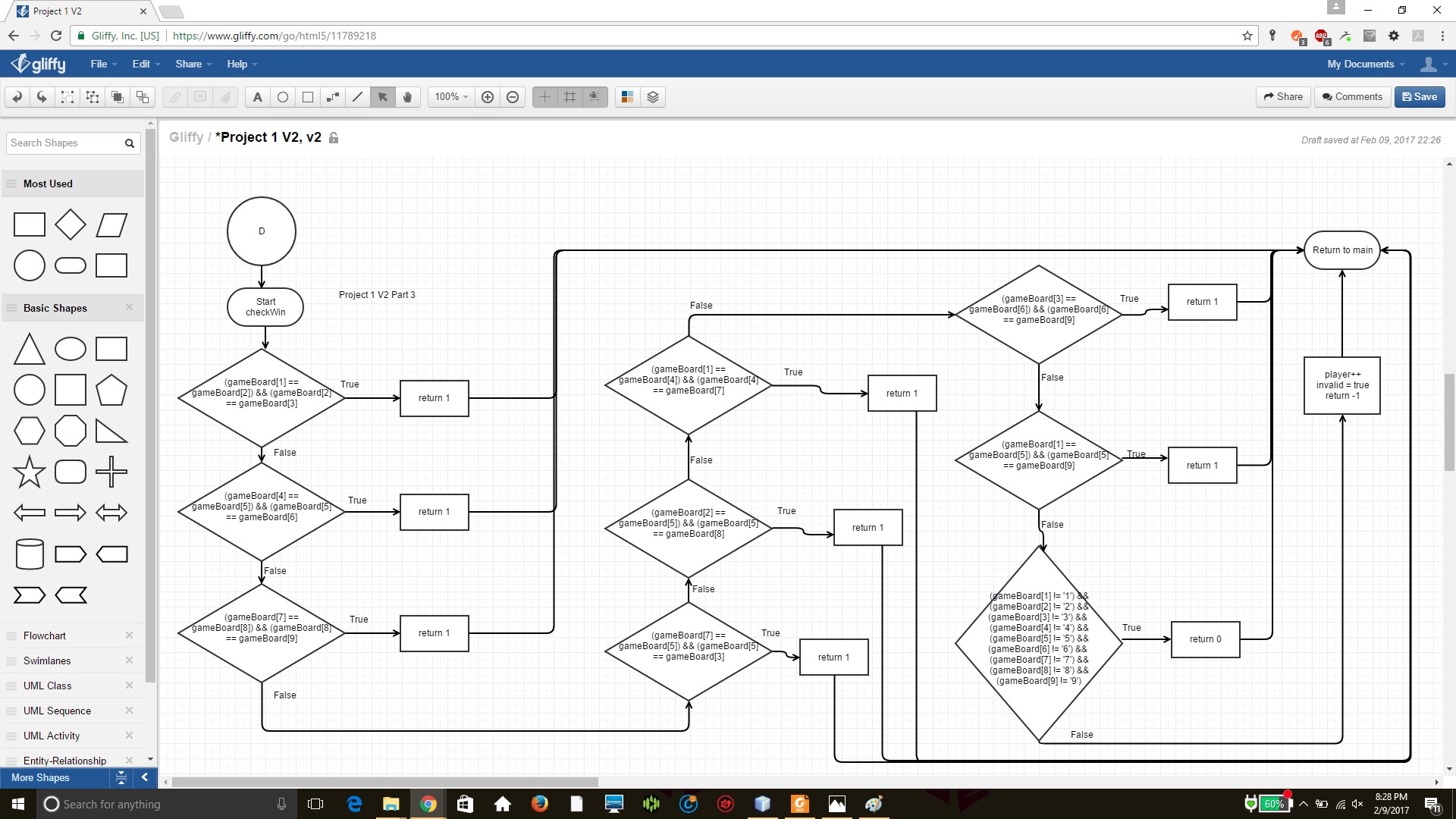
Part 1



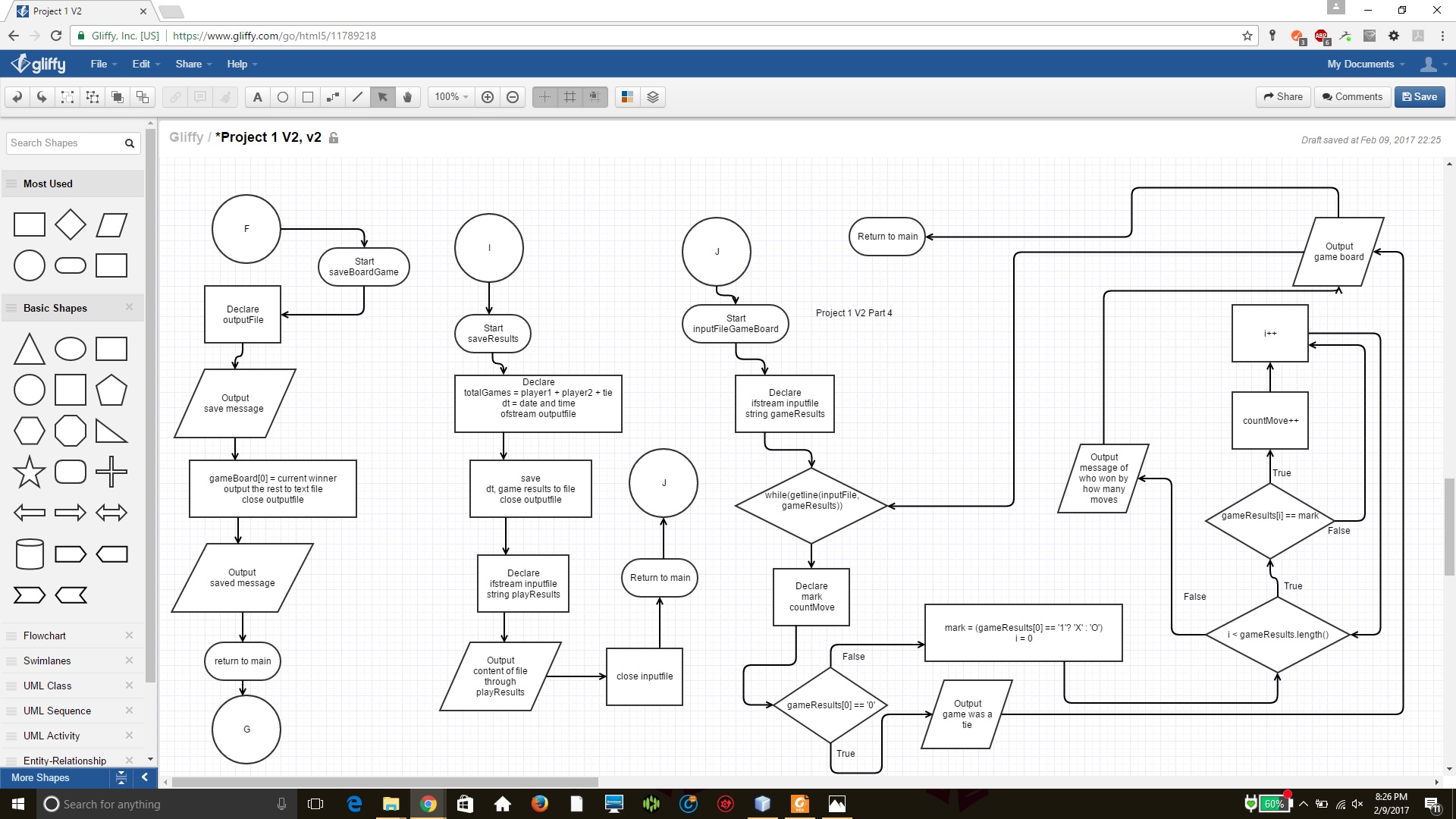
Part 2



Part 3



Part 4



**Pseudo-code**

Initialize board

Random player is selected

Player enters a number

Check if input is valid

Check if any player won or tie game

If someone won or tie game exit to show who one

Else next player’s turn

Show result of game

Save game board

Ask if player wants to play again

If yes return to initialize board

If no go to show results of all games

Display results of all games

Save result to file

Output content form file to console

Display past game boards

**Major variable**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Variable Name** | **Description** | **Location** |
| Float | totalGames | Hold total amount of games played | main() |
| Char | playGame | Hold answer as to if player wants to play again | main() |
| Char | gameBoard[] | Hold winner and game board | main() |
| unsigned | Seed | Equal to time(0), hold amount of seconds available at the time it is initialized | main() |
| Int | totalGames | Hold all wins between player 1, player 2, and tie game | saveResults() |
| char | Choice | Holds which player the user chooses | playerInput() |
| char | Mark | Replaces the selected square with either X or O | playerInput() |
| integer | Player1Score | Hold player 1 score | main() |
| integer | Player2Score | Hold player 1 score | main() |
| integer | tieGame | Identify how many tie games occurred | main() |
| integer | Player | Used to identity player 1 or player 2 | main() |
| integer | Winner | Identify if a player one or a tie game | main() |
| Boolean | Invalid | Check if player’s move was valid | main() |
| Integer | maxValue | Hold highest range for random number | main() |
| Integer | minValue | Hold lowest range for random number | main() |
| Time\_t | Now | Hold current time | saveResults() |
| Char\* | dt | Hold current time in string form | saveResults() |
| Osftream | outputFile | Link to file to save content to | saveResults()  saveGameBoard() |
| Instream | inputFile | Link to file to read content from | saveResults()  saveGameBoard()  inputFileGameBoard() |
| String | playResults | Store each line from file and output to screen | saveResults() |
| String | gameResults | Store each line from file and output to screen | inputFileGameBoard() |

**C++ Construct**

|  |  |  |
| --- | --- | --- |
| Chapter | New syntax and keyword | Location |
| 2 | Cout | Main() |
|  | Libraries (Iostream, iomanip, fstream, string) |  |
|  | Variables/literals |  |
|  | Integer |  |
|  | Character |  |
|  | Float |  |
|  | Bool |  |
|  | Variables 7 letters or less |  |
|  | Arithmetic operators |  |
|  | Comments 20%+ |  |
|  | Named constants |  |
| 3 | Cin |  |
|  | Math expression |  |
|  | Type casting |  |
|  | Formating output |  |
| 4 | Relational operators |  |
|  | If |  |
|  | If-else |  |
|  | Nesting |  |
|  | Logical operators |  |
|  | Validate user input |  |
|  | Conditional operator |  |
| 5 | Increment |  |
|  | While |  |
|  | Do-while |  |
|  | File input/output both |  |
| 6 | Prototypes |  |
|  | Pass by values |  |
|  | Returning values |  |
|  | Static local variables |  |
|  | Default arguments |  |
|  | Reference variables |  |
| 7 | Array initization |  |
|  | Arrays in function arguments |  |
| 8 | Search linear/binary |  |

/\*

File: main.cpp

Author: Ghislain Muberwa

Created on February 7, 2017, 6:39 PM

Purpose: Implement a two player Tic Tac Toe game. Upgrade the game with array

\*/

//System Libraries

#include <iostream>

#include <iomanip>

#include <string> //String class

#include <cstdlib> //Timer class

#include <cmath> //Math class

#include <fstream>

#include <ctime>

using namespace std;

//User Libraries

//Global Constants

//Such as PI, Vc, -> Math/Science values

//as well as conversions from system of units to

//another

//Function Prototypes

int checkWin(int , int &, bool &, char []); //verify if there is a winner

void drawGameBoard(char []); //Draw the game board

void playerInput(int, bool &, char []); //Obtain player input

void saveResults(int, int, int);

void saveGameBoard(int, char []); //Save board and show best

void inputFileGameBoard(); //Read gameBoard.txt content and display board

//Executable code begins here!!!

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

//Declare Variables

//Holds how many wins each player won

int player1Score = 0; //Holds how many games player 1 won

int player2Score = 0; //Holds how many games player 2 won

int tieGame = 0; //Holds tie games

float totalGames; //How many games were played

char playGame; //Hold response of if player wants to play again

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

int winner = -1;

do{

//Create game board

char gameBoard[10] = { 'n', '1', '2', '3', '4', '5', '6', '7', '8', '9'};

//Implement random generator to select who goes first, player 1 or player 2

unsigned seed = time(0);

srand(seed);

//Minimize choice to two possibility

int maxValue = 2;

int minValue = 1;

//indicate player's turn

int player = (rand() % (maxValue - minValue + 1)) + minValue;

int winner = -1; //Verify who the winner is

bool invalid = true; //Check that move was valid

//Message to tell what the game is

cout << "Welcome to Tic Tac Toe. This is a two player game where "

<< "each player will try to match either three X or O in a line. Player 1 will be X and Player 2 will be O.\n\n";

do{

cout << "Player 1 = X Player 2 = O" << endl;

//Draw current game board

drawGameBoard(gameBoard);

//Obtain player input

playerInput(player, invalid, gameBoard);

//Check if there is a winner

winner = checkWin(winner, player, invalid, gameBoard);

}while(winner == -1);

//Output who won

cout << endl;

if(winner == 0){

cout << "Game is a tie" << endl;

tieGame++;

}

else{

((player)%2) ? player1Score++ : player2Score++;

cout << "The winner is Player " << ((player)%2 ? 1 : 2) << ", this is the winning board" << endl;

}

//Draw winning game board

drawGameBoard(gameBoard);

cout << endl;

//Save the winning game board with the winner in place

saveGameBoard(((player)%2 ? 1 : 2), gameBoard);

//Ask if player wants to play again

cout << "Do you want to play again (Y for yes, N for no): ";

cin >> playGame;

while((playGame != 'Y') && (playGame != 'y') && (playGame != 'N') && (playGame != 'n')){

cout << "Please enter Y for yes or N for no: ";

cout << playGame;

}

}while((playGame == 'Y') || (playGame == 'y'));

//Output results of all games

totalGames = player1Score + player2Score + tieGame;

cout << "at the end of all the games, this is the final score:\n";

cout << "Player 1: " << player1Score << " Winning ratio: " << ((player1Score/(float)totalGames)\*100) << "%" << endl;

cout << "Player 2: " << player2Score << " Winning ratio: " << ((player2Score/(float)totalGames)\*100) << "%" << endl;

cout << "Tie game: " << tieGame << " tie ratio: " << (tieGame/(float)totalGames) << "%" << endl;

cout << endl;

//Save final game

saveResults(player1Score, player2Score, tieGame);

//Display winning game board

inputFileGameBoard();

//Exit stage right!

return 0;

}

/\*Check if there is a winner\*/

int checkWin(int winner, int& player, bool& invalid, char gameBoard[]){

//Check if there is a winner

if((gameBoard[1] == gameBoard[2]) && (gameBoard[2] == gameBoard[3])) //First row match

return 1;

else if((gameBoard[4] == gameBoard[5]) && (gameBoard[5] == gameBoard[6])) //Second row match

return 1;

else if((gameBoard[7] == gameBoard[8]) && (gameBoard[8] == gameBoard[9])) //Third row match

return 1;

else if((gameBoard[1] == gameBoard[4]) && (gameBoard[4] == gameBoard[7])) //First column match

return 1;

else if((gameBoard[2] == gameBoard[5]) && (gameBoard[5] == gameBoard[8])) //Second column match

return 1;

else if((gameBoard[3] == gameBoard[6]) && (gameBoard[6] == gameBoard[9])) //Third column match

return 1;

else if((gameBoard[1] == gameBoard[5]) && (gameBoard[5] == gameBoard[9])) //Diagonal down match

return 1;

else if((gameBoard[7] == gameBoard[5]) && (gameBoard[5] == gameBoard[3])) //Diagonal up match

return 1;

else if((gameBoard[1] != '1') && (gameBoard[2] != '2') && (gameBoard[3] != '3') &&

(gameBoard[4] != '4') && (gameBoard[5] != '5') && (gameBoard[6] != '6') &&

(gameBoard[7] != '7') && (gameBoard[8] != '8') && (gameBoard[9] != '9'))

return 0;

player++; //Increment player variable to allow next player to play

invalid = true;

return -1;

}

/\*Draw the current board\*/

void drawGameBoard(char gameBoard[]){

cout << " | | " << endl;

cout << " " << gameBoard[1] << " | " << gameBoard[2] << " | " << gameBoard[3] << endl;

cout << "\_\_\_\_\_\_\_|\_\_\_\_\_\_\_|\_\_\_\_\_\_\_" << endl;

cout << " | | " << endl;

cout << " " << gameBoard[4] << " | " << gameBoard[5] << " | " << gameBoard[6] << endl;

cout << "\_\_\_\_\_\_\_|\_\_\_\_\_\_\_|\_\_\_\_\_\_\_" << endl;

cout << " | | " << endl;

cout << " " << gameBoard[7] << " | " << gameBoard[8] << " | " << gameBoard[9] << endl;

}

void playerInput(int player, bool& invalid, char gameBoard[]){

char choice; //Hold user's choice

char mark; //Change square to either X or O

player = (player%2) ? 1 : 2; //Identity current player

cout << "Current player: " << player << " enter a number: ";

cin >> choice; //Obtain player's choice;

//Check if input is appropriate

while((choice < '1') && (choice > '9')){

cout << "Please enter a number between 1 and 9: ";

cin >> choice;

}

mark = (player == 1) ? 'X' : 'O'; //Use appropriate mark for player

//Find which square was selected and replace it with appropriate mark

while(invalid){ //Stay in same turn until valid choice is implemented

if((choice == '1') && gameBoard[1] == '1'){

gameBoard[1] = mark;

invalid = false;

}

else if((choice == '2') && gameBoard[2] == '2'){

gameBoard[2] = mark;

invalid = false;

}

else if((choice == '3') && gameBoard[3] == '3'){

gameBoard[3] = mark;

invalid = false;

}

else if((choice == '4') && gameBoard[4] == '4'){

gameBoard[4] = mark;

invalid = false;

}

else if((choice == '5') && gameBoard[5] == '5'){

gameBoard[5] = mark;

invalid = false;

}

else if((choice == '6') && gameBoard[6] == '6'){

gameBoard[6] = mark;

invalid = false;

}

else if((choice == '7') && gameBoard[7] == '7'){

gameBoard[7] = mark;

invalid = false;

}

else if((choice == '8') && gameBoard[8] == '8'){

gameBoard[8] = mark;

invalid = false;

}

else if((choice == '9') && gameBoard[9] == '9'){

gameBoard[9] = mark;

invalid = false;

}

else{

cout << "Invalid move please try again: ";

cin >> choice;

invalid = true;

}

}

}

/\*Save results of game and display content\*/

void saveResults(int player1, int player2, int tie){

//Declare variable

int totalGames = player1 + player2 + tie;

//Save final game

time\_t now = time(0);

char\* dt = ctime(&now);

ofstream outputFile;

outputFile.open("saveGame.txt", ios::app);

cout << "Saving scores..." << endl;

outputFile << "Last game saved: " << dt;

outputFile << "Player 1: " << player1 << " Winning ratio: " << ((player1/(float)totalGames)\*100) << "%\r\n";

outputFile << "Player 2: " << player2 << " Winning ratio: " << ((player2/(float)totalGames)\*100) << "%\r\n";

outputFile << "Tie game: " << tie << " tie ratio: " << (tie/(float)totalGames) << "%\r\n";

outputFile << "\r\n";

cout << "Done.\n\n";

//Close file

outputFile.close();

//Reading file

string playResults;

ifstream inputFile;

inputFile.open("saveGame.txt");

//Output file content

cout << "Result from previous games:\n";

while(getline(inputFile, playResults)){

cout << playResults << endl;

}

cout << endl;

//Close file

inputFile.close();

}

void saveGameBoard(int player, char gameBoard[]){

ofstream outputFile; //Open link to output file

outputFile.open("gameBoard.txt", ios::app);

cout << "Saving game board..." << endl;

gameBoard[0] = '0' + player; //Convert single int into char

outputFile << gameBoard[0];

for(int i = 1; i < 10; i++)

outputFile << gameBoard[i];

outputFile << "\r\n";

outputFile.close();

}

void inputFileGameBoard(){

ifstream inputFile; //Open link to existing file

inputFile.open("gameBoard.txt");

cout << "Past game Boards" << endl;

string gameResults; //Hold each gameboards as line

while(getline(inputFile, gameResults)){

char mark;

int countMove = 0; //used count how many moves they won by

if(gameResults[0] == '0') //Check who won

cout << "Game was a tie" << endl;

else{

//Search for how many moves they won by

mark = (gameResults[0] == '1'? 'X' : 'O'); //Hold mark

for(int i = 1; i < gameResults.length(); i++){

if(gameResults[i] == mark)

countMove++;

}

cout << "Winner of this game was player " << gameResults[0]

<< "(" << mark << ") with " << countMove << " moves." << endl;

}

//Display winning board

cout << " | | " << endl;

cout << " " << gameResults[1] << " | " << gameResults[2] << " | " << gameResults[3] << endl;

cout << "\_\_\_\_\_\_\_|\_\_\_\_\_\_\_|\_\_\_\_\_\_\_" << endl;

cout << " | | " << endl;

cout << " " << gameResults[4] << " | " << gameResults[5] << " | " << gameResults[6] << endl;

cout << "\_\_\_\_\_\_\_|\_\_\_\_\_\_\_|\_\_\_\_\_\_\_" << endl;

cout << " | | " << endl;

cout << " " << gameResults[7] << " | " << gameResults[8] << " | " << gameResults[9] << endl;

}

}