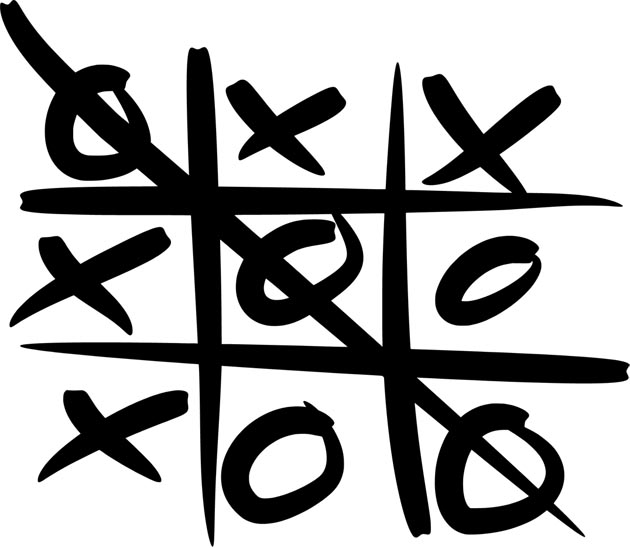
**PROJECT 1**

TIC TAC TOE



1See References

Anh Vu

CSC 5 – 46023

July 18th, 2014

**TABLE OF CONTENTS**

**Cover**……………………………………………………………………………………….…….1

**Introduction**……………………………………………………………………………………...3

**Rules**…………………………………………………………………………………...…3

**Design Details**................................................................................................................................ 4

**Research**.........................................................................................................................................7

**Flowchart**........................................................................................................................................8

**Variable List**.................................................................................................................................18

**Topics Covered**............................................................................................................................18

**Code**..............................................................................................................................................20

**Credits and References**................................................................................................................26

**INTRODUCTION**

Tic-Tac-Toe is a game in which two players, represented as either X or O, mark a 3 x 3 grid until one player wins. The objective of the game is to mark 3 spaces that are horizontally, vertically, or diagonally adjacent to one another before the other player does.

**Rules –**

* Each player will take turns marking the grid. No player will mark the grid more than once at a time.
* You must only mark the empty spaces in the grid.
* Many times you will find that neither player wins. This game is considered a draw, or a cat’s game.

**Instructions –**

* The program will randomly decide which player (‘X’ or ‘O’) will play first
* Each space in the 3 x 3 grid will be represented by a number. Please select the numbered space in which you would like to place your mark. Your mark will be placed on the grid after each time you input.
* The game will continue to run until 1) one person wins, or 2) all spaces are occupied and no one wins.
* If you wish to see your total scores, see the text file names "scores"
* Enjoy the game!

**DESIGN DETAILS**

**Approach -**

In order to successfully code Tic-Tac-Toe, I first played a game and broke the game down in small steps in order to fully understand the sub components of the game and to ultimately implement them in a code. Below are the steps I initially drafted:

1. Draw and output a 3x3 board.
2. Designate each player as either 'X' or 'O'
3. Allow users to take turns and to mark the board with their respective marks.
4. Conditions for the game to end
5. Ways which players can win
6. Keeping scores and outputting them so user can see

**Overview -**

At the end of my project, I had developed a game that had encompassed all of the steps that I had originally thought of. (Step 1) My program outputs a board each time a player makes a move. The board I created consists of a set of variables that represents each space. An example of my board is below:

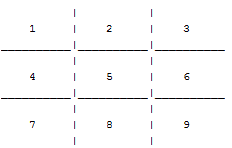


Figure 1. My tic-tac-toe board. In order to make a move, each player is required to input the number where he or she wants to place his or her mark.

If the space is already taken up, the program prompts the user to input another number. (Step 2) My program also designates Player 1 as 'X' and Player 2 as 'O' and randomly chooses which player will be able to play first. Then, the program switches between each player from one move to the next. (Step 3) When the player chooses a number to place his mark, the number in the grid will be substituted with the mark. An example is shown below:

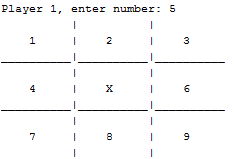


Figure 2. When player 1 (X) chooses number 5, that specific space is then replaced with player 1's mark.

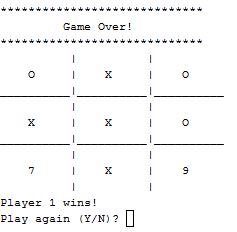
(Step 4) Of course, when playing any game there are conditions to signal the end of the game. The game can end in nine different ways. Three ways the game can is if there are the same marks adjacent to one another in each of the three horizontal rows of the board (e.g. spaces 1, 2, and 3). Another three ways the game can end is the same as the aforementioned condition, but it applies to each of the three vertical columns. Two ways that will end the game is if the same marks are adjacent to one another diagonally. The last way the game can end is if all the spaces are taken up. In this case, neither player wins and the game is considered a draw. Below shows an example of an output to signal the end of the game. Also note that I prompt the user to decide whether he or she wants to play again.

Figure 3. Player 1 wins because he/she placed 3 marks adjacent to one another vertically before the other player could. The program also asks the user if he or she wants to play again.

C:\Users\user\Desktop\Capture.PNG(Step 5) As you can see in *figure 3*, the program is able to determine who wins. The condition that determines who wins is which player is able to end the game first. (Step 6) When each game ends, the scores for each player are subsequently added depending on which player wins. Scores are tallied and output on a text file titled "scores." Below is a sample of a the text file:

Figure 4. Player 1 has one win while Player 2 has none.

The details of my program will be further discussed throughout my report.

**RESEARCH**

In order to code my game, I had to research and study the following topics that were not covered in class thus far:

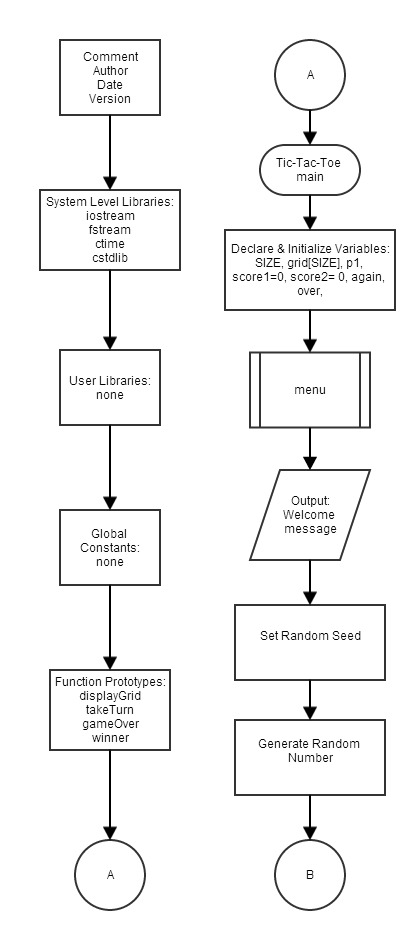
1. **Arrays -**

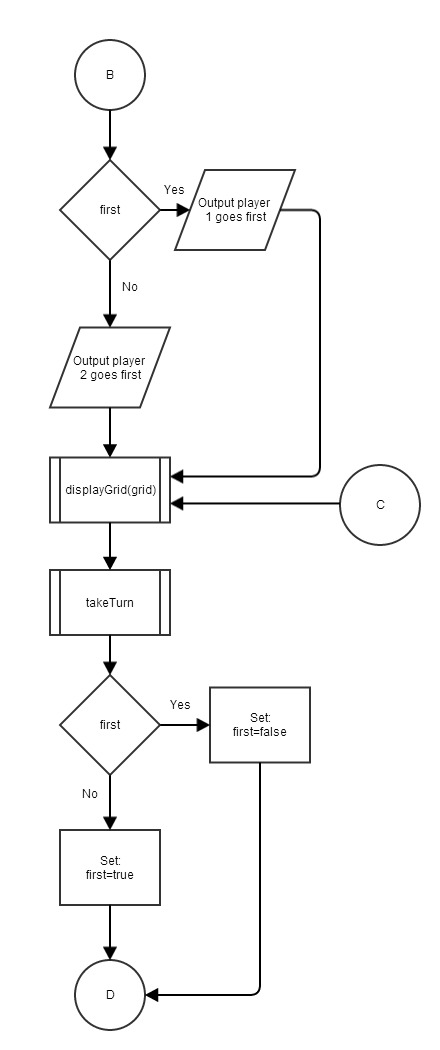
I used arrays so I could hold the variables that would represent each space on the board. When the user inputs where they want their mark, that variable in the array would be replaced with the user's respective mark. I mostly used arrays in order to hold a lot of variables on my tic-tac-toe board; this made it very easy for me to reassign each space with a mark throughout the game.

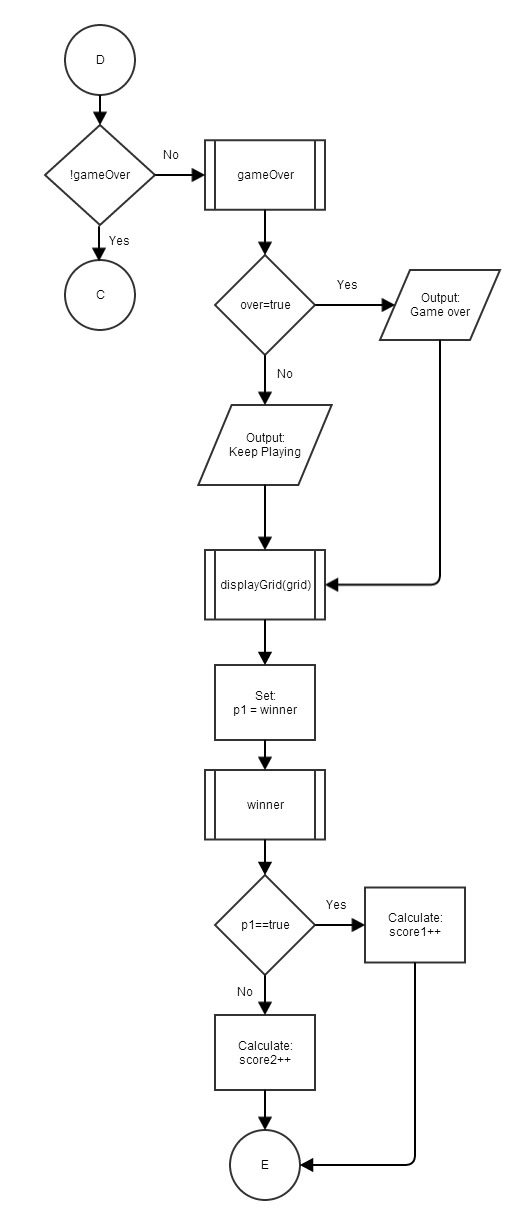
1. **Functions -**

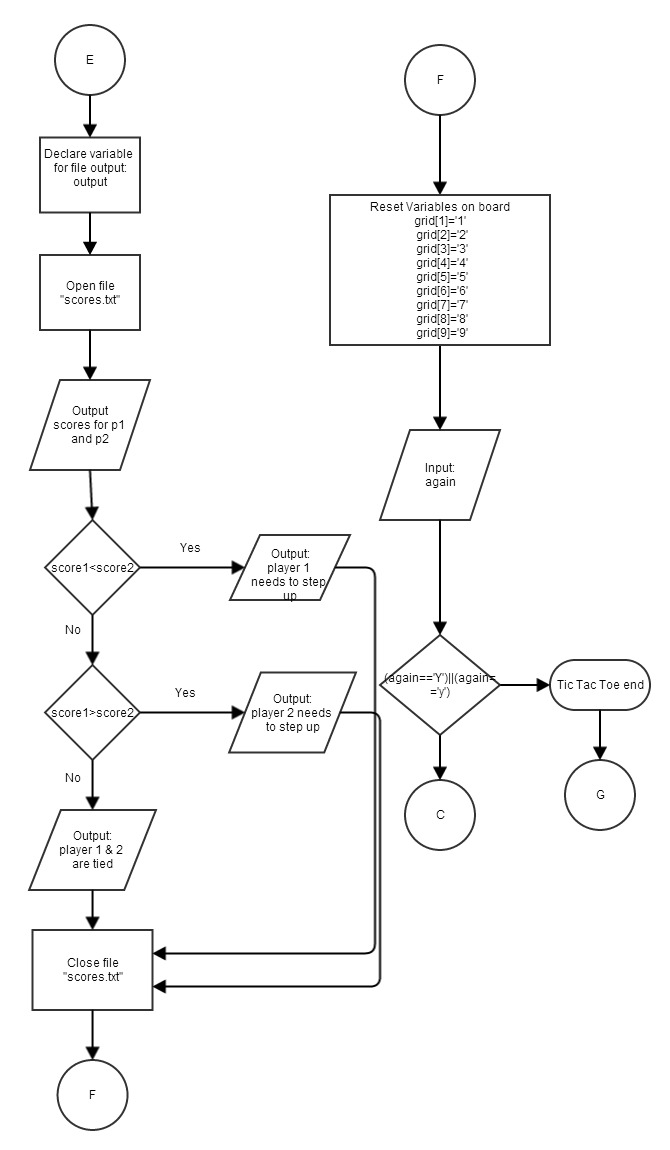
Without functions, my code would have been unnecessarily long. I would have to repeat my code to output a grid every time a player makes a move. The same applies for determining the winner or whether the game is over. Instead of repeating the same lines of code over and over again, declaring a function and defining them would be much easier. Instead of repeating lines of code, I could call functions that would perform these codes. It greatly simplified the layout of my code and made it much easier to read and understand. I have separate functions for printing out the grid, determining whether the game is over, determining the winner, and outputting the right marks on each space.

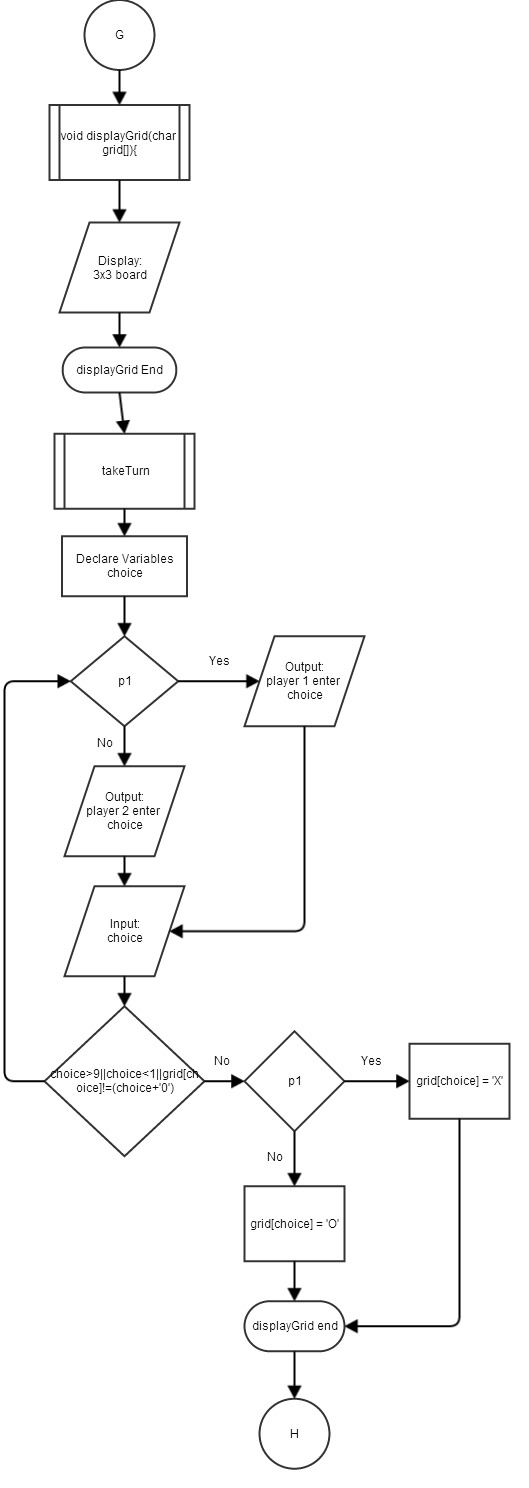
**FLOWCHART**

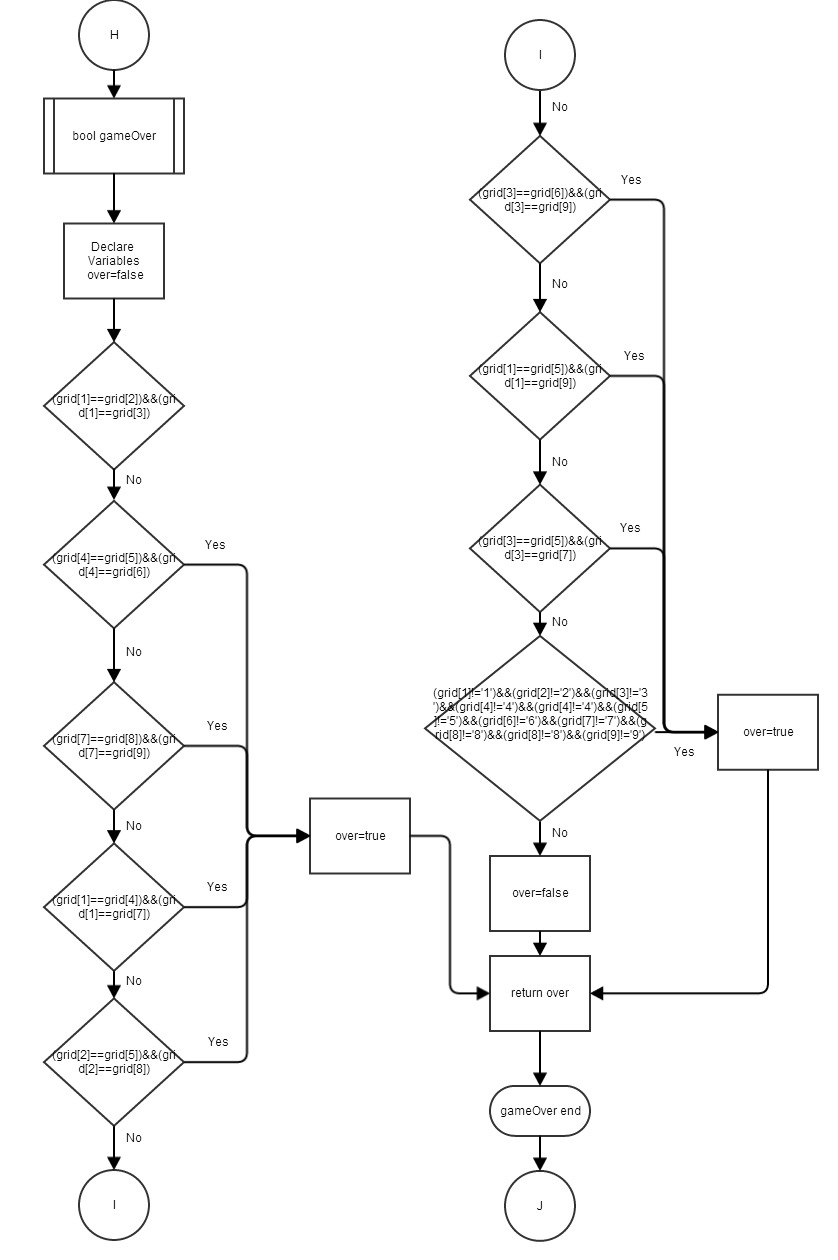
****

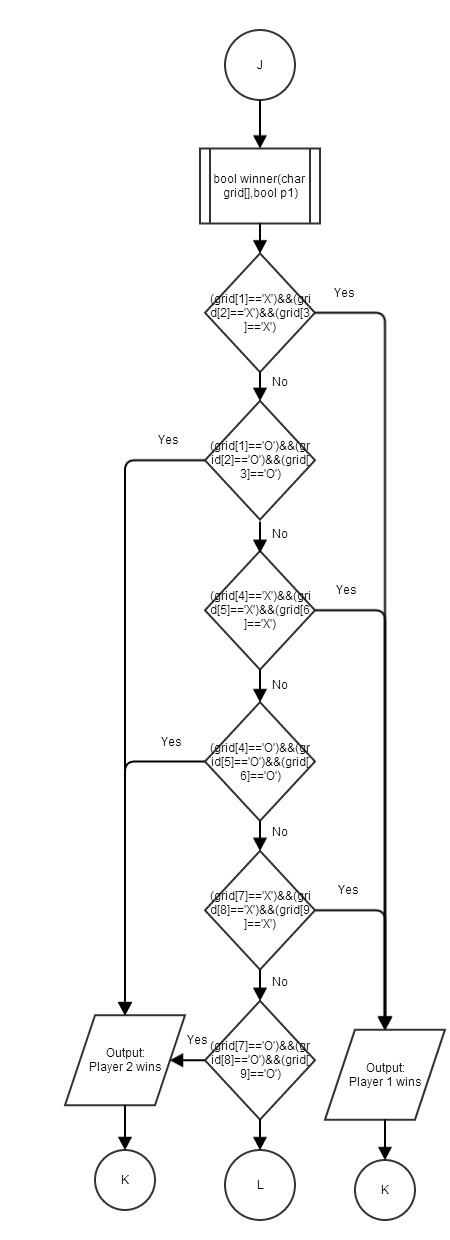
****

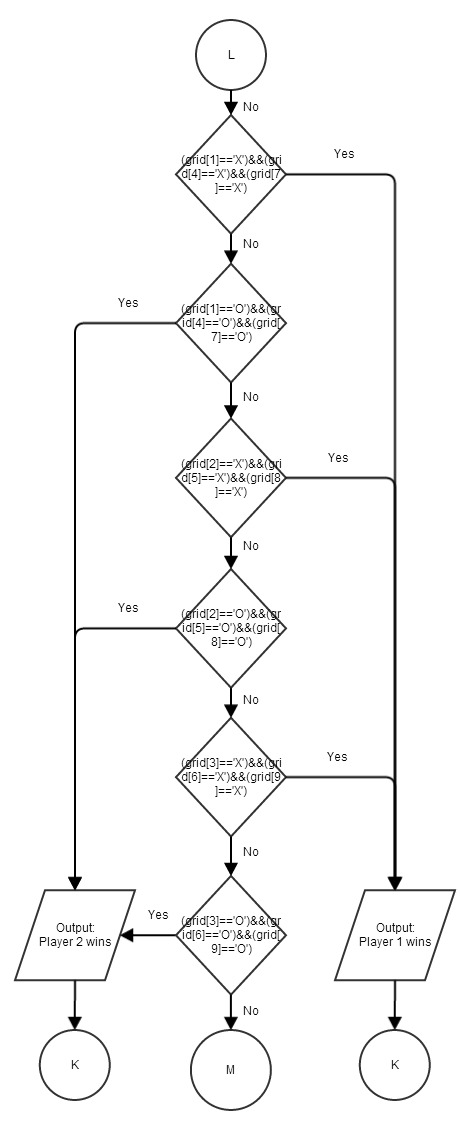
****

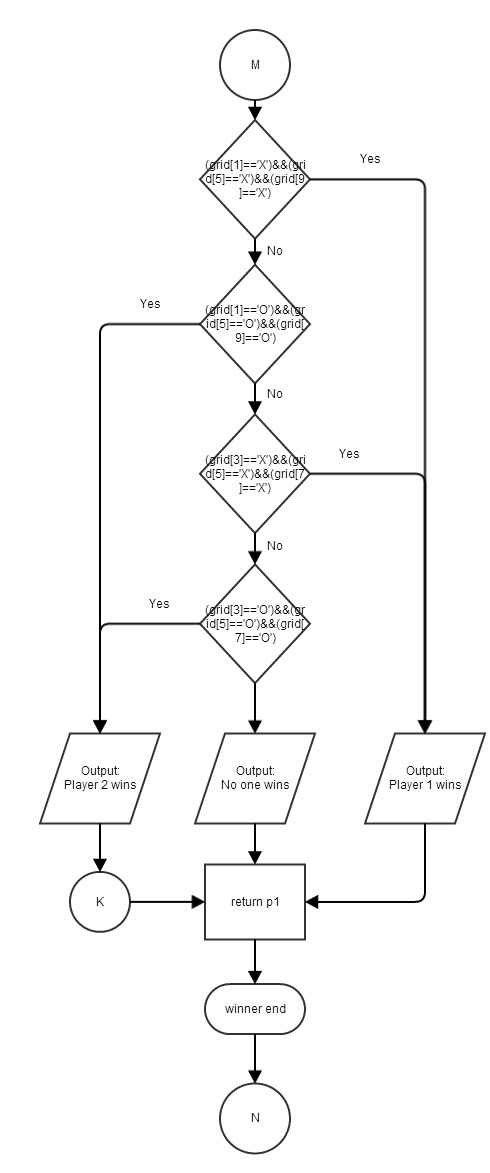
****

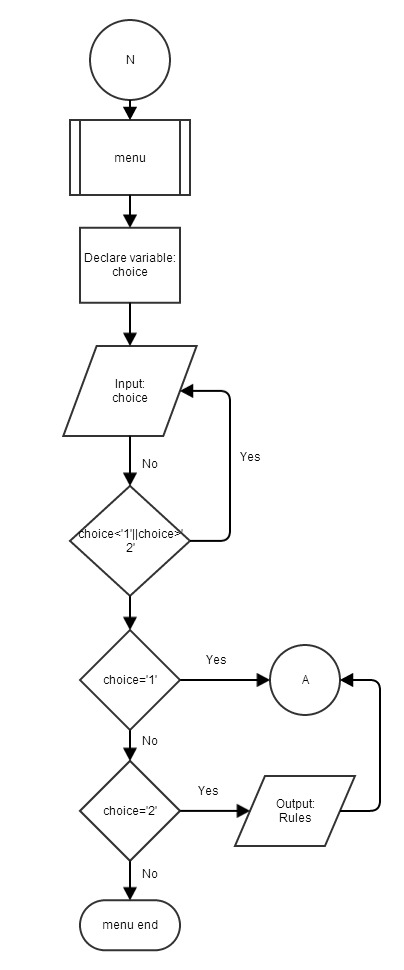
****

****

****







**VARIABLE LIST**

|  |  |
| --- | --- |
| **Variables & System Libraries** | **Purpose** |
| #include <iostream> | Used for cout and cin |
| #include <ctime> | Necessary for setting random seed |
| #include <cstdlib> | Generating random number |
| #include <fstream> | File output |
| const int SIZE=10; | Size of array |
| char grid[SIZE]={'x','1','2','3','4','5','6','7','8','9'} | Array used to hold variables for spaces on the tic tac toe board |
| bool p1 | Decides who's turn it is and which player is which mark |
| unsigned short score1=0 | Used to count score for player 1 |
| unsigned short score2=0 | Count score for player 2 |
| char again | Option that allows user to play game again. |
| bool over | Determine whether the game is over |
| Unsigned short choice | Where user wants to place his or her mark. Also used to allows user to choose from menu |
| Short win | Determine which player wins. |

**TOPICS COVERED**

|  |  |
| --- | --- |
| **Topic** | **Examples** |
| Primitive Data Types | ***Boolean*** (Line 177)  ***Char*** (Line 36)  ***Int*** (Line 30)  ***Short*** (Line 34)  \*I did not use floats, I couldn't think of a way to incorporate a necessary float into my program. |
| System Level Libraries | ***Iostream***  ***Cstdlib***  ***Ctime***  ***Fstream*** |
| Operators | Lines 181-199  211-256  (&&, ||, ==, ++,etc) |
| Conditionals | ***Do While*** (Line 64)  ***If else*** (Line 92) |
| Menu | Line 265 |
| Arrays | Line 32 |
| Functions | Line 256 |

**CODE**

1 /\*

2 \* File: main.cpp

3 \* Author: Anh Vu

4 \* Purpose: Project 1 - Tic Tac Toe!

5 \* Created on July 11, 2014, 11:15 AM

6 \*/

7

8 //System Level Libraries

9 #include <iostream>

10 #include <ctime>

11 #include <cstdlib>

12 #include <fstream>

13 using namespace std;

14

15 //User Libraries

16

17 //Global Constants

18

19 //Function Prototypes

20 void **displayGrid**(char []); //Displays Tic-Tac-Toe board

21 void **takeTurn**(char[], bool); //Places appropriate mark on board

22 bool **gameOver**(char[]); //Determines whether or not game is over

23 short **winner**(char[], short); //Determines winner

24 void **menu**(); //Displays menu

25

26

27 //Execution Begins Here:

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

29 //Declare and Initialize Variables

30 const int SIZE=10;

31 //Array used for board

32 char grid[SIZE]={'x','1','2','3','4','5','6','7','8','9'};

33 bool p1; //Used to determine who's turn it is

34 short win; //Used to determining winner

35 unsigned short score1=0; //Calculate score for player 1

36 unsigned short score2=0; //Calculate score for player 2

37 char again; //Whether players want to play again

38 bool over; //Determines if game is over

39

40 //Display Menu

41 menu();

42

43 //Welcome player and output which player is what mark

44 cout<<**"**Welcome to Tic-Tac-Toe! Who will play first will be randomly chosen.**"**;

45 cout<<**"** May the best man win!**"**<<endl;

46 cout<<**"**Player 1= X**"**<<endl;

47 cout<<**"**Player 2= O**"**<<endl<<endl;

48

49 //Set random seed

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

51

52 //Determine who will go first

53 bool first=rand()%2;

54

55 //Output who will go first

56 if(first){

57 cout<<**"**Congratulations Player 1, you get to play first!**"**;

58 cout<<endl<<endl;

59 }else{

60 cout<<**"**Congratulations Player 2, you get to play first!**"**;

61 cout<<endl<<endl;

62 }

63

64 //Game Play

65 //Reiterates game until players decide to stop

66 do{

67 //Game continues until game is over

68 do{

69 displayGrid(grid); //Display Grid

70 takeTurn(grid,first); //User takes turn

71 //Switches between players 1 & 2

72 if(first){

73 first=false;

74 }else{

75 first=true;

76 }

77 }while(!gameOver(grid));

78

79 //Determines if game is over & outputs game over message

80 gameOver(grid);

81 if (over=true){

82 cout<<**"\n**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***\n"**;

83 cout<<**"** Game Over! **\n"**;

84 cout<<**"**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***\n"**;

85 }

86

87 //Display final board that ends game

88 displayGrid(grid);

89

90 //Displays winner

91 win = winner(grid,win);

92

93 //Keep track of scores

94 if (win==1){

95 score1++;

96 }else if (win==0){

97 score2++;

98 }else if (win==-1){

99 score1+=0;

100 score2+=0;

101 }

102

103 //Output scores to file

104 ofstream output;

105 output.open (**"**scores.txt**"**);

106 output<<**"**Player 1: **"**<<score1<<**"** wins & **"**

107 <<**"**Player 2: **"**<<score2<<**"** wins...........**"**;

108

109 if(score1<score2)

110 output<<**"**Player 1, you need to step up your game!**\n"**;

111 else if (score2<score1)

112 output<<**"**Player 2, you need to step up your game!**\n"**;

113 else

114 output<<**"**Player 1 and Player 2, you are tied.**\n"**;

115

116 //Close File

117 output.close ();

118

119 //Reset board so user can play again

120 grid[1]='1';

121 grid[2]='2';

122 grid[3]='3';

123 grid[4]='4';

124 grid[5]='5';

125 grid[6]='6';

126 grid[7]='7';

127 grid[8]='8';

128 grid[9]='9';

129

130 //Ask if player wants to play again

131 cout<<**"**Play again (Y/N)? **"**;

132 cin>>again;

133 }while((again=='Y')||(again=='y'));

134

135 //Exit Stage Right!

136 return 0;

137

138 }

139

140 //Function displays grid

141 void **displayGrid**(char grid[]){

142 //Board with variables at designated positions

143 cout<<**"** | | **"**<<endl;

144 cout<<**"** **"**<<grid[1]<<**"** | **"**<<grid[2]<<**"** | **"**<<grid[3]<<endl;

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

146 cout<<**"** | | **"**<<endl;

147 cout<<**"** **"**<<grid[4]<<**"** | **"**<<grid[5]<<**"** | **"**<<grid[6]<<endl;

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

149 cout<<**"** | | **"**<<endl;

150 cout<<**"** **"**<<grid[7]<<**"** | **"**<<grid[8]<<**"** | **"**<<grid[9]<<endl;

151 cout<<**"** | | **"**<<endl;

152 }

153

154 //Function places mark on board

155 void **takeTurn**(char grid[], bool p1){

156 unsigned short choice; //Where user wants to place mark (1,2,3...9)

157

158 //Gather Data Input

159 //Takes user's choices as long as spaces are available

160 do{

161 if(p1){

162 cout<<**"**Player 1, **"**;

163 }else{

164 cout<<**"**Player 2, **"**;

165 }

166 cout<<**"**enter number: **"**;

167 cin>>choice;

168 }while(choice>9||choice<1||grid[choice]!=(choice+'0'));

169

170 //If player one makes mark, place an X. If player 2, place O.

171 if(p1){

172 grid[choice] = 'X';

173 }

174 else{

175 grid[choice] = 'O';

176 }

177

178 }

179

180 //Determines whether game is over

181 bool **gameOver**(char grid[]){

182 //Declare Variables

183 bool over=false;

184

185 //Determine if game is over

186 //Game is over if one of the players wins, or if all the spaces are taken up

187 if((grid[1]==grid[2])&&(grid[1]==grid[3]))

188 over=true;

189 else if((grid[4]==grid[5])&&(grid[4]==grid[6]))

190 over=true;

191 else if((grid[7]==grid[8])&&(grid[7]==grid[9]))

192 over=true;

193 else if((grid[1]==grid[4])&&(grid[1]==grid[7]))

194 over=true;

195 else if((grid[2]==grid[5])&&(grid[2]==grid[8]))

196 over=true;

197 else if((grid[3]==grid[6])&&(grid[3]==grid[9]))

198 over=true;

199 else if((grid[1]==grid[5])&&(grid[1]==grid[9]))

200 over=true;

201 else if((grid[3]==grid[5])&&(grid[3]==grid[7]))

202 over=true;

203 else if ((grid[1]!='1')&&(grid[2]!='2')&&(grid[3]!='3')&&(grid[4]!='4')&&

204 (grid[4]!='4')&&(grid[5]!='5')&&(grid[6]!='6')&&(grid[7]!='7')&&

205 (grid[8]!='8')&&(grid[8]!='8')&&(grid[9]!='9'))

206 over=true;

207 else

208 over=false;

209

210 //Over=true, Game over

211 //Over=false, game not over

212 return over;

213 }

214

215 //Determines winner

216 short **winner**(char grid[],short win){

217 win=-1;

218 if((grid[1]=='X')&&(grid[2]=='X')&&(grid[3]=='X')){

219 cout<<**"**Player 1 wins!**"**<<endl;

220 win=1;

221 }else if((grid[1]=='O')&&(grid[2]=='O')&&(grid[3]=='O')){

222 cout<<**"**Player 2 wins!**"**<<endl;

223 win=0;

224 }else if((grid[4]=='X')&&(grid[5]=='X')&&(grid[6]=='X')){

225 cout<<**"**Player 1 wins!**"**<<endl;

226 win=1;

227 }else if((grid[4]=='O')&&(grid[5]=='O')&&(grid[6]=='O')){

228 cout<<**"**Player 2 wins!**"**<<endl;

229 win=0;

230 }else if((grid[7]=='X')&&(grid[8]=='X')&&(grid[9]=='X')){

231 cout<<**"**Player 1 wins!**"**<<endl;

232 win=1;

233 }else if((grid[7]=='O')&&(grid[8]=='O')&&(grid[9]=='O')){

234 cout<<**"**Player 2 wins!**"**<<endl;

235 win=0;

236 }else if((grid[1]=='X')&&(grid[4]=='X')&&(grid[7]=='X')){

237 cout<<**"**Player 1 wins!**"**<<endl;

238 win=1;

239 }else if((grid[1]=='O')&&(grid[4]=='O')&&(grid[7]=='O')){

240 cout<<**"**Player 2 wins!**"**<<endl;

241 win=0;

242 }else if((grid[2]=='X')&&(grid[5]=='X')&&(grid[8]=='X')){

243 cout<<**"**Player 1 wins!**"**<<endl;

244 win=1;

245 }else if((grid[2]=='O')&&(grid[5]=='O')&&(grid[8]=='O')){

246 cout<<**"**Player 2 wins!**"**<<endl;

247 win=0;

248 }else if((grid[3]=='X')&&(grid[6]=='X')&&(grid[9]=='X')){

249 cout<<**"**Player 1 wins!**"**<<endl;

250 win=1;

251 }else if((grid[3]=='O')&&(grid[6]=='O')&&(grid[9]=='O')){

252 cout<<**"**Player 2 wins!**"**<<endl;

253 win=0;

254 }else if((grid[1]=='X')&&(grid[5]=='X')&&(grid[9]=='X')){

255 cout<<**"**Player 1 wins!**"**<<endl;

256 win=1;

257 }else if((grid[1]=='O')&&(grid[5]=='O')&&(grid[9]=='O')){

258 cout<<**"**Player 2 wins!**"**<<endl;

259 win=0;

260 }else if((grid[3]=='X')&&(grid[5]=='X')&&(grid[7]=='X')){

261 cout<<**"**Player 1 wins!**"**<<endl;

262 win=1;

263 }else if((grid[3]=='O')&&(grid[5]=='O')&&(grid[7]=='O')){

264 cout<<**"**Player 2 wins!**"**<<endl;

265 win=0;

266 }else{

267 cout<<**"**No one wins.**"**<<endl;

268 win=-1;

269 }

270

271 //win=1, player 1 wins

272 //win=0, player 2 wins

273 //win=-1, draw

274 return win;

275

276 }

277 void **menu**(){

278 char choice;

279 //Prompt user to input choice

280 do{

281 cout<<**"**Main Menu: **"**<<endl;

282 cout<<**"**[1] Play Game**"**<<endl;

283 cout<<**"**[2] Rules**"**<<endl;

284 cin>>choice;

285 }while(choice<'1'||choice>'2');

286

287 //Output options

288 switch(choice){

289 case '1':

290 cout<<endl;

291 break;

292 case '2':

293 cout<<**"**Rules: **"**<<endl;

294 cout<<**"**1. Decide who will be Player 1 or Player 1**"**<<endl;

295 cout<<**"**2. When making your move, choose the number where you want**"**;

296 cout<<**"** to place your mark.**"**<<endl;

297 cout<<**"**3. Try to get all three of your marks in a row, **"**;

298 cout<<**"** column, or diagonal row before the other player does.**\n"**;

299 cout<<**"**4. After each game, decide if you want to play again.**\n"**;

300 cout<<**"**4. If you want to see a tally of your scores, see the **"**;

301 cout<<**"**text file entitled scores.**"**<<endl;

302 cout<<**"**5. Have fun!**"**<<endl<<endl;

303 break;

304 }

305 }

306

307

**REFERENCES**

1"Tic Tac Toe." [Http://kidavalanche.wordpress.com/2010/02/03/tic-tac-toe-magic-trick prediction/](Http://kidavalanche.wordpress.com/2010/02/03/tic-tac-toe-magic-trick-prediction/). N.p., n.d. Web.

\*Bradd Carey helped me a tremendous amount understanding arrays and functions, so a large amount of my code was successful because of the things he taught me.