**VIET NAM NATIONAL UNIVERSITY,HO CHI MINH CITY**

**UNIVERSITY OF SCIENCE**

**FACULTY OF: INFORMATION TECHNOLOGY**



**Project report**

**| Instructor |**

**Mr. Trương Toàn Thịnh**

**Mr. Nguyễn Lê Hoàng Dũng**

**Mr. Mai Anh Tuấn**

**Mr. Nguyễn Thành An**

Hồ Chí Minh city – 2021

Caro Game

Contents

[How to play. 3](#_Toc79105668)

[1) Starting up. 3](#_Toc79105669)

[2) Play game. 3](#_Toc79105670)

[3) Save game. 5](#_Toc79105671)

[4) Load game. 6](#_Toc79105672)

[5) Instruction. 7](#_Toc79105673)

[6) Exit. 7](#_Toc79105674)

[Note. 8](#_Toc79105675)

[Implementation. 8](#_Toc79105676)

[1) Set up common variables, structure, functions. 8](#_Toc79105677)

[***a.*** ***In use library.*** 8](#_Toc79105678)

[***b.*** ***Global variables.*** 9](#_Toc79105679)

[***c.*** ***Common Functions*** 10](#_Toc79105680)

[2) Menu and in-game screen. 12](#_Toc79105681)

[3) Processing the game. 13](#_Toc79105682)

[4) Saving the game. 17](#_Toc79105683)

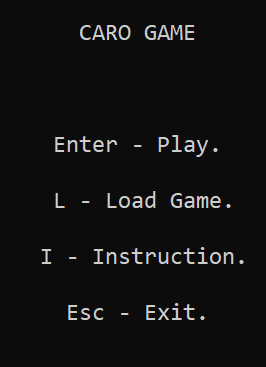
[5) Loading game. 21](#_Toc79105684)

[6) Timer. 24](#_Toc79105685)

[7) The main function. 24](#_Toc79105686)

# How to play.

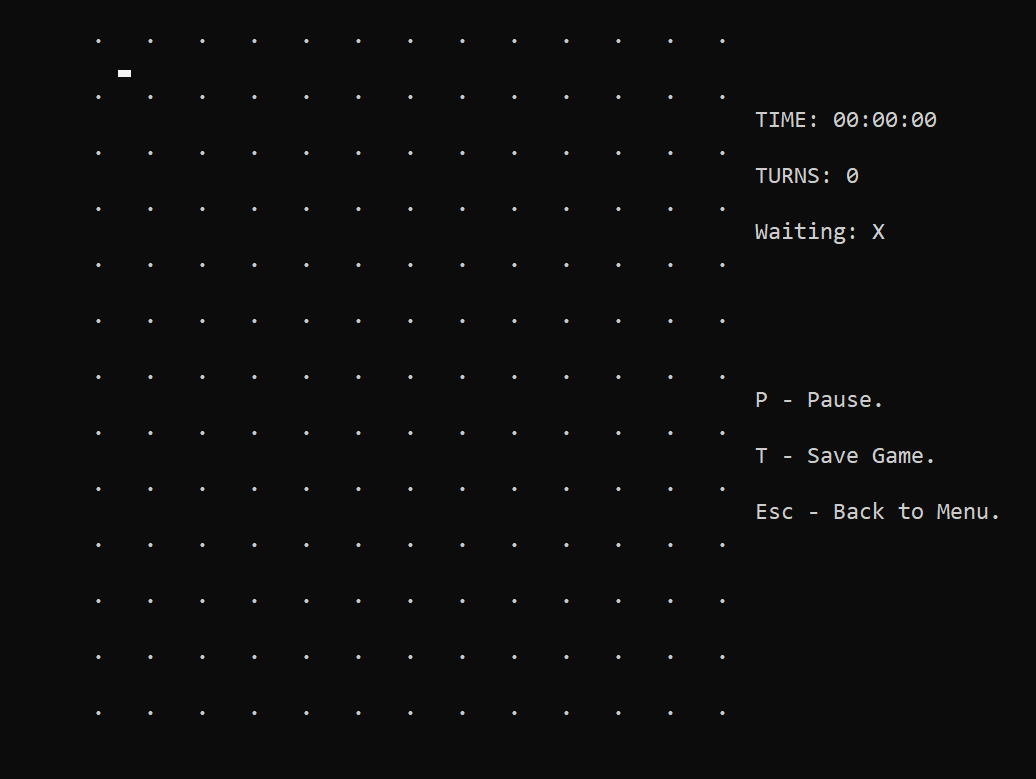
1. **Starting up.**

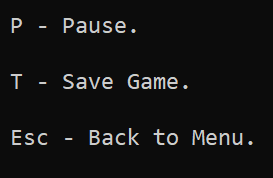
****

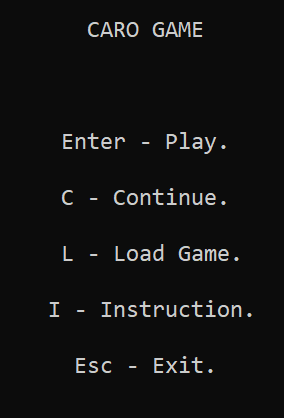
* When the game starts up, the main menu should be display on the user’s screen and each option all have its respective input precede of it.
* From this, player can choose which option they want to:
  + Start a new game.
  + Load a game from files.L
  + Read instruction *“How to play”*.
  + Exit the game.

1. **Play game.**

To start the game, press the *Enter* button while on the main menu. The table of the chess should be appeared:



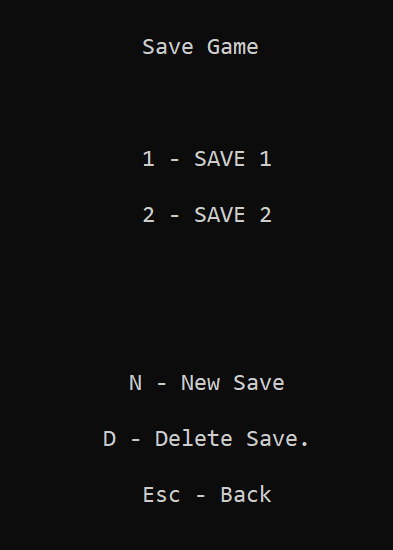
* Now, you can play the game by using *W / A / S / D* to move the curent cursor up / left / down / right and *Enter* to make a move in the current cell.
* From the most left side, there are some indexes, and some addictional operations when the players are playing the game.
  + The vaule after *TIME:* is the current time of the game.
  + The value after *TURNS:* is the number of moves that players have been made.
  + The symbols *‘X’* or *‘O’* after the *Waiting:* shows the current turn of X or O player.
  + Some accessibilities that players can use while playing the game by pressing the respective button like:
    - *P* – to pause the game
    - *T* – to save the current process into a file.
    - *Esc* – Return to the main menu.



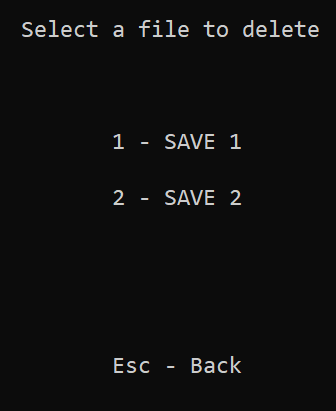
* When the player press *Esc* while in a game, it will bring back to the main menu and this time, it will appear the *‘C - Continue’* option:
* So to come back to the game, simily press *C* and the game should be loaded up again.

1. **Save game.**

While players want to stop playing but the game has not finished yet. Player can press the *T* button to open the save game menu.



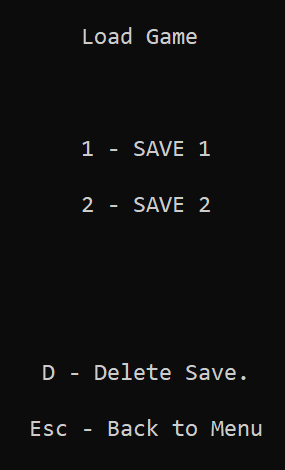
* From here, the user is allowed to choose the way to save the process.
* If the there is no file yet, the program should display *“EMPTY!”*. **
* If the user wants to save into an existed savefile, simply press the niche file’s index to overwrite into it.
* Player can save the process into a brand new file by pressing *“N”* **. It should start creating a new savefile with the following index of the previous file.
* When the a saving action is made, the progress bar should be displayed. 
* If the process of saving is done properly, it should display the “SUCCEED!” message. 
* The user can also delete a file by using the *‘D’* button . The deleting savefile menu should be display:



* From here, the user can select the file by pressing the respective index of it to delete. If they want to go back, just simply press the *Esc*.
* When the user are done, just press the *Esc* button to back to the game. 

1. **Load game.**

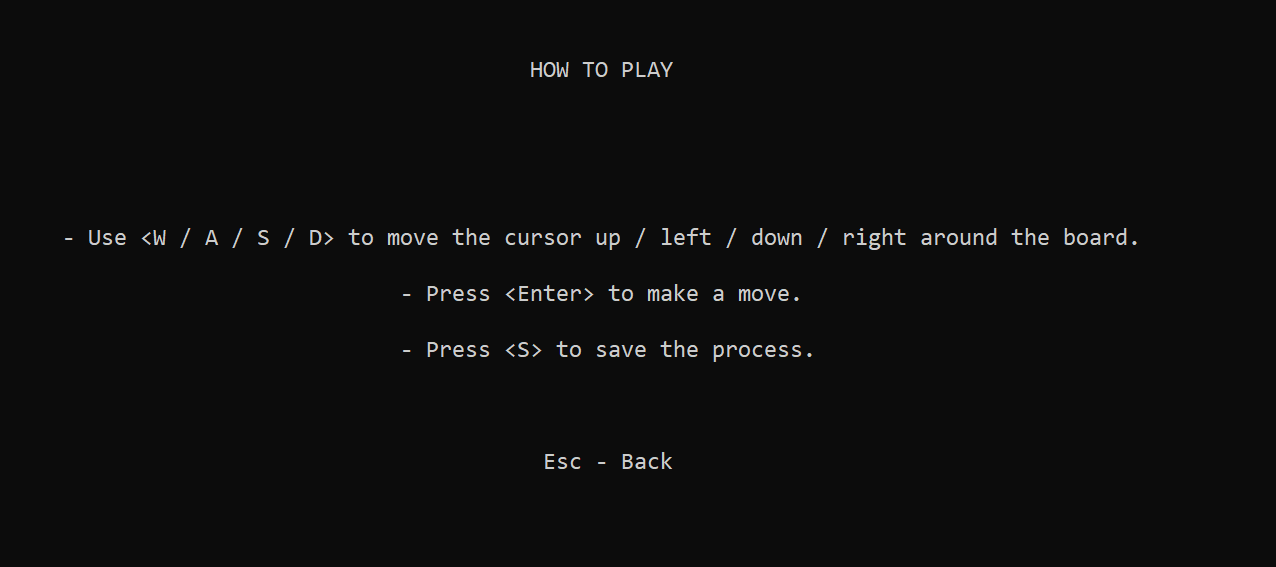
To load the game, press the *L* button while on the main menu. The load game menu should be appeared:



* From here, the user is allowed to choose the savefile to load.
* The user can also delete a file by using the *‘D’* button . The deleting savefile process in load game is similar with save game.
* When the a saving action is made, the progress bar should be displayed . When it is done, the chess board should be displayed on screen and players can continue playing.

1. **Instruction.**

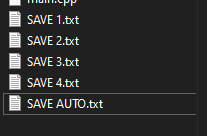
To get to know how to play the game, press the *I* button while on the main menu. The *“HOW TO PLAY”* tab should be appeared:



1. **Exit.**

To exit the game, press the *Esc* button while on the main menu. The message  will pop up on the screen and the program will terminate.

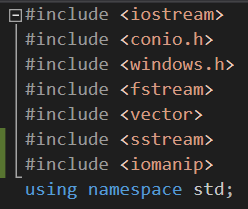
# Note.

* + - **Files.**
    - The game source code file is *main.cpp*.
    - Some save files: *SAVE X.txt* and *SAVE AUTO.txt*.
    - **Classify functions.**
* All the functions are classified in one specific group for easily managing and maintaining.
  + - **Saving files.**
* **All the progress saved files are text files and have the same name format - *“SAVE [FILE\_NUM].txt”*.
* Except for the auto save file is always have the name - *“SAVE AUTO.txt”*.
* Other text files with different name formats will not be read whatsoever.
* The auto save is just a temporary save is easily got replaced.

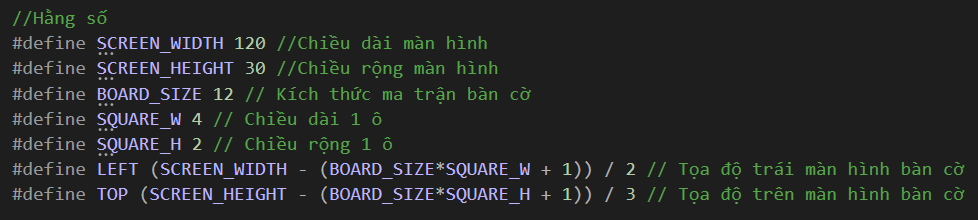
# Implementation.

1. **Set up common variables, structure, functions.**
   1. ***In use library.***

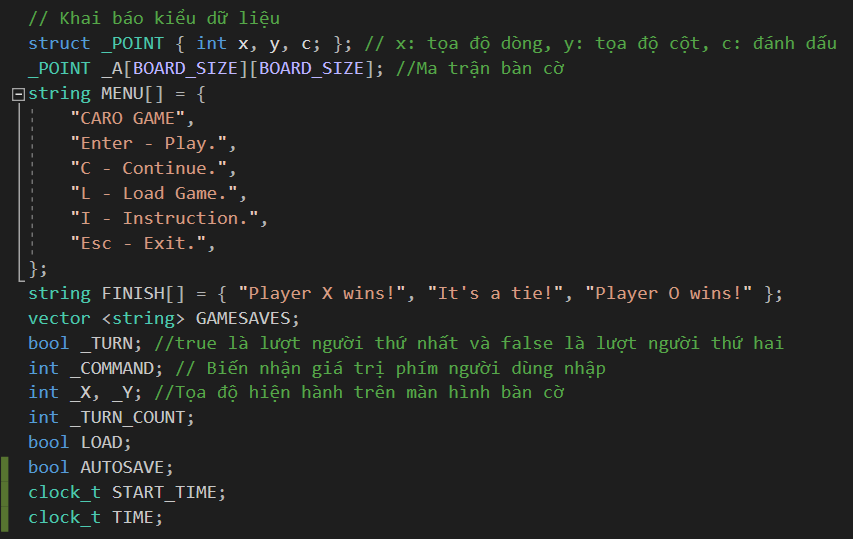
Before starting the implementation, there are 6 built-in library that will be needed to use many utility functions in the process of developing the game include: *iostream.h*, *conio.h*, *windows.h*,*fstream, vector, sstream*.



* 1. ***Global variables.***
     + **Const variables.**

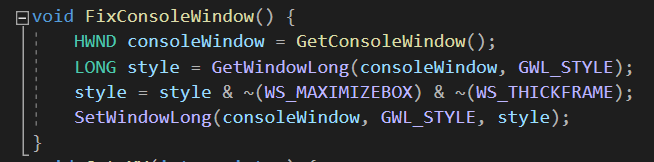


* *#define SCREEN\_WIDTH 120*–width of the screen[[1]](#footnote-1).
* *#define SCREEN\_HEIGHT 30 –* height of the screen1.
* *#define BOARD\_SIZE 12*-definding the size of the square chessboard (The default is 12x12).
* *#define SQUARE\_W 4* – width of a square chess piece.
* *#define SQUARE\_H 2* – height of a square chess piece.
* *#define LEFT (SCREEN\_WIDTH - (BOARD\_SIZE\*4 + 1)) / 2*–is the coordinate from the left of the screen.
* *#define TOP (SCREEN\_HEIGHT - (BOARD\_SIZE\*2 + 1)) / 3 -* is the coordinate from the top of the screen.
  + - **Utility Variables and Structures.**

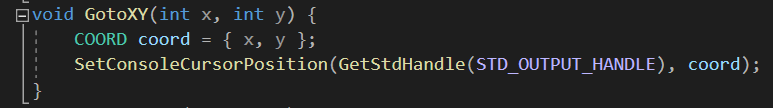


* *struct \_POINT { int x, y, c; };* - define the pointer structure that has 2 integer coordinate *x*, *y* and the *c* will represent whether the cell is checked or not.
* *\_POINT \_A[BOARD\_SIZE][BOARD\_SIZE];* - The 2D array that contains elements and each elements stand for 1 chess piece.
* *string MENU[] = {…};* - The strings that need to be print on screen when user is at the menu.
* *string FINISH[] = {…};* - The strings that need to be print on screen when the game is over.
* *vector <string> GAMESAVES;* - a 1D vector array which elements contain a name of the save files that is currently available.
* *bool \_TURN;* - to know which player’s current turn. (1 is X, 0 is O)
* *int \_COMMAND; -* contains that user typed in.
* *int \_X, \_Y;* - The current x and y coordinate in the game.
* *int \_TURN\_COUNT;* - contains the number of moves has made.
* *bool LOAD;* - notifies the program to load from a file do not to reset the data.
* *bool AUTOSAVE;* - if it is true, it means that the file *“SAVE AUTO.txt”* is available.
* *clock\_t START\_TIME;* - Save the time that from the start of the game.
* *clock\_t TIME;* - Time in game.
  1. ***Common Functions***

*void FixConsoleWindow()*

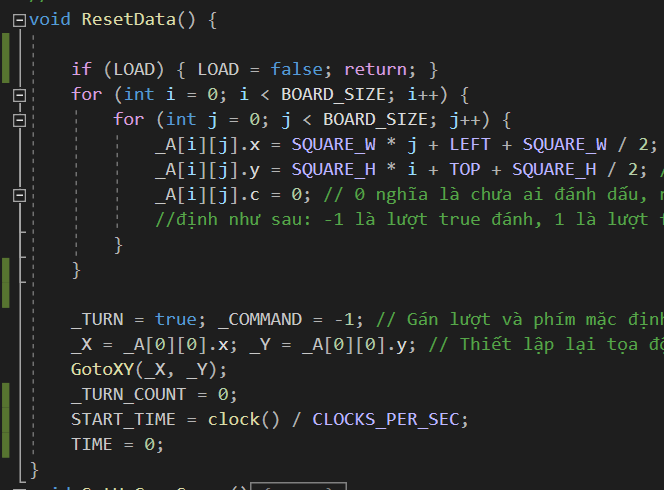


* This function will force the console window to be fixed for the players cannot resize the window while the game is running.



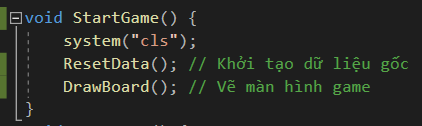
* Jump to the respective x and y coordinate to print on screen something.

*void ResetData();*



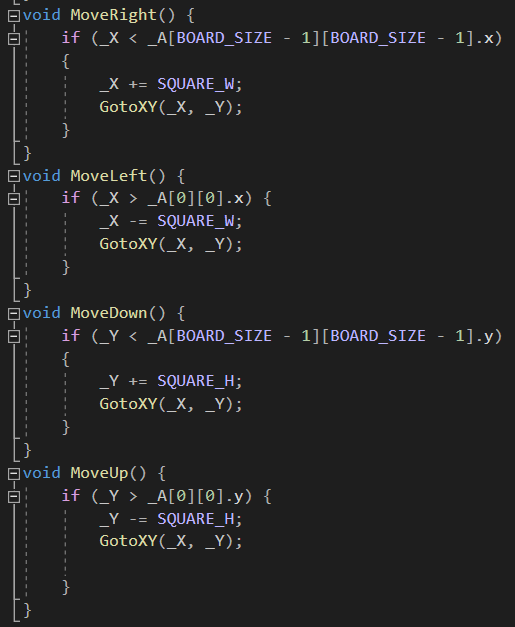
* Set up and initialize every global variables and the board array.
* The first if statement is used to notify the program to know that it is loading from the file, do not reinitialize the data.

*bool StartGame();*



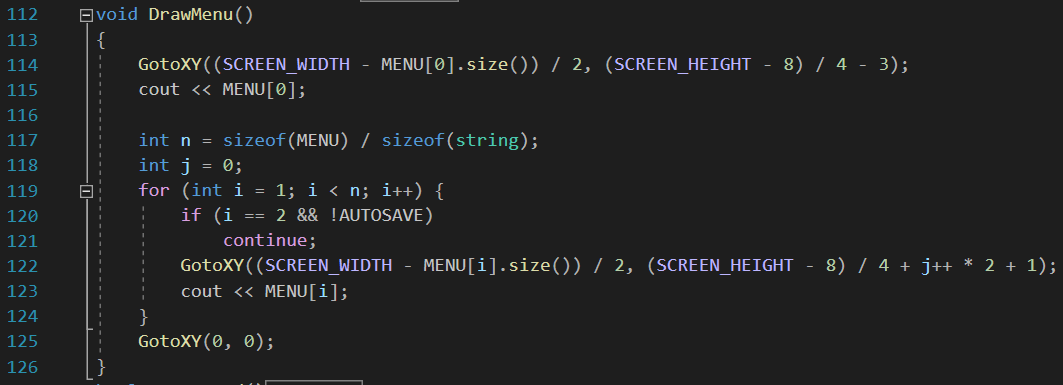
* A warp function will be in the process every time the program starts a game to make sure every thing is set up.

*Moving cursor functions*

* These functions is executed while the player traversing along squares.
* They increase the cursor‘s current position value along X-axis or Y-axis every time the player make a move.

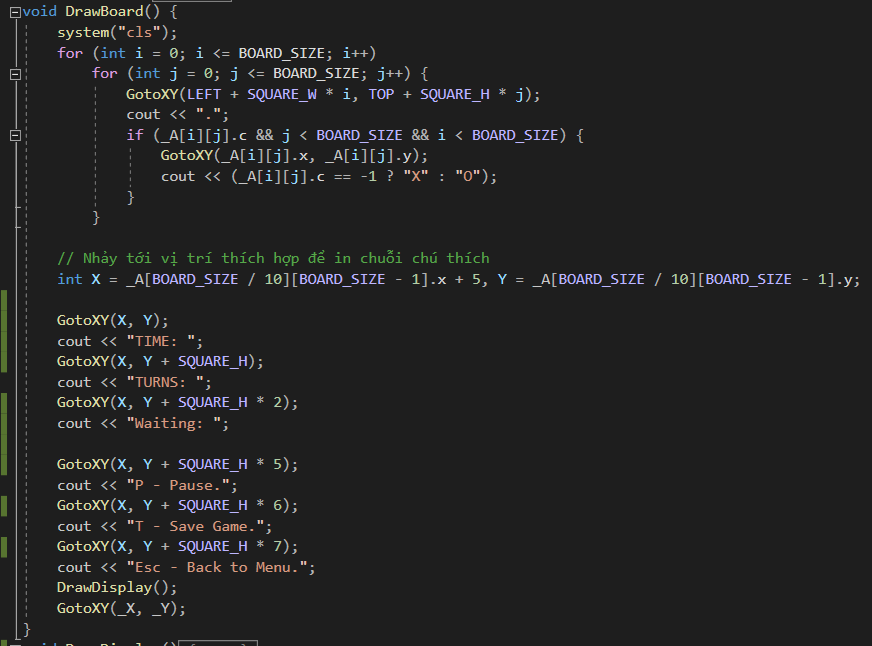
1. **Menu and in-game screen.**

*void DrawMenu();*



* Display all the strings of the menu on the screen whenever the user is at the menu.
* It use *GotoXY* functions to jump to a niche coordinates on the screen to print the strings.
* The first two lines print out the game name and the for loop prints the rest of the menu strings.
* The string on *MENU[2]* *("C - Continue.")* will only be printed when the auto save file is available. The if *(line 120 - 121)* will recognize and refuse to print the string.

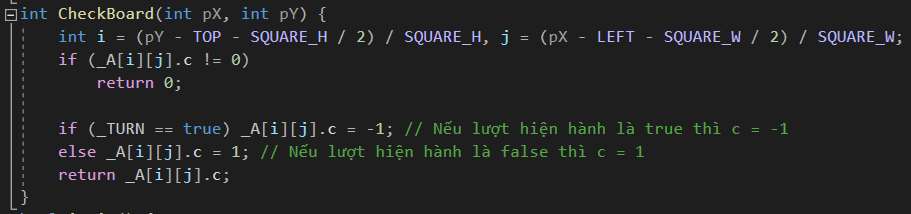
*bool DrawBoard();*



* The first part of this function will handle the process of print the board *(line 129 - 138)* by using a dot as the boundary between square chess pieces.
* The second part display the necessary displayments for indexes or supportive interface for the userto interact. *(line 141 – 153)*.

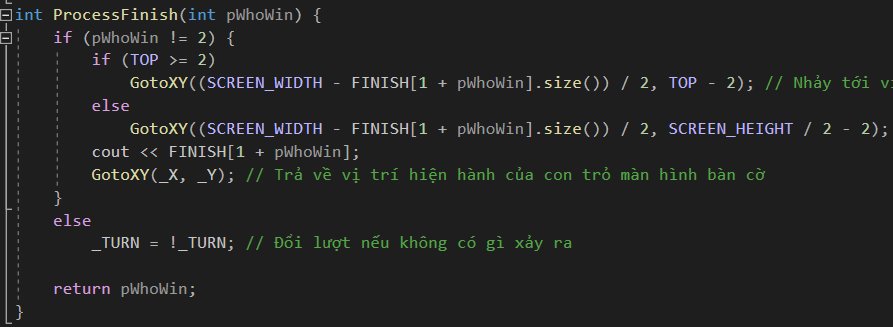
1. **Processing the game.**

*int CheckBoard(int pX, int pY)*

****

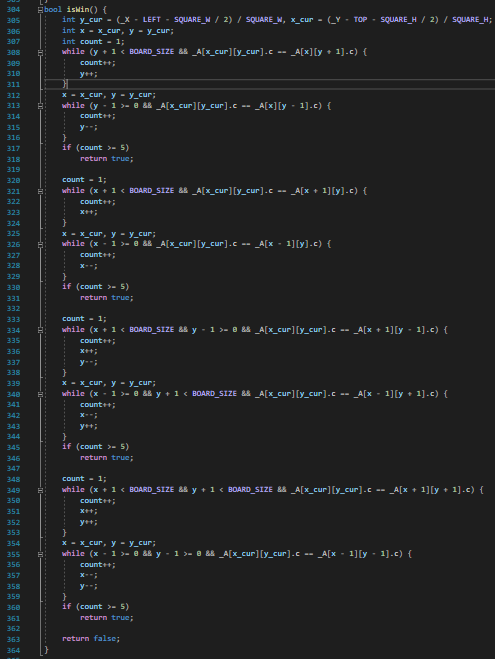
* Is used to check whether the player makes the move on the available square. If so, it will change it into a checked square. If not, it will do nothing and return 0.

*int ProcessFinish(int pWhoWin)*



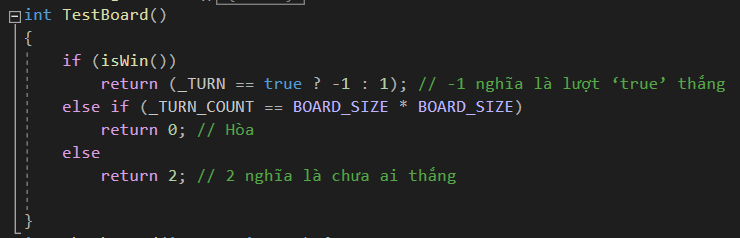
* Print out the *FINISH* strings when the game is finish.
* If no player has won, the game will switch turn and continue.

*bool isWin()*



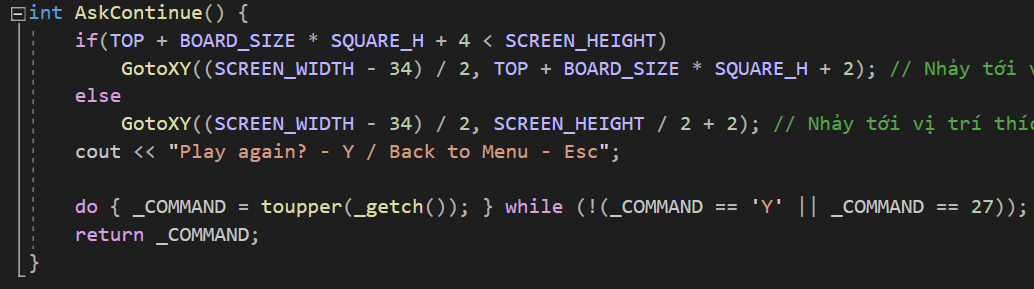
* The algorithm that return the winner by checking all the cases:
  + *line (308 – 318)* check 5 consecutive cells along the vertical row.
  + *line (320 – 331)* check 5 consecutive cells along the horizontal row.
  + *line (333 – 346)* check 5 consecutive cells along the sub diagonal.
  + *line (348 – 361)* check 5 consecutive cells along the main diagonal.
  + If none of the cases above are true, it mean no player has won yet.

*int TestBoard()*



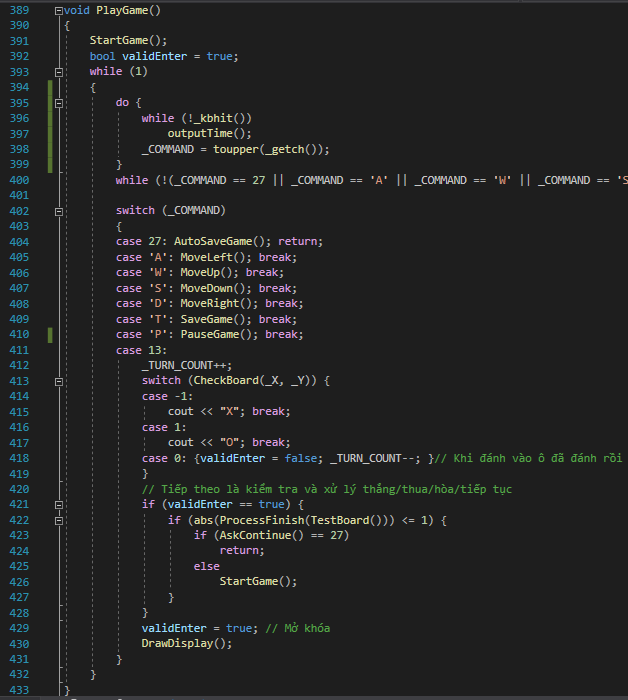
* Depend on the winner that is returned from the *isWin()* functions to decide the action.
* If the number of turns are equal with the number of squares *(BOARD\_SIZE\* BOARD\_SIZE)* so it must be a tie.
* If none of the above, it will return 2 which mean no player has won.

*int AskContinue();*



* Everytime after a game is over, the program will let the players to choose whether they want to play again or return to the menu by using this function.
* The input must be *‘Y’/’y’* or ‘*N’/’n’* to select the option.
* The return will be the option of the players.

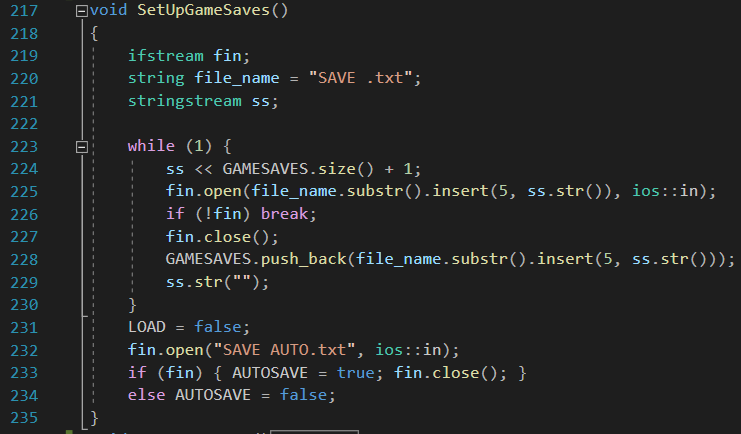
*void PlayGame()*



* When the user choose to play a new game. This function will be executed.
* First, It will build the game by running the *StartGame()* functions
* Then, it start receiving player inputs through the *\_COMMAND* variable and make the right move for each input by using switch statement.
* If the \_COMMAND contains one of the four moving keys *(‘W’, ‘A’, ‘S’, ‘D’)*, the moving cursor functions will be executed.
* The others will be executed with the respective functions like *‘T’* is for saving the game, *‘Esc’/27* is for automatically saving and back to the menu.
* If the \_COMMAND contains the enter key (13), It mean that the player want to make a move in a specific square. So the process of handling it will be operated:
  + *(line 412 - 419)* the \_TURN\_COUNT will count up 1 index, next it will use *CheckBoard* function to check if the move is valid to print the mark or reduce the turn counter back.
  + *(line 421 – 428)* If the move is in a valid cell, it will continue to check whether the game is over by calling the *TestBoard()* and *ProcessFinish()* functions to process the conclusion. Then, It will call *AskContinue()* to take the player’s permission to start the game again.
* Afterall, the DrawDisplay() is called to update the utility indexes.

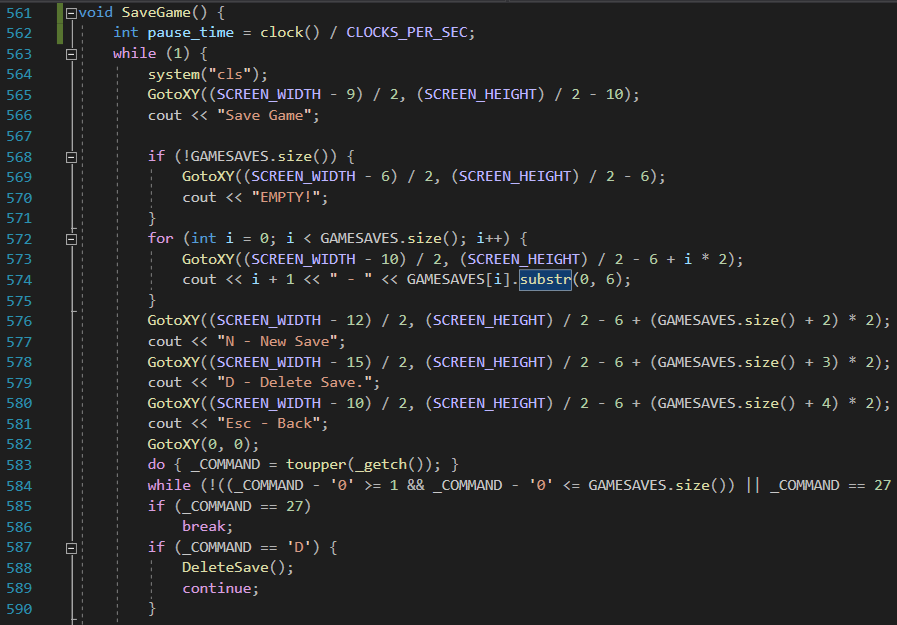
1. **Saving the game.**

*void SetUpGameSaves()*

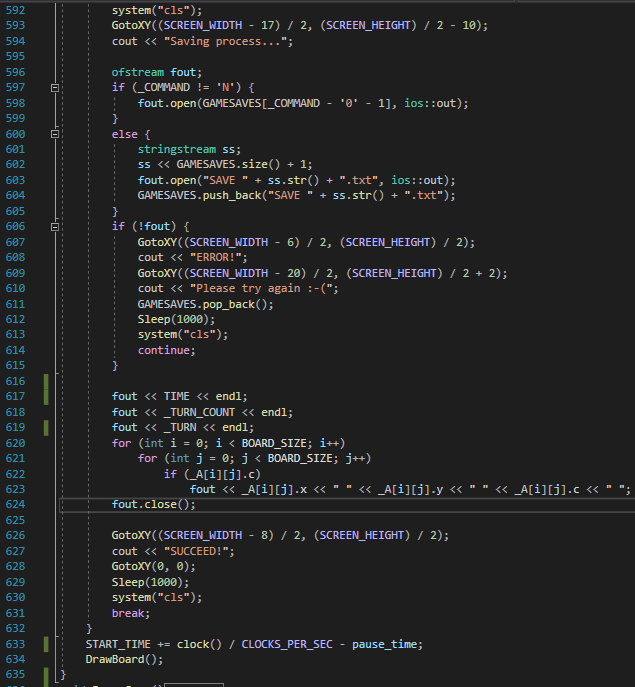
**

* This function will set up all the concepts of saving when the program first executes:
  + *(line 223 – 230)* to check and make all the valid existed save files appears available by pushing back the name strings into the *GAMESAVES* vector.
  + *(line 232 – 234)* to check whether the auto save file is currently exist or not.

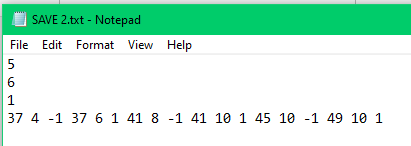
*void SaveGame()*

****

* The first part of the function prints out all the save files for the user to choose which file they want to save into or make a new save file.
* N is for creating a new save file, D is for deleting a save file.
* If there is no save file left. The *”EMPTY!”* notification will be displayed instead of the file names.

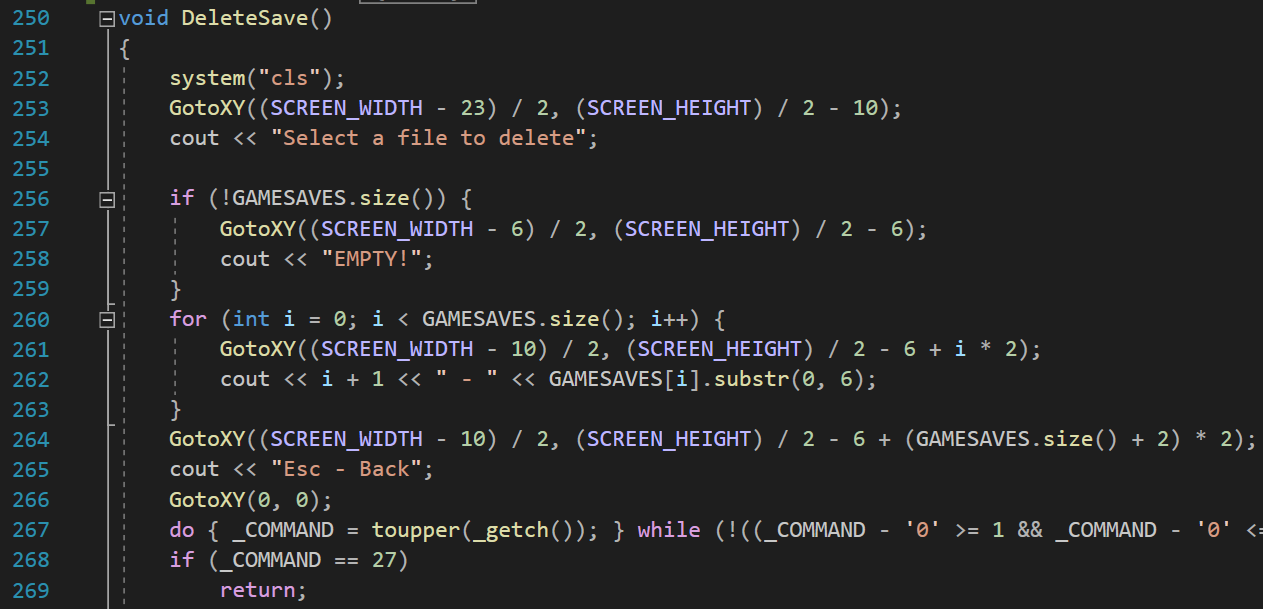
****

* After, the player has entered the file number, the process of saving will be executed:
  + *(line 596 – 605)* the function will try to open the user’s file name. However, if the user want to save into a new file, the function will create the new file with the following index number automatically.
  + *(line 606 – 615)* checks whether the file is opened properly. If the file is corrupted and can not be read, the ”ERROR!” notification will be displayed.
  + *(line 617 – 623)* prints out the all the process of the game with a specific format:

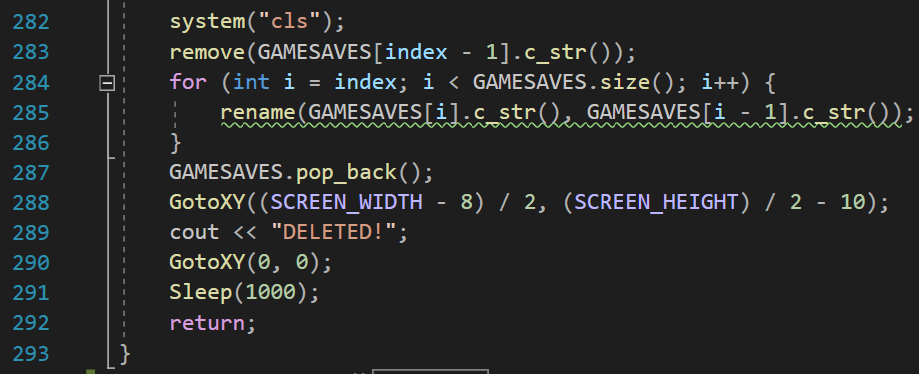


* + - The first line is the time of the game.
    - The second line is the number of moves the players has made (\_TURN\_COUNT).
    - The third line is the player’s current turn (\_TURN).
    - The last line is a long string of numbers each 3 number are the indexes of a cell, include the x and y coordinates and the value that represents the square is checked by which player.
* If the process of saving works properly, the ”SUCCEED!” notification will be displayed.

*void DeleteSave()*

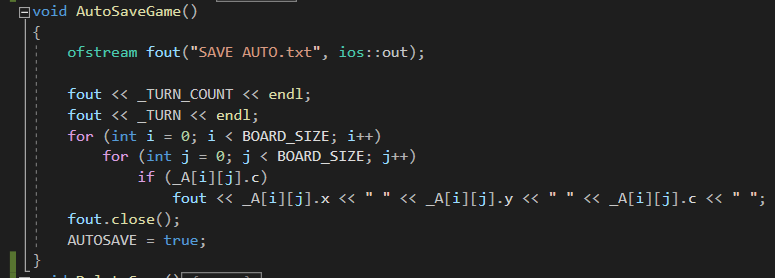


* The first part of the function prints out all the save files for the user to choose which file they want to delete.
* If there is no save file left. The *”EMPTY!”* notification will be displayed instead of the file names.
* Esc will bring back to the previous files menu.



* Next, it will delete the file with the respective user’s input name.
* Then, the for loop will rename files in order to make a sequence of files with consecutive indexes and pop back the last element in the vector array.
* When it is done, the program will display *“DELETED!”*.

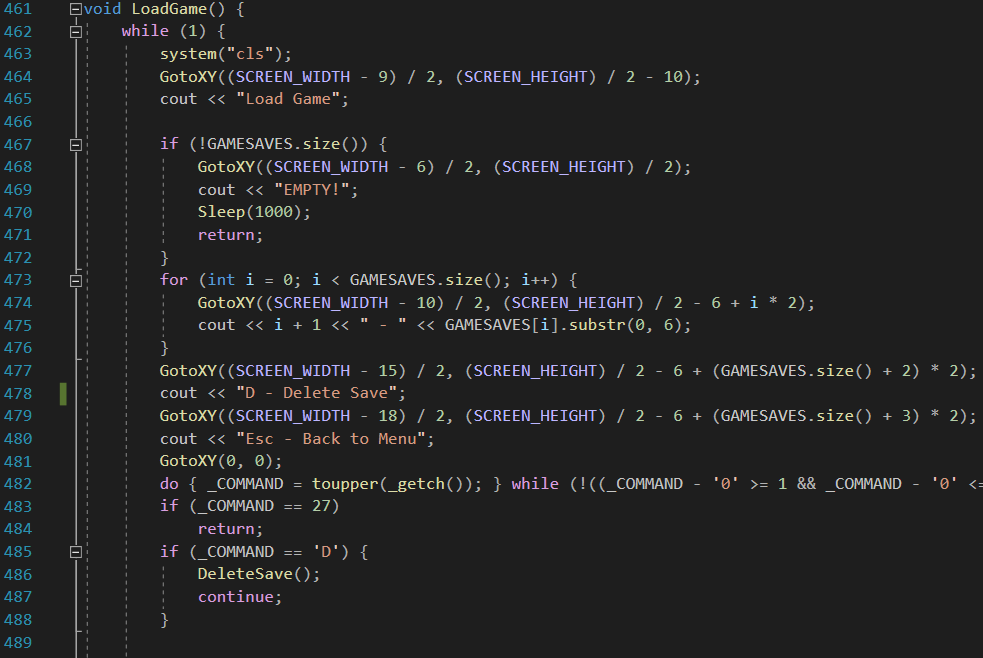
*void AutoSaveGame()*



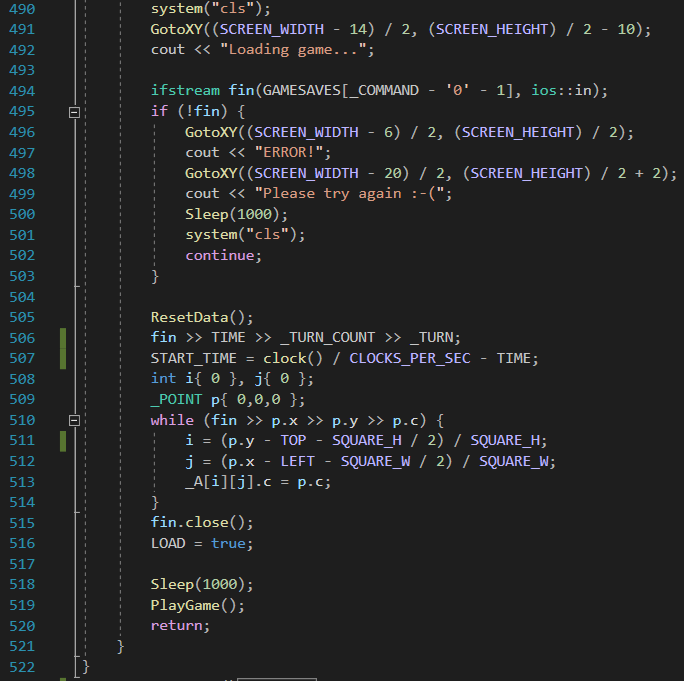
* This special function will help the players get back to the game after they return to the menu by save automatically.
* The file’s name contains the process is different from the rest which is *“SAVE AUTO.txt”*.
* The process is similar with the normal saving process above.
* After saving properly, the *AUTOSAVE* must be change into true which is necessary for the program to know the autosave file have been available.

1. **Loading game.**

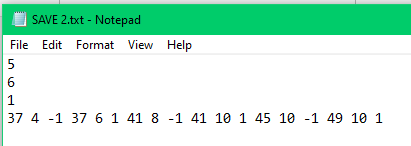
*void LoadGame()*

****

* The first part of the function prints out all the save files for the user to choose which file they want to load.
* D is for deleting a save file.
* If there is no save file left. The *”EMPTY!”* notification will be displayed instead of the file names and return back to menu.

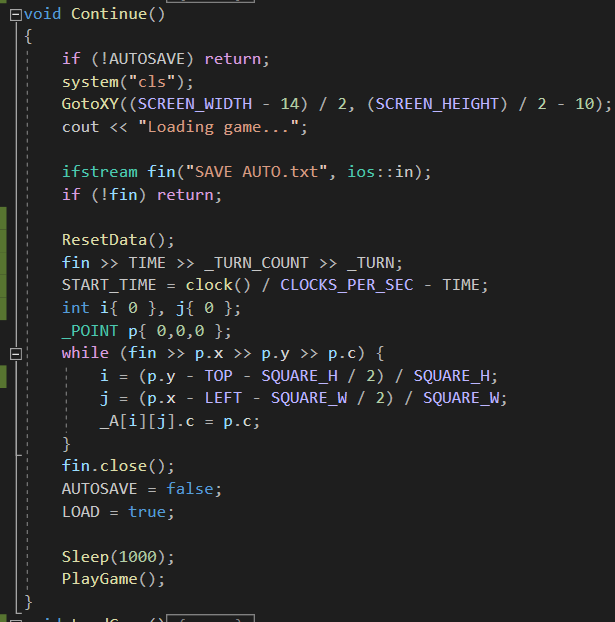
****

* After, the player has entered the file number, the process of saving will be executed:
  + *(line 494)* the function will try to open the user’s file name.
  + *(line 495 – 503)* checks whether the file is opened properly. If the file is corrupted and can not be read, the ”ERROR!” notification will be displayed.
  + *(line 506 – 514)* reset data and read all the all the process of the game with a specific format:



* + - The first line is the time of the game.
    - The second line is the number of moves the players has made (\_TURN\_COUNT).
    - The third line is the player’s current turn (\_TURN).
    - The last line is a long string of numbers each 3 number are the indexes of a cell, include the x and y coordinates and the value that represents the square is checked by which player.
* Also, change the *LOAD* into true so that it will not reset the data after read the file.

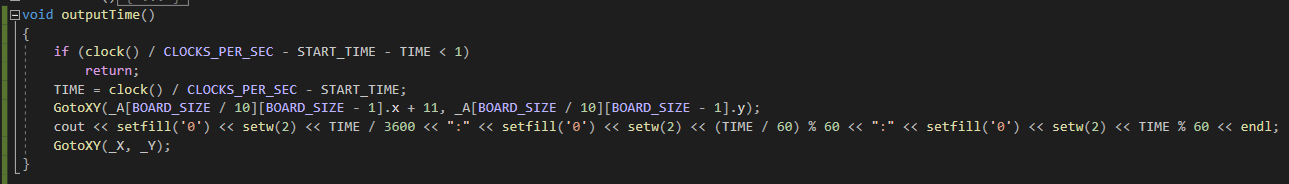
*void Continue()*



* The *Continue()* is one of the main operation at the menu.
* It helps players to load back the process where they have left off after pressing *Esc* by reading the *“SAVE AUTO.txt”* file.
* The process and format of reading the file is very similar to the *LoadGame()* function.
* After reading the file properly, change *AUTOSAVE* to false to make the file unavailable and *LOAD* to true to start the game with the load from file mode.

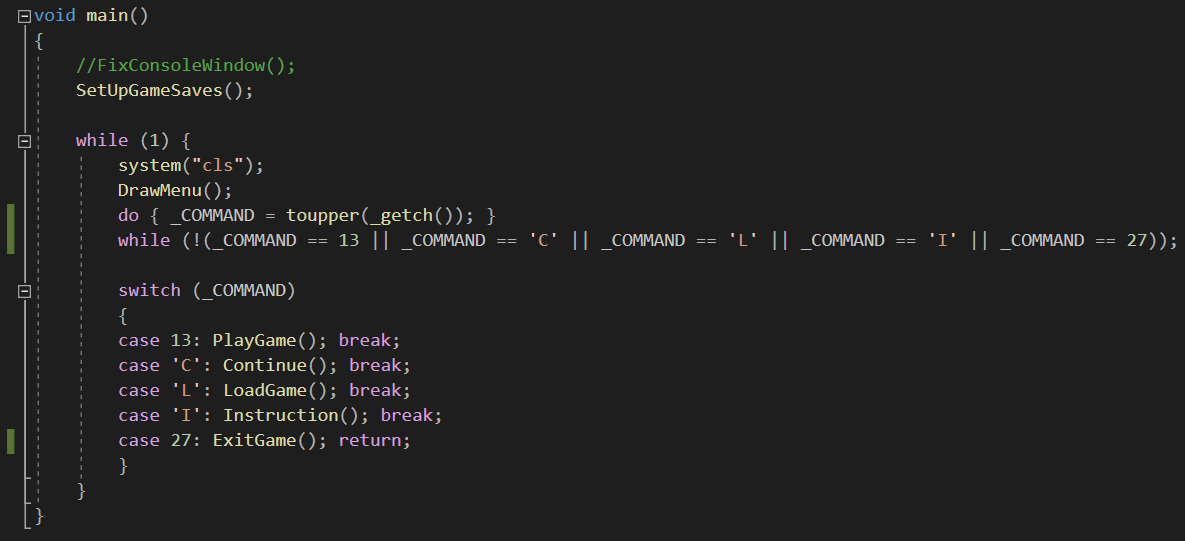
1. **Timer.**

*void outputTime();*

****

* This function will update the time every 1 second and display it next to the *TIME:* value.
* It is used in the *PlayGame()* function.

1. **The main function.**



* The main function is used to control all the operations of the main menu.

1. The size of the screen is depend the current console screen size of user. Typically, It have the size of 120x30. [↑](#footnote-ref-1)