TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN NHẬP MÔN LẬP TRÌNH
……..o0o……..

ĐỒ ÁN HỌC KỲ I

Năm học: 2019 - 2020

Game Caro trên Console Window

Nhóm 4:
- Ngô Trọng Đức – 19120061
- Lê Gia Minh – 19120094
- Nguyễn Minh Phúc – 19120119
- Võ Tiến Dũng – 19120202
- Võ Văn Thắng – 1512527

Giáo viên hướng dẫn: Thầy Trương Toàn Thịnh

Mục lục

1. Lời dẫn

Mọi thứ đều có bắt đầu, và đồ án Caro này cũng như vậy, trong quá trình hoàn thiện đồ án, nhóm đã hợp tác tốt, mọi người tiếp thu rất nhanh những thứ nhóm trưởng đề ra.

2. Giới thiệu trò chơi

Các chức năng chính của trò chơi bao gồm:

+ Bắt đầu trò chơi
+ Save/Load game
+ Bảng xếp hạng
+ Hướng dẫn
+ Thoát game

Về phần trò chơi, người chơi sẽ sử dụng các phím W/A/D/S hoặc các phím mũi tên để di chuyển trên bàn cờ, phím Enter/Space để đánh dấu X/O vào bàn cờ Caro, phím Esc để sử dụng các chức năng phụ khi cần thiết.

Sau đây là giới thiệu về trò chơi cũng như các nhóm hàm mà trò chơi cần.

3. Xây dựng trò chơi

Bước 0: Nhóm hàm Data, nơi chứa những hằng số, cấu trúc cần dùng xuyên suốt chương trình.

```
1  //Console Window
2  #define WIDTH 1320
3  #define HEIGHT 700
4  #define X_CENTER WIDTH / 16 + 4
5  #define Y_CENTER HEIGHT / 32 - 4
6  //Chess Board
7  #define BOARD_SIZE 13
8  #define LEFT 10
9  #define TOP 1
10 #define HORIZONTAL_DISTANCE 4
11 #define VERTICAL_DISTANCE 2
12 //ASCII
13 #define SPACE 32
14 #define ENTER 13
15 #define ESC 27
16 #define ARROW_UP 72
17 #define ARROW_DOWN 80
18 #define ARROW_LEFT 75
19 #define ARROW_RIGHT 77
20 #define TOP_LEFT (char)218
21 #define TOP_RIGHT (char)191
22 #define BOTTOM_LEFT (char)192
23 #define BOTTOM_RIGHT (char)217
24 #define TOP_CROSS (char)194
25 #define BOTTOM_CROSS (char)193
26 #define    LEFT_CROSS (char)195
27 #define RIGHT_CROSS (char)180
28 #define CROSS (char)197
29 #define HORIZONTAL_LINE (char)196
30 #define VERTICAL_LINE (char)179
31 //Playing Constances
32 #define FIRST true
33 #define SECOND false
34 #define P_X -1
35 #define P_O 1
36 //Built-in Libraries
37 #include <iostream>
38 #include <Windows.h>
39 #include <conio.h>
40 #include <string>
41 #include <fstream>
42 #include <vector>
43 //File stuffs
44 #define SAVED_LIST "savedlist.txt" //Ten cac file game duoc
    save
45 #define PLAYER_LIST "PlayerList.txt" //Thong tin cac player
    da choi game
46 //Namespaces
```

```
47  using std::cout;
48  using std::cin;
49  using std::string;
50  //Structs Declaration
51  struct _POINT
52  {
53        int x; //Hoanh do o co
54        int y; //Tung o co
55        int c; /*Kiem tra danh hay chua c = -1: luot X danh
56                                                    c = 1: luot
    O danh
57                                                    c = 0: chua
    danh*/
58  };
59  struct _MENU
60  {
61        int options;      //So chuc nang cua Menu
62        int x;                  // Toa do cua diem bat
63        int y;                  //dau cua chuc nang dau tien
64        int cursorColor; //Mau con tro chon menu
65  };
66  struct _PLAYER
67  {
68        string name; //Ten nguoi choi
69        int wins = 0;     //Dem so tran thang
70        int rank; //Xep hang cua nguoi choi
71
72        bool operator>(_PLAYER other);
73        bool operator<(_PLAYER other);
74        bool operator==(_PLAYER other);
75  };
76  #pragma comment (lib,"winmm.lib") //dung cho PlaySoundA()
```

- Ý nghĩa của các hằng số như trong comment.
- Định nghĩa các hàm chồng toán tử so sánh trong cấu trúc _PLAYER:

```
1   bool _PLAYER::operator>(_PLAYER other)
2   {
3         if (this->wins > other.wins) return true;
4         else return false;
5   }
6
7   bool _PLAYER::operator<(_PLAYER other)
8   {
9         if (this->wins < other.wins) return true;
10        else return false;
11  }
12
13  bool _PLAYER::operator==(_PLAYER other)
14  {
15        if (this->name == other.name) return true;
16        else return false;
17  }
```

- Mục đích của các toán tử so sánh này là để sắp xếp thứ hạng cho người chơi Caro bằng cách so sánh số trận thắng tính đến thời điểm chơi.

Bước 1: Đầu tiên là tạo cửa sổ console với kích thước là hằng số cho trước (trong nhóm Data) (hai hàm này thuộc nhóm hàm View).

- Hàm tạo cửa sổ Console:0

```
1  void CreateConsoleWindow()
2  {
3      HWND consoleWindow = GetConsoleWindow();
4      RECT r;
5      HANDLE hConsole;
6
7      hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
8      SetConsoleTextAttribute(hConsole, 240);
9      GetWindowRect(consoleWindow, &r);
10     MoveWindow(consoleWindow, 0, 0, WIDTH, HEIGHT, TRUE);
11 }
```

GetConsoleWindow trả về con trỏ cửa sổ Window để thực hiện những thao tác tới Window.
Hàm GetWindowRect để đưa dữ liệu về kích thước Window vào con trỏ r kiểu RECT.
Hàm MoveWindow chỉ nơi bắt đầu vẽ cửa sổ, chiều rộng và cao của cửa sổ…
Tạo biến `HANDLE hConsole` để dung biến này chỉnh sửa thuộc tính của các ký tự trên mành hình console qua lênh `SetConsoleTextAttribute(_MENU, int);`

- Hàm cố định cửa sổ Window:

```
1  void FixConsoleWindow()
2  {
3      HWND consoleWindow = GetConsoleWindow();
4      LONG style = GetWindowLong(consoleWindow, GWL_STYLE);
5      style = style & ~(WS_MAXIMIZEBOX) & ~(WS_THICKFRAME);
6      SetWindowLong(consoleWindow, GWL_STYLE, style);
7  }
```

GWL_STYLE được xem là dấu hiệu để hàm GetWindowLong lấy các đặc tính của cửa sổ Console. Hàm trả về một số kiểu long, ta có thể hiệu chỉnh lại tại dòng số 5. Hàm này để không cho người dùng tự thay đổi kích thước cửa sổ hiện hành.

Bước 2: Xây dựng tiếp nhóm hàm View:

- **Hàm di chuyển trên Console:**

```
1  void GotoXY(int x, int y)
2  {
3      COORD coord;
4
5      coord.X = x;
6      coord.Y = y;
7      SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),
coord);
```

```
8  }
```

GotoXY(): di chuyển trên màn hình Console, với tọa độ là kiểu Coord và các thuộc tính (X là hoành độ, Y là tung độ), sử dụng hàm **SetConsoleCursorPosition()** để đưa con trỏ trên Console đến tọa độ coord được truyền vào.

- Các hàm liên quan đến bài trí, hiệu ứng:

```
1  void DrawBox(int color, int width, int height, int x, int y)
2  {
3        SetColor(color);
4        for (int i = 0; i < height; i++)
5        {
6              GotoXY(x, y + i);
7              for (int j = 0; j < width; j++)
8              {
9                    cout << " ";
10             }
11       }
12       SetColor(240);
13 }
14
15 void SetColor(int color)
16 {
17       HANDLE hConsole;
18
19       hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
20       SetConsoleTextAttribute(hConsole, color);
21 }
22
23 void PrintText(string text, int color, int x, int y)
24 {
25       GotoXY(x, y);
26       SetColor(color);
27       cout << text;
28       SetColor(240);
29 }
30 void DrawBigText(string filename, int color, int x, int y)
31 {
32       std::fstream textFile;
33       textFile.open(filename.c_str(), std::fstream::in);
34       string line;
35       std::vector<std::string> line1;
36       int tempY = y;
37       while (getline(textFile, line)) line1.push_back(line);
38       if (filename == "XWin.txt" || filename == "OWin.txt" ||
   filename == "Draw.txt")
39       {
40             int count = 0;
41             while (count <= 48)
42             {
43                   for (int i = 0; i < line1.size(); i++)
```

```
44                          PrintText(line1[i], color + count%10, x, y++);
45                  y = tempY;
46                  Sleep(100);
47                  for (int i = 0; i < line1.size(); i++)
48                  {
49                          string templine = "";
50                          for (int j = 0; j < line1[i].length(); j++) templine += ' ';
51                          PrintText(templine, 240, x, y++);
52                  }
53                  Sleep(100);
54                  y = tempY;
55                  count++;
56              }
57      }
58      for (int i = 0; i < line1.size(); i++)
59      PrintText(line1[i], color, x, y++);
60      textFile.close();
61  }
```

- **DrawBox():** Vẽ hộp có màu, tham số truyền vô là color (màu của hộp), độ rộng, độ cao hộp, và tọa độ điểm bắt đầu vẽ hộp. Vẽ bằng cách in ra các ký tự " " (khoảng trắng) liên tục với màu nền được truyền vô.
- **SetColor():** Chỉnh màu cho các ký tự tiếp theo được output trên Console, HANDLE là con trỏ liên quan tới xử lý sự kiện trên Console Window, **GetStdHandle()** sẽ trả về con trỏ đó. Và hàm **SetConsoleTextAttribute()** sẽ thay đổi thuộc tính màu đó trên cửa sổ Console.
- **PrintText():** In chuỗi ký tự được truyền vô với các tham số gồm màu, tọa độ bắt đầu in chuỗi ký tự, giúp cho việc thể hiện chữ trên Console trở nên linh hoạt hơn.
- **DrawBigText():** In "chữ cỡ bự" lên màn hình dựa vào file txt đã được tạo sẵn nhờ sự trợ giúp của công cụ chuyển từ hình ảnh sang ASCII có trên mạng (xem ở phần nguồn tham khảo). Ngoài ra

- Các hàm Menu:

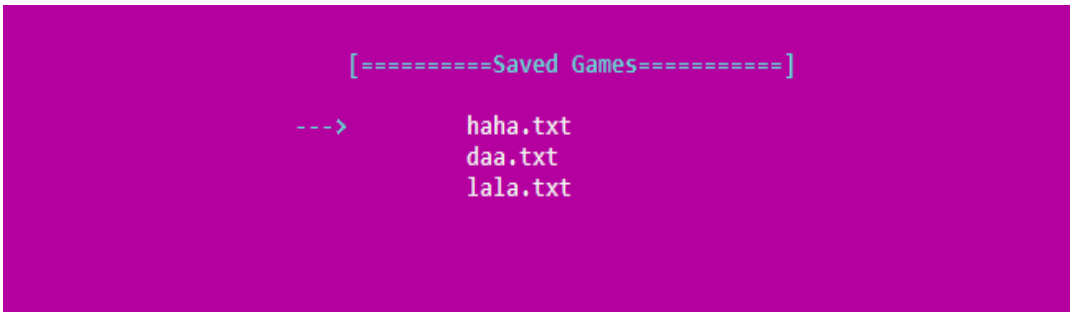
```
1   _MENU MainMenu()
2   {
3       _MENU menu;
4
5       menu.options = 5;
6       menu.x = X_CENTER - 13;
7       menu.y = Y_CENTER + 10;
8       menu.cursorColor = 219;
9
10      system("cls");
11
12      DrawBigText("Logo.txt", 253, 12, 1);
13      DrawBox(221, 50, 13, X_CENTER - 25, Y_CENTER + 5);
14      PrintText("**************************", 219, menu.x, menu.y - 3);
```

```
15      PrintText("*    Welcome to Caroro    *", 219, menu.x, menu.y - 2);
16      PrintText("**************************", 219, menu.x, menu.y - 1);
17      PrintText("         Start Game        ", 219, menu.x, menu.y);
18      PrintText("         Load Game         ", 219, menu.x, menu.y + 1);
19      PrintText("          Ranking          ", 219, menu.x, menu.y + 2);
20      PrintText("           Help            ", 219, menu.x, menu.y + 3);
21      PrintText("         Exit Game         ", 219, menu.x, menu.y + 4);
22
23      return menu;
24  }
25
26  _MENU LoadingMenu()
27  {
28      _MENU menu;
29      string name;
30
31
32      std::vector<string> files;
33      files = LoadFiles();
34
35      menu.options = files.size();
36      menu.x = X_CENTER - 15;
37      menu.y = Y_CENTER - files.size() / 2;
38      menu.cursorColor = 219;
39
40      DrawBox(221, 100, menu.options + 10, X_CENTER - 50, Y_CENTER - 5);
41      PrintText("[==========Saved Games==========]", 219, menu.x, menu.y - 2);
42      for (int i = 0; i < files.size(); i++)
43      {
44          name = "          " + files.at(i);
45          PrintText(name, 223, menu.x, menu.y + i);
46      }
47
48      return menu;
49  }
50
51  _MENU EscMenu(_POINT _A[][BOARD_SIZE])
52  {
53      _MENU menu;
54
55      menu.options = 3;
56      menu.x = _A[0][BOARD_SIZE - 1].x + 65;
```

```
57      menu.y = Y_CENTER + 3;
58      menu.cursorColor = 75;
59
60      //DrawBoard(1, 1, 62, 25, menu.x - 23, menu.y - 19);
61      DrawBox(75, 63, 25, menu.x - 23, menu.y - 19);
62      DrawBigText("EscLogo.txt", 75, menu.x - 22, menu.y -
    17);
63      PrintText("    Continue    ", 75, menu.x, menu.y);
64      PrintText("    Save game   ", 75, menu.x, menu.y + 1);
65      PrintText("    Exit game   ", 75, menu.x, menu.y + 2);
66
67      return menu;
68  }
69
70  _MENU YesNoMenu(int x, int y)
71  {
72      _MENU menu;
73
74      menu.options = 2;
75      menu.x = x;
76      menu.y = y;
77      menu.cursorColor = 249;
78
79      PrintText("Yes", 245, menu.x, menu.y);
80      PrintText("No", 245, menu.x, menu.y + 1);
81
82      return menu;
83  }
```

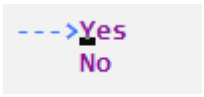
- Các hàm Menu đều trả về giá trị là 1 cấu trúc _MENU, cấu trúc này có các thuộc tính như đã nêu trong Data Header. Các hàm Menu sẽ vẽ các khung, hộp màu cũng như in các chức năng lên màn hình để người chơi đọc và sử dụng, mỗi chức năng nằm trên một hàng, giá trị trả về sẽ được truyền vô hàm **SelectMenu()** trong nhóm hàm Control (sẽ giải thích sau).

MainMenu(): Menu chính trước khi vô game



LoadingMenu(): Danh sách cái Game đã Save để người chơi Load lên chơi tiếp

EscMenu(): Menu hiện ra khi người chơi ấn Esc trong khi đang chơi game (tham số là mảng _A truyền vào với mục đích để lấy tọa độ ngoài cùng bên phải của bàn cờ để canh vị trí in Menu)



YesNoMenu(): Menu để chấp nhận hoặc từ chối (Yes/No), tham số truyền vào là tọa độ bắt đầu vẽ Menu này.

- ## Các hàm phục vụ vẽ bảng (Bàn cờ):

```
1  void DrawTableLine(int numOfCol, char mainSym, char subSym, int width)
2  {
3      for (int i = 0; i < numOfCol - 1; i++)
4      {
5          for (int i = 0; i < width; i++) cout << mainSym;
6          cout << subSym;
7      }
8      for (int i = 0; i < width; i++) cout << mainSym;
9  }
10
11 void DrawBoard(int row, int col, int width, int height, int x, int y)
12 {
13     GotoXY(x, y);
14     cout << TOP_LEFT;
15     DrawTableLine(col, HORIZONTAL_LINE, TOP_CROSS, width);
16     cout << TOP_RIGHT;
17
```

```
18      for (int i = 0; i < (row - 1) * (height + 1); i++)
19      {
20          GotoXY(x, y + i + 1);
21          if ((i + 1) % (height + 1) != 0)
22          {
23              cout << VERTICAL_LINE;
24              DrawTableLine(col, SPACE, VERTICAL_LINE,
    width);
25              cout << VERTICAL_LINE;
26          }
27          else
28          {
29              cout << LEFT_CROSS;
30              DrawTableLine(col, HORIZONTAL_LINE, CROSS,
    width);
31              cout << RIGHT_CROSS;
32          }
33      }
34      for (int i = 0; i < height; i++)
35      {
36          GotoXY(x, y + (row - 1) * (height + 1) + 1 + i);
37          cout << VERTICAL_LINE;
38          DrawTableLine(col, SPACE, VERTICAL_LINE, width);
39          cout << VERTICAL_LINE;
40      }
41
42      GotoXY(x, y + (row - 1) * (height + 1) + 1 + height);
43      cout << BOTTOM_LEFT;
44      DrawTableLine(col, HORIZONTAL_LINE, BOTTOM_CROSS,
    width);
45      cout << BOTTOM_RIGHT;
46  }
```

- Hàm **DrawTableLine()** dung để vẽ 1 hàng trên bảng với các tham số nhận vào là số cột, ký tự chính cho hang, ký tự phụ (ký tự phân cách giữa các cột), độ dài 1 cột
- Hàm **DrawBoard()** dùng để vẽ bảng với các tham số truyền vào là số hang, số cột, độ rộng cột, độ cao hàng và tọa độ nơi bắt đầu vẽ bảng
- Các hàm vẽ trên theo một cấu trúc chung là: vẽ 1 dãy ký tự chính (dựa trên độ rộng cho trước), sau đó vẽ 1 ký tự phân cách, lặp lại như vậy n lần, và cuối cùng là vẽ them 1 lần dãy ký tự chính nữa, như vậy là ta sẽ có n + 1 cột,. Như vậy để vẽ 10 cột thì t sẽ lặp cấu trúc 1 dãy ký tự chính – 1 ký tự phân cách 9 lần, sau đó vẽ thêm 1 dãy ký tự chính. Lúc vẽ bảng chính ta sẽ thêm 1 ký tự bên trái ngoài cùng trước cấu trúc và thêm 1 ký tự ngoài cùng bên phải sau cấu trúc trên..

- Hàm đánh dấu vào bàn cờ dựa trên dữ liệu có sẵn:

```
1   void DrawLoaded(_POINT _A[][BOARD_SIZE])
2   {
3       for (int i = 0; i < BOARD_SIZE; i++)
4       {
5           for (int j = 0; j < BOARD_SIZE; j++)
6               if (_A[i][j].c == P_X)
```

```
7                   {
8                       PrintText("X", 252, _A[i][j].x,
_A[i][j].y);
9                   }
10                  else if (_A[i][j].c == P_O)
11                  {
12                      PrintText("O", 250, _A[i][j].x,
_A[i][j].y);
13                  }
14      }
15  }
```

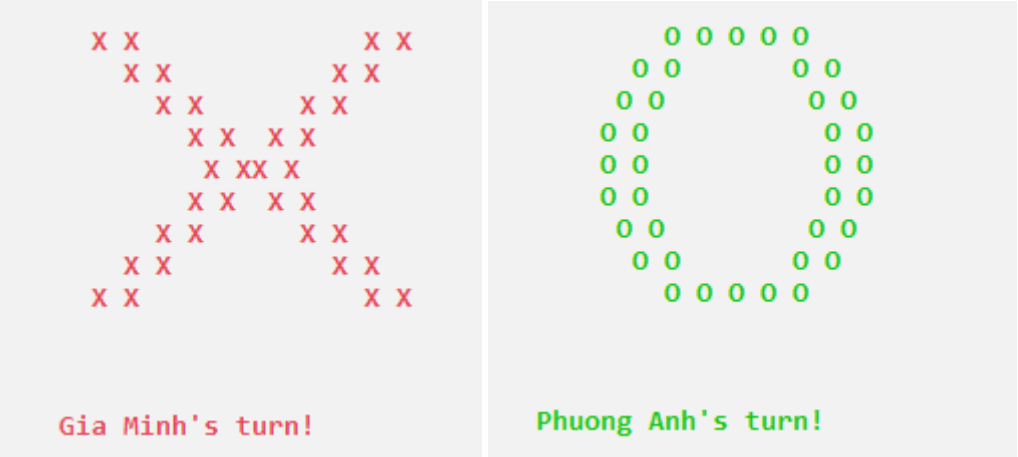
- Hàm **DrawLoaded()** dùng để đánh dấu lên bàn cờ "X" "O" với dữ liệu có sẵn trong mảng _A, hàm này được áp dụng trong việc load game đã lưu.

- Các hàm hiển thị:

```
1   void ShowTurn(_POINT _A[][BOARD_SIZE], _PLAYER _PLAYER1,
_PLAYER _PLAYER2, bool _TURN)
2   {
3       int start = _A[0][BOARD_SIZE - 1].x + 12;
4
5       DrawBox(255, 30, 10, start, 2);
6
7       DrawBigText((_TURN) ? "X.txt" : "O.txt", (_TURN) ? 252
: 250, start, 2);
8
9       DrawBox(255, 20, 1, start - 2, 14);
10      PrintText(((_TURN) ? _PLAYER1.name : _PLAYER2.name) +
"'s turn!", (_TURN) ? 252 : 250, start - 2, 14);
11  }
12
13  void ShowPlayerInfo(_POINT _A[][BOARD_SIZE], _PLAYER
_PLAYER1, _PLAYER _PLAYER2)
14  {
15      int start = _A[0][BOARD_SIZE - 1].x + 4;
16
17      DrawBoard(3, 3, 10, 1, start, 17);
18
19      PrintText(_PLAYER1.name, 253, start + 12, 18);
20      PrintText(_PLAYER2.name, 253, start + 23, 18);
21      PrintText("Win games", 253, start + 1, 20);
22      PrintText(std::to_string(_PLAYER1.wins), 253, start +
12, 20);
23      PrintText(std::to_string(_PLAYER2.wins), 253, start +
23, 20);
24      PrintText("Rank", 253, start + 1, 22);
25      PrintText(std::to_string(_PLAYER1.rank), 253, start +
12, 22);
26      PrintText(std::to_string(_PLAYER2.rank), 253, start +
23, 22);
27  }
28
```

```
29  void ShowHelp()
30  {
31      int x = X_CENTER - 15;
32      int y = Y_CENTER - 10;
33      string line;
34      int i = 0;
35
36      std::fstream helpFile;
37      helpFile.open("Help.txt", std::fstream::in);
38
39      system("cls");
40      while (getline(helpFile, line))
41      {
42          PrintText(line, 243, x, y + i);
43          i++;
44      }
45  }
46
47  void ShowRank()
48  {
49      int x = X_CENTER - 10;
50      int y = Y_CENTER - 10;
51      std::vector<_PLAYER> players = GetPlayerList();
52
53      system("cls");
54
55      PrintText("********************", 253, x, y);
56      PrintText("* ~TOP 10 PLAYERS~ *", 253, x, y + 1);
57      PrintText("********************", 253, x, y + 2);
58      PrintText("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~", 253, x - 10, y + 4);
59      PrintText("[Master King] " + players.at(0).name, 252, x - 2, y + 6);
60      PrintText("[King] " + players.at(1).name, 251, x - 2, y + 7);
61      for (int i = 2; i < 10; i++)
62      {
63          PrintText("[Master] " + players.at(i).name, 249, x - 2, y + 6 + i);
64      }
65
66      PrintText("-Press Esc to turn back-", 253, x - 2, y + 16);
67  }
```

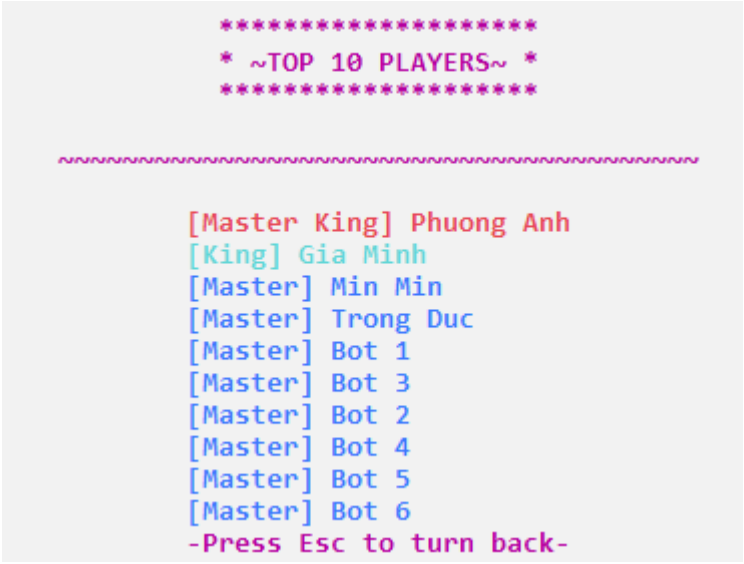
ShowTurn(): Hiển thị lượt của người chơi, cho biết đến lượt người chơi nào.



	Gia Minh	Phuong Anh
Win games	76	77
Rank	2	1

ShowPlayerInfo(): Hiển thị thông tin của người chơi trong trận đấu đó, bao gồm kẻ bảng, hiển thị tên, số trận thắng, và xếp hạng

ShowHelp(): Hiển thị hướng dẫn cũng như luật thắng thua, lấy từ file (“Help.txt”)



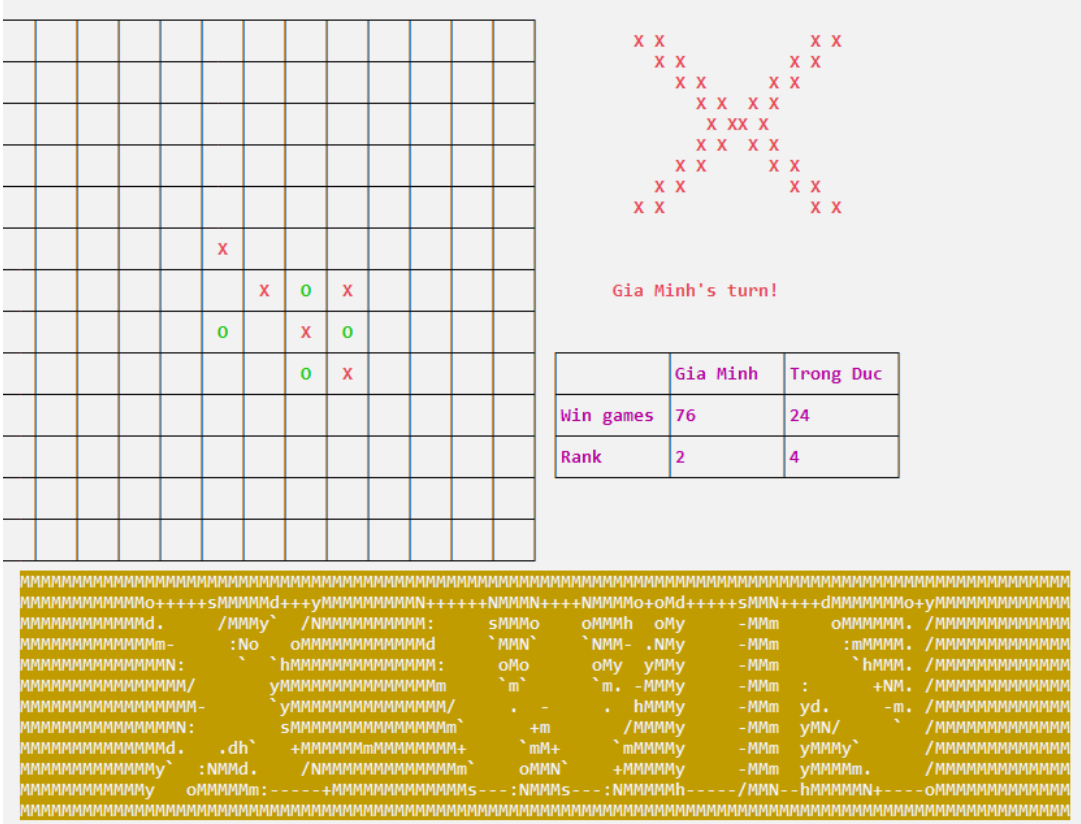
ShowRank(): Hiển thị xếp hạng của 10 người chơi đứng đầu (thắng nhiều nhất), danh sách người chơi đã được sắp xếp là vector players được return từ hàm **GetPlayerList()** (giải thích trong nhóm hàm Model).

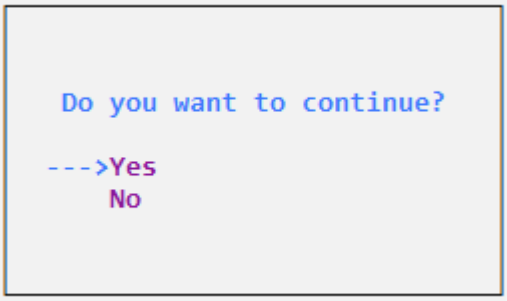
- **Các hàm hiển thị kết quả:**

```
1  int ProcessFinish(_POINT _A[][BOARD_SIZE], _PLAYER& _PLAYER1, _PLAYER& _PLAYER2, bool& _TURN, int pWhoWin)
2  {
3      switch (pWhoWin)
4      {
5      case P_X:
6          _PLAYER1.wins++;
7          DrawBox(111, 100, 12, _A[0][0].x + 20, _A[BOARD_SIZE - 1][0].y + 2);
8          DrawBigText("XWin.txt", 111, _A[0][0].x + 20, _A[BOARD_SIZE - 1][0].y + 2);
9          SavePlayer(_PLAYER1);
10         break;
11     case P_O:
12         _PLAYER2.wins++;
13         DrawBox(111, 100, 12, _A[0][0].x + 20, _A[BOARD_SIZE - 1][0].y + 2);
14         DrawBigText("OWin.txt", 111, _A[0][0].x + 20, _A[BOARD_SIZE - 1][0].y + 2);
15         SavePlayer(_PLAYER2);
16         break;
17     case 0:
18         DrawBox(111, 100, 12, _A[0][0].x + 20, _A[BOARD_SIZE - 1][0].y + 2);
19         DrawBigText("Draw.txt", 111, _A[0][0].x + 20, _A[BOARD_SIZE - 1][0].y + 2);
20         break;
21     case 2:
22         _TURN = !_TURN;
23         ShowTurn(_A, _PLAYER1, _PLAYER2, _TURN);
24         break;
25     }
26     return pWhoWin;
```

```
27  }
28
29  bool AskContinue(int x, int y)
30  {
31      int option;
32
33      option = SelectMenu(YesNoMenu(x, y));
34      switch (RunYesNoMenu(option))
35      {
36      case 'Y':
37          return true;
38      case 'N':
39          return false;
40      }
41  }
```

ProcessFinish(): Hiển thị các hiệu ứng khi người chơi thắng, thua, hòa sau khi kết thúc ván đấu, tăng số trận thắng của người chơi đã thắng và SavePlayer() (giải thích trong nhóm hàm Model) để lưu thông tin người chơi đã thắng lại vô file danh sách người chơi ("PlayerList.txt").





AskContinue(): Hỏi người chơi có muốn tiếp tục chơi hay không (sử dụng YesNoMenu()) để trả lời, hàm thực thi chức năng Menu sẽ được mô tả trong nhóm hàm Control.

Bước 3: Xây dựng nhóm hàm Model:

- **Các hàm liên quan đến xử lý dữ liệu:**

```
1  void ResetData(_POINT _A[][BOARD_SIZE], _PLAYER & _PLAYER1, _PLAYER & _PLAYER2, bool & _TURN, int & _COMMAND, int & _X, int & _Y)
2  {
3      //Khoi tao du lieu ban co
4      for (int i = 0; i < BOARD_SIZE; i++)
5      {
6          for (int j = 0; j < BOARD_SIZE; j++)
7          {
8              //Trung hoanh do va tung do cua ma tran ban co voi console
9              _A[i][j].x = LEFT + j * HORIZONTAL_DISTANCE + 2;
10             _A[i][j].y = TOP + i * VERTICAL_DISTANCE + 1;
11
12             _A[i][j].c = 0; //Tat ca vi tri deu chua danh
13         }
14     }
15     //Khoi tao luot choi
16     _TURN = FIRST;
17     //Khoi tao phim
18     _COMMAND = -1;
19     //Khoi tao vi tri ban dau
20     _X = _A[0][0].x;
21     _Y = _A[0][0].y;
22 }
23
24 void SaveData(string filename, _POINT _A[][BOARD_SIZE], _PLAYER _PLAYER1, _PLAYER _PLAYER2, bool _TURN)
25 {
26     std::fstream saveFile;
27     saveFile.open(filename, std::fstream::out);
28
29     SavePlayer(_PLAYER1);
30     SavePlayer(_PLAYER2);
31
32     saveFile << _PLAYER1.name << "\n";
33     saveFile << _PLAYER2.name << "\n";
34
35     saveFile << _TURN << "\n";
36
37     for (int i = 0; i < BOARD_SIZE; i++)
38     {
39         for (int j = 0; j < BOARD_SIZE; j++)
40         {
```

```
41                  saveFile << _A[i][j].c << " ";
42              }
43              saveFile << "\n";
44          }
45
46          saveFile.close();
47  }
48
49  void LoadData(string filename, _POINT _A[][BOARD_SIZE],
    _PLAYER & _PLAYER1, _PLAYER & _PLAYER2, bool & _TURN, int &
    _COMMAND, int & _X, int & _Y)
50  {
51          std::ifstream loadFile;
52          loadFile.open(filename.c_str());
53
54          getline(loadFile, _PLAYER1.name);
55          _PLAYER1 = LoadPlayer(_PLAYER1);
56
57          getline(loadFile, _PLAYER2.name);
58          _PLAYER2 = LoadPlayer(_PLAYER2);
59          loadFile >> _TURN;
60
61          for (int i = 0; i < BOARD_SIZE; i++)
62          {
63              for (int j = 0; j < BOARD_SIZE; j++)
64              {
65                  _A[i][j].x = HORIZONTAL_DISTANCE * j + LEFT
    + 2; //Trung voi hoanh do ban co
66                  _A[i][j].y = VERTICAL_DISTANCE * i + TOP +
    1; //Trung voi tung do ban co
67                  loadFile >> _A[i][j].c;
68              }
69          }
70          loadFile.close();
71
72          _COMMAND = -1;
73
74          //Thiet lap lai toa do ban dau
75          _X = _A[0][0].x;
76          _Y = _A[0][0].y;
77  }
78
79  std::vector<string> LoadFiles()
80  {
81          std::vector<string> files;
82          string filename;
83
84          std::fstream savedFile;
85          savedFile.open(SAVED_LIST, std::fstream::in);
86
87          while (savedFile >> filename)
```

```
88      {
89            files.push_back(filename);
90      }
91      savedFile.close();
92
93      return files;
94  }
95
96  bool CheckFileExistence(string filename)
97  {
98      string name; // filename cua cac file da luu trong
    savedlist.txt
99      std::fstream savedFile;
100     savedFile.open(SAVED_LIST, std::fstream::in);
101
102     while (savedFile >> name)
103     {
104           if (name == filename)
105           {
106                 savedFile.close();
107                 return true;
108           }
109     }
110
111     savedFile.close();
112     return false;
113 }
114
115
116
```

- Hàm **ResetData()** dùng để khởi tạo dữ liệu bàn cờ (trùng tọa độ x, y của mảng _A với tọa độ bàn cờ trên Console), lượt chơi, biến nhận lệnh nhập từ bàn phím (Khởi tạo là -1 đồng nghĩa chưa có lệnh nào được nhập từ bàn phím), vị trí con nháy ban đầu (trùng với ô cờ đầu tiên trong ma trận bàn cờ Caro)
- Hàm **SaveData()** dùng để lưu dữ liệu bao gồm ma trận bàn cờ, tên người chơi, tọa độ và dấu (kí hiệu X và O) của các ô cờ, lượt đi tại thời điểm lưu.
- Hàm **LoadData()** dùng để tải dữ liệu bao gồm ma trận bàn cờ, người chơi, tọa độ và cờ (X, O) của các ô cờ, lượt đi đã được lưu trong file lưu tương ứng.

- Hàm đánh dấu vào ma trận bàn cờ:

```
1   int CheckBoard(_POINT _A[][BOARD_SIZE], bool _TURN, int cRow,
    int cCol)
2   {
3       if (_A[cRow][cCol].c == 0)
4       {
5             _A[cRow][cCol].c = (_TURN) ? P_X : P_O;
6             return _A[cRow][cCol].c;
7       }
8       else return 0;
9   }
```

- Hàm **CheckBoard()** dùng để đánh dấu vô bàn cờ, bằng cách kiểm tra xem ô đó đã được đánh hay chưa (nếu chưa thì thuộc tính c của mảng _A tại ô đó sẽ = 0). Nếu chưa đánh thì xét _TURN để trả về giá trị c phù hợp, còn nếu đánh vào ô đánh rồi thì trả về 0.

- Các hàm kiểm tra thắng, thua, hòa:

```
1  bool CheckWin(_POINT _A[][BOARD_SIZE], int currentRow, int currentCol)
2  {
3      if (HorizontalCheck(_A, currentRow, currentCol) || VerticalCheck(_A, currentRow, currentCol) || BackwardSlashCheck(_A, currentRow, currentCol) || ForwardSlashCheck(_A, currentRow, currentCol))
4          return true;
5      else return false;
6  }
7
8  bool HorizontalCheck(_POINT _A[][BOARD_SIZE], int currentRow, int currentCol)
9  {
10     int countWin = 1;
11     bool blockRight = false;
12     bool blockLeft = false;
13     bool wall = false;
14     int count = 1;
15     bool keepCounting = true;
16
17     if (currentCol == 0 || currentCol == BOARD_SIZE - 1) wall = true;
18
19     while (currentCol + count < BOARD_SIZE)
20     {
21         if ((_A[currentRow][currentCol + count].c != 0) && (_A[currentRow][currentCol + count].c != _A[currentRow][currentCol].c))
22         {
23             blockRight = true;
24             break;
25         }
26         else if ((_A[currentRow][currentCol + count].c == _A[currentRow][currentCol].c) && keepCounting)
27         {
28             if (currentCol + count == BOARD_SIZE - 1) wall = true;
29             countWin++;
30             count++;
31         }
32         else
33         {
34             count++;
35             keepCounting = false;
```

```
36              }
37        }
38
39        count = 1;
40        keepCounting = true;
41
42        while (currentCol - count > -1)
43        {
44              if ((_A[currentRow][currentCol - count].c != 0) &&
(_A[currentRow][currentCol - count].c !=
_A[currentRow][currentCol].c))
45              {
46                    blockLeft = true;
47                    break;
48              }
49              else if ((_A[currentRow][currentCol - count].c ==
_A[currentRow][currentCol].c) && keepCounting)
50              {
51                    if (currentCol - count == 0) wall = true;
52                    countWin++;
53                    count++;
54              }
55              else
56              {
57                    count++;
58                    keepCounting = false;
59              }
60        }
61
62        if (!blockLeft && !blockRight && !wall && (countWin ==
4))
63              return true;
64        else if (countWin >= 5 && !(blockRight && blockLeft))
65              return true;
66        else
67              return false;
68  }
69
70  bool VerticalCheck(_POINT _A[][BOARD_SIZE], int currentRow,
int currentCol)
71  {
72        int countWin = 1;
73        bool blockUp = false;
74        bool blockDown = false;
75        bool wall = false;
76        int count = 1;
77        bool keepCounting = true;
78
79        if (currentRow == 0 || currentRow == BOARD_SIZE - 1)
wall = true;
80
```

```
81      while (currentRow + count < BOARD_SIZE)
82      {
83          if ((_A[currentRow + count][currentCol].c != 0) &&
        (_A[currentRow + count][currentCol].c !=
        _A[currentRow][currentCol].c))
84          {
85              blockDown = true;
86              break;
87          }
88          else if ((_A[currentRow + count][currentCol].c ==
        _A[currentRow][currentCol].c) && keepCounting)
89          {
90              if (currentRow + count == BOARD_SIZE - 1)
        wall = true;
91              countWin++;
92              count++;
93          }
94          else
95          {
96              count++;
97              keepCounting = false;
98          }
99      }
100
101     count = 1;
102     keepCounting = true;
103
104     while (currentRow - count > -1)
105     {
106         if ((_A[currentRow - count][currentCol].c != 0) &&
        (_A[currentRow - count][currentCol].c !=
        _A[currentRow][currentCol].c))
107         {
108             blockUp = true;
109             break;
110         }
111         else if ((_A[currentRow - count][currentCol].c ==
        _A[currentRow][currentCol].c) && keepCounting)
112         {
113             if (currentRow - count == 0) wall = true;
114             countWin++;
115             count++;
116         }
117         else
118         {
119             count++;
120             keepCounting = false;
121         }
122     }
123
124     if (!blockUp && !blockDown && !wall && (countWin == 4))
```

```
125             return true;
126         else if (countWin >= 5 && !(blockUp && blockDown))
127             return true;
128         else
129             return false;
130 }
131
132 bool ForwardSlashCheck(_POINT _A[][BOARD_SIZE], int currentRow, int currentCol)
133 {
134         int countWin = 1;
135         bool blockUp = false;
136         bool blockDown = false;
137         bool wall = false;
138         int count = 1;
139         bool keepCounting = true;
140
141         if (currentCol == 0 || currentCol == BOARD_SIZE - 1 || currentRow == 0 || currentRow == BOARD_SIZE - 1) wall = true;
142
143         while (currentRow + count < BOARD_SIZE && currentCol - count > -1)
144         {
145             if ((_A[currentRow + count][currentCol - count].c != 0) && (_A[currentRow + count][currentCol - count].c != _A[currentRow][currentCol].c))
146             {
147                 blockDown = true;
148                 break;
149             }
150             else if ((_A[currentRow + count][currentCol - count].c == _A[currentRow][currentCol].c) && keepCounting)
151             {
152                 if (currentRow + count == BOARD_SIZE - 1 || currentCol - count == 0) wall = true;
153                 countWin++;
154                 count++;
155             }
156             else
157             {
158                 count++;
159                 keepCounting = false;
160             }
161         }
162
163         count = 1;
164         keepCounting = true;
165
166         while (currentRow - count > -1 && currentCol + count < BOARD_SIZE)
167         {
```

```
168             if ((_A[currentRow - count][currentCol + count].c != 0) && (_A[currentRow - count][currentCol + count].c != _A[currentRow][currentCol].c))
169             {
170                 blockUp = true;
171                 break;
172             }
173             else if ((_A[currentRow - count][currentCol + count].c == _A[currentRow][currentCol].c) && keepCounting)
174             {
175                 if (currentCol + count == BOARD_SIZE - 1 || currentRow - count == 0) wall = true;
176                 countWin++;
177                 count++;
178             }
179             else
180             {
181                 count++;
182                 keepCounting = false;
183             }
184         }
185
186         if (!blockUp && !blockDown && !wall && (countWin == 4))
187             return true;
188         else if (countWin >= 5 && !(blockUp && blockDown))
189             return true;
190         else
191             return false;
192 }
193
194 bool BackwardSlashCheck(_POINT _A[][BOARD_SIZE], int currentRow, int currentCol)
195 {
196         int countWin = 1;
197         bool blockUp = false;
198         bool blockDown = false;
199         bool wall = false;
200         int count = 1;
201         bool keepCounting = true;
202
203         if (currentCol == 0 || currentCol == BOARD_SIZE - 1 || currentRow == 0 || currentRow == BOARD_SIZE - 1) wall = true;
204
205         while (currentRow + count < BOARD_SIZE && currentCol + count < BOARD_SIZE)
206         {
207             if ((_A[currentRow + count][currentCol + count].c != 0) && (_A[currentRow + count][currentCol + count].c != _A[currentRow][currentCol].c))
208             {
209                 blockDown = true;
```

```
210                 break;
211           }
212           else if ((_A[currentRow + count][currentCol + count].c == _A[currentRow][currentCol].c) && keepCounting)
213           {
214                 if (currentRow + count == BOARD_SIZE - 1 || currentCol + count == BOARD_SIZE - 1) wall = true;
215                 countWin++;
216                 count++;
217           }
218           else
219           {
220                 count++;
221                 keepCounting = false;
222           }
223     }
224
225     count = 1;
226     keepCounting = true;
227
228     while (currentRow - count > -1 && currentCol - count > -1)
229     {
230           if ((_A[currentRow - count][currentCol - count].c != 0) && (_A[currentRow - count][currentCol - count].c != _A[currentRow][currentCol].c))
231           {
232                 blockUp = true;
233                 break;
234           }
235           else if ((_A[currentRow - count][currentCol - count].c == _A[currentRow][currentCol].c) && keepCounting)
236           {
237                 if (currentRow - count == 0 || currentCol - count == 0) wall = true;
238                 countWin++;
239                 count++;
240           }
241           else
242           {
243                 count++;
244                 keepCounting = false;
245           }
246     }
247
248     if (!blockUp && !blockDown && !wall && (countWin == 4))
249           return true;
250     else if (countWin >= 5 && !(blockUp && blockDown))
251           return true;
252     else
253           return false;
```

```
254 }
255
256 bool CheckTie(_POINT _A[][BOARD_SIZE])
257 {
258     for (int i = 0; i < BOARD_SIZE; i++)
259     {
260         for (int j = 0; j < BOARD_SIZE; j++)
261         {
262             if (_A[i][j].c == 0) return false;
263         }
264     }
265     return true;
266 }
267
268 int TestBoard(_POINT _A[][BOARD_SIZE], bool _TURN, int cRow,
    int cCol)
269 {
270     if (CheckWin(_A, cRow, cCol))
271         return (_TURN == FIRST ? P_X : P_O); //-1 nghia la
    luot 'true' thang
272     else if (CheckTie(_A)) return 0;
273     else return 2;
274
275 }
```

- Hàm **HorizontalCheck()** dạng luận lý dùng để kiểm tra điều kiện thắng theo hàng ngang với các tham số truyền vào là ma trận bàn cờ caro, chỉ số hàng và cột hiện tại của ô cờ vừa đánh. Khi người chơi vừa thực hiện việc đánh dấu vào bàn cờ, hàm sẽ được sử dụng để kiểm tra xem người chơi đã đủ điều kiện thắng hay chưa. Để trực quan hơn, ta có thể hiểu như sau:
  - Các biến được sử dụng:
    + countWin: biến đếm số quân thắng, được khởi tạo giá trị ban đầu là 1 do được kiểm tra từ ô hiện tại.
    + blockRight (dạng luận lý): được sử dụng làm dấu hiệu bị chặn bởi quân cờ đối phương ở bên phải, tương tự blockLeft
    + wall (dạng luận lý): được sử dụng làm dấu hiệu bị chặn bởi khung của bàn cờ
    + count: biến được cộng thêm vào chỉ số hàng, cột hiện tại để chuyển sang vị trí tiếp theo trên bàn cờ
    + keepCounting (dạng luận lý): được sử dụng để duy trì việc đếm số quân giống nhau và liên tiếp nhau (không có ô trống ở giữa).

  - **Giải thích:** Đầu tiên ta sẽ kiểm tra bên phải ô đang xét. Cho vòng lặp chạy đến khi đụng rìa bàn cờ. Nếu ô đang xét khác quân cờ so với ô ban đầu, blockRight chuyển thành true và break vòng lặp. Còn nếu ô đang xét là quân cờ giống với quân ban đầu và keepCounting vẫn đang là true (chưa gặp khoảng trống nào suốt quá trình xét) thì tăng countWin lên 1 và tăng count 1 đơn vị. Còn nếu gặp khoảng trắng, chuyển keepCounting thành false và vẫn cho count tăng 1 đơn vị.

- Các hàm **VerticalCheck(), BackwardSlashCheck(), ForwardSlashCheck()** tương tự như hàm trên.

- Hàm **CheckWin()** (dạng luận lý) được dùng để kiểm tra điều kiện thắng, có giá trị true nếu một trong 4 hàm kiểm tra trên có giá trị true, ngược lại, có giá trị false.

- Hàm **CheckTie()** dạng luận lý, là hàm dùng để kiểm tra hòa khi bàn cờ đã được đánh hết nhưng chưa có bên nào thắng

- Hàm **TestBoard()** là hàm kiểu int, được áp dụng như sau: Nếu hàm **CheckWin()** có giá trị true thì hàm sẽ được trả về giá trị -1 nếu lượt là lượt của người chơi 1, trả về 1 nếu là lượt của người chơi 2; còn không nếu hàm **CheckTie()** có giá trị true, hàm sẽ được trả về giá trị 0; trả về giá trị 2 nếu không có gì xảy ra.

- Các hàm xử lý dữ liệu người chơi:

```
1  void SetPlayer(_PLAYER& _PLAYER1, _PLAYER& _PLAYER2)
2  {
3        system("cls");
4        do
5        {
6              PrintText("Enter Player1's name (2-10 characters):
   ", 252, X_CENTER - 24, Y_CENTER);
7              std::getline(cin, _PLAYER1.name);
8              DrawBox(255, 100, 1, X_CENTER, Y_CENTER);
9        } while (_PLAYER1.name.length() < 2 ||
   _PLAYER1.name.length() > 10);
10       do
11       {
12             PrintText("Enter Player2's name (2-10 characters):
   ", 250, X_CENTER - 24, Y_CENTER);
13             std::getline(cin, _PLAYER2.name);
14             DrawBox(255, 100, 1, X_CENTER, Y_CENTER);
15       } while (_PLAYER2.name.length() < 2 ||
   _PLAYER2.name.length() > 10 || _PLAYER2 == _PLAYER1);
16
17       _PLAYER1 = LoadPlayer(_PLAYER1);
18       _PLAYER2 = LoadPlayer(_PLAYER2);
19 }
20
21 std::vector<_PLAYER> GetPlayerList()
22 {
23       _PLAYER player;
24       std::vector<_PLAYER> players;
25       std::fstream playerList;
26       std::string clear;
27       std::string name;
28       playerList.open(PLAYER_LIST, std::fstream::in);
29
30       while (getline(playerList, player.name))
31       {
32             playerList >> player.wins;
33             getline(playerList, clear);
34             players.push_back(player);
35       }
36
37       playerList.close();
38
```

```
39      return players;
40  }
41
42  int CheckPlayerExistence(_PLAYER player)
43  {
44      std::vector<_PLAYER> players = GetPlayerList();
45
46      for (int i = 0; i < players.size(); i++)
47      {
48          if (players.at(i) == player) return i;
49      }
50
51      return -1;
52  }
53
54  void SavePlayer(_PLAYER player)
55  {
56      int exist = CheckPlayerExistence(player);
57      std::vector<_PLAYER> players = GetPlayerList();
58      std::fstream playerList;
59      playerList.open(PLAYER_LIST, std::fstream::out);
60
61      if (exist == -1) players.push_back(player);
62      else players.at(exist).wins = player.wins;
63
64
65      SortPlayerList(players);
66      for (int i = 0; i < players.size(); i++)
67      {
68          playerList << players.at(i).name << "\n" <<
    players.at(i).wins << "\n";
69      }
70
71      playerList.close();
72
73  }
74
75  _PLAYER LoadPlayer(_PLAYER player)
76  {
77      std::vector<_PLAYER> players;
78      players = GetPlayerList();
79      int exist = CheckPlayerExistence(player);
80
81      if (exist == -1)
82      {
83          player.wins = 0;
84          SavePlayer(player);
85          return player;
86      }
87      else return players.at(exist);
88  }
```

```
89
90  void SetPlayerRank(_PLAYER& player)
91  {
92      int exist = CheckPlayerExistence(player);
93      player.rank = exist + 1;
94  }
```

- Đa số các câu lệnh trong hàm **SetPlayer()** dùng để tạo dữ liệu người chơi bằng cách cho người dùng nhập tên người chơi, sau đó gán 2 tham chiếu người chơi cho hàm **LoadPlayer()** (hàm này trả về một _PLAYER, chi tiết nêu ở dưới)
- Hàm **GetPlayerList()** là hàm trả về vector<_PLAYER>. Hàm này đầu tiên tạo 1 biến kiểu _PLAYER có tên là player, lại tiếp tục tạo 1 đối tượng fstream với tên là playerList, tạo 1 biến kiểu vector có tên là players (đây là thứ sẽ được trả về), tạo biến string tạm thời là clear (biến clear này dùng để lấy các kí tự "\n" còn lưu lại trong bộ nhớ đệm sau khi chúng ta lấy giá trị số trận thắng của mỗi người chơi trong file). Vòng while không có gì khác biệt ngoài việc điều kiện của vòng while là hàm getline có luồng truyền là playerList vào biến player.name, nếu thực hiện được, đi vào vòng while lấy số trận thắng của biến player. Sau khi có tên và thuộc tính số trận thắng, ta đẩy biến player vào vector players và đóng file rồi trả về vector players.
- Hàm **CheckPlayerExistence()** là hàm kiểm tra xem người chơi được truyền vào có tồn tại trong file hay chưa, nếu tồn tại thì trả về chỉ số của người chơi đó trong file, nếu chưa thì trả về -1.
- Hàm **SavePlayer()** có tham số truyền vào là player kiểu _PLAYER. Tạo 1 biến exist và gọi hàm **CheckPlayerExistence()**;
- Tạo 1 vector kiểu _PLAYER có tên players và gọi hàm **GetPlayerList()** để lấy dữ liệu cho biến này. Tạo đối tượng fstream có tên playerList dùng phương thức open mở file. Ta xét điều kiện nếu exist bằng -1 thì player chưa có trong bảng xếp hạng thì ta đẩy giá trị player này vào vector players, ngược lại thì cập nhật số trận thắng mới cho player đã được lưu trong bảng xếp hạng. Sau đó gọi hàm **SortPlayerList()** (nêu ở dưới) sắp xếp lại các phần tử trong vector, rồi dùng vòng for để ghi đè lên file playerList một bảng xếp hạng mới.
- Hàm **LoadPlayer()** dùng để tải dữ liệu người chơi dựa trên tham số truyền vào là player kiểu _PLAYER). Tạo 1 vector kiểu _PLAYER có tên là players, gọi hàm **GetPlayerList()** để lấy dữ liệu cho biến này. Tạo 1 biến exist và gọi hàm **CheckPlayerExistence()** để trả về giá trị cho biến exist. Ta xét điều kiện nếu exist có giá trị là -1 tức là player chưa có sẵn, ta gán số trận thắng của player là 0, gọi hàm **SavePlayer()** để thực hiện thao tác lưu player mới và trả về giá trị player. Ngược lại, trả về giá trị players tại vị trí exist trong vector<_PLAYER> players.
- Hàm **SetPlayerRank()** dùng để cập nhật thứ hạng của player dựa trên chỉ số vị trí trong danh sách người chơi, những thứ có trong hàm này đã được nói ở trên, nhóm tác giả xin không đề cập để tránh gây mất thời gian cũng như gây rối trong báo cáo.

- Các hàm lấy chỉ số cột hàng trên bàn cờ:

```
1   int GetRowIndex(int pY)
2   {
3       int rowIndex;
4
5       rowIndex = (pY - TOP - 1) / VERTICAL_DISTANCE;
6
7       return rowIndex;
8   }
9
10  int GetColIndex(int pX)
```

```
{
	int colIndex;

	colIndex = (pX - LEFT - 2) / HORIZONTAL_DISTANCE;

	return colIndex;
}
```

- Hàm **GetRowIndex()** dùng để lấy chỉ số hàng hiện tại, cụ thể như sau: hàm sẽ truyền tham số là tung độ của ô cờ hiện tại trừ cho khoảng cách từ rìa trên màn hình console đến rìa trên bàn cờ, sau đó lấy hiệu trên chia khoảng cách theo phương thẳng đứng giữa các ô cờ.

- Hàm **GetColIndex()** tương tự như **GetRowIndex().**

- Hàm sắp xếp người chơi:

```
void SortPlayerList(std::vector<_PLAYER>& playerList)
{
	_PLAYER key;
	int j;

	for (int i = 1; i < playerList.size(); i++)
	{
		key = playerList.at(i);
		j = i - 1;
		while (j >= 0 && playerList.at(j) < key)
		{
			playerList.at(j + 1) = playerList.at(j);
			j--;
		}
		playerList.at(j + 1) = key;
	}
}
```

- Hàm **SortPlayerList()** với tham số truyền vào là 1 vector kiểu _PLAYER. Ở đây nhóm tác giả dùng phương pháp sắp xếp chèn (Insertion Sort) tham khảo trên wikipedia bằng toán tử >, < đã được định nghĩa trong struct _PLAYER.

Về kiểu vector, đây là kiểu dữ liệu mảng động cho phép khai báo mà không cần phải biết trước kích thước cụ thể, và có thể thêm phần tử vô cũng như truy xuất phần tử dễ dàng tương tự mảng. Phương thức push_bac(<phần tử>) được dung để thêm phần tử vô mảng vector, phương thức at(<index>) tương tự như a[index] trong mảng bình thường.

Bước 4: Xây dựng nhóm hàm Control để vận dụng các hàm đã xây dựng trong View và Model:

- Hàm chọn tính năng Menu:

```
int SelectMenu(_MENU menu)
{
	int cursor = 1;
	char key;

	PrintText("--->", menu.cursorColor, menu.x - 4,
menu.y);
```

```
7
8       do
9       {
10              key = _getch();
11              if (key == ARROW_UP && cursor > 1)
12              {
13                      PrintText("    ", menu.cursorColor, menu.x - 4, menu.y + cursor - 1);
14                      cursor--;
15                      PrintText("--->", menu.cursorColor, menu.x - 4, menu.y + cursor - 1);
16              }
17              else if (key == ARROW_DOWN && cursor < menu.options)
18              {
19                      PrintText("    ", menu.cursorColor, menu.x - 4, menu.y + cursor - 1);
20                      cursor++;
21                      PrintText("--->", menu.cursorColor, menu.x - 4, menu.y + cursor - 1);
22              }
23              else if (key == ESC)
24              {
25                      return -1;
26              }
27      } while (key != ENTER);
28
29      return cursor;
30  }
```

- Hàm **SelectMenu():** đây là hàm tổng quát được xài cho TẤT CẢ các loại Menu trong chương trình. Hàm này có 2 chức năng: Đầu tiên là hiển thị con trỏ người dùng đang ở là một chuỗi "--->" . Hàm dùng một vòng lặp do while với điều kiện dừng là biến key (nhập vào nhờ hàm _getch() – hàm để dừng màn hình lấy ký tự tiếp theo nhập từ bàn phím) khác phím ENTER.
- Khởi tạo một biến cursor = 1 (ý chỉ bạn đang ở lựa chọn đầu tiên)
- Trong vòng lặp Do-While key là mũi tên đi lên (ARROW-UP) và cursor > 1 thì di chuyển dấu "--->" đi xuống (viết chuỗi " " ở hàng trước đè lên để xóa) 1 tọa độ (1 hàng)
- Ngược lại với ARROW_DOWN
- Nếu key == ENTER thì return giá trị cursor.

- Các hàm chạy tính năng Menu:

```
1   void RunMainMenu(_POINT _A[][BOARD_SIZE], _PLAYER& _PLAYER1, _PLAYER& _PLAYER2, bool& _TURN, int& _COMMAND, int& _X, int& _Y, bool& run, int option)
2   {
3       int loadOption;
4       switch (option)
5       {
6       case 1:
7               SetPlayer(_PLAYER1, _PLAYER2);
```

```
8           StartGame(_A, _PLAYER1, _PLAYER2, _TURN, _COMMAND, _X, _Y);
9           RunGame(_A, _PLAYER1, _PLAYER2, _TURN, _COMMAND, _X, _Y);
10          break;
11      case 2:
12          system("cls");
13          loadOption = SelectMenu(LoadingMenu());
14          if (loadOption == -1) break;
15          else
16          {
17              LoadGame(RunLoadingMenu(loadOption), _A, _PLAYER1, _PLAYER2, _TURN, _COMMAND, _X, _Y);
18              RunGame(_A, _PLAYER1, _PLAYER2, _TURN, _COMMAND, _X, _Y);
19              break;
20          }
21      case 3:
22          do
23          {
24              ShowRank();
25          } while (_getch() != ESC);
26          break;
27      case 4:
28          do
29          {
30              ShowHelp();
31          } while (_getch() != ESC);
32          break;
33      case 5:case -1:
34          ExitGame(run);
35          break;
36      }
37  }
38
39  void RunEscMenu(_POINT _A[][BOARD_SIZE], _PLAYER _PLAYER1, _PLAYER _PLAYER2, bool _TURN, int option, bool& runGame)
40  {
41          switch (option)
42      {
43      case 1: case -1:
44          system("cls");
45          DrawBoard(BOARD_SIZE, BOARD_SIZE, 3, 1, LEFT, TOP);
46          DrawLoaded(_A);
47          ShowTurn(_A, _PLAYER1, _PLAYER2, _TURN);
48          break;
49      case 2:
50          SaveGame(_A, _PLAYER1, _PLAYER2, _TURN);
51          runGame = false;
52          break;
```

```
53          case 3:
54                  runGame = false;
55                  break;
56          }
57  }
58
59  string RunLoadingMenu(int option)
60  {
61          string filename;
62          std::vector<string> files;
63
64          files = LoadFiles();
65          filename = files.at(option - 1);
66
67          return filename;
68  }
69
70  char RunYesNoMenu(int option)
71  {
72          switch (option)
73          {
74          case 1:
75                  return 'Y';
76          case 2:
77                  return 'N';
78          }
79  }
```

- Hàm **RunMainMenu()** dùng để thực thi những chức năng mà bạn đã chọn khi bắt đầu chương trình với các tham số truyền vào lần lượt là ma trận bàn cờ, dữ liệu 2 người chơi, biến nhận lệnh nhập từ bàn phím, tọa độ con nháy, lượt chơi, biến chạy chương trình (dạng luận lý) và biến lệnh đã chọn từ menu chính.

- Hàm **RunEscMenu()** tương tự như **hàm RunMainMenu()** nhưng thay biến chạy chương trình (dạng luận lý) thành biến chạy game (dạng luận lý). Hàm này được thực thi khi 2 người đang chơi nhưng có một trong 2 người nhấn nút ESC, nút ESC sẽ gọi hàm **EscMenu()** và gọi hàm **SelectMenu()** cho hàm **EscMenu()** dưới dạng **Select(EscMenu())** để trả ra giá trị lựa chọn rồi truyền vào tham trị option trong hàm **RunEscMenu()**

- **RunLoadingMenu()** cũng tương tự. Hàm này gọi hàm **SelectMenu()** với tham trị **LoadingMenu()**, Hàm này có tham số duy nhất là option mình lựa chọn để trả về giá trị chuỗi, chuỗi này là tên file text mà chúng ta lựa chọn để load lại game dang chơi dở dang trước đó, hàm này tạo một vector và dùng hàm **LoadFiles()** để lấy dữ liệu trong files SavedList.txt và đẩy vào vector<string> vừa tạo theo từng dòng. Nếu lựa chọn là "i" thì trả về string ở vị trí thứ "i – 1" để hàm **LoadGame()** có thể khởi tạo và bắt đầu chạy game theo dữ liệu đã lưu.

- **RunYesNoMenu()** có một tham trị option. Và hàm này trả về ký tự 'Y' nếu như option là 1 và ngược lại thì là 'N' nếu như option là 2. Và tất nhiên option này là giá trị của **SelectMenu(YesNoMenu()).**

- Hàm bắt đầu Game:

```
1  void StartGame(_POINT _A[][BOARD_SIZE], _PLAYER& _PLAYER1, _PLAYER& _PLAYER2, bool& _TURN, int& _COMMAND, int& _X, int& _Y)
2  {
3        SetPlayerRank(_PLAYER1);
4        SetPlayerRank(_PLAYER2);
5        ResetData(_A, _PLAYER1, _PLAYER2, _TURN, _COMMAND, _X, _Y);
6        system("cls");
7        DrawBoard(BOARD_SIZE, BOARD_SIZE, 3, 1, LEFT, TOP);
8        ShowTurn(_A, _PLAYER1, _PLAYER2, _TURN);
9        GotoXY(_X, _Y);
10 }
```

- Hàm **StartGame()** lấy xếp hạng của người chơi (để đưa lên bảng thông tin người chơi trong lúc chơi game) (hàm **SetPlayerRank()** sẽ được nói rõ trong nhóm hàm Model), bắt đầu trò chơi mới với bàn cờ trống bằng các hàm **ResetData()** (trong Model) và các hàm hiển thị.

- Hàm chạy Game:

```
1  void RunGame(_POINT _A[][BOARD_SIZE], _PLAYER & _PLAYER1, _PLAYER & _PLAYER2, bool & _TURN, int & _COMMAND, int & _X, int & _Y)
2  {
3        bool validEnter = true;
4        bool runGame = true;
5        int escOption;
6
7        PlaySoundA("NhacGame.wav", NULL, SND_ASYNC | SND_LOOP);
8
9        while (runGame)
10       {
11             ShowPlayerInfo(_A, _PLAYER1, _PLAYER2);
12             GotoXY(_X, _Y);
13             _COMMAND = toupper(_getch());
14             if (_COMMAND == ESC)
15             {
16                   escOption = SelectMenu(EscMenu(_A));
17                   RunEscMenu(_A, _PLAYER1, _PLAYER2, _TURN, escOption, runGame);
18             }
19             else
20             {
21                   if (_COMMAND == 'W' || _COMMAND == ARROW_UP) MoveUp(_A, _X, _Y);
22                   else if (_COMMAND == 'S' || _COMMAND == ARROW_DOWN) MoveDown(_A, _X, _Y);
23                   else if (_COMMAND == 'A' || _COMMAND == ARROW_LEFT) MoveLeft(_A, _X, _Y);
24                   else if (_COMMAND == 'D' || _COMMAND == ARROW_RIGHT) MoveRight(_A, _X, _Y);
25                   else if (_COMMAND == ENTER || _COMMAND == SPACE)
```

```
26                  {
27                      switch (CheckBoard(_A, _TURN,
GetRowIndex(_Y), GetColIndex(_X)))
28                      {
29                      case P_X:
30                          PrintText("X", 252, _X, _Y);
31                          break;
32                      case P_O:
33                          PrintText("O", 250, _X, _Y);
34                          break;
35                      case 0:
36                          validEnter = false;
37                          break;
38                      }
39                      if (validEnter)
40                      {
41                          switch (ProcessFinish(_A, _PLAYER1,
_PLAYER2, _TURN, TestBoard(_A, _TURN, GetRowIndex(_Y),
GetColIndex(_X))))
42                          {
43                          case P_X:case P_O:case 0:
44                              DrawBoard(1, 1, 30, 8,
_A[0][BOARD_SIZE - 1].x + 58, Y_CENTER - 7);
45                              PrintText("Do you want to
continue?", 249, _A[0][BOARD_SIZE - 1].x + 62, Y_CENTER - 4);
46                              if
(AskContinue(_A[0][BOARD_SIZE - 1].x + 65, Y_CENTER - 2))
47                              {
48                                  StartGame(_A, _PLAYER1,
_PLAYER2, _TURN, _COMMAND, _X, _Y);
49                                  PlaySoundA("NhacGame.wav",
NULL, SND_ASYNC | SND_LOOP);
50                              }
51                              else runGame = false;
52                              break;
53                          }
54                      }
55                  }
56                  validEnter = true;
57              }
58          }
59          PlaySoundA("NoSound.wav", NULL, SND_ASYNC | SND_LOOP);
60  }
```

- Hàm **RunGame()** chạy trò chơi với các tham số truyền vào là ma trận bàn cờ, dữ liệu người chơi 1, dữ liệu người chơi 2, lượt chơi, biến nhận lệnh nhập từ bàn phím, tọa độ con nháy hiện tại. Biến validEnter để kiểm tra xem người dùng có đánh cờ vào ô hợp lệ hay không (đánh vào ô chưa đánh), sau đó chạy game trong 1 vòng lặp nơi người chơi sử dụng phím để thao tác.

- Các hàm Save/Load/ExitGame:

```
1  void SaveGame(_POINT _A[][BOARD_SIZE], _PLAYER _PLAYER1,
_PLAYER _PLAYER2, bool _TURN)
```

```
{
	bool overwrite = false;
	bool save = true;
	string filename;
	char key;
	int option;

	do
	{
		system("cls");
		PrintText("Nhap ten muon luu game: ", 245, X_CENTER - 30, Y_CENTER);
		std::getline(cin, filename);
		filename += ".txt";
		if (CheckFileExistence(filename))
		{
			PrintText("Ten da ton tai", 245, X_CENTER - 30, Y_CENTER + 2);
			PrintText("Ban co muon luu de?", 245, X_CENTER - 30, Y_CENTER + 4);
			option = SelectMenu(YesNoMenu(X_CENTER - 15, Y_CENTER + 5));
			key = RunYesNoMenu(option);
			if (key == 'Y')
			{
				overwrite = true;
				save = false;
			}
		}
		else save = false;
	} while (save);

	SaveData(filename, _A, _PLAYER1, _PLAYER2, _TURN);

	if (!overwrite)
	{
		std::fstream saveFile;
		saveFile.open(SAVED_LIST, std::fstream::app);

		saveFile << filename << "\n";

		saveFile.close();
	}
}

void LoadGame(string filename, _POINT _A[][BOARD_SIZE], _PLAYER & _PLAYER1, _PLAYER & _PLAYER2, bool & _TURN, int & _COMMAND, int & _X, int & _Y)
{
	system("cls");
```

```
46      LoadData(filename, _A, _PLAYER1, _PLAYER2, _TURN, _COMMAND, _X, _Y);
47      SetPlayerRank(_PLAYER1);
48      SetPlayerRank(_PLAYER2);
49      DrawBoard(BOARD_SIZE, BOARD_SIZE, 3, 1, LEFT, TOP);
50      ShowTurn(_A, _PLAYER1, _PLAYER2, _TURN);
51      DrawLoaded(_A);
52      GotoXY(_X, _Y);
53  }
54
55  void ExitGame(bool & run)
56  {
57      run = false;
58      system("cls");
59      DrawBigText("End.txt", 94, 15, Y_CENTER - 7);
60      Sleep(3000);
61  }
```

- Hàm **SaveGame()**: Bắt đầu quá trình lưu trò chơi, đầu tiên là cho người dùng nhập tên trò chơi để lưu lại, sau đó kiểm tra tên đã tồn tại hay chưa (hàm **CheckFileExistence()**, trong Model), nếu tồn tại thì hỏi người chơi có muốn lưu đè hay không (biến overwrite để kiểm tra sự lưu đè đó). Sau đó gọi hàm **SaveData()** (trong Model) để lưu dữ liệu vào file, đồng thời nếu trc đó người chơi không lưu đè thì sẽ thêm tên file vào danh sách file lưu (SavedList.txt).

- Hàm **LoadGame()**: Bắt đầu quá trình load trò chơi, **LoadData()** để load dữ liệu bàn cờ cũng như người chơi đã được lưu trước đó (hàm nói rõ trong Model), sau đó là vẽ bảng và vẽ cờ "X" "O" tương ứng lên bàn cờ dựa trên dữ liệu đã được lưu.

- Hàm **ExitGame():** Thoát Game, hiện hiệu ứng "Thank you" cảm ơn người chơi, dừng chạy vòng lặp menu chính ở hàm **main()** để thoát chương trình.

- Các hàm di chuyển trên bàn cờ Caro:

```
1   bool MoveLeft(_POINT _A[][BOARD_SIZE], int& _X, int& _Y)
2   {
3       if (_X > _A[0][0].x)
4       {
5           _X -= HORIZONTAL_DISTANCE;
6           GotoXY(_X, _Y);
7           return true;
8       }
9       return false;
10  }
11
12  bool MoveRight(_POINT _A[][BOARD_SIZE], int& _X, int& _Y)
13  {
14      if (_X < _A[BOARD_SIZE - 1][BOARD_SIZE - 1].x)
15      {
16          _X += HORIZONTAL_DISTANCE;
17          GotoXY(_X, _Y);
```

```
18              return true;
19        }
20        return false;
21  }
22
23  bool MoveDown(_POINT _A[][BOARD_SIZE], int& _X, int& _Y)
24  {
25        if (_Y < _A[BOARD_SIZE - 1][BOARD_SIZE - 1].y)
26        {
27              _Y += VERTICAL_DISTANCE;
28              GotoXY(_X, _Y);
29              return true;
30        }
31        return false;
32  }
33
34  bool MoveUp(_POINT _A[][BOARD_SIZE], int& _X, int& _Y)
35  {
36        if (_Y > _A[0][0].y)
37        {
38              _Y -= VERTICAL_DISTANCE;
39              GotoXY(_X, _Y);
40              return true;
41        }
42        return false;
43  }
```

- Hàm **MoveLeft()** dùng để di chuyển con trỏ bàn cờ Caro bằng cách di chuyển dấu nháy sang trái 1 khoảng bằng khoảng cách giữa các ô cờ với các tham số nhận vào là ma trận bàn cờ, tọa độ con nháy hiện tại. Tương tự với các hàm **MoveRight():** sang phải; **MoveDown():** đi xuống; **MoveUp():** đi lên.

Bước 5: Xây dựng hàm Main:

```
1   int main()
2   {
3         _POINT _A[BOARD_SIZE][BOARD_SIZE];
4         _PLAYER _PLAYER1, _PLAYER2;
5         bool _TURN;
6         int _COMMAND;
7         int _X, _Y;
8         int option;
9         bool run = true;
10
11
12        CreateConsoleWindow();
13        FixConsoleWindow();
14
15        do
16        {
17              option = SelectMenu(MainMenu());
18              RunMainMenu(_A, _PLAYER1, _PLAYER2, _TURN,
    _COMMAND, _X, _Y, run, option);
```

```
19        } while (run);
20  }
```

Vậy là các nhóm hàm đã hoàn tất, bây giờ ta sẽ chạy chương trình chính trong hàm main(), đầu tiên là khai báo các biến cần sử dụng xuyên suốt chương trình, sau đó gọi 2 hàm CreateConsoleWindow() và FixConsoleWindow() để khởi tạo cửa sổ ban đầu. Và sau đó là tạo 1 vòng lặp chạy Menu chính, quy tắc chạy Menu cũng giống như đã nêu ở các hàm trong Control.

Lưu ý: Trong hàm **`RunGame()`** và cũng như một số hàm khác, ta xài một câu lệnh chơi nhạc như câu lệnh ở dòng 7 mục hàm **`RunGame()`** trang 35 của tài liệu này, với các tham số truyền vào là tên file (chuỗi theo mã ASCII), một biến NULL, và các cờ như SND_ASYNC để nhạc chạy và thực hiện câu lệnh tiếp theo, SND_LOOP với việc chạy xong sẽ chạy lại từ đầu file nhạc đó.

Và các file.wav chỉ có thể chơi được khi ta có dòng lệnh `#pragma comment(lib,"winmm.lib")` đã được nêu ra trong file View.h

4. Lời kết:

- Các thành viên trong nhóm suốt thời gian qua đã cùng nhau lập kế hoạch cho đồ án, cải tiến cũng như thêm bớt các chức năng.
- Đã có rất nhiều ý kiến cũng như các chức năng hay được đưa ra như (tạo hệ thống đăng nhập bằng cách nhập tên tài khoản và mật khẩu, hay là tạo chế độ đánh với máy, hay là cho người chơi chọn play list nhạc yêu thích phát trong quá trình chơi game của mình,...). Nhưng vì để tránh gây rối cũng như phức tạp và đi xa khỏi mục tiêu Nhập môn lập trình, nhóm chúng em đã quyết định làm một game caro trên Console đơn giản nhưng cũng không kém phần sinh động, qua đó cũng giúp cho các thành viên không quá áp lực thời gian trong cái môn học khác cũng như giúp các thành viên tập trung thật vững cơ bản của Nhập môn lập trình.
- Tuy code của nhóm vẫn còn nhiều chỗ chưa được tối ưu, liền mạch cũng như dễ hiểu ở mức tối đa; nhưng nhóm đã làm rất tốt trong việc giao tiếp cũng như đạt được gần như mọi mục tiêu đã đề ra và hoàn thiện tốt sản phẩm cuối cùng của mình.

5. Tài liệu tham khảo:

- **Tài liệu Youtube:**
  Kỹ thuật lập trình C/C++ - Làm Game Đua Xe Need For Speed
  Link dẫn:
  https://www.youtube.com/playlist?list=PLq9VOPepajsD4KilefuA6tgJVf1hTJHJq

  The Cherno Project, Playlist C++
  Link dẫn:
  https://www.youtube.com/playlist?list=PLlrATfBNZ98dudnM48yfGUldqGD0S4FFb

- **Tài liệu C++ tham khảo:** sách Fundamentals of C++ Programming của Richard L. Halterman.
- **Các công cụ chuyển đổi:**
  - Big text
    Link dẫn:
    https://www.text-image.com/convert/ascii.html?fbclid=IwAR1Qk4vXKMTgQINRk6xHMYLiZuw7KIqPlpeLgjJXyTd5iuQsG7XPfXmb3Is
  - Chuyển đổi mp3 sang WAV
    Link dẫn:
    https://convertio.co/vn/mp3-wav/?fbclid=IwAR1lqHTF5yPf3yqA_x3VQ8WXlc3q1994Rf3VtA1_DpGRZWyIoiFHTOP-U0s
- Và quan trọng nhất, Stack Overflow đã giúp chúng em rất nhiều trong công đoạn debug.