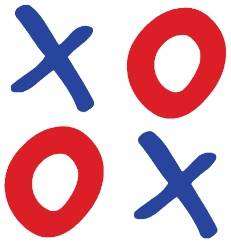
TRƯỜNG ĐẠI HỌC ĐÀ LẠT

KHOA TOÁN - TIN HỌC

TRẦN ĐẠT TÍN

MSSV: 2011345

VIẾT CHƯƠNG TRÌNH TRÒ CHƠI CARO TRÊN WINDOWS FORM VỚI C#



BÀI TẬP LỚN HỌC PHẦN

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG (TN2301D)

ĐÀ LẠT 2022

Tóm tắt

Có lẽ khi đặt câu hỏi "bạn có biết cờ caro không?" với bất kỳ người Việt Nam nào, câu trả lời nhận được sẽ là có. Bởi đây là trò chơi gắn liền với tuổi học trò của mỗi người. Chỉ với một tờ giấy và hai cây bút, luật chơi đơn giản, lũ học sinh sinh viên có thể mê say hàng giờ liền không biết chán. Cờ caro không chỉ là một trò chơi có tính giải trí cao mà còn đầy tính trí tuệ và hấp dẫn.

Thế nhưng it ai biết rằng, môn cờ tưởng chừng đơn giản nhất trong các loại cờ này lại có một lịch sử lâu đời và cả một lý thuyết chặt chẽ, có Tạp chí chuyên đề, có những chương trình máy tính, có những giải vô địch quốc gia và quốc tế và có cả Liên đoàn ca-rô thế giới.

Cờ caro chính là môn cờ logic lâu đời và cổ xưa nhất trên Trái Đất. Cờ caro đã được sáng tạo từ nhiều nền văn minh khác nhau một cách độc lập. Nó bắt đầu xuất hiện từ năm 2000 trước CN ở sông Hoàng Hà, Trung Quốc. Một số nhà khoa học đã tìm thấy bằng chứng chứng minh Caro đã được phát minh ở Hy lạp cổ đại và ở Châu Mỹ trước thời Colombo.

Bài tiểu luận được chia làm 3 phần:

- Chương 1: Giới thiệu luật chơi cơ bản của Tetris sử dụng trong tiểu luận này.
- Chương 2: Ý tưởng lập trình game Caro trên Windows form và một số kĩ thuật trong ngôn ngữ C# liên quan đến ý tưởng này.
- Chương 3: Tổng quan cấu trúc chương trình.

Mục lục

Chương 1

Tổng quan về trò chơi Caro

1.1 Tên gọi

Cờ ca-rô, trong tiếng Việt *ca-rô* (hay *sọc ca-rô*) được phiên âm từ *carreau* trong tiếng Pháp là các các ô vuông được sắp đều trên một mặt bề mặt. Cờ ca-rô trong tiếng Triều Tiên là omok 오목, tiếng Trung là 五子棋 và trong tiếng Nhật là 五目並べ (gomokunarabe). Tiếng Anh sử dụng lại tiếng Nhật, gọi là Gomoku. Gomoku đã tồn tại ở Nhật Bản từ trước thời Duy Tân Minh Trị (1868). "Go" trong "Gomokunarabe" tiếng Nhật nghĩa là *năm*, "moku" là nghĩa là *quân cờ* và "narabe" có nghĩa là *được xếp hàng*.

1.2 Ý tưởng

Mỗi người chơi sẽ được dùng viên đá cùng màu (hoặc có thể dùng một ký hiệu nếu chơi trên các môi trường khác). Các người chơi thay phiên nhau đặt một viên đá lên trên một bảng trống có kẻ các ô vuông. Người chiến thắng là người chơi đầu tiên tạo thành một chuỗi liên tục gồm năm viên đá theo chiều ngang, chiều dọc hoặc đường chéo. Việc đặt sao cho một chuỗi có nhiều hơn năm viên đá cùng loại được tạo ra sẽ không được xem là chiến thắng. Có nhiều cách chơi quy định rằng mỗi chuỗi 5 cùng màu được xem là không hợp lệ nếu hai đầu chuỗi được đặt bởi hai viên đá khác. Trong bài tiểu luận tác giả sẽ không sử dụng luật này.

Chương 2

Ý tưởng giải thuật, các bài toán cốt lõi

2.1 Visual Studio và Windows Form

Visual Studio hay Microsoft Visual Studio là một môi trường phát triển tích hợp (IDE) từ Microsoft. Visual Studio còn được gọi là "Trình soạn thảo mã nhiều người sử dụng nhất thế giới", được dùng để lập trình C++ và C# là chính. Nó được sử dụng để phát triển chương trình máy tính cho Microsoft Windows, cũng như các trang web, các ứng dụng web và các dịch vụ web. Visual Studio sử dụng nền tảng phát triển phần mềm của Microsoft như Windows API, Windows Forms, Windows Presentation Foundation, Windows Store và Microsoft Silverlight.

Windows form hay còn gọi là **Winform** là thuật ngữ mô tả một ứng dụng được viết trên .NET FrameWork và có giao diện đồ hoạ.

Để hiểu sâu hơn thì Windows Form là một thư viện lớp đồ họa, mã nguồn mở và hoàn toàn miễn phí. Từ năm 2003 thì windows form được xem là một phần của microsoft. Phần mềm này sẽ cung cấp nền tảng để viết những lập trình phong phú dành cho máy tính bàn, laptop, máy tính bảng,... được coi như một sự thay thế cho thư viện lớp nền tảng Microsoft Foundation của C++.

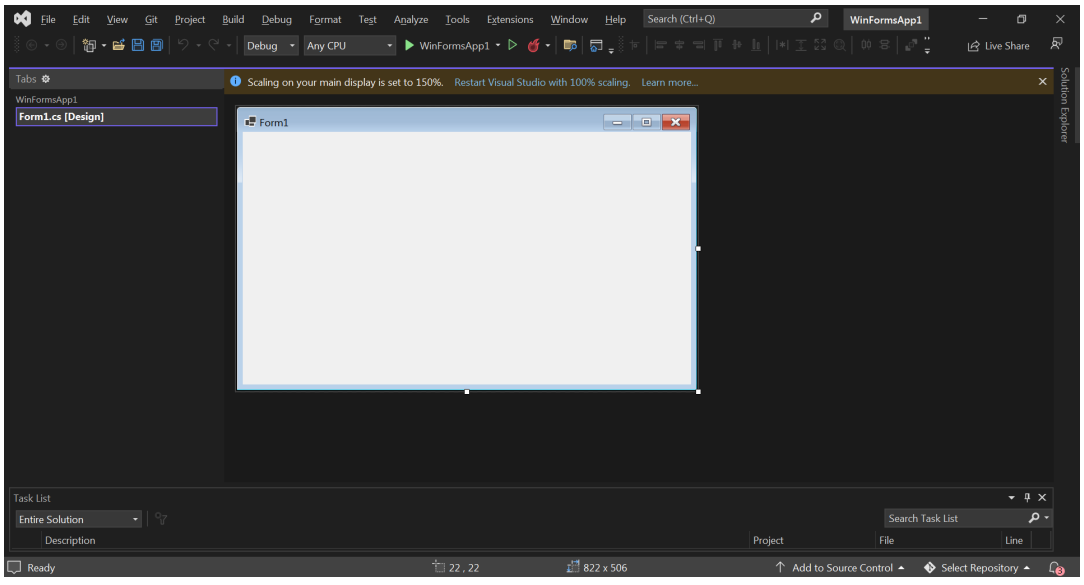
Tuy nhiên đến năm 2014 thì Windows Form chính thức bị Microsoft khai

tử. Windows Form có tuổi thọ khá lâu đời, chính vì thế nên chúng bị khai tử. Nhưng ở Việt Nam thì chúng vẫn được tồn tại và phát triển. Lý do là vì những ưu điểm mà chúng mang lại. Những lập trình viên C# ở đời đầu cũng đều được học và sử dụng về Windows Form. Bởi vì:

- Giao diện có thể kéo thả dễ học và sử dụng.
- Gắn các event cho các button chỉ cần double click. Hỗ trợ nhiều event như click, hover,. . .
- Viết code trực quan hơn: Có thể lấy text từ textbox và show dữ liệu bằng messagebox, kết nối grid bằng SQL.

Bên cạnh những ưu điểm của Windows Form thì chúng cũng có nhiều nhược điểm chưa được khắc phục sau đây:

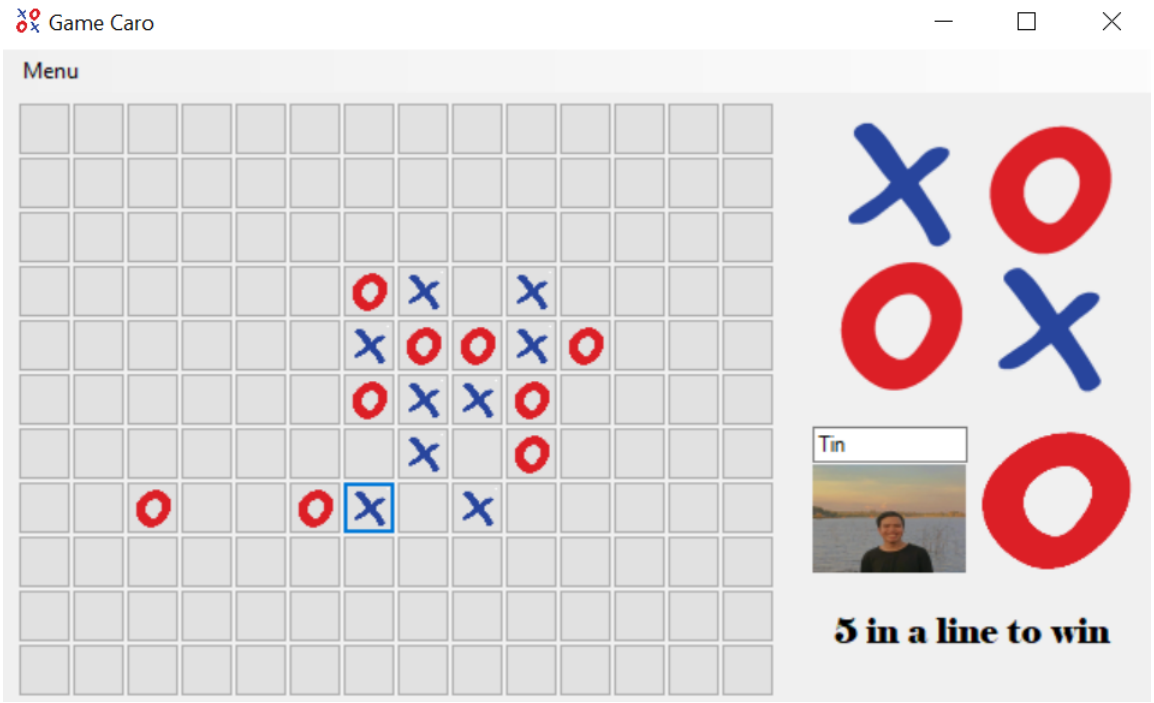
- Phần mềm chỉ có thể chạy trên nền tảng Windows Form.
- Windows Form chỉ phù hợp cho ứng dụng trên desktop (ứng dụng quản lý thông tin và tương tác trực tiếp với người dùng).
- Đồ họa không cao nên giao diện sẽ thiếu đi tính trực quan, không thân thiện đối với người dùng.



Hình 2.1: *Môi trường làm việc của project Windows Form trên Visual Studio*

2.2 Giao diện đồ hoạ (Graphical User Interface - GUI)

Giao diện đồ họa hay giao diện người dùng trong tiếng Anh gọi tắt là *GUI (Graphical User Interface)* là một thuật ngữ trong ngành công nghiệp máy tính. Đó là một cách giao tiếp với máy tính hay các thiết bị điện tử bằng hình ảnh và chữ viết thay vì chỉ là các dòng lệnh đơn thuần. GUI được sử dụng phổ biến trong máy tính, các thiết bị cầm tay, các thiết bị đa phương tiện, hoặc các linh kiện điện tử trong văn phòng, nhà ở,... Trong bài tiểu luận này game Caro sẽ được chạy trên giao diện đồ hoạ.

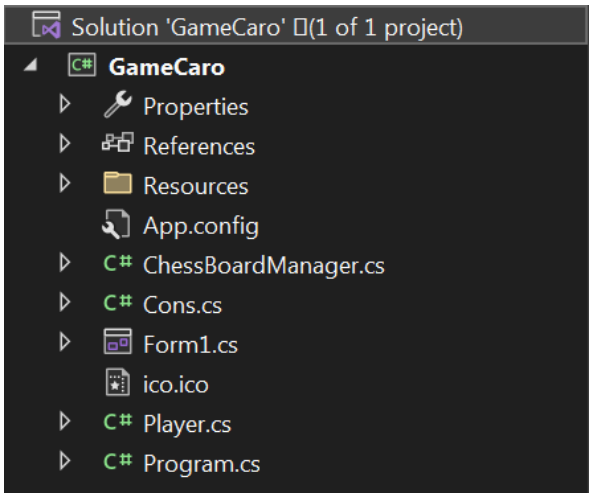


Hình 2.2: *Trò chơi Caro trên giao diện đồ hoạ (Graphical User Interface - GUI)*

2.3 Tổng quan về một project, trong Winform

Project (dự án) là cấp độ quản lý mã nguồn quan trọng nhất của C#. Mỗi project sau khi biên dịch sẽ tạo ra một chương trình. Mỗi project mặc định đều chứa:

- **Các file mã nguồn:** là các file văn bản có phần mở rộng .cs (viết tắt của C sharp).
- **Các file cấu hình:** là file xml có phần mở rộng .config
- **Các thư viện được tham chiếu tới (References):** là danh sách các file thư viện chuẩn của .NET framework, hoặc thư viện từ các hãng thứ ba, hoặc chính các project khác, chứa các class được sử dụng bởi các class trong project này.
- **Các thuộc tính (Properties):** bao gồm nhiều các loại thông tin khác nhau quyết định những tính chất quan trọng của project, như phiên bản của .NET framework được sử dụng, loại chương trình mà dự án này sẽ được dịch thành, các tài nguyên được sử dụng trong project, cấu hình của ứng dụng,... Visual Studio cung cấp giao diện đồ hoạ để có thể dễ dàng quản lý các thông tin này. Giao diện này mở ra khi click đúp vào mục Properties của project.



Hình 2.3: *Solution và Project trong C#*

Tất cả các thành phần của một project đều đặt chung trong một thư mục cùng tên với project.

Solution (giải pháp) là cấp độ quản lý mã nguồn cao nhất trong C# cho phép quản lý tập trung nhiều project. Mỗi solution trong C# có thể chứa nhiều

project. Nếu solution không chứa project nào, nó gọi là Empty Solution. Tại mỗi thời điểm Visual Studio chỉ có thể mở một solution.

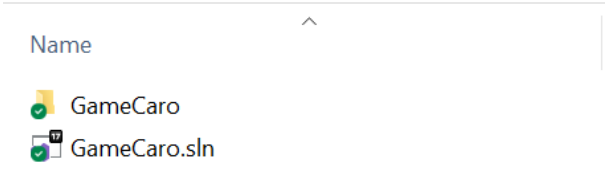
Trên giao diện Visual Studio, solution và các project của nó được hiển thị trong một cửa sổ riêng gọi là Solution Explorer. Cửa sổ này hiển thị tất cả các thành phần trong dự án C# theo cấu trúc cây, với solution làm gốc, các project là các nhánh trực tiếp xuất phát từ gốc này.

Ngoài ra để tiện lợi trong việc quản lý các project thành viên, solution cho phép tạo thêm các thư mục, gọi là Solution Folder, trong đó lại có thể chứa có project khác. Cấu trúc quản lý này cho phép quản lý một số lượng lớn project một cách dễ dàng. Các project trong cùng một solution thường có quan hệ nhất định với nhau.

2.4 Cấu trúc file/thư mục của C# project

Nếu ở giai đoạn tạo project chúng ta đánh dấu lựa chọn "Create directory for solution", Visual Studio sẽ tạo ra một thư mục có tên được xác định trong mục "Solution name". Mỗi project được tạo ra sẽ đặt trong một thư mục con của thư mục solution và có cùng tên với project. Tất cả file của một project sẽ nằm trong thư mục này.

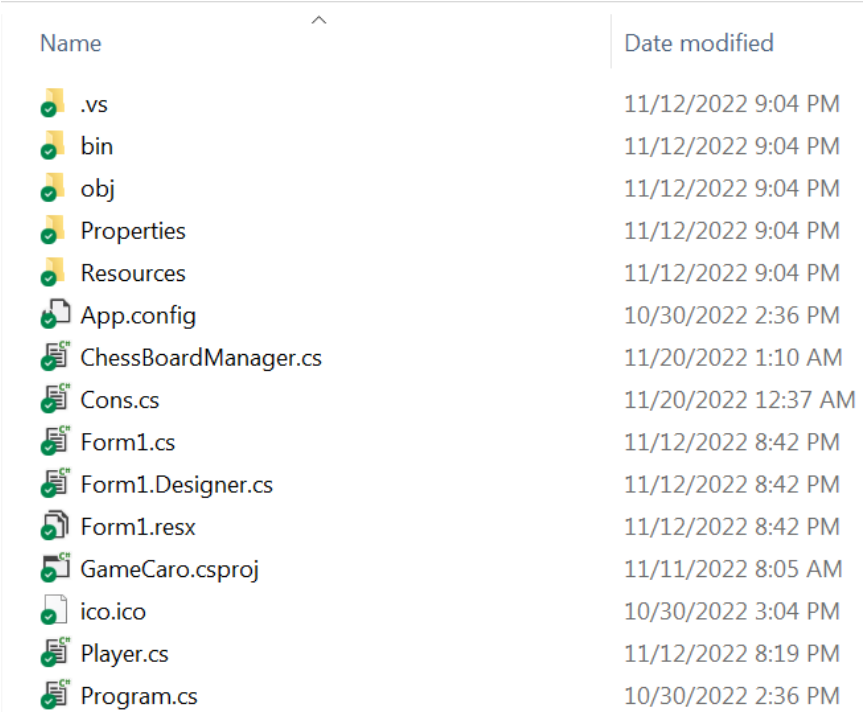
Toàn bộ cấu hình của solution được lưu trong một file có phần mở rộng **.sln** nằm trong thư mục chứa solution. Thông tin cấu hình của dự án được lưu trong file có tên trùng với tên dự án và phần mở rộng **.csproj**.



Hình 2.4: *Thư mục Solution*

Sau khi biên dịch project thành công, trong thư mục của nó sẽ xuất hiện một thư mục con có tên là *bin*. Thư mục này không xuất hiện trong cấu trúc dự án hiển thị ở Solution Explorer.

Nếu biên dịch ở chế độ debug, trong thư mục bin sẽ xuất hiện thư mục con



Hình 2.5: *Thư mục Project*

Debug. File chương trình sau khi biên dịch (ở chế độ debug) xong sẽ xuất hiện trong thư mục này.

Nếu chạy chương trình ở chế độ debug (phím F5), thư mục Debug này sẽ trở thành thư mục hiện hành của chương trình đang chạy thử. Tất cả các file cấu hình và file tài nguyên (nếu có) cũng sẽ được tự động copy vào thư mục này.

Hình 2.6: *Thư mục bin*

Như vậy, nếu bạn muốn tìm file chạy của chương trình sau khi biên dịch, hãy mở theo đường dẫn **{tên đường dẫn}\{tên project}\bin\{Debug}**.

2.5 Một số control trong Winform

Trong phần này tác sẽ sẽ giới thiệu một số control trong C# sử dụng trong dự án

Panel : Cho phép chứa đựng các điều khiển khác như: Button, RadioButton, Checkbox

PictureBox : Cho phép hiển thị hình ảnh dạng bitmap, metafile, icon, JPEG, GIF.

TextBox : để nhập dữ liệu đầu vào, ngoài ra còn có thể dùng để xuất dữ liệu. Các ô tài khoản mật khẩu trên mạng xã hội, các ô nhập thông tin trong phần mềm, các ô lưu tên cho highscore khi chơi game, chúng ta có thể thấy textbox ở khắp mọi ứng dụng hiện nay.

MenuStrip : là một điều khiển cho phép lập trình viên xây dựng hệ thống Menu trên Form.

Label : hiển thị nội dung, chữ.

Button : là một thành phần tương tác cho phép người dùng giao tiếp với một ứng dụng. Lớp Button (Nút) kế thừa trực tiếp từ lớp ButtonBase. Có thể nhấp vào Button bằng cách sử dụng chuột, phím ENTER hoặc phím cách nếu Button đã được đặt tiêu điểm.

2.6 Tạo bàn cờ

Một bàn cờ thực chất là một ma trận mà mỗi ô là một Button (nút bấm). Ma trận này được khởi tạo như sau `Matrix = new List<List<Button>>` Các Button này có các điều chỉnh kích thước và toạ độ trên Form. Ở mỗi vòng lặp để xác định được vị trí cần tạo của Button mới ta cần một biến `oldButton` để lưu vị trí của Button trước đó. Và để tiện trong việc tạo ra ván game mới khi sử dụng chương trình, trước khi khởi tạo bàn cờ phương thức `DrawChessBoard()` sẽ thực hiện xoá hết các Control trên Panel bằng lệnh `ChessBoard.Controls.Clear()`.

```
public void DrawChessBoard()
{
  ChessBoard.Controls.Clear();
  Matrix = new List<List<Button>>();
  Button oldButton = new Button() { Width = 0, Height = 0,
 Location = new Point(0, 0) };
  for (int i = 0; i < Cons.CHESS_BOARD_HEIGHT; i++)
  {
    Matrix.Add(new List<Button>());
    for (int j = 0; j < Cons.CHESS_BOARD_WIDTH; j++)
    {
      Button btn = new Button()
      {
        Width = Cons.CHESS_WIDTH,
        Height = Cons.CHESS_HEIGHT,
        Location = new Point(oldButton.Location.X + oldButton.
 Width, oldButton.Location.Y),
        BackgroundImageLayout = ImageLayout.Stretch,
        Tag = i.ToString()
      };

      btn.Click += btn_Click ;

      ChessBoard.Controls.Add(btn);

      Matrix[i].Add(btn);

      oldButton = btn;
    }
    oldButton.Location = new Point(0, oldButton.Location.Y +
 Cons.CHESS_HEIGHT);
    oldButton.Width = 0;
    oldButton.Height = 0;
  }
}
```

Đoạn mã 2.1: Phương thức `DrawChessBoard()`

Khi các người chơi thực hiện thao tác click vào các Button trên bàn cờ thì dòng 21 ở Đoạn mã sẽ thực hiện phương thức `btn_Click()` thực hiện việc đổi hình ảnh nền của Button sang hình ảnh hiển thị quân cờ của người chơi hiện tại. Đồng thời gọi các phương thức `Mark()` thực hiện đổi người chơi và kí hiệu quân cờ của người chơi sau đó phương thức `ChangePlayer()` được gọi để chỉnh các thông tin người chơi vừa được đổi. Cuối cùng phương thức `btn_Click()` gọi hàm `isEndgame()` để kiểm tra xem lần chơi cuối cùng đã kết thúc game chưa.

```
1  void btn_Click(object sender, EventArgs e)
2  {
3    Button btn = sender as Button;
4
5    if (btn.BackgroundImage != null)
6      return;
7
8    Mark(btn);
9
10   ChangePlayer();
11
12   if (isEndGame(btn))
13   {
14     EndGame();
15   }
16 }
```

Đoạn mã 2.2: Phương thức `btn_Click()`

2.7 Xử lý đổi người chơi

Trong hàm khởi tạo của class `ChessBoardManager` ta sẽ tạo ra một List có kiểu `Player`, lớp `Player` này sẽ chứa các thuộc tính liên quan đến thông tin người chơi. Lý do tạo List mà không tạo ra luôn hai người chơi cố định là để dễ dàng mở rộng chương trình sau này nếu muốn nhiều người chơi cùng lúc. Ta tạo thêm một biến `CurrentPlayer` kiểu `int` được gán giá trị là 0, dùng để lưu lượt của người chơi ở đây 0 tức là đang đến lượt người đầu tiên (vị trí 0 trong List). Và

cũng nghĩa là chương trình mặc định cho người đầu chơi đầu tiên chơi List chơi trước.

Việc đổi người chơi sẽ được thực thi khi thực hiện việc click vào các Button trên Panel bàn cờ. Khi đó ta sẽ thực hiện hai phương thức là `Mark()` và `ChangePlayer()` trong phương thức `btn_Click()` 2.6.

```
1  private void Mark(Button btn)
2  {
3    btn.BackgroundImage = Player[CurrentPlayer].Mark;
4    CurrentPlayer = CurrentPlayer == 1 ? 0 : 1;
5  }
6
7  private void ChangePlayer()
8  {
9    PlayerName.Text = Player[CurrentPlayer].Name;
10
11   PlayerMark.Image = Player[CurrentPlayer].Mark;
12
13   PlayerAvatar.Image = Player[CurrentPlayer].Avatar;
14 }
```

Đoạn mã 2.3: Phương thức `btn_Click()`

Phương thức `Mark()` sẽ đổi hình ảnh của Button được truyền vào (tức là Button vừa được click) dựa theo `CurrentPlayer` tức là thứ tự của người chơi trong `List<Player>`. Vì chương trình chỉ cài đặt với hai người chơi nên nếu `CurrentPlayer` là 0 thì sẽ đổi sang 1 và ngược lại.

Tiếp đó phương thức `ChangePlayer()` thực hiện hiển thị các thông tin người chơi lên màn hình bao gồm: tên, kí hiệu quân cờ, hình đại diện.

2.8 Xử lý thắng thua

Sau khi mỗi người chơi thực hiện đánh một quân cờ (tức là click vào một ô Button trên Panel bàn cờ) thì như trong phương thức `btn_Click()` 2.6. Ta sẽ gọi phương thức `isEndgame()` kiểm tra trò chơi đã có người chiến thắng chưa. Phương thức `isEndGame()` sẽ trả về là `True` nếu một trong bốn trường hợp

chiến thắng thoả mãn là đường chéo chính, đường chéo phụ, đường trục tung và đường trục hoành.

```
private bool isEndGame(Button btn){
  return isEndHorizontal(btn) || isEndVertical(btn) ||
 isMainDiag(btn) || isSubDiag(btn);
}
```

Đoạn mã 2.4: Phương thức `isEndGame()`

Trước tiên ta cần phương thức `GetChessPoint()` trả về toạ độ thuộc kiểu `Point` của một Button trong ma trận bàn cờ.

```
private Point GetChessPoint(Button btn)
{
  int vertical = Convert.ToInt32(btn.Tag);
  int horizontal = Matrix[vertical].IndexOf(btn);

  Point point = new Point(horizontal,vertical);

  return point;
}
```

Đoạn mã 2.5: Phương thức `isEndGame()`

Để kiểm tra chiến thắng theo chiều ngang mỗi khi có người chơi click vào một Button. Trước tiên ta sẽ lấy toạ độ của Button đó. Sau đó từ vị trí Button vừa được click vào ta sẽ thực hiện từng vòng lặp. Vòng lặp thứ nhất sẽ duyệt qua từ Button vừa click sang tất cả các Button bên phải nếu gặp một Button mà thuộc tính `BackgroundImage` khác với Button vừa click thì vòng lặp sẽ dừng lại. Tại vòng lặp thứ hai thực hiện đếm từ Button bên trái Button vừa click với điều kiện như vòng lặp phía trước. Cuối cùng ta cộng lại số lần đếm được các Button có cùng `BackgroundImage` ở ở hai vòng lặp. Nếu kết quả chính xác bằng 5 thì sẽ trả về `True`.

```
private bool isEndHorizontal(Button btn)
{
  Point point = GetChessPoint(btn);

```

```
5     int countLeft = 0;
6     for (int i = point.X; i >= point.X-5; i--)
7     {
8       if (i < 0)
9         break;
10      if (Matrix[point.Y][i].BackgroundImage == btn.
    BackgroundImage)
11      {
12        countLeft++;
13      }
14      else
15        break;
16    }
17
18    int countRight = 0;
19    for (int i = point.X+1; i < point.X + 1+ 5; i++)
20    {
21      if (i > Cons.CHESS_BOARD_WIDTH-1)
22        break;
23      if (Matrix[point.Y][i].BackgroundImage == btn.
    BackgroundImage)
24      {
25        countRight++;
26      }
27      else
28        break;
29    }
30
31    return countLeft + countRight == 5;
32  }
```

Đoạn mã 2.6: Phương thức `isEndHorizontal()` kiểm tra chiến thắng theo trục hoành

Với ý tưởng đó ta hoàn toàn có thể lặp lại cho các phương thức kiểm tra theo trục tung, đường chéo chính và đường chéo phụ qua lần lượt các phương thức `isEndVertical()`, `isMainDiag()`, `isSubDiag(Button btn)`, chi tiết tại [4].

Chương 3

Tổng quan cấu trúc chương trình

3.1 Khai báo lớp đối tượng Player

Lớp đối tượng này quản lý các thuộc tính liên quan đến thông tin của người chơi. Cụ thể:

- `name`: lưu tên của người chơi thuộc kiểu `string`.
- `mark`: lưu kí hiệu mà người chơi dùng thuộc kiểu `Image`.
- `Avatar`: lưu hình đại diện của người chơi thuộc kiểu `Image`.

3.2 Khai báo lớp đối tượng ChessBoardManager

Lớp đối tượng này quản lý các thuộc tính liên quan đến các tác vụ trên bàn cờ. Cụ thể:

- `chessBoard`: lưu tên của người chơi thuộc kiểu `Panel`.
- `player`: lưu trữ mảng tất cả các người chơi thuộc kiểu `Player`.

- `currentPlayer`: lưu giá trị vị trí của người chơi hiện tại trong mảng `player`.
- `playerName`: lưu trữ tên của người chơi hiển thị trên bảng thông tin thuộc kiểu `TextBox`.
- `playerMark`: lưu trữ kí hiệu quân cờ của người chơi hiển thị trên bảng thông tin thuộc kiểu `PictureBox`
- `playerAvatar`: lưu trữ ảnh đại diện của người chơi hiển thị trên bảng thông tin thuộc kiểu `PictureBox`.
- `matrix`: là mảng hai chiều lưu các Button được dùng làm các ô trên bàn cờ.

Cùng với một số phương thức như:

- `ChessBoardManager()`: khởi tạo thuộc tính của đối tượng thành các giá trị mong muốn hoặc mang giá trị mặc định tại thời điểm tạo đối tượng.
- `DrawChessBoard()`: thực hiện việc vẽ bản cờ với phương pháp đã thảo luận ở 2.6
- `btn_Click()`: sự kiện sau khi thực hiện thao tác click chuột vào các Button của các người chơi
- `Mark()`: sẽ đổi hình ảnh của Button được truyền vào (tức là Button vừa được click) dựa theo `CurrentPlayer` tức là thứ tự của người chơi trong `List<Player>`.
- `ChangePlayer()`: thực hiện hiển thị các thông tin người chơi lên màn hình bào gồm: tên, kí hiệu quân cờ, hình đại diện.
- `GetChessPoint()`: trả về toạ độ trên thuộc tính `matrix`.
- `isEndHorizontal()`: kiểm tra chiến thắng theo chiều ngang.
- `isEndVertical()`: kiểm tra chiến thắng theo chiều dọc.
- `isMainDiag()`: kiểm tra chiến thắng theo đường chéo chính.

- `isSubDiag()`: kiểm tra chiến thắng theo đường chéo phụ.
- `isEndGame()`: trả về `True` khi một trong các kiểm tra chiến thắng trả về `True`.
- `EndGame()`: Thực hiện xuất lệnh thông báo thúc trò chơi khi một trong các kiểm tra chiến thắng trả về `True`.

3.3 Khai báo lớp đối tượng Form1

Khi thực hiện việc thiết kế form trong C#, thì Visual Studio sẽ tiến hành tách Form thành 2 file "Form1.cs" xử lý nghiệp vụ, events... và "Form1.Designer.cs" thực hiện việc khai báo và thiết kế control. Trong class `Form1` ta có các thuộc tính

- `ChessBoard`: là bàn cờ thuộc kiểu `ChessBoardManager`.
- `pnlChessBoard`: là Panel của bàn cờ.
- `panelLogo`: là Panel của Logo.
- `panelInfo`: là Panel của khung thông tin người chơi.
- `label5ltw`: là Label hiển thị dòng "5 line to win" trong chương trình.
- `txbPlayerName`: là TextBox khung xuất hiển thị thông tin người chơi.
- `pctbLogo`: là PictureBox hiển thị hình ảnh logo trò chơi.
- `pctbMark`: là PictureBox hiển thị hình ảnh quân cờ.
- `pctbAvt`: là PictureBox hiển thị hình ảnh đại diện người chơi.
- `menuStrip`: là MenuStrip xây dựng Menu trên Form, cùng với các `ToolStripMenuItem` trên Menu.

Cùng với các phương thức sau:

- `InitializeComponent()`: dùng để khởi tạo các đối tượng có trên form như TextBox, PictureBox, Panel,...

- `Dispose()`: dùng để giải phóng các tài nguyên không được quản lý bất kì khi nào nó được gọi.
- `Form1()`: phương thức khởi tạo trước tiên gọi `InitializeComponent()` sau đó truyền các thuộc tính trên Form cần thiết hiển thị trên bàn cờ vào `ChessBoard`, sau đó thực hiển vẽ bàn cờ qua hàm `NewGame()`

Kết luận

Trong nội dung bài tiểu luận này tác giả đã giới thiệu qua trò chơi caro hay gomuku. Sau đó trình bày môi trường code trên Visual Studio, thư viện Windows Form, các giải thuật trong game Caro, và nếu tổng quan của cấu trúc chương trình.

Ngoài ra trong tương lai chương trình có thể cải tiến thêm về các luật chơi biến thể, kết nối với các máy tính khác qua mạng LAN, thêm giới hạn thời gian cho người chơi...

Tài liệu tham khảo

[1] Cấu trúc dự án c# - solution, project, file mã nguồn. https://tuhocict.com/solution-project-source-namespace/. Accessed: 2022-11-21.

[2] Gomoku. https://en.wikipedia.org/wiki/Gomoku. Accessed: 2022-11-21.

[3] Lập trình game caro với c# winform. https://howkteam.vn/course/lap-trinh-game-caro-voi-c-winform-14. Accessed: 2022-11-21.

[4] Trần Đạt Tín - Mã nguồn trình bày trong tiểu luận 2022. https://github.com/trandattin/caro-csharp. Accessed: 2022-03-25.

[5] Thanh, T. D. Viết game tetris chạy trên giao diện dòng lệnh (console user interface). *Trường Đại học Đà Lạt* (2022).