TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN
MÔN: **NHẬP MÔN LẬP TRÌNH**

HƯỚNG DẪN ĐỒ ÁN

CARO

TP.HCM, ngày 23 tháng 09 năm 2018

MỤC LỤC

1 Giới thiệu

Trong phần đồ án này ta sẽ phối hợp các kĩ thuật và cấu trúc dữ liệu cơ bản để xây dựng một trò chơi đơn giản, cờ caro.

Để thực hiện được đồ án này ta cần các kiến thức cơ bản như: xử lý tập tin, handle, cấu trúc dữ liệu mảng một/hai chiều…

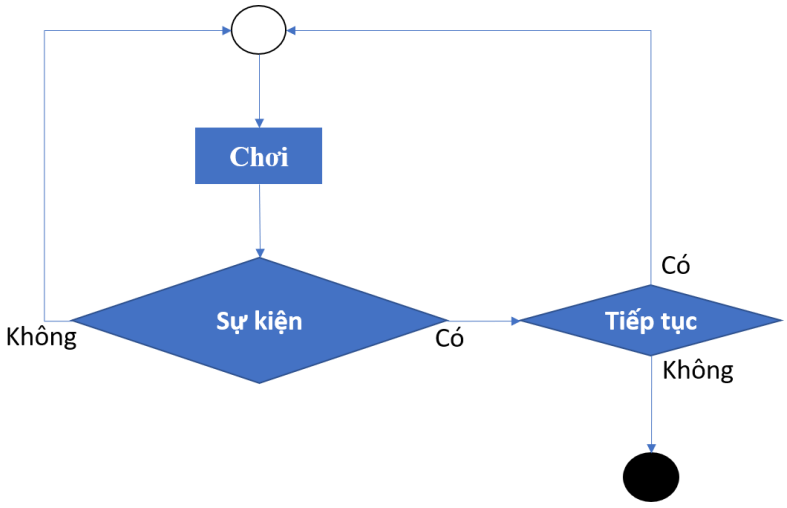
Phần hướng dẫn giúp sinh viên xây dựng trò chơi ở mức độ cơ bản, các em tự nghiên cứu để hoàn thiện một cách tốt nhất có thể.

2 Kịch bản trò chơi

Lúc đầu khi vào game sẽ xuất hiện bảng cờ caro, người chơi sẽ dùng các phím ‘W’, ‘A’, ‘S’, ‘D’ để điều chỉnh hướng di chuyển. Khi người chơi nhấn phím ‘enter’ thì sẽ xuất hiện dấu ‘X’ hoặc ‘O’ tùy vào lượt.

Khi một trong hai người chiến thắng theo luật caro thì màn hình xuất hiện dòng chữ chúc mừng người chiến thắng. Sau đó sẽ hỏi người dùng muốn tiếp tục chơi hay không, nếu chọn phím “y” thì chương trình khởi động lại dữ liệu từ đầu, còn nhấn phím bất kì thì thoát chương trình.

Trường hợp khi vị trí bàn cờ đã kín chỗ thì màn hình xuất hiện dòng chữ ‘Hai ben hoa nhau’. Sau đó hỏi người dùng có muốn thoát hay chơi tiếp tương tự như trên.



Hình 1: Sơ đồ kịch bản trò chơi

3 Các bước xây dựng trò chơi

Trong phần này ta sẽ lần lượt đi qua các bước xây dựng trò chơi. Lưu ý đây chỉ là một gợi ý lập trình, sinh viên có thể tự thiết kế mẫu phù hợp trong quá trình làm đồ án.

Bước 1: Trong bước này ta sẽ cố định màn hình với kích thước thích hợp, làm điều này giúp tránh trường hợp người dùng tự co giãn màn hình sẽ gây khó khăn trong quá trình xử lý.

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

Trong đoạn mã trên, kiểu HWND là một con trỏ trỏ tới chính cửa sổ Console. Để làm việc với các đối tượng đồ họa này, ta cần có những kiểu như thế. Cờ GWL_STYLE được xem là dấu hiệu để hàm GetWindowLong lấy các đặc tính mà cửa sổ Console đang có. Kết quả trả về của hàm GetWindowLong là một số kiểu long, ta sẽ hiệu chỉnh tại dòng số 4. Ý nghĩa là để làm mờ đi nút maximize và không cho người dùng thay đổi kích thước cửa sổ hiện hành. Sau khi đã hiệu chỉnh xong, ta dùng hàm SetWindowLong để gán kết quả hiệu chỉnh trở lại. Ta có thể thử nghiệm hàm trên và tự xem kết quả.

Bước 2: Trong trò chơi sẽ có rất nhiều vị trí mà ta muốn in tại đó, vì vậy ta cần có khả năng di chuyển tới tất cả các vị trí trong màn hình console.

Dòng	// Hàm nhóm View
1	`void GotoXY(int x, int y) {`
2	`    COORD coord;`
3	`    coord.X = x;`
4	`    coord.Y = y;`
5	`    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);`
6	`}`

Trong đoạn mã này ta sử dụng struct _COORD (COORD), đây là một cấu trúc dành xử lý cho tọa độ trên màn hình console. Ta gán hoành độ và tung độ cho biến coord sau đó thiết lập vị trí lên màn hình bằng hàm SetConsoleCursorPosition. Lưu ý: hàm này cần một đối tượng chính là màn hình console (màn hình đen), vì vậy ta cũng cần có một con trỏ trỏ tới đối tượng này (HANDLE thực chất là void*). Ta có được bằng cách gọi hàm GetStdHandle với tham số là một cờ STD_OUTPUT_HANDLE.

Bước 3: Tiếp theo ta cần có dữ liệu để phục vụ trò chơi, để đơn giản ta sử dụng biến toàn cục. Ý nghĩa của từng biến và hằng số sinh viên tự xem trong bảng.

Dòng	
1	`//Hằng số`
2	`#define BOARD_SIZE 12 // Kích thước ma trận bàn cờ`
3	`#define LEFT 3 // Tọa độ trái màn hình bàn cờ`
4	`#define TOP 1 // Tọa độ trên màn hình bàn cờ`
5	`// Khai báo kiểu dữ liệu`
6	`struct _POINT {int x, y, c;}; // x: tọa độ dòng, y: tọa độ cột, c: đánh dấu`
7	`_POINT _A[BOARD_SIZE][BOARD_SIZE]; //Ma trận bàn cờ`
8	`bool _TURN; //true là lượt người thứ nhất và false là lượt người thứ hai`
9	`int _COMMAND; // Biến nhận giá trị phím người dùng nhập`
10	`int _X, _Y; //Tọa độ hiện hành trên màn hình bàn cờ`

Bước 4: Bước tiếp theo ta sẽ xây dựng hàm ResetData, mục tiêu của hàm này là để thiết lập dữ liệu về trạng thái ban đầu. Ta thiết lập lại dữ liệu bàn cờ bằng giá trị tọa độ trên màn hình và biến c đại diện cho việc đánh dấu hay chưa đánh dấu.

Dòng	
1	`//Hàm khởi tạo dữ liệu mặc định ban đầu cho ma trận bàn cờ (hàm nhóm Model)`
2	`void ResetData() {`
3	`  for(int i = 0 ; i < BOARD_SIZE ; i++){`
4	`    for(int j = 0 ; j < BOARD_SIZE ; j++){`
5	`      _A[i][j].x = 4 * j + LEFT + 2; // Trùng với hoành độ màn hình bàn cờ`
6	`      _A[i][j].y = 2 * i + TOP + 1; // Trùng với tung độ màn hình bàn cờ`
7	`      _A[i][j].c = 0; // 0 nghĩa là chưa ai đánh dấu, nếu đánh dấu phải theo quy //định như sau: -1 là lượt true đánh, 1 là lượt false đánh`
8	`    }`
9	`  }`
10	`  _TURN = true; _COMMAND = -1; // Gán lượt và phím mặc định`
11	`  _X = _A[0][0].x; _Y = _A[0][0].y; // Thiết lập lại tọa độ hiện hành ban đầu`
12	`}`

Bước 5: Công việc tiếp theo ta cần xây dựng là vẽ một hình vuông bao quanh làm phạm vi. Hàm này vẽ hình vuông dài và rộng là BOARD_SIZE.

Dòng	**// Hàm nhóm View**
1	`void DrawBoard(int pSize){`
2	`    for(int i = 0; i <= pSize; i++){`
3	`      for(int j = 0; j <= pSize; j++){`
4	`        GotoXY(LEFT + 4 * i, TOP + 2 * j);`
5	`        printf(".");`
6	`      }`

7	}
8	}

Bước 6: Tiếp theo ta sẽ xây dựng hàm StartGame(), hàm này thực chất là tập các công việc cần làm trước khi vào trò chơi

Dòng	**// Hàm nhóm Control**
1	`void StartGame() {`
2	`   system("cls");`
3	`   ResetData(); // Khởi tạo dữ liệu gốc`
4	`   DrawBoard(BOARD_SIZE); // Vẽ màn hình game`
5	`}`

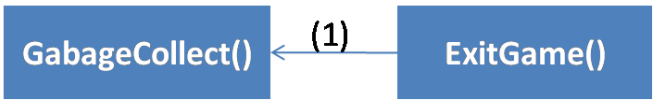
Dòng mã đầu tiên để xóa trắng màn hình, dòng mã thứ hai là lời gọi hàm ResetData() nhằm khôi phục dữ liệu mặc định ban đầu. Kế đó là gọi hàm DrawBoard() để vẽ hình chữ nhật xung quanh.

Hình 2: Sơ đồ gọi hàm từ StartGame()

Bước 7: Ngoài hàm StartGame(), ta cần xây dựng hai hàm ExitGame() và GabageCollect() thực hiện chức năng thoát và dọn dẹp tài nguyên khi dừng trò chơi.

Dòng	
1	`//Hàm dọn dẹp tài nguyên (hàm nhóm Model)`
2	`void GabageCollect()`
3	`{`
4	`  // Dọn dẹp tài nguyên nếu có khai báo con trỏ`
5	`}`
6	`//Hàm thoát game (hàm nhóm Control)`
7	`void ExitGame() {`
8	`   system("cls");`
9	`   GabageCollect();`
10	`   //Có thể lưu game trước khi exit`
11	`}`

Trong hàm ExitGame() ta thực hiện xóa trắng màn hình và dọn dẹp tài nguyên. Lưu ý: Nếu ứng dụng không sử dụng biến con trỏ thì hàm GabageCollect() để trống.



Hình 3: Sơ đồ gọi hàm từ ExitGame()

Bước 8: Tiếp theo ta xây dựng hàm xử lý khi người chơi thắng/thua. Khi người chơi thắng/thua/hòa, hàm trên đơn giản in ra dòng chữ báo hiệu. Ngoài ra xây dựng them hàm tiện ích AskContinue() để nhận phím quyết định có tiếp tục hay không của người dùng.

Dòng	// Hàm nhóm View
1	//Hàm xử lý khi người chơi thua
2	int ProcessFinish(int pWhoWin) {

3	GotoXY(0, _A[BOARD_SIZE - 1][BOARD_SIZE - 1].y + 2); // Nhảy tới vị trí
	// thích hợp để in chuỗi thắng/thua/hòa

4	switch(pWhoWin){
5	case -1:
6	printf("Nguoi choi %d da thang va nguoi choi %d da thua\n", true, false);
7	break;
8	case 1:
9	printf("Nguoi choi %d da thang va nguoi choi %d da thua\n", false, true);
10	break;
11	case 0:
12	printf("Nguoi choi %d da hoa nguoi choi %d\n", false, true);
13	break;
14	case 2:
15	_TURN = !_TURN; // Đổi lượt nếu không có gì xảy ra
16	}
17	GotoXY(_X, _Y); // Trả về vị trí hiện hành của con trỏ màn hình bàn cờ
18	return pWhoWin;
19	}
20	int AskContinue()
21	GotoXY(0, _A[BOARD_SIZE - 1][BOARD_SIZE - 1].y + 4);
22	printf("Nhan 'y/n' de tiep tuc/dung: ");
23	return toupper(getch());
24	}

Bước 9: Để biết được xem có người thắng/thua hoặc hòa nhau, ta cần có hàm kiểm tra. Lưu ý ta tự quy định đầu ra hàm này, ví dụ hàm trả ra 0 có nghĩa là hòa, -1 có nghĩa lượt 'true' thắng, 1 có nghĩa lượt 'false' thắng và 2 có nghĩa là chưa ai thắng.

Dòng	// Hàm nhóm Model
1	//Hàm kiểm tra xem có người thắng/thua hay hòa
2	int TestBoard()
3	{
4	if(<Ma trận đầy>) return 0; // Hòa
5	else {
6	if (<tồn tại điều kiện thắng theo luật caro>)
7	return (_TURN == true ? -1 : 1); // -1 nghĩa là lượt 'true' thắng

```
8        Else
9          return 2; // 2 nghĩa là chưa ai thắng
10     }
11  }
```

Bước 10: Ta cần xây dựng hàm đánh dấu vào ma trận bàn cờ khi người chơi nhấn phím ‘enter’.

```
Dòng  // Hàm nhóm Model
1   int CheckBoard(int pX, int pY){
2     for(int i = 0; i < BOARD_SIZE; i++){
3       for(int j = 0; j < BOARD_SIZE; j++){
4         if(_A[i][j].x == pX && _A[i][j].y == pY && _A[i][j].c == 0){
5          if(_TURN == true) _A[i][j].c = -1; // Nếu lượt hiện hành là true thì c = -1
6          else __A[i][j].c = 1; // Nếu lượt hiện hành là false thì c = 1
7          return _A[i][j].c;
8        }
9      }
10   }
11   return 0;
12  }
```

Bước 11: Tiếp theo ta sẽ xây dựng các hàm di chuyển trên màn hình bàn cờ.

```
Dòng  // Hàm nhóm Control
1   void MoveRight() {
2       if (_X < _A[BOARD_SIZE - 1][BOARD_SIZE - 1].x)
3       {
4           _X += 4
5           GotoXY(_X, _Y);
7       }
8   }
9   void MoveLeft() {
10      if (_X > _A[0][0].x) {
11          _X -= 4;
12          GotoXY(_X, _Y);
14      }
15  }
16  void MoveDown() {
17      if (_Y < _A[BOARD_SIZE - 1][BOARD_SIZE - 1].y)
18      {
19          _Y += 2;
20          GotoXY(_X, _Y);
22      }
```

```
23  }
24  void MoveUp() {
25      if (_Y > _A[0][0].y) {
26          _Y -= 2;
27          GotoXY(_X, _Y);
29      }
30  }
```

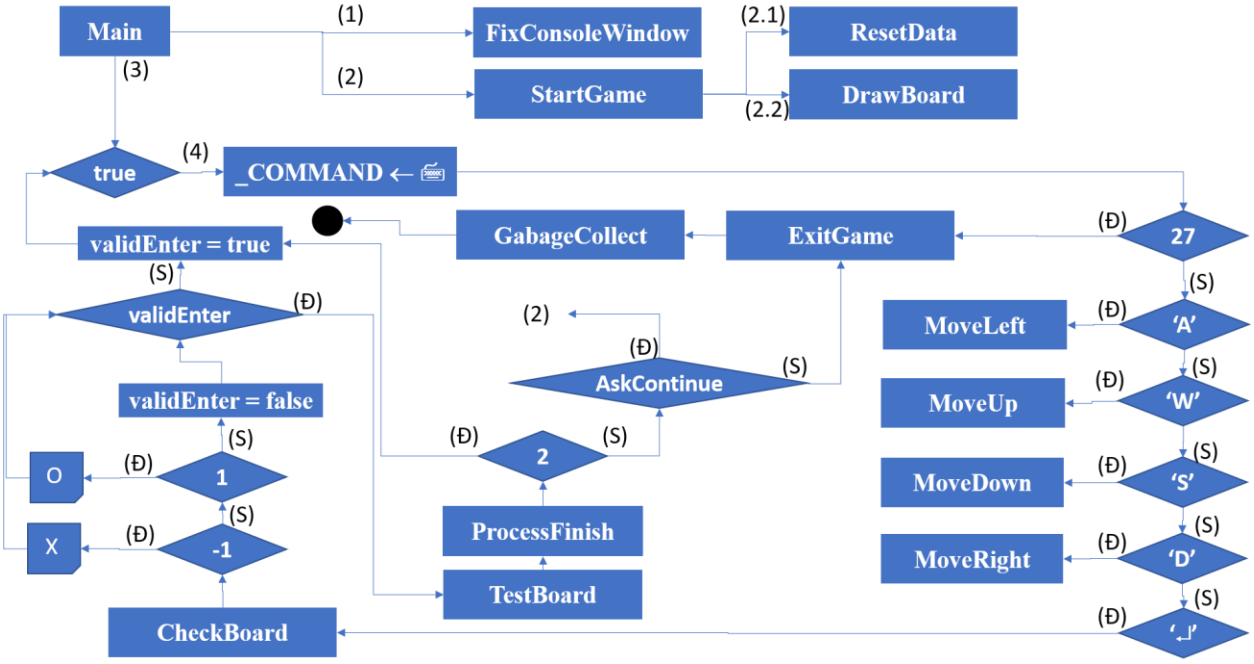
Trong quá trình di chuyển trên màn hình bàn cờ, nếu vượt quá phạm vi thì ta không xử lý, ngược lại ta thực hiện cập nhật tọa độ vị trí mới trên màn hình bàn cờ

Bước 12: Cuối cùng ta xây dựng hàm main để thực hiện tiếp nhận phím từ người dùng.

```
Dòng
1   void main()
2   {
3       FixConsoleWindow();
4       StartGame();
        bool validEnter = true;
5       while (1)
6       {
7           _COMMAND = toupper(getch());
8           if (_COMMAND == 27)
9           {
10              ExitGame();
11              return;
12          }
13          else {
14              if (_COMMAND == 'A') MoveLeft();
15              else if (_COMMAND == 'W') MoveUp();
16              else if (_COMMAND == 'S') MoveDown();
17              else if (_COMMAND == 'D') MoveRight();
18              else if (_COMMAND == 13){// Người dùng đánh dấu trên màn hình bàn cờ
19                  switch(CheckBoard(_X, _Y)){
20                      case -1:
21                          printf("X"); break;
22                      case 1:
23                          printf("O"); break;
24                      case 0: validEnter = false; // Khi đánh vào ô đã đánh rồi
25                  }
26                  // Tiếp theo là kiểm tra và xử lý thắng/thua/hòa/tiếp tục
27                  if(validEnter == true){
28                    switch(ProcessFinish(TestBoard())){
```

```
29                          case -1: case 1: case 0:
30                              if(AskContinue() != 'Y'){
31                                  ExitGame(); return 0;
32                              }
33                              else StartGame();
34                      }
35                  }
36                  validEnter = true; // Mở khóa
37              }
38          }
39  }
```

Trong hàm main, bước đầu tiên là ta cố định màn hình ngăn ngừa người chơi thay đổi kích thước sẽ gây vỡ giao diện chương trình. Sau đó ta gọi StartGame() để thực hiện chuẩn bị dữ liệu cho màn chơi. Hàm main() **liên tục** chờ phím từ người chơi, sau đó tùy vào phím người dùng chọn chương trình sẽ có phản ứng thích hợp.



Hình 4: Sơ đồ gọi hàm từ hàm main ()

4 YÊU CẦU ĐỒ ÁN

Trong phần hướng dẫn trên ta còn thiếu một vài chức năng cơ bản

4.1 Xử lý lưu/tải trò chơi (save/load) - 3đ

Trong hướng dẫn chưa xử lý việc lưu/tải trò chơi. Ta cần cài đặt thêm tính năng này. Khi người dùng nhấn phím 'L' thì sẽ hiện dòng chữ yêu cầu người dùng nhập tên tập tin muốn lưu trạng thái hiện hành của trò chơi. Khi người dùng nhấn phím 'T' thì sẽ hiện dòng chữ yêu cầu người dùng nhập tên tập tin muốn tải lại.

4.2 Nhận biết thắng/thua/hòa - 3đ

Ta cần bổ sung tính năng kiểm tra quy luật thắng/thua /hòa trong caro. Từ đó sẽ kiểm tra sau mỗi bước đi của người chơi

4.3 Xử lý hiệu ứng thắng/thua/hòa - 2đ

Trong hướng dẫn, khi thắng/thua/hòa chỉ hiển thị dòng chữ báo hiệu đơn giản. Ta hãy cài đặt hiệu ứng giúp sinh động hơn

4.4 Xử lý giao diện màn hình khi chơi - 1đ

Trong quá trình chơi, cho hiển thị các thông số của hai người chơi, ví dụ người chơi thứ nhất đã đánh bao nhiêu bước, người chơi thứ hai đã thua mấy ván… Sinh viên tự tổ chức giao diện màn hình sao cho rõ ràng, sinh động.

4.5 Xử lý màn hình chính - 1đ

Trước khi vào trò chơi, hiển thị danh sách menu, ví dụ như "New Game", "Load Game", "Settings", … Như vậy sẽ giúp chương trình caro hoàn thiện và giống thực tế một trò chơi hơn.