BÁO CÁO PROJECT LỚN / KĨ THUẬT LẬP TRÌNH

Những sinh viên tham gia thực hiện : (Lớp: 010412410106 – CN2303CLCA)

Họ và tên:	MSSV:
Nguyễn Huy Lai	094205001839
Trần Minh Thiện	079205010726
Phạm Trương Anh Tú	051205000251

Ý tưởng của nhóm :

Tạo ra một game puzzle dạng 4x4 sử dụng giao diện text dễ nhìn , mục tiêu là đưa các số từ 1-15 về vị trí đúng để giành chiến thắng , trò chơi bao gồm phần Bắt đầu chơi - Hướng dẫn – AutoRun - Thoát

Phần hướng dẫn sẽ hướng dẫn người chơi luật và cách chơi của game .

Khi chọn Bắt đầu chúng ta có thể tạo “người chơi mới” , đặt tên nhân vật hoặc sử dụng “người dùng” mà chúng ta đã tạo và lưu trước đó .

AutoRun sẽ tự động giải câu đố và giúp chúng ta giành chiến thắng .

Trò chơi còn bao gồm cả các cấp độ khác nhau từ level 1-5 , mỗi level sẽ cung cấp một độ khó khác nhau để tạo thử thách cho người chơi , sau khi tạo nhân vật thì sẽ bắt đầu từ level 1 và phải hoàn thành câu đố để đạt đến những level cao hơn .

Phương án thực hiện :

Project trên được lập trình theo hướng đối tượng và hướng thủ tục , bao gồm các thành phần chính như :

Header File :

#include<iostream>: Luồng nhập/ xuất tiêu chuẩn.

#include<windows.h>: API Windows cho các chức năng liên quan đến console.

#include<iomanip>: Các bộ chuyển đổi định dạng nhập/ xuất.

#include<fstream>: Luồng tệp để đọc và ghi tệp.

#include<queue> và **#include<vector>**: Containers để xếp hàng và lưu trữ dữ liệu.

#include<algorithm>: Các hàm thuật toán như **swap()**.

Bảng Trò Chơi :

Bảng trò chơi được biểu diễn bằng một mảng 2D board[][], trong đó mỗi phần tử đại diện cho một ô trong trò chơi xếp hình.

Hàm Random Move :

Hàm random_move tạo một bước di chuyển ngẫu nhiên (lên, xuống, trái hoặc phải) cho ô trống trong trò chơi. Bước di chuyển không được vượt qua ranh giới của trò chơi.

Hàm Auto Run :

Hàm auto_run giúp chương trình hoạt động một cách tự động và thực hiện các bước di chuyển ngẫu nhiên cho đến khi trò chơi được giải quyết hoặc số lần di chuyển đã đạt đến tối đa.

Class player :

Có một lớp Player với các thuộc tính name và level. Nó được sử dụng để lưu trữ thông tin người chơi.

Hàm main :

Hàm main là điểm bắt đầu của chương trình.

Chương trình bắt đầu bằng cách hiển thị một menu chính (Bắt đầu, Hướng dẫn, Thoát, Auto run) sử dụng hàm disp_o_menu.

Tùy thuộc vào lựa chọn của người dùng, chương trình có thể bắt đầu một trò chơi mới, cung cấp hướng dẫn, thực hiện chạy tự động hoặc thoát.

Cấp Độ Trò Chơi và Tiến Triển:

Các cấp độ khác nhau của trò chơi được biểu diễn bởi các mảng 2D khác nhau (level1, level2, ...).

Tiến triển của người chơi được lưu trong một tệp có tên "data.txt."

Chương trình cho phép người chơi tạo một tài khoản mới hoặc tải một tài khoản đã tồn tại, lưu trữ và cập nhật dữ liệu người chơi trong tệp.

Hàm Game play :

Hàm game_play thực hiện cái tương tác trong trò chơi , cho phép người chơi thực hiện các bước di chuyển bằng các phím mũi tên cho đến khi trò chơi được giải quyết hoặc người chơi quyết định thoát.

Hàm Hiển Thị Menu:

Các hàm disp_o_menu xử lý hiển thị và điều hướng của các menu khác nhau, chẳng hạn như menu chính, lựa chọn cấp độ và lựa chọn người chơi.

Hàm Sleep:

Hàm Sleep được sử dụng để tạo ra các đợt trễ để kiểm soát tốc độ của trò chơi.

Thuộc Tính Màu Sắc Của Console:

SetConsoleTextAttribute được sử dụng để đặt thuộc tính màu của văn bản trên console.

Thoát và Khởi Động Lại:

Chương trình cho phép người chơi thoát hoặc khởi động lại sau khi hoàn thành một cấp độ.

Các kiến thức được sử dụng trong project :

- C++ Basics:

Sử dụng cú pháp cơ bản của C++, bao gồm khai báo biến, mảng, hàm, lệnh điều kiện, vòng lặp, và toán tử.

- File I/O (Đọc và Ghi Tệp):

  Sử dụng thư viện fstream để đọc và ghi dữ liệu vào tệp tin, trong trường hợp này, để lưu trạng thái của người chơi.

- Windows API (API Windows):

  Sử dụng windows.h để tương tác với console và thực hiện các chức năng như đặt màu sắc văn bản, thiết lập thuộc tính của console.

- Random Number Generation (Sinh Số Ngẫu Nhiên):

  Sử dụng hàm rand() để tạo số ngẫu nhiên trong việc thực hiện các bước di chuyển ngẫu nhiên trong trò chơi.

- Thư viện Standard Template Library (STL):

  Sử dụng các container như vector và queue từ STL để quản lý và lưu trữ dữ liệu.

- Hướng Đối Tượng (Object-Oriented Programming - OOP):

  Sử dụng lớp Player để đại diện cho thông tin của người chơi, áp dụng các khái niệm OOP như đóng gói và truy cập dữ liệu qua các phương thức.

- **Xử Lý Sự Kiện Bàn Phím:**

  Sử dụng hàm GetAsyncKeyState để theo dõi trạng thái các phím bàn phím và thực hiện các hành động tương ứng.

- **Lập Trình Đa Luồng (Multithreading):**

  Sử dụng hàm Sleep để tạo đợt trễ trong trò chơi, có thể coi là một hình thức của lập trình đa luồng đơn giản để kiểm soát tốc độ của trò chơi.

- **Algorithm Design (Thiết Kế Thuật Toán):**

  Thực hiện thuật toán xáo trộn bảng để bắt đầu trò chơi và kiểm tra trạng thái hoàn thành của trò chơi.

- **Game Logic Design (Thiết Kế Logic Trò Chơi):**

  Xây dựng logic của game, kiểm tra điều kiện chiến thắng, và quản lý quá trình tiến triển qua các cấp độ.

- **User Interface Design (Thiết Kế Giao Diện Người Dùng):**

  Thiết kế các menu và giao diện người dùng sử dụng các hàm như disp_o_menu để hiển thị và điều hướng qua các tùy chọn.

Từ những kiến thức trên nhóm em bắt đầu viết project như sau :

Project có 5 đối tượng chính :

1/ Player (Người Chơi):

Đối tượng này đại diện cho thông tin của người chơi, bao gồm tên và cấp độ hiện tại. Có các phương thức để nhập và hiển thị thông tin người chơi.

2/ Game (Trò Chơi):

Đối tượng quản lý logic của trò chơi, bao gồm các hàm để thực hiện trò chơi, kiểm tra điều kiện chiến thắng, và quản lý các cấp độ khác nhau.

3/ Menu (Menu):

Đối tượng quản lý hiển thị và xử lý menu trong trò chơi, bao gồm các hàm để hiển thị menu, chấp nhận đầu vào từ người chơi và thực hiện các hành động tương ứng.

4/ Board (Bảng):

Đối tượng này đại diện cho bảng trò chơi, bao gồm cấu trúc dữ liệu lưu trữ các ô và các phương thức để hiển thị bảng, di chuyển ô trống, và kiểm tra trạng thái chiến thắng.

5/ AutoRunner (Chạy Tự Động):

Đối tượng quản lý quá trình chạy tự động của trò chơi, bao gồm các hàm để thực hiện các bước di chuyển ngẫu nhiên và kiểm tra trạng thái chiến thắng.

Project và giải thích :

```
#include <iostream>
#include <windows.h>
#include <iomanip>
#include <fstream>
#include <queue>
#include <vector>
#include <algorithm>

using namespace std;

// Trạng thái mục tiêu của trò chơi
int board[][4] = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}, {13, 14, 15, 99}};

// Hàm di chuyển ngẫu nhiên một ô trong trò chơi
int random_move(int level[][4], int& ptr1, int& ptr2) {
    int direction;
    while (true) {
        direction = rand() % 4;
        // Kiểm tra xem di chuyển có hợp lệ hay không
        if ((direction == 0 && ptr1 > 0) ||
            (direction == 1 && ptr1 < 3) ||
```

```
        (direction == 2 && ptr2 > 0) ||
        (direction == 3 && ptr2 < 3)) {
        break;
    }
}
// Thực hiện di chuyển dựa trên hướng đã chọn
switch (direction) {
    case 0:
        swap(level[ptr1][ptr2], level[ptr1 - 1][ptr2]);
        ptr1--;
        break;
    case 1:
        swap(level[ptr1][ptr2], level[ptr1 + 1][ptr2]);
        ptr1++;
        break;
    case 2:
        swap(level[ptr1][ptr2], level[ptr1][ptr2 - 1]);
        ptr2--;
        break;
    case 3:
        swap(level[ptr1][ptr2], level[ptr1][ptr2 + 1]);
        ptr2++;
        break;
```

```
    }
    return direction;  // Trả về hướng di chuyển
}

// Hàm chạy tự động trò chơi cho một cấp độ
int auto_run(int level[][4], int correct, int l) {
    int win = correct, ptr1, ptr2;
    int moveCount = 0;
    // Tìm vị trí của ô trống
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            if (level[i][j] == 99) {
                ptr1 = i;
                ptr2 = j;
                break;
            }
        }
    }
    // Lặp cho đến khi trò chơi hoàn thành
    while (win != 16) {
        int direction = random_move(level, ptr1, ptr2);
        // Hiển thị trạng thái trò chơi và thông tin
        system("cls");
```

```
        cout << "Auto Run - Game level " << l << endl << endl;
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                if (level[i][j] == 99) {

SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 11);
                    cout << setw(3) << "__";

SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 15);
                } else {
                    cout << setw(3) << level[i][j];
                }
            }
            cout << endl;
        }
        cout << "Correct positions=" << setw(2) << win << " Incorrect= " <<
setw(2) << 16 - win;
        Sleep(200);
        // Kiểm tra lại số ô đúng sau mỗi bước di chuyển
        win = 0;
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                if (level[i][j] == board[i][j]) {
                    win++;
```

```
            }
        }
    }
    moveCount++;
    // Điều kiện dừng nếu số bước di chuyển vượt quá giới hạn
    if (moveCount >= 1000) {
        break;
    }
  }
  // Hiển thị kết quả khi trò chơi hoàn thành
  system("cls");
  cout << endl << "Auto Run completed in " << moveCount << " moves!"
<< endl;
  Sleep(200);
  return 0;
}

// Định nghĩa lớp Player
class Player {
public:
  char name[40];
  int level;
};
```

```
int main() {
    SetConsoleOutputCP(CP_UTF8);

    // Khai báo các hàm
    void showconst(int num[][4]);
    int game_play(int level[][4], int correct, int l);
    int disp_o_menu(char item[][40], int n, char* string);
    int disp_o_menu(int level);
    Player* disp_o_menu(Player* ptr, int n);
    Player data[40], *ptr[40];
    char MMenu[][40] = {"Bắt đầu", "Hướng dẫn", "Thoát", "Auto run"};
    char SMenu[][40] = {"Người mới", "Đã có tk"};
    char yes_no[][40] = {"Có", "Không"};
    int level1[][4] = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 12, 15}, {13, 14, 11, 99}};
    int level2[][4] = {{1, 6, 2, 3}, {5, 10, 4, 9}, {8, 14, 7, 12}, {13, 15, 11, 99}};
    int level3[][4] = {{1, 6, 3, 5}, {4, 6, 2, 8}, {13, 10, 11, 12}, {9, 14, 15, 99}};
    int level4[][4] = {{1, 7, 3, 6}, {5, 4, 2, 8}, {9, 13, 11, 12}, {10, 14, 15, 99}};
    int level5[][4] = {{1, 4, 3, 7}, {5, 6, 2, 8}, {9, 10, 13, 12}, {11, 14, 15, 99}};
    int correct[5] = {2, 13, 4, 4, 5};
    int ret;

    // Menu chính
```

```
menu:
    int choice1 = disp_o_menu(MMenu, 4, "Main Menu");
    if (choice1 == 0) {
        // Menu lựa chọn người chơi mới hoặc sử dụng tài khoản đã có
        Sleep(100);
        int choice2 = disp_o_menu(SMenu, 2, "Người dùng");
        fstream file;
        char name[40];
        int level, levels;

        // Đọc dữ liệu từ tệp tin
        file.open("data.txt", ios::app | ios::in | ios::out);
        int id = 0;
        while (file >> name >> level) {
            strcpy(data[id].name, name);
            data[id].level = level;
            ptr[id] = &data[id];
            id++;
        }
        file.close();

        if (choice2 == 0) {
            // Tạo người chơi mới
```

```
        int out = 1;
        do {
            system("cls");
            SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
9);
            cout << "Nhập tên mà bạn muốn nhập. ";
            cin >> name;
            out = 1;
            // Kiểm tra tên có tồn tại hay không
            for (int j = 0; j < id; j++) {
                if (!strcmp(data[j].name, name)) {
                    out = 0;
                    cout << "Tên của bạn đã tồn tại!" << endl;
                    Sleep(900);
                    break;
                }
            }
        } while (out != 1);

        // Chờ người chơi nhấn Enter để tiếp tục
        while (GetAsyncKeyState(VK_RETURN) != 0)
            cout << ".....";
        while (GetAsyncKeyState(VK_ESCAPE) != 0);
```

```
    level = 1;
    levels = level;
} else {
    // Sử dụng tài khoản đã có
    cout << "Đang tải menu..";
    cout << ".";
    Sleep(200);
    cout << ".";
    Sleep(200);

    // Chọn người chơi từ danh sách
    Player* select = disp_o_menu(ptr[0], id);
    strcpy(name, select->name);

    system("cls");
    cout << "đang xử lý...";
    cout << ".";
    Sleep(200);
    level = select->level;
    levels = disp_o_menu(level);
    Sleep(500);
}
```

```
system("cls");
do {
    system("cls");
    char a;
    switch (levels) {
        case 1:
            ret = game_play(level1, correct[0], 1);
            break;
        case 2:
            ret = game_play(level2, correct[1], 2);
            break;
        case 3:
            ret = game_play(level3, correct[2], 3);
            break;
        case 4:
            ret = game_play(level4, correct[3], 4);
            break;
        case 5:
            ret = game_play(level5, correct[4], 5);
            break;
    }
```

```
        // Xử lý khi trò chơi kết thúc
        if (ret == 1000) {
            system("cls");
            cout << "Bạn ";
            Sleep(250);
            cout << "đang ";
            Sleep(250);
            cout << "muốn ";
            Sleep(250);
            cout << "thoát ";
            Sleep(250);
            while (GetAsyncKeyState(VK_RETURN) != 0)
                cout << ".....";
            goto menu;
        }

        levels++;

        // Lưu tiến trình nếu chơi đến cấp độ mới
        if (levels > level && levels != 6) {
            level = levels;
            int c = disp_o_menu(yes_no, 2, "Bạn có muốn lưu tiến trình trò chơi lại ?");
```

```
            if (c == 0) {
                cout << "Đang lưu.";
                int x = id;
                cout << ".";
                file.open("data.txt", ios::out);
                cout << ".";
                file.close();
                cout << ".";
                file.open("data.txt", ios::out | ios::app);
                cout << ".";
                while ((x - 1) >= 0) {
                    if (!strcmp(data[x - 1].name, name)) {
                        data[x - 1].level = level;
                    }
                    file << data[x - 1].name << endl << data[x - 1].level <<
endl;
                    x--;
                }
                if (choice2 == 0) {
                    file << name << endl << level << endl;
                }
                cout << ".";
                file.close();
```

```
            } else {
                while (GetAsyncKeyState(VK_RETURN) != 0);
            }
        }

        // Kết thúc trò chơi khi chơi đến cấp độ cuối cùng
        if (levels > 5) {
            cout << endl << "Game completed";
            Sleep(40);
            break;
        }

        while (GetAsyncKeyState(VK_RETURN) != 0);
        int c = disp_o_menu(yes_no, 2, "Bạn có muốn chơi cấp độ tiếp theo?");
        if (c == 1) {
            while (GetAsyncKeyState(VK_RETURN) != 0);
            goto menu;
        }
    } while (1);
} else if (choice1 == 1) {
// Hiển thị hướng dẫn chói
system("cls");
```

```
                SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),2);
                cout<< "----------Hướng dẫn cách chơi----------"<<endl<<endl;
                SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),3);
                cout<< "-----Ở đây sẽ có ngẫu nhiên số từ 1-15."<<endl;
                cout<< "-----Nhiệm vụ của bạn là làm sao để nó về đúng vị trí của nó."<<endl;
                cout<< "-----Ví dụ:"<<endl;
                showconst(level1);
                SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),3);
                cout<<endl<<"-----trở thành..."<<endl<<endl;
                showconst(board);
                cout<< "-----Chỉ duy nhất ô trống là bạn có thể điều khiển qua lại với nhau."<<endl;
                cout<< "-----Hãy sử dụng phím mũi tên <^> để di chuyển ô trống đó"<<endl;
                cout<<"          Good luck for you!!!          "<<endl;
        system("pause");
                while(GetAsyncKeyState(VK_RETURN)!=0);
                goto menu;
          }
          else if(choice1==3){
// Lựa chọn chức năng autoRun
        system("cls");
        ret = auto_run(level1, correct[0], 1);
          }
```

```
        else {
                return 0;
        }
}

// Hàm hiển thị trạng thái hiện tại của trò chơi trên màn hình console
void showconst(int num[][4]) {
   SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 6);  // Thiết lập màu chữ trên console là màu vàng
   for (int i = 0; i < 4; i++) {
      for (int j = 0; j < 4; j++) {
         if (num[i][j] == 99)
            cout << setw(3) << "__";  // Hiển thị ô trống
         else
            cout << setw(3) << num[i][j];  // Hiển thị số trong ô
      }
      cout << endl;
   }
   SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 4);  // Thiết lập màu chữ trên console là màu đỏ
}

// Hàm hiển thị menu chọn người chơi và trả về con trỏ đến người chơi được chọn
Player* disp_o_menu(Player* ptr, int n) {
```

```
    int pointer = 0;
    for (;;) {
        Player* ptrt = ptr;
        system("cls");  // Xóa màn hình console
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 2);  // Thiết lập màu chữ trên console là màu xanh lá cây
        cout << setw(40) << "Name" << setw(5) << " Level" << endl;
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 15);  // Thiết lập màu chữ trên console là màu trắng
        for (int i = 0; i < n; i++) {
            if (i == pointer)
                SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 11);  // Thiết lập màu chữ trên console là màu vàng
            cout << setw(40) << ptrt->name << " " << setw(5) << ptrt->level << endl;
            SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 15);  // Thiết lập màu chữ trên console là màu trắng
            ptrt++;
        }
        while (true) {
            if (GetAsyncKeyState(VK_UP) != 0) {
                pointer -= 1;
                if (pointer == -1)
                    pointer = n - 1;
                break;
            } else if (GetAsyncKeyState(VK_DOWN) != 0) {
                pointer += 1;
```

```
            if (pointer == n)
                pointer = 0;
            break;
        } else if (GetAsyncKeyState(VK_RETURN) != 0)
            return ptr + pointer;
    }
    Sleep(120);
  }
}

// Hàm hiển thị menu với danh sách các mục (items) và trả về chỉ số của mục được chọn
int disp_o_menu(char item[][40], int n, char* string) {
  int pointer = 0;
  for (;;) {
    system("cls");  // Xóa màn hình console
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 2);  // Thiết lập màu chữ trên console là màu xanh lá cây
    cout << string << endl << endl;
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 15);  // Thiết lập màu chữ trên console là màu trắng
    for (int i = 0; i < n; i++) {
      if (i == pointer)
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 11);  // Thiết lập màu chữ trên console là màu vàng
      cout << item[i] << endl;
```

```
            SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 15);  // Thiết lập màu chữ trên console là màu trắng
        }
        while (true) {
            if (GetAsyncKeyState(VK_UP) != 0) {
                pointer -= 1;
                if (pointer == -1)
                    pointer = n - 1;
                break;
            } else if (GetAsyncKeyState(VK_DOWN) != 0) {
                pointer += 1;
                if (pointer == n)
                    pointer = 0;
                break;
            } else if (GetAsyncKeyState(VK_RETURN) != 0)
                return pointer;
        }
        Sleep(120);
    }
}

// Hàm hiển thị menu chọn cấp độ (level) của trò chơi và trả về cấp độ được chọn
int disp_o_menu(int level) {
    int pointer = 0;
    for (;;) {
```

```
        system("cls");  // Xóa màn hình console
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 2);  // Thiết lập màu chữ trên console là màu xanh lá cây
        cout << "Select Level" << endl;
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 15);  // Thiết lập màu chữ trên console là màu trắng
        for (int i = 0; i < level; i++) {
            if (i == pointer)
                SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 11);  // Thiết lập màu chữ trên console là màu vàng
            cout << "level " << i + 1 << endl;
            SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 15);  // Thiết lập màu chữ trên console là màu trắng
        }
        for (int i = level; i < 5; i++) {
            SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 8);  // Thiết lập màu chữ trên console là màu xám
            cout << "level " << i + 1 << endl;
        }
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 15);  // Thiết lập màu chữ trên console là màu trắng
        while (true) {
            if (GetAsyncKeyState(VK_UP) != 0) {
                pointer -= 1;
                if (pointer == -1)
                    pointer = level - 1;
```

```
            break;
        } else if (GetAsyncKeyState(VK_DOWN) != 0) {
            pointer += 1;
            if (pointer == level)
                pointer = 0;
            break;
        } else if (GetAsyncKeyState(VK_RETURN) != 0) {
            return pointer + 1;
        }
    }
    Sleep(120);
  }
}

// Hàm thực hiện quá trình chơi game cho một cấp độ cụ thể
int game_play(int level[][4], int correct, int l) {
  int win = correct, ptr1, ptr2;
  for (; win != 16;) {
    system("cls");  // Xóa màn hình console
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 2);  // Thiết lập màu chữ trên console là màu xanh lá cây
    cout << "Game level " << l;
    cout << endl << endl;
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 15);  // Thiết lập màu chữ trên console là màu trắng
```

```
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            if (level[i][j] == 99) {
                SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 11);
// Thiết lập màu chữ trên console là màu vàng
                cout << setw(3) << "__";  // Hiển thị ô trống
                ptr1 = i;
                ptr2 = j;
                SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 15);
// Thiết lập màu chữ trên console là màu trắng
            } else
                cout << setw(3) << level[i][j];  // Hiển thị số trong ô
        }
        cout << endl;
    }
    cout << "correct positions=" << setw(2) << win << " Incorrct= " << setw(2) <<
16 - win;
    while (true) {
        if (GetAsyncKeyState(VK_UP) != 0) {
            if (ptr1 > 0 && ptr1 <= 3) {
                level[ptr1][ptr2] = level[ptr1 - 1][ptr2];
                level[ptr1 - 1][ptr2] = 99;
                ptr1--;
            }
            break;
```

```
} else if (GetAsyncKeyState(VK_DOWN) != 0) {
    if (ptr1 >= 0 && ptr1 < 3) {
        level[ptr1][ptr2] = level[ptr1 + 1][ptr2];
        level[ptr1 + 1][ptr2] = 99;
        ptr1++;
    }
    break;
} else if (GetAsyncKeyState(VK_LEFT) != 0) {
    if (ptr2 > 0 && ptr2 <= 3) {
        level[ptr1][ptr2] = level[ptr1][ptr2 - 1];
        level[ptr1][ptr2 - 1] = 99;
        ptr2--;
    }
    break;
} else if (GetAsyncKeyState(VK_RIGHT) != 0) {
    if (ptr2 >= 0 && ptr2 < 3) {
        level[ptr1][ptr2] = level[ptr1][ptr2 + 1];
        level[ptr1][ptr2 + 1] = 99;
        ptr2++;
    }
    break;
} else if (GetAsyncKeyState(VK_ESCAPE) != 0) {
    return 1000;
}
```

```
    }
    win = 0;
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            if (level[i][j] == board[i][j])
                win++;
        }
    }
    Sleep(200);
}
system("cls");  // Xóa màn hình console
cout << endl << "P";
Sleep(100);
cout << "u";
Sleep(100);
cout << "z";
Sleep(200);
cout << "z";
Sleep(200);
cout << "l";
Sleep(200);
cout << "e ";
Sleep(200);
cout << "c";
```

```
    Sleep(200);
    cout << "o";
    Sleep(200);
    cout << "m";
    Sleep(200);
    cout << "p";
    Sleep(200);
    cout << "l";
    Sleep(200);
    cout << "e";
    Sleep(200);
    cout << "t";
    Sleep(200);
    cout << "e";
    Sleep(200);
    cout << "d";
    Sleep(200);
    return 0;
}
```

---KẾT THÚC BÀI REPORT---