ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN MÔN HỌC
KỸ THUẬT LẬP TRÌNH

NHÓM 14

| Đề tài |

GAME RẮN SĂN MỒI TRÊN CONSOLE WINDOW

| Giảng viên hướng dẫn |

Trương Toàn Thịnh

| Nhóm thực hiện |

Hoàng Thị Khôn – 21120485

Nguyễn Huy Hoàng – 21120458

Nguyễn Quốc Hưng – 21120464

Nguyễn Văn Hậu – 21120449

Nguyễn Ngọc Như Huyền – 21120475

LỜI MỞ ĐẦU

Ngôn ngữ C/C++ là những ngôn ngữ nền tảng nhất của khoa học máy tính và lập trình. Nếu bạn học tốt ngôn ngữ lập trình C/C++, bạn sẽ thu được nhiều lợi ích, thậm chí nếu sau này bạn không sử dụng chúng nữa. Chúng sẽ mang lại cho bạn cái nhìn sâu sắc vào trong điểm bắt đầu và nguồn gốc của khoa học máy tính và lập trình máy tính. Cả hai ngôn ngữ đều có những ưu điểm và nhược điểm của chúng. Vậy nên, trong học kỳ 1 và học kỳ 2 của ngành Công nghệ thông tin khoa Công nghệ thông tin, trường Đại học Khoa học Tự nhiên – ĐHQG TP. HCM, sinh viên đã được tiếp xúc và tìm hiểu về ngôn ngữ C/C++.

Với việc vận dụng những kiến thức về ngôn C/C++ qua các bài giảng của giảng viên kết hợp với các tài liệu tham khảo về ngôn ngữ lập trình C/C++ năng cao chúng em đã ứng dụng làm game nhỏ là **game rắn săn mồi** trên màn hình console window.

TP. Hồ Chí Minh, ngày... tháng... năm 2022

Nhóm sinh viên thực hiện

Hoàng Thị Khôn – Nguyễn Huy Hoàng

Nguyễn Quốc Hưng – Nguyễn Văn Hậu

Nguyễn Ngọc Như Huyền

MỤC LỤC

GIỚI THIỆU

Giới thiệu trò chơi:

a. Giới thiệu:

Rắn săn mồi là game nhóm phát triển trên ngôn ngữ lập trình C++. Game là những con số xếp liên tiếp nhau đóng vai trò như một con rắn di chuyển trên màn hình console đơn giản, game có lối chơi đơn giản và có sự hấp dẫn đối với người chơi nhờ những tính năng như sound, animation, level up,…

b. Các chức năng chính:

- ✓ Bắt đầu New Game.
- ✓ Save/ Load(Lưu và Tải lại).
- ✓ Bảng cài đặt
- ✓ Bảng hướng dẫn.
- ✓ Thoát game.

Mô tả trò chơi:

- Bắt đầu mỗi level, con rắn sẽ xuất hiện trên ngẫu nhiên trên màn hình console.
- Người chơi có 4 phím di chuyển:

Phím	**Mô tả**
A	Di chuyển sang trái
S	Di chuyển xuống dưới
D	Di chuyển sang phải
W	Di chuyển lên trên

- Có nhiều level khác nhau, qua mỗi level sẽ tăng độ khó và tốc độ rắn.
- Điểm được cộng 1 điểm khi mỗi lần rắn ăn thức ăn
- Mỗi khi rắn ăn sẽ được cộng điểm ngược lại bị đâm vào vật cản hoặc vào thân của mình → game over → lưu lại thông tin của người chơi.
- Khi ăn hết thức ăn ở từng level thì sẽ được chuyển lên level tiếp theo.
    - Tốc độ tăng theo level.
    - Độ dài rắn: khi bắt đầu 1 level mới thì độ dài rắn trở về trạng thái ban đầu.
- Level: Gồm có 4 level. Khi bắt đầu chơi mặc định chơi ở level 1. Sau khi ăn đủ thức ăn quy định của từng level thì người chơi được qua level tiếp theo.

CÁC NHÓM HÀM

Nhóm hàm DATA:

- Là nơi chứa những hằng số, cấu trúc cần dùng xuyên suốt chương trình

```
1  #pragma once
2  #pragma comment(lib, "Winmm.lib") //Dùng để gọi âm thanh trong game
3  #pragma warning(disable: 4996)
4  #define MAX_SIZE_SNAKE 40 //Kích thước tối đa của rắn
5  #define MAX_SIZE_FOOD 4 //Thức ăn tối đa trong 1 màn
6  #define MAX_SIZE_OBSTACLE 5 //Chướng ngại vật tối đa trong 1 level
7  #define MAX_SIZE_GATE 4 //Số cổng tối đa
8  #define MAX_SPEED 2 //Tốc độ tối đa của rắn
9  #include <windows.h> //for HWND
10 #include <iostream>
11 #include <time.h> //for srand(time(NULL));
12 #include <conio.h> //for getch()
13 #include <thread>
14 #include "dirent.h"
15 #include <vector>
16 #include <iomanip>
17 #include <mutex>
18 #include <string>
19 #include <fstream> //for read file
20 #include <Mmsystem.h>
21 using namespace std;
22 //thong tin file in ra phan load
23 typedef struct save_info {
24       char name[16];
25       char timestr[11];
26       int level, score;
27 };
```

Nhóm hàm hiển thị màn hình Console:

- **Hàm cố định cửa sổ Window**

```
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  }
```

- 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. Hàm này để không cho người dùng tự thay đổi kích thước cửa sổ hiện hành.

Nhóm hàm vẽ và đồ họa:

- **Hàm di chuyển trên console**

```
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 }
```

- **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 tới đồ họa, giao diện

```
1 void setTextColor(int color)
2 {
3 SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
  color);
4 }
```

```
1  //change fontsize
2  void fontsize(int a, int b) {
3       HANDLE out = GetStdHandle(STD_OUTPUT_HANDLE);
4       PCONSOLE_FONT_INFOEX lpConsoleCurrentFontEx = new
  CONSOLE_FONT_INFOEX();
5       lpConsoleCurrentFontEx->cbSize =
  sizeof(CONSOLE_FONT_INFOEX);
6       GetCurrentConsoleFontEx(out, 0,
  lpConsoleCurrentFontEx);
7       lpConsoleCurrentFontEx->dwFontSize.X = a;
8       lpConsoleCurrentFontEx->dwFontSize.Y = b;
9       SetCurrentConsoleFontEx(out, 0,
  lpConsoleCurrentFontEx);
10 }
```

- Ngoài ra còn có các hàm phụ trợ khác để vẽ hình ảnh rắn, chữ, intro,...lên màn hình console cho game thêm phần đặc sắc
- **setTextColor():** 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.

Nhóm hàm hiển thị MENU ban đầu:

- Màn hình MENU ban đầu

```
1 // menu stuff
2 string menu_list[5] = { "Play","Load
  game","Setting","Instruction","Exit" };
3 int menu_list_element_count;
```

```
4   int menu_choice;
```

- **Từng chức năng của menu**

```
1   void ProcessSave() {
2       int HEIGH_CONSOLE = 20;
3       char name[20];
4       GotoXY(0, HEIGH_CONSOLE + 2);
5       cout << "Enter name: ";
6       fflush(stdin);
7       cin.getline(name, 20);
8       cout << name;
9       SaveData(name);
10  }
```

- **ProcessSave():** Hàm lưu tên file chơi tối đa 20 ký tự.

```
1   void ProcessLoad() {
2       char k;
3       int Cur_Choice = 0, Cur_element = 0;
4       int num = file.size();
5
6       if (num - 1 < max_file_shown) max_file_shown = num -
    1;
7       GotoXY(0, 0); cout << num << max_file_shown;
8       //in cot
9       setTextColor(15);
10      GotoXY(x_filesave, y_filesave - 2);
11      cout << setw(14) << "NAME" << setw(20) << "LEVEL" <<
    setw(20) << "SCORE" << setw(26) << "DATE";
12      while (true)
13      {
14          setTextColor(15);
15          printlist(Cur_element - Cur_Choice, Cur_element
    - Cur_Choice + max_file_shown);
16          setTextColor(11);
17          GotoXY(x_filesave, y_filesave + Cur_Choice * 2);
18          cout << ">>" << setw(14) <<
    file[Cur_element].name << setw(20) <<
    file[Cur_element].level << setw(20) <<
    file[Cur_element].score << setw(26) <<
    file[Cur_element].timestr;
19          k = toupper(_getch());
20          if (k == 'W') {
21              Cur_element--;
22              if (Cur_Choice > 0) Cur_Choice--;
23          }
```

```
24              if (k == 'S') {
25                  Cur_element++;
26                  if (Cur_Choice < max_file_shown)
Cur_Choice++;
27              }
28              if (k == '\r' || k == 'D') {
29                  strcpy(Name, file[Cur_element].name);
30                  LoadData(Name);
31                  system("cls");
32                  setTextColor(7);
33                  if (FOOD_INDEX == -1)
DrawGate(gate[GATE_INDEX].x, gate[GATE_INDEX].y, 2, 2);
34                  DrawMapLv(LEVEL);
35                  Start();
36                  DrawSnakeAndFood('*');
37                  getch();
38                  return;
39              }
40              if (k == ESC || k == 'A') {
41                  Delete_detail_board();
42                  return;
43              }
44              if (Cur_element < 0) {
45                  Cur_element = num - 1;
46                  Cur_Choice = max_file_shown;
47              }
48              if (Cur_element > num - 1) {
49                  Cur_element = 0;
50                  Cur_Choice = 0;
51              }
52              Sleep(50);
53      }
54      return;
55  }
```

- **ProcessLoad():** Hàm hiển thị file đã lưu và cho người chơi chọn file, load file. Chọn file bằng cách sử dụng phím W và S (tương ứng lên và xuống). Nhấn phím A hoặc ESC để thoát khỏi Load game. Nhấn phím Enter hoặc D để chọn file cần chơi.

```
1   //Bắt đầu game
2   void ProcessStart() {
3       int x_mid_detail_board = x_filesave + 35,
y_mid_detail_board = y_filesave;
4       if (Score == 0) {
5           GotoXY(x_mid_detail_board, y_mid_detail_board);
6           setTextColor(11);
```

```
7               cout << ">> New Game <<";
8               while (1) {
9                       char c = _toupper(getch());
10                      if (c == '\r' || c == 'D') {
11                              NewGame();
12                              //system("cls");
13                              break;
14                      }
15                      else if (c == ESC || c == 'A') {
16                              Delete_detail_board();
17                              return;
18                      }
19                      Sleep(50);
20              }
21      }
22      else {
23              int cur_choice = 0;
24              string choice[2] = { "   New Game   ",
25                                                   "   Continue   " };
26              string choice_[2] = { ">> New Game <<",
27                                                   ">> Continue <<" };
28              while (1)
29              {
30                      GotoXY(x_mid_detail_board,
y_mid_detail_board + 3 * cur_choice);
31                      setTextColor(11);
32                      cout << choice_[cur_choice];
33                      GotoXY(x_mid_detail_board,
y_mid_detail_board + 3 * (1 - cur_choice));
34                      setTextColor(15);
35                      cout << choice[1 - cur_choice];
36                      char c = _toupper(getch());
37                      if (c == 'S') cur_choice++;
38                      else if (c == 'W') cur_choice--;
39                      if (c == '\r' || c == 'D') {
40                              if (cur_choice == 0) {
41                                      NewGame();
42                              }
43                              else {
44                                      system("cls");
45                                      if (FOOD_INDEX == -1)
DrawGate(gate[GATE_INDEX].x, gate[GATE_INDEX].y, 2, 2);
46                                      DrawMapLv(LEVEL);
47                              }
48                              break;
49                      }
50                      else if (c == ESC || c == 'A') {
51                              Delete_detail_board();
```

```
52                          return;
53                    }
54                    if (cur_choice > 1) cur_choice = 0;
55                    else if (cur_choice < 0) cur_choice = 1;
56                    Sleep(50);
57              }
58        }
59        Start();
60  }
```

- **ProcessStart():** Hàm bắt đầu màn chơi. Nhấn phím Enter hoặc D để tiến hành nhập tên file thông qua hàm NewGame() nếu đây là màn chơi mới, ngược lại là chế độ chơi tiếp màn chơi chưa hoàn thành. Nhấn phím A hoặc ESC để thoát khỏi chế độ Start.

```
1   //Cài đặt âm thanh
2   void ProcessSetting() {
3         int x_mid_detail_board = x_filesave + 35,
    y_mid_detail_board = y_filesave;
4         GotoXY(x_mid_detail_board, y_mid_detail_board);
5         cout << "Sound";
6         char c;
7         while (1) {
8               GotoXY(x_mid_detail_board + 5,
    y_mid_detail_board);
9               setTextColor(11);
10              if (sound_ == 1) cout << "  on ";
11              else cout << "  off";
12              c = _toupper(getch());
13              if (c == '\r' || c == 'D') {
14                    sound_ = 1 - sound_;
15              }
16              else if (c == ESC || c == 'A') {
17                    Delete_detail_board();
18                    return;
19              }
20              Sleep(50);
21        }
22  }
```

- **ProcessSetting():** Hàm bật, tắt âm thanh game. Sử dụng phím Enter hoặc D để bật, tắt âm thanh; phím ESC hoặc A để thoát khỏi chế độ Setting.

```
1   //Mở hướng dẫn chơi game
2   void ProcessInstruction() {
3         setTextColor(11);
4         GotoXY(x_filesave, y_filesave - 2);
5         cout << char(30) << "       : W";
6         GotoXY(x_filesave, y_filesave);
7         cout << char(31) << "       : S";
8         GotoXY(x_filesave, y_filesave + 2);
```

```
9       cout << char(16) << "       : A";
10      GotoXY(x_filesave, y_filesave + 4);
11      cout << char(17) << "       : D";
12      GotoXY(x_filesave, y_filesave + 6);
13      cout << "Confirm : D/Enter";
14      GotoXY(x_filesave, y_filesave + 8);
15      cout << "Back    : A/Esc";
16      char c;
17      c = _toupper(getch());
18      if (c == ESC || c == 'A') {
19          Delete_detail_board();
20          return;
21      }
22  }
```

- **ProcessInstruction():** Chế độ hướng dẫn người chơi game bao gồm các phím điều khiển rắn (W, A, S, D) , phím chọn (D/ENTER), phím thoát (A/ESC).

- **Hàm điều khiển chức năng của MENU**

```
1   void Menu()
2   {
3       int HEIGH_CONSOLE = 29, WIDTH_CONSOLE = 118;
4       if (sound_ == 1) PlaySound(TEXT("menu.wav"), NULL,
    SND_ASYNC);
5       int menu_choice = 0;
6       char c;
7       int check = 0;
8       Draw_Board(0, 0, WIDTH_CONSOLE, HEIGH_CONSOLE);
9       Draw_menu_board();
10      setTextColor(15);
11      for (int i = 0; i < 5; i++) {
12          GotoXY(12 - pos_calc(menu_list[i]), y_menu + 3 +
    i * 4);
13          cout << menu_list[i];
14      }
15      //draw detail menu board (right)
16      Draw_Board(x_menu + 18, y_menu, WIDTH_CONSOLE - 26,
    HEIGH_CONSOLE - 7);
17      //divide detail table
18      setTextColor(9);
19      GotoXY(x_filesave - 2, y_filesave + (6) * 2);
20      cout << char(204);
21      GotoXY(x_filesave - 2 + WIDTH_CONSOLE - 26, y_filesave
    + (6) * 2);
22      cout << char(185);
23      GotoXY(x_filesave - 1, y_filesave + (6) * 2);
24      for (int i = 1; i < WIDTH_CONSOLE - 26; i++) cout <<
    char(205);
```

```
25      while (menu_run == 1)
26      {
27          setTextColor(11);
28          GotoXY(12 - pos_calc(menu_list[menu_choice]),
y_menu + 3 + menu_choice * 4);
29          cout << menu_list[menu_choice];
30          setTextColor(15);
31          if (_kbhit() == true)
32          {
33              c = toupper(_getch());
34              if (c == 'W') {
35                  GotoXY(12 -
pos_calc(menu_list[menu_choice]), y_menu + 3 + menu_choice
* 4);
36                  cout << menu_list[menu_choice];
37                  menu_choice--;
38              }
39              if (c == 'S') {
40                  GotoXY(12 -
pos_calc(menu_list[menu_choice]), y_menu + 3 + menu_choice
* 4);
41                  cout << menu_list[menu_choice];
42                  menu_choice++;
43              }
44              if (c == '\r' || c == 'D') {
45                  if (menu_choice == 4)//EXIT
46                      exit(0);
47                  if (menu_choice == 0) {//PLAY
48                      ProcessStart();
49                      if (sound_ == 1) PlaySound(NULL,
NULL, SND_ASYNC);
50                      if (STATE == 1) return;
51                  }
52                  if (menu_choice == 1) {//LOAD GAME
53                      ProcessLoad();
54                      if (STATE == 1) return;
55                  }
56                  if (menu_choice == 2)//setting
57                  {
58                      ProcessSetting();
59                  }
60                  if (menu_choice == 3)//instruction
61                  {
62                      ProcessInstruction();
63                  }
64              }
65          }
66          if (menu_choice < 0)
```

```
67                  menu_choice = 4;
68              if (menu_choice > 4)
69                  menu_choice = 0;
70              Sleep(50);
71          }
72  }
```

- **Menu():** Hàm vẽ giao diện menu và cho phép người chơi chọn các chế độ (Play, Exit, Load game, Setting, Instruction).

```
1   //Lựa chọn sau khi rắn chết
2   void DeadOption() {
3       if (end_game == 0) DeadAnimation();
4       getch();
5       if (sound_ == 1 && end_game == 0)
    PlaySound(TEXT("gameover.wav"), NULL, SND_ASYNC);
6
7       string death_option[3] = { " Play again "," Menu "," 
    Exit " };
8       string death_option_[3] = { "  Play again  ","  Menu 
    ","  Exit  " };
9       int x_over = 15, y_over = 15;
10      int cur_choice = 0;
11      system("cls");
12      if (end_game == 0) draw_gameover();
13      else Draw_endgame();
14      char c;
15      while (1) {
16          setTextColor(14);
17          for (int i = 0; i < 3; i++) {
18              GotoXY(x_over + 10 -
    pos_calc(death_option_[i]), y_over + i * 3 + 4);
19              cout << death_option_[i];
20          }
21          GotoXY(x_over + 10 -
    pos_calc(death_option_[cur_choice]), y_over + cur_choice *
    3 + 4);
22          cout << char(175) << death_option[cur_choice] <<
    char(174);
23          c = toupper(getch());
24          if (c == 'W') {
25              cur_choice--;
26          }
27          else if (c == 'S') {
28              cur_choice++;
29          }
30          else if (c == '\r' || c == 'D') {
31              if (cur_choice == 2) {
32                  ExitGame();
```

```
33                  }
34                  else if (cur_choice == 0) {
35                        system("cls");
36                        ResetData();
37                        Start();
38                        if (sound_ == 1) PlaySound(NULL,
    NULL, SND_ASYNC);
39                  }
40                  else if (cur_choice == 1)
41                  {
42                        system("cls");
43                        back_to_menu = 1;
44                  }
45                  return;
46            }
47            if (cur_choice < 0) cur_choice = 2;
48            if (cur_choice > 2) cur_choice = 0;
49            Sleep(50);
50      }
51      setTextColor(12);
52  }
```

- **DeadOption():** Hàm lựa chọn sau khi rắn chết (Play again, Menu, Exit).

Hàm điều khiển trong game:

```
1   // Thread
2   void ThreadFunc() {
3       while (back_to_menu == 0) {
4             if (STATE == 1)
5             {
6                   ClearSnakeAndFood(' ');
7                   switch (MOVING) {
8                   case 'A':
9                         MoveLeft();
10                        break;
11                  case 'D':
12                        MoveRight();
13                        break;
14                  case 'S':
15                        MoveDown();
16                        break;
17                  case 'W':
18                        MoveUp();
19                        break;
20                  }
21                  DrawSnakeAndFood('*');
22
23                  if (LEVEL == 4) {
```

```
24                          MoveSpider();
25                          Sleep(160 - 15 * LEVEL);
26                          m.lock();
27                          GotoXY(nhen_x + 2, nhen_y - 2);
28                          cout << "     ";
29                          GotoXY(nhen_x, nhen_y - 1);
30                          cout << "         ";
31                          GotoXY(nhen_x + 1, nhen_y);
32                          cout << "       ";
33                          GotoXY(nhen_x + 2, nhen_y + 1);
34                          cout << "     ";
35                          nhen_x++;
36                          m.unlock();
37                          if (nhen_x == 68) {
38                               nhen_x = 10;
39                          }
40                    }
41                    else Sleep(160 - 15 * LEVEL);
42               }
43          }
44     }
```

- **ThreadFunc():** Hàm tạo luồng chuyển động cho rắn và nhện.

```
1    //Khởi chạy rắn
2    void Run() {
3         int temp;
4         thread t(ThreadFunc);
5         HANDLE handle_t = t.native_handle();
6         back_to_menu = 0;
7         while (1) {
8              temp = toupper(_getch());
9              if (STATE == 1) {
10                  if (temp == 'P') {
11                       PauseGame(handle_t);
12                  }
13                  else if (temp == 'M') {
14                       PauseGame(handle_t);
15                       STATE = 0;
16                       system("cls");
17                       back_to_menu = 1;
18                       t.detach();
19                       return;
20                  }
21                  else if (temp == 'L') {
22                       SaveData(Name);
23                  }
24                  else if (temp == 27) {
```

```
25                          ExitGame();
26                          return;
27                    }
28                    else {
29                          ResumeThread(handle_t);
30                          if ((temp != CHAR_LOCK) && (temp == 'D' || temp == 'A' || temp == 'W' || temp == 'S'))
31                          {
32                                if (temp == 'D') CHAR_LOCK = 'A';
33                                else if (temp == 'W') CHAR_LOCK = 'S';
34                                else if (temp == 'S') CHAR_LOCK = 'W';
35                                else CHAR_LOCK = 'D';
36                                MOVING = temp;
37                                Sleep(160 - 15 * LEVEL);
38                          }
39                    }
40              }
41              else {
42                    DeadOption();
43                    if (back_to_menu == 1) {
44                          t.detach();
45                          return;
46                    }
47              }
48        }
49  }
```

- **Run():** Hàm cho phép người chơi ngừng game (nhấn phím P), thoát game trở về menu (nhấn phím M), lưu game (nhấn phím L), thoát game (nhấn phím ESC).

I. Nhóm hàm dùng để điều khiển rắn, vẽ khung, thức ăn, chướng ngại vật, mê cung, cổng qua màn trong game

- **Các biến toàn cục được sử dụng trong Game (ý nghĩa của các biến như comment).**

```
1  //GLOBAL variables
2  POINT snake[40]; //Mảng lưu vị trí từng phần của rắn
3  POINT food[4]; //Mảng lưu vị trí thức ăn
4  POINT obstacle[5]; //Mảng lưu vị trí chướng ngại vật level 1
5  POINT gate[4]; //Mảng lưu vị trí cổng
6  int MSSV[40] = { 2,1,1,2,0,4,4,9,
7  2,1,1,2,0,4,5,8,
```

```
8   2,1,1,2,0,4,6,4,
9   2,1,1,2,0,4,7,5,
10  2,1,1,2,0,4,8,5 }; //Mảng lưu MSSV của các thành viên
11  int CHAR_LOCK; //Xác định hướng mà rắn không thể di chuyển (mỗi một thời điểm sẽ có 1 hướng mà rắn không thể di chuyển)
12  int MOVING; //Xác định hướng di chuyển của rắn (mỗi một thời điểm sẽ có 3 hướng mà rắn có thể di chuyển)
13  int SPEED = 1; //Tốc độ di chuyển của rắn
14  int HEIGH_CONSOLE = 29, WIDTH_CONSOLE = 118;
    //Chiều cao và chiều rộng của màn hình console
15  int FOOD_INDEX; //Chỉ số thức ăn
16  int GATE_INDEX; //Chỉ số cổng
17  int SIZE_SNAKE; //Kích thước của rắn
18  int STATE; //Trạng thái của rắn: sống hay chết
19  int LEVEL; //Màn chơi
20  int Score; //Điểm số
21  int threadrun = 1;
22  //dieu kien de chay menu
23  int menu_run = 1;
24  int ESC = 27;
25  int back_to_menu = 0;
26  char Name[10];
27  //Thông số Spider
28  int nhen_x = 50;
29  int nhen_y = 10;
30  // sound on off
31  int sound_ = 1;
32  int end_game = 0;
```

- **Hàm vẽ khung trò chơi, chướng ngại vật, mê cung, cổng qua màn**

```
1   void DrawBoard(int x, int y, int width, int height, int curPosX, int curPosY) {
2       GotoXY(x, y);
3       setTextColor(9);
4       for (int i = 0; i <= width; i++) cout << char(219);
5       GotoXY(x, height + y);
6       for (int i = 0; i <= width; i++) cout << char(219);
7       for (int i = y + 1; i < height + y; i++) {
8           GotoXY(x, i); cout << char(219);
9           GotoXY(x + width, i); cout << char(219);
10      }
11      setTextColor(7);
12      GotoXY(curPosX, curPosY);
13  }
```

```
1  void DrawObstacle(int x, int y, int width, int height) {
2        GotoXY(x, y);
3        for (int i = x; i < width + x; i++) {
4              for (int j = y; j <= height + y; j++) {
5                    GotoXY(i, j); cout << char(219);
6              }
7        }
8  }
9
10 //Vẽ nhện
11 bool DrawSpider() {
12       bool flag = true;
13       if (flag) {
14             GotoXY(nhen_x + 2, nhen_y - 2);
15             cout << "/ _ \\";
16             GotoXY(nhen_x, nhen_y - 1);
17             cout << "\\_\\(_)/_/";
18             GotoXY(nhen_x + 1, nhen_y);
19             cout << "_//o\\\\_";
20             GotoXY(nhen_x + 2, nhen_y + 1);
21             cout << "/   \\";
22       }
23       return flag;
24 }
```

```
1  //Vẽ cổng
2  void DrawGate(int x, int y, int width, int height) {
3        GotoXY(x, y);
4        for (int i = 0; i <= width; i++)
5              cout << char(219);
6        for (int i = y + 1; i < y + height; i++) {
7              GotoXY(x, i);
8              cout << char(219);
9        }
10       for (int i = y + 1; i < y + height; i++) {
11             GotoXY(x + width, i);
12             cout << char(219);
13       }
14 }
```

```
1  //Vẽ màn chơi
2  void DrawMapLv(int level_index) {
3        switch (level_index) {
4        case 2:
5              //Draw Obstacle
6              for (int i = 0; i < MAX_SIZE_OBSTACLE; i++) {
```

```
7                   DrawObstacle(obstacle[i].x, obstacle[i].y,
5, 7);
8             }
9             break;
10        case 3:
11            // Draw Obstacle
12            for (int i = 4; i < 8; i++) {
13                  GotoXY(40, i);
14                  cout << char(219);
15                  GotoXY(46, i);
16                  cout << char(219);
17            }
18            for (int i = 12; i < 16; i++) {
19                  GotoXY(40, i);
20                  cout << char(219);
21                  GotoXY(46, i);
22                  cout << char(219);
23            }
24            for (int i = 30; i <= 40; i++) {
25                  GotoXY(i, 8);
26                  cout << char(219);
27                  GotoXY(i, 12);
28                  cout << char(219);
29            }
30            for (int i = 46; i <= 56; i++) {
31                  GotoXY(i, 8);
32                  cout << char(219);
33                  GotoXY(i, 12);
34                  cout << char(219);
35            }
36            break;
37        case 4:
38            DrawSpider(); //Vẽ nhện
39            // tường ngang
40            for (int i = 23; i <= 35; i++) {
41                  GotoXY(i, 5);
42                  cout << char(219);
43                  GotoXY(i, 15);
44                  cout << char(219);
45            }
46            for (int i = 51; i <= 63; i++) {
47                  GotoXY(i, 5);
48                  cout << char(219);
49                  GotoXY(i, 15);
50                  cout << char(219);
51            }
52            // 4 cục đá
```

```
53              GotoXY(8 + 7, 0 + 2);
54              cout << char(219);
55              GotoXY(78 - 7, 0 + 2);
56              cout << char(219);
57              GotoXY(8 + 7, 20 - 2);
58              cout << char(219);
59              GotoXY(78 - 7, 20 - 2);
60              cout << char(219);
61          }
62      }
```

- **DrawBoard():** Hàm vẽ các khung trong game.
- **DrawObstacle():** Hàm vẽ chướng ngại vật level 2.
- **DrawSpider():** Hàm vẽ nhện.
- **DrawGate():** Hàm vẽ cổng.
- **DrawMapLv():** Hàm vẽ các màn chơi.

- Các hàm liên quan đến rắn và thức ăn

```
1   //Vẽ rắn và thức ăn
2   void DrawSnakeAndFood(char str) {
3       if (FOOD_INDEX != -1) {
4           GotoXY(food[FOOD_INDEX].x, food[FOOD_INDEX].y);
5           printf("%c", str);
6       }
7       for (int i = 0; i < SIZE_SNAKE; i++) {
8           GotoXY(snake[i].x, snake[i].y);
9           printf("%d", MSSV[SIZE_SNAKE - 1 - i]);
10      }
11  }
```

```
1   //Di chuyển
2   void MoveSpider() {
3       DrawSpider();
4   }
5   void MoveRight() {
6       if (LEVEL == 1) {
7           if (snake[SIZE_SNAKE - 1].x + 1 == WIDTH_CONSOLE
    + 8
8               || SnakeTouchBody(snake[SIZE_SNAKE - 1].x +
    1, snake[SIZE_SNAKE - 1].y) == true
9               || SnakeTouchGate(snake[SIZE_SNAKE - 1].x +
    1, snake[SIZE_SNAKE - 1].y, 2, 2 == true)) {
10              ProcessDead();
11          }
12          else {
13              if (snake[SIZE_SNAKE - 1].x + 1 ==
    food[FOOD_INDEX].x && snake[SIZE_SNAKE - 1].y ==
    food[FOOD_INDEX].y) {
14                  Eat();
```

```
15                  }
16                  for (int i = 0; i < SIZE_SNAKE - 1; i++) {
17                        snake[i].x = snake[i + 1].x;
18                        snake[i].y = snake[i + 1].y;
19                  }
20                  snake[SIZE_SNAKE - 1].x++;
21            }
22      }
23      if (LEVEL == 2) {
24            if (snake[SIZE_SNAKE - 1].x + 1 == WIDTH_CONSOLE
      + 8
25                  || SnakeTouchBody(snake[SIZE_SNAKE - 1].x +
      1, snake[SIZE_SNAKE - 1].y) == true
26                  || SnakeTouchObstacle(snake[SIZE_SNAKE -
      1].x + 1, snake[SIZE_SNAKE - 1].y, 5, 7) == true
27                  || SnakeTouchGate(snake[SIZE_SNAKE - 1].x +
      1, snake[SIZE_SNAKE - 1].y, 2, 2) == true) {
28                  ProcessDead();
29            }
30            else {
31                  if (snake[SIZE_SNAKE - 1].x + 1 ==
      food[FOOD_INDEX].x && snake[SIZE_SNAKE - 1].y ==
      food[FOOD_INDEX].y) {
32                        Eat();
33                  }
34                  for (int i = 0; i < SIZE_SNAKE - 1; i++) {
35                        snake[i].x = snake[i + 1].x;
36                        snake[i].y = snake[i + 1].y;
37                  }
38                  snake[SIZE_SNAKE - 1].x++;
39            }
40      }
41      if (LEVEL == 3) {
42            if (snake[SIZE_SNAKE - 1].x + 1 == WIDTH_CONSOLE
      + 8
43                  || SnakeTouchBody(snake[SIZE_SNAKE - 1].x +
      1, snake[SIZE_SNAKE - 1].y) == true
44                  || SnakeTouch_Lv3(snake[SIZE_SNAKE - 1].x,
      snake[SIZE_SNAKE - 1].y) == true
45                  || SnakeTouchGate(snake[SIZE_SNAKE - 1].x +
      1, snake[SIZE_SNAKE - 1].y, 2, 2) == true) {
46                  ProcessDead();
47            }
48            else {
49                  if (snake[SIZE_SNAKE - 1].x + 1 ==
      food[FOOD_INDEX].x && snake[SIZE_SNAKE - 1].y ==
      food[FOOD_INDEX].y) {
50                        Eat();
```

```
51                  }
52                  for (int i = 0; i < SIZE_SNAKE - 1; i++) {
53                        snake[i].x = snake[i + 1].x;
54                        snake[i].y = snake[i + 1].y;
55                  }
56                  snake[SIZE_SNAKE - 1].x++;
57            }
58      }
59      if (LEVEL == 4) {
60            if (snake[SIZE_SNAKE - 1].x + 1 == WIDTH_CONSOLE
      + 8
61                  || SnakeTouchBody(snake[SIZE_SNAKE - 1].x +
      1, snake[SIZE_SNAKE - 1].y) == true
62                  || SnakeTouchSpider(snake[SIZE_SNAKE - 1].x
      + 1, snake[SIZE_SNAKE - 1].y) == true
63                  || SnakeTouch_Lv4(snake[SIZE_SNAKE - 1].x,
      snake[SIZE_SNAKE - 1].y) == true
64                  || SnakeTouchGate(snake[SIZE_SNAKE - 1].x +
      1, snake[SIZE_SNAKE - 1].y, 2, 2) == true) {
65                  ProcessDead();
66            }
67            else {
68                  if (snake[SIZE_SNAKE - 1].x + 1 ==
      food[FOOD_INDEX].x && snake[SIZE_SNAKE - 1].y ==
      food[FOOD_INDEX].y) {
69                        Eat();
70                  }
71                  for (int i = 0; i < SIZE_SNAKE - 1; i++) {
72                        snake[i].x = snake[i + 1].x;
73                        snake[i].y = snake[i + 1].y;
74                  }
75                  snake[SIZE_SNAKE - 1].x++;
76            }
77      }
78
79  }
80  void MoveLeft() {
81      if (LEVEL == 1) {
82            if (snake[SIZE_SNAKE - 1].x - 1 == 8
83                  || SnakeTouchBody(snake[SIZE_SNAKE - 1].x -
      1, snake[SIZE_SNAKE - 1].y) == true
84                  || SnakeTouchGate(snake[SIZE_SNAKE - 1].x -
      1, snake[SIZE_SNAKE - 1].y, 2, 2) == true) {
85                  ProcessDead();
86            }
87            else {
```

```
88                  if (snake[SIZE_SNAKE - 1].x - 1 == food[FOOD_INDEX].x && snake[SIZE_SNAKE - 1].y == food[FOOD_INDEX].y) {
89                      Eat();
90                  }
91                  for (int i = 0; i < SIZE_SNAKE - 1; i++) {
92                      snake[i].x = snake[i + 1].x;
93                      snake[i].y = snake[i + 1].y;
94                  }
95                  snake[SIZE_SNAKE - 1].x--;
96              }
97          }
98          if (LEVEL == 2) {
99              if (snake[SIZE_SNAKE - 1].x - 1 == 8
100                 || SnakeTouchBody(snake[SIZE_SNAKE - 1].x - 1, snake[SIZE_SNAKE - 1].y) == true
101                 || SnakeTouchObstacle(snake[SIZE_SNAKE - 1].x - 1, snake[SIZE_SNAKE - 1].y, 5, 7) == true
102                 || SnakeTouchGate(snake[SIZE_SNAKE - 1].x - 1, snake[SIZE_SNAKE - 1].y, 2, 2) == true) {
103                 ProcessDead();
104             }
105             else {
106                 if (snake[SIZE_SNAKE - 1].x - 1 == food[FOOD_INDEX].x && snake[SIZE_SNAKE - 1].y == food[FOOD_INDEX].y) {
107                     Eat();
108                 }
109                 for (int i = 0; i < SIZE_SNAKE - 1; i++) {
110                     snake[i].x = snake[i + 1].x;
111                     snake[i].y = snake[i + 1].y;
112                 }
113                 snake[SIZE_SNAKE - 1].x--;
114             }
115         }
116         if (LEVEL == 3) {
117             if (snake[SIZE_SNAKE - 1].x - 1 == 8
118                 || SnakeTouchBody(snake[SIZE_SNAKE - 1].x - 1, snake[SIZE_SNAKE - 1].y) == true
119                 || SnakeTouch_Lv3(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y) == true
120                 || SnakeTouchGate(snake[SIZE_SNAKE - 1].x - 1, snake[SIZE_SNAKE - 1].y, 2, 2) == true) {
121                 ProcessDead();
122             }
123             else {
```

```
124                 if (snake[SIZE_SNAKE - 1].x - 1 == food[FOOD_INDEX].x && snake[SIZE_SNAKE - 1].y == food[FOOD_INDEX].y) {
125                     Eat();
126                 }
127                 for (int i = 0; i < SIZE_SNAKE - 1; i++) {
128                     snake[i].x = snake[i + 1].x;
129                     snake[i].y = snake[i + 1].y;
130                 }
131                 snake[SIZE_SNAKE - 1].x--;
132             }
133         }
134         if (LEVEL == 4) {
135             if (snake[SIZE_SNAKE - 1].x - 1 == 8
136                 || SnakeTouchBody(snake[SIZE_SNAKE - 1].x - 1, snake[SIZE_SNAKE - 1].y) == true
137                 || SnakeTouchSpider(snake[SIZE_SNAKE - 1].x - 1, snake[SIZE_SNAKE - 1].y) == true
138                 || SnakeTouch_Lv4(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y) == true
139                 || SnakeTouchGate(snake[SIZE_SNAKE - 1].x - 1, snake[SIZE_SNAKE - 1].y, 2, 2) == true) {
140                 ProcessDead();
141             }
142             else {
143                 if (snake[SIZE_SNAKE - 1].x - 1 == food[FOOD_INDEX].x && snake[SIZE_SNAKE - 1].y == food[FOOD_INDEX].y) {
144                     Eat();
145                 }
146                 for (int i = 0; i < SIZE_SNAKE - 1; i++) {
147                     snake[i].x = snake[i + 1].x;
148                     snake[i].y = snake[i + 1].y;
149                 }
150                 snake[SIZE_SNAKE - 1].x--;
151             }
152         }
153 }
154 void MoveDown() {
155         if (LEVEL == 1) {
156             if (snake[SIZE_SNAKE - 1].y + 1 == HEIGH_CONSOLE
157                 || SnakeTouchBody(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y + 1) == true
158                 || SnakeTouchGate(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y + 1, 2, 2) == true) {
159                 ProcessDead();
160             }
161             else {
```

```
162                 if (snake[SIZE_SNAKE - 1].x == food[FOOD_INDEX].x && snake[SIZE_SNAKE - 1].y + 1 == food[FOOD_INDEX].y) {
163                     Eat();
164                 }
165                 for (int i = 0; i < SIZE_SNAKE - 1; i++) {
166                     snake[i].x = snake[i + 1].x;
167                     snake[i].y = snake[i + 1].y;
168                 }
169                 snake[SIZE_SNAKE - 1].y++;
170             }
171         }
172         if (LEVEL == 2) {
173             if (snake[SIZE_SNAKE - 1].y + 1 == HEIGH_CONSOLE
174                 || SnakeTouchBody(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y + 1) == true
175                 || SnakeTouchObstacle(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y + 1, 5, 7) == true
176                 || SnakeTouchGate(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y + 1, 2, 2) == true) {
177                 ProcessDead();
178             }
179             else {
180                 if (snake[SIZE_SNAKE - 1].x == food[FOOD_INDEX].x && snake[SIZE_SNAKE - 1].y + 1 == food[FOOD_INDEX].y) {
181                     Eat();
182                 }
183                 for (int i = 0; i < SIZE_SNAKE - 1; i++) {
184                     snake[i].x = snake[i + 1].x;
185                     snake[i].y = snake[i + 1].y;
186                 }
187                 snake[SIZE_SNAKE - 1].y++;
188             }
189         }
190         if (LEVEL == 3) {
191             if (snake[SIZE_SNAKE - 1].y + 1 == HEIGH_CONSOLE
192                 || SnakeTouchBody(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y + 1) == true
193                 || SnakeTouch_Lv3(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y) == true
194                 || SnakeTouchGate(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y + 1, 2, 2) == true) {
195                 ProcessDead();
196             }
197             else {
```

```
198                 if    (snake[SIZE_SNAKE    -    1].x    ==
food[FOOD_INDEX].x  &&  snake[SIZE_SNAKE  -  1].y  +  1  ==
food[FOOD_INDEX].y) {
199                       Eat();
200                 }
201                 for (int i = 0; i < SIZE_SNAKE - 1; i++) {
202                       snake[i].x = snake[i + 1].x;
203                       snake[i].y = snake[i + 1].y;
204                 }
205                 snake[SIZE_SNAKE - 1].y++;
206           }
207     }
208     if (LEVEL == 4) {
209           if (snake[SIZE_SNAKE - 1].y + 1 == HEIGH_CONSOLE
210                 || SnakeTouchBody(snake[SIZE_SNAKE  -  1].x,
snake[SIZE_SNAKE - 1].y + 1) == true
211                 || SnakeTouchSpider(snake[SIZE_SNAKE - 1].x,
snake[SIZE_SNAKE - 1].y) == true
212                 || SnakeTouch_Lv4(snake[SIZE_SNAKE  -  1].x,
snake[SIZE_SNAKE - 1].y) == true
213                 || SnakeTouchGate(snake[SIZE_SNAKE  -  1].x,
snake[SIZE_SNAKE - 1].y + 1, 2, 2) == true) {
214                 ProcessDead();
215           }
216           else {
217                 if    (snake[SIZE_SNAKE    -    1].x    ==
food[FOOD_INDEX].x  &&  snake[SIZE_SNAKE  -  1].y  +  1  ==
food[FOOD_INDEX].y) {
218                       Eat();
219                 }
220                 for (int i = 0; i < SIZE_SNAKE - 1; i++) {
221                       snake[i].x = snake[i + 1].x;
222                       snake[i].y = snake[i + 1].y;
223                 }
224                 snake[SIZE_SNAKE - 1].y++;
225           }
226     }
227 }
228 void MoveUp() {
229     if (LEVEL == 1) {
230           if (snake[SIZE_SNAKE - 1].y - 1 == 0
231                 || SnakeTouchBody(snake[SIZE_SNAKE  -  1].x,
snake[SIZE_SNAKE - 1].y - 1) == true
232                 || SnakeTouchGate(snake[SIZE_SNAKE  -  1].x,
snake[SIZE_SNAKE - 1].y, 2, 2) == true) {
233                 ProcessDead();
234           }
235           //Điều kiện rắn vào cổng
```

```
236             else if (snake[SIZE_SNAKE - 1].x == gate[GATE_INDEX].x + 1 && snake[SIZE_SNAKE - 1].y == gate[GATE_INDEX].y + 1 && FOOD_INDEX == -1) {
237                 GotoXY(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y);
238                 printf("%c", ' ');
239                 SIZE_SNAKE--;
240                 if (SIZE_SNAKE == 0) {
241                     //Level up
242                     SIZE_SNAKE = 10;
243                     LevelUp(LEVEL);
244                 }
245             }
246             else {
247                 if (snake[SIZE_SNAKE - 1].x == food[FOOD_INDEX].x && snake[SIZE_SNAKE - 1].y - 1 == food[FOOD_INDEX].y) {
248                     Eat();
249                 }
250                 for (int i = 0; i < SIZE_SNAKE - 1; i++) {
251                     snake[i].x = snake[i + 1].x;
252                     snake[i].y = snake[i + 1].y;
253                 }
254                 snake[SIZE_SNAKE - 1].y--;
255             }
256         }
257         else if (LEVEL == 2) {
258             if (snake[SIZE_SNAKE - 1].y - 1 == 0
259                 || SnakeTouchBody(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y - 1) == true
260                 || SnakeTouchObstacle(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y - 1, 5, 7) == true
261                 || SnakeTouchGate(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y, 2, 2) == true) {
262                 ProcessDead();
263             }
264             else if (snake[SIZE_SNAKE - 1].x == gate[GATE_INDEX].x + 1 && snake[SIZE_SNAKE - 1].y == gate[GATE_INDEX].y + 1 && FOOD_INDEX == -1) {
265                 GotoXY(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y);
266                 printf("%c", ' ');
267                 SIZE_SNAKE--;
268                 if (SIZE_SNAKE == 0) {
269                     //Level up
270                     SIZE_SNAKE = 14;
271                     LevelUp(LEVEL);
272                 }
```

```
273             }
274             else {
275                 if (snake[SIZE_SNAKE - 1].x == food[FOOD_INDEX].x && snake[SIZE_SNAKE - 1].y - 1 == food[FOOD_INDEX].y) {
276                     Eat();
277                 }
278                 for (int i = 0; i < SIZE_SNAKE - 1; i++) {
279                     snake[i].x = snake[i + 1].x;
280                     snake[i].y = snake[i + 1].y;
281                 }
282                 snake[SIZE_SNAKE - 1].y--;
283             }
284         }
285         else if (LEVEL == 3) {
286             if (snake[SIZE_SNAKE - 1].y - 1 == 0
287                 || SnakeTouchBody(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y - 1) == true
288                 || SnakeTouch_Lv3(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y) == true
289                 || SnakeTouchGate(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y, 2, 2) == true) {
290                 ProcessDead();
291             }
292             else if (snake[SIZE_SNAKE - 1].x == gate[GATE_INDEX].x + 1 && snake[SIZE_SNAKE - 1].y == gate[GATE_INDEX].y + 1 && FOOD_INDEX == -1) {
293                 GotoXY(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y);
294                 printf("%c", ' ');
295                 SIZE_SNAKE--;
296                 if (SIZE_SNAKE == 0) {
297                     //Level up
298                     SIZE_SNAKE = 18;
299                     LevelUp(LEVEL);
300                 }
301             }
302             else {
303                 if (snake[SIZE_SNAKE - 1].x == food[FOOD_INDEX].x && snake[SIZE_SNAKE - 1].y - 1 == food[FOOD_INDEX].y) {
304                     Eat();
305                 }
306                 for (int i = 0; i < SIZE_SNAKE - 1; i++) {
307                     snake[i].x = snake[i + 1].x;
308                     snake[i].y = snake[i + 1].y;
309                 }
310                 snake[SIZE_SNAKE - 1].y--;
```

```
311             }
312         }
313         else if (LEVEL == 4) {
314             if (snake[SIZE_SNAKE - 1].y - 1 == 0
315                 || SnakeTouchBody(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y - 1) == true
316                 || SnakeTouchSpider(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y) == true
317                 || SnakeTouch_Lv4(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y) == true
318                 || SnakeTouchGate(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y, 2, 2) == true) {
319                 ProcessDead();
320             }
321             else if (snake[SIZE_SNAKE - 1].x == gate[GATE_INDEX].x + 1 && snake[SIZE_SNAKE - 1].y == gate[GATE_INDEX].y + 1 && FOOD_INDEX == -1) {
322                 GotoXY(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y);
323                 printf("%c", ' ');
324                 SIZE_SNAKE--;
325                 if (SIZE_SNAKE == 0) {
326                     STATE = 0;
327                     end_game = 1;
328                 }
329             }
330             else {
331                 if (snake[SIZE_SNAKE - 1].x == food[FOOD_INDEX].x && snake[SIZE_SNAKE - 1].y - 1 == food[FOOD_INDEX].y) {
332                     Eat();
333                 }
334                 for (int i = 0; i < SIZE_SNAKE - 1; i++) {
335                     snake[i].x = snake[i + 1].x;
336                     snake[i].y = snake[i + 1].y;
337                 }
338                 snake[SIZE_SNAKE - 1].y--;
339             }
340         }
341 }
```

- **Move():** Hàm di chuyển rắn trong đó có các hàm kiểm tra rắn chạm tường, chướng ngại vật, cổng, tự chạm thân, nếu đúng rắn sẽ chết. Nếu rắn ăn thức ăn thì gọi hàm Eat() và thêm vị trí cho kích thước mới của rắn. Khi cổng xuất hiện và rắn chui vào khe giữa cổng thì xóa và thu dần kích thước rắn cho đến khi rắn mất.

```
1 //Rắn ăn thức ăn
2 void Eat() {
```

```
3       if (sound_ == 1) PlaySound(TEXT("eat.wav"), NULL, SND_ASYNC);
4       GotoXY(93, HEIGH_CONSOLE - 5);
5       Score++; //Tăng điểm lên 1
6       cout << Score;
7       snake[SIZE_SNAKE] = food[FOOD_INDEX];
8       if (FOOD_INDEX == MAX_SIZE_FOOD - 1)
9       {
10            FOOD_INDEX = -1;
11            //Tạo cổng
12            GenerateGate(2, 2);
13            DrawGate(gate[GATE_INDEX].x, gate[GATE_INDEX].y, 2, 2);
14      }
15      else {
16            FOOD_INDEX++; //Chỉ số thức ăn tăng lên 1
17            GenerateFood(LEVEL); //Khởi tạo vị trí thức ăn mới
18            SIZE_SNAKE++; //Rắn dài thêm
19      }
20  }
```

- **Eat():** Hàm xử lý dữ liệu khi rắn ăn thức ăn. Sau khi ăn điểm, chỉ số thức ăn, kích thước rắn sẽ tăng lên 1; vị trí rắn trùng với thức ăn; tạo thức ăn mới. Khi ăn đủ 4 thức ăn sẽ tạo cổng để qua màn mới.

- Xử lý va chạm chướng ngại vật, xử lý chết,...

```
1   /*Rắn chạm*/
2   //Rắn chạm thân
3   bool SnakeTouchBody(int x, int y)
4   {
5       for (int i = 1; i < SIZE_SNAKE; i++)
6       {
7           if (x == snake[i].x && y == snake[i].y)
8           {
9               return true;
10          }
11      }
12      return false;
13  }
14
15  //Rắn chạm chướng ngại vật level 2
16  bool SnakeTouchObstacle(int x, int y, int width, int height)
17  {
18      for (int i = 0; i < MAX_SIZE_OBSTACLE; i++) {
19          if (x >= obstacle[i].x && x < obstacle[i].x + width
```

```
20                  && y >= obstacle[i].y && y <= obstacle[i].y + height)
21                  return true;
22          }
23          return false;
24  }
25
26  //Rắn chạm chướng ngại vật level 3
27  bool SnakeTouch_Lv3(int x, int y) {
28          bool flag2 = false;
29          if (
30                  // cham thanh doc tren
31                  (x == 40 && y >= 4 && y <= 8)
32                  || (x == 46 && y >= 4 && y <= 8)
33                  // cham thanh doc duoi
34                  || (x == 40 && y >= 12 && y < 16)
35                  || (x == 46 && y >= 12 && y < 16)
36                  // cham thanh ngang trai
37                  || (x >= 30 && x <= 40 && y == 8)
38                  || (x >= 30 && x <= 40 && y == 12)
39                  // cham thanh ngang phai
40                  || (x >= 46 && x <= 56 && y == 8)
41                  || (x >= 46 && x <= 56 && y == 12)
42                  ) {
43                  flag2 = true;
44          }
45          return flag2;
46  }
47
48  //Rắn chạm chướng ngại vật level 4
49  bool SnakeTouch_Lv4(int x, int y) {
50          bool flag3 = false;
51          if ((x >= 23 && x <= 35 && y == 5)
52                  || (x >= 51 && x <= 63 && y == 15)
53                  || (x == 15 && y == 2) || (x == 71 && y == 2)
54                  || (x == 15 && y == 18) || (x == 71 && y == 18))
    {
55                  flag3 = true;
56          }
57          return flag3;
58  }
59
60  //Rắn chạm cổng
61  bool SnakeTouchGate(int x, int y, int width, int height) {
62          if (FOOD_INDEX < MAX_SIZE_FOOD - 1 && FOOD_INDEX != -
    1) return false;
63          if (x >= gate[GATE_INDEX].x && x <= gate[GATE_INDEX].x
    && y == gate[GATE_INDEX].y)
```

```
64      {
65          return true;
66      }
67      if ((x == gate[GATE_INDEX].x || x == gate[GATE_INDEX].x + width)
68          && (y >= gate[GATE_INDEX].y && y < gate[GATE_INDEX].y + height))
69      {
70          return true;
71      }
72      return false;
73  }
74
75  //Rắn chạm nhện
76  bool SnakeTouchSpider(int x, int y) {
77      bool flag = false;
78      if ((x == nhen_x && y == nhen_y - 2) || (x == nhen_x && y == nhen_y)
79          || (x == nhen_x && y == nhen_y + 1) || (x == nhen_x + 1 && y == nhen_y + 1) || (x == nhen_x + 2 && y == nhen_y + 1)
80          || (x == nhen_x + 2 && y == nhen_y) || (x == nhen_x + 1 && y == nhen_y - 2)
81          || (x == nhen_x + 2 && y == nhen_y - 2) || (x == nhen_x + 1 && y == nhen_y - 1)) {
82          flag = true;
83      }
84      return flag;
85  }
```

- Các hàm điều kiện rắn chết được thiết kế dựa vào việc xác định vị trí đầu rắn không trùng với vị trí vẽ chướng ngại vật.

```
1  //Rắn chết
2  void ProcessDead() {
3      STATE = 0;
4      if (sound_ == 1) PlaySound(TEXT("dead.wav"), NULL, SND_ASYNC);
5      DrawGameOver();
6  }
```

- **ProcessDead():** Hàm xác định rắn chết, khởi tạo trạng thái rắn về 0 và vẽ menu sau khi rắn chết để người chơi chọn chơi tiếp hoặc về lại menu chính hoặc thoát game.

Nhóm hàm liên quan đến dữ liệu game

- **Hàm dùng để khởi tạo dữ liệu ban đầu cho game**

```
1  //Chức năng
```

```
2  void ResetData() {
3  CHAR_LOCK = 'A', MOVING = 'D', SPEED = 1; FOOD_INDEX = 0,
   GATE_INDEX = 0,
4  WIDTH_CONSOLE = 70, HEIGH_CONSOLE = 20, SIZE_SNAKE = 6;
   Score = 0, LEVEL = 1, nhen_x = 50;
5  snake[0] = { 10, 10 }; snake[1] = { 11, 10 };
6  snake[2] = { 12, 10 }; snake[3] = { 13, 10 };
7  ,
8  obstacle[0] = { 18, 1 }; obstacle[1] = { 41, 1 };
9  obstacle[2] = { 64, 1 }; obstacle[3] = { 29, HEIGH_CONSOLE
   - 8 }; obstacle[4] = { 53, HEIGH_CONSOLE - 8 };
10 DrawMapLv(LEVEL);
11 GenerateFood(LEVEL);
12 }
```

- **ResetData():** Hàm khởi tạo giá trị đầu cho các biến ở dòng 3 và 4, tọa độ rắn, tọa độ chướng ngại vật level 2 và vẽ màn chơi, thức ăn đầu tiên.

- Hàm dùng để lưu Game:

```
1  void SaveData(string name) {
2          ofstream Data;
3          name = "save/" + name;
4          Data.open(name);
5          Data << LEVEL << " " << Score << " " << MOVING << " " <<
   SIZE_SNAKE << " " << FOOD_INDEX << " " << SPEED << " ";
6          for (int i = 0; i < SIZE_SNAKE; i++) Data << snake[i] << " ";
7          Data << food[FOOD_INDEX] << " ";
8          if (FOOD_INDEX == -1) Data << gate[GATE_INDEX];
9          Data.close();
10 }
```

- SaveData(): Hàm lưu các dữ liệu khi chơi game (màn chơi, điểm số, hướng rắn đang di chuyển, kích thước rắn, chỉ số thức ăn, tốc độ rắn, vị trí cổng).

- Hàm dùng để Load Game từ một tệp tin đã lưu từ trước:

```
1  void LoadData(string name) {
```

```
2       ifstream Data;
3       name = "save/" + name;
4       Data.open(name);
5       Data >> LEVEL >> Score >> MOVING >> SIZE_SNAKE >>
FOOD_INDEX >> SPEED;
6       for (int i = 0; i < SIZE_SNAKE; i++) Data >>
snake[i];
7       Data >> food[FOOD_INDEX];
8       if (FOOD_INDEX == -1) Data >> gate[GATE_INDEX];
9       Data.close();
10  }
```

- **LoadData():** Hàm load file đã lưu trước đó bằng cách đọc file.

```
1   //nhap sanh sach file luu
2   void listFiles(const char* dirname) {
3       DIR* dir = opendir(dirname);
4       if (dir == NULL) {
5           return;
6       }
7       struct dirent* entity;
8       entity = readdir(dir);
9       entity = readdir(dir);
10      entity = readdir(dir);
11      while (entity != NULL) {
12          strcpy(tempfile.name, entity->d_name);
13          readinfo();
14          file.push_back(tempfile);
15          entity = readdir(dir);
16      }
17      closedir(dir);
18  }
19
20  //in danh sach file saved
21  void printlist(int begin, int end) {
22      for (int i = begin; i <= end; i++) {
23          GotoXY(x_filesave, y_filesave + (i - begin) *
2);
24          cout << setw(14) << file[i].name << setw(20) <<
file[i].level << setw(20) << file[i].score << setw(26) <<
file[i].timestr << "  ";
25      }
26  }
27
28  void Delete_detail_board() {
```

```
29      string blank = "
";
30      for (int i = 0; i < 15; i++) {
31          GotoXY(x_filesave, y_filesave - 4 + i);
32          cout << blank;
33      }
34  }
```

- **listFiles():** Hàm sử dụng thư viện dirent.h đọc hai thông tin đầu tiên của từng file load và ngày lưu file.
- **printlist():** Hàm in danh sách file đã lưu.
- **Delete_detail_board():** Hàm quay về màn hình chọn menu.

II. Các hàm phụ trợ khác (PauseGame, ExitGame,...):

```
1   // Thoát game
2   void ExitGame() {
3       system("cls");
4       _endthreadex(0);
5   }
6   // Tạm dừng khi đang chơi
7   void PauseGame(HANDLE t) {
8       SuspendThread(t);
9   }
```

- **ExitGame():** Xóa màn hình và kết thúc luồng thread chính.
- **PauseGame():** Dừng màn hình chơi và ngưng thread.

III. Hàm main()

```
1   void main() {
2       setTextColor(10);
3       ShowCur(0);
4       Draw_Newgame_intro();
5       FixconsoleWindow();
6       listFiles("save");
7       ResetData();
8       while (1) {
9           Menu();
10          Run();
11      }
12  }
```

- Đây là nơi viết nên tiến trình chính của Game. Bắt đầu cố định màn hình Console, kích thước màn hình, sau đó gọi hàm ResetData(), sau đó là gọi hàm Menu() để người chơi chọn các chế độ chơi, tùy chỉnh Game và cuối cùng là gọi hàm Run() để bắt đầu một tiến trình Game.

LỜI KẾT

Chúng em xin gửi lời cảm ơn chân thành nhất tới giảng viên hướng dẫn - thầy Trương Toàn Thịnh đã có những hướng dẫn cho nhóm chúng em, tạo điều kiện cho chúng em nghiên cứu thêm về ngôn ngữ C/C++ sâu hơn trong học kỳ này và những lưu ý về nội dung và báo cáo để nhóm chúng em hoàn thành tốt nhất.

Do khả năng còn hạn chế về kiến thức chuyên môn nên trong quá trình làm game có thể không tranh khỏi những thiếu sót. Chúng em rất mong nhận được sự góp ý, đánh giá của thầy để đề tài của chúng em được hoàn thiện hơn.

Tài hiệu là tâm huyết của nhóm sinh viên thực hiện. Nếu trích dẫn vui lòng kèm theo nguồn!

- Nguồn tài liệu tham khảo:

- ✓ Stack Overflow.
- ✓ Hướng dẫn chạy âm thanh trong C++: https://www.stdio.vn/article/chay
- ✓ file-wav-voi-windows-h-Y4lmL
- ✓ Tài liệu về hướng dẫn đồ án của thầy Trương Toàn Thịnh