

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

-00-



BÁO CÁO ĐỒ ÁN
TRÒ CHƠI RĂN SĂN MỒI
(SNAKE GAME)
MÔN HỌC: KỸ THUẬT LẬP TRÌNH

Thành viên:

NGUYỄN TRUNG NGUYỄN – 20127404
LÊ ĐẶNG MINH KHÔI – 20127213
ĐOÀN ÁNH DƯƠNG – 20127474
NGUYỄN PHƯỢNG KHANH – 20127204
PHẠM HUY CƯỜNG THỊNH – 20127335

Giảng viên:

TRƯỜNG TOÀN THỊNH

MỤC LỤC

1. TỔNG QUAN	4
1.1. Mô tả đồ án	4
1.2. Lời cảm ơn	4
1.3. Đánh giá mức độ hoàn thành	4
2. LUU ĐỒ (FLOW CHART)	5
2.1. Lưu đồ vận hành menu chính.....	5
2.2. Lưu đồ vận hành game.....	6
2.3. Lưu đồ vận hành save/load game.....	7
3. CÁC HÀM CHÍNH TRONG GAME	8
3.1. KHÓI HÀM MENU	8
3.1.1. void snakeBoard().....	8
3.1.2. void PrintMainMenu()	9
3.1.3. void SaveMenu()	9
3.1.4. void MainMenu()	10
3.2. KHÓI HÀM QUẢN LÝ CONSOLE VÀ OUTPUT	11
3.2.1. void DrawBoard(int x, int y, int width, int height, int color)	11
3.2.2. void statusBoard().....	11
3.2.3. void foodOut(bool type)	12
3.2.4. void snakeOut(bool type).....	12
3.2.5. void gateOut()	13
3.2.6. void wallOut()	13
3.2.7. void ScrUpdate().....	15
3.2.8. void BoardRfsh()	15
3.2.9. void deadAnimate()	15

3.2.10. void printRound (int LEVEL_STATE)	16
3.3. KHỐI HÀM CƠ CHẾ GAME	16
3.3.1. void gateGeneration()	16
3.3.2. void deleteGate().....	17
3.3.3. void wallGeneration()	17
3.3.4. void ProgressDead()	19
3.3.5. void setup()	19
3.3.6. void Update()	20
3.3.7. void lvUP()	20
3.3.8. bool samePoint (POINT a, POINT b).....	21
3.3.9. bool pointStatus(int x, int y)	Error! Bookmark not defined.
3.3.10. bool pointValid(POINT target[], int nTarget).....	21
3.4. KHỐI HÀM SAVE/LOAD GAME	21
3.4.1. string* loadRawDATA(int& n)	22
3.4.2. void LoadGame (string a)	22
3.4.3. void SaveGame(string a).....	23
3.4.4. void YesNoScreen().....	23
3.4.5. void YesNoDisable()	23
3.4.6. void datList()	24
3.4.7. void Control 2(string a).....	24
3.4.8. void Control3(string a).....	25
3.4.9. void Continue()	25
3.5. HÀM VÀ CÁC BIẾN TOÀN CỤC QUẢN LÝ TOÀN BỘ GAME	26
3.5.1. void game()	26
3.5.2. Global Variables	28
4. TÀI LIỆU THAM KHẢO.....	29

1. TỔNG QUAN

1.1. Mô tả đồ án

Rắn săn mồi (Snake) gắn liền với tên tuổi huyền thoại Nokia từng là kỉ niệm tuổi thơ của không biết bao nhiêu thế hệ 9x, 2k thời kỳ đầu. Chỉ với một chiếc màn hình điện thoại đen trắng và những phím bấm nhỏ đặc trưng của dòng điện thoại cục gạch đã khiến bao người trong chúng ta mê mẩn. Với mong muốn tái hiện lại tuổi thơ và ứng dụng những kiến thức được học vào trong thực tế nhóm chúng em xin thiết kế lại trò chơi con rắn bằng ngôn ngữ C++.

1.2. Lời cảm ơn

Nhóm chúng em xin gửi lời cảm ơn chân thành đến thầy Trương Toàn Thịnh, giảng viên lý thuyết và những người bạn đã tận tình hướng dẫn, giúp đỡ và góp ý để chúng em có thể hoàn thành đồ án này.

1.3. Đánh giá mức độ hoàn thành

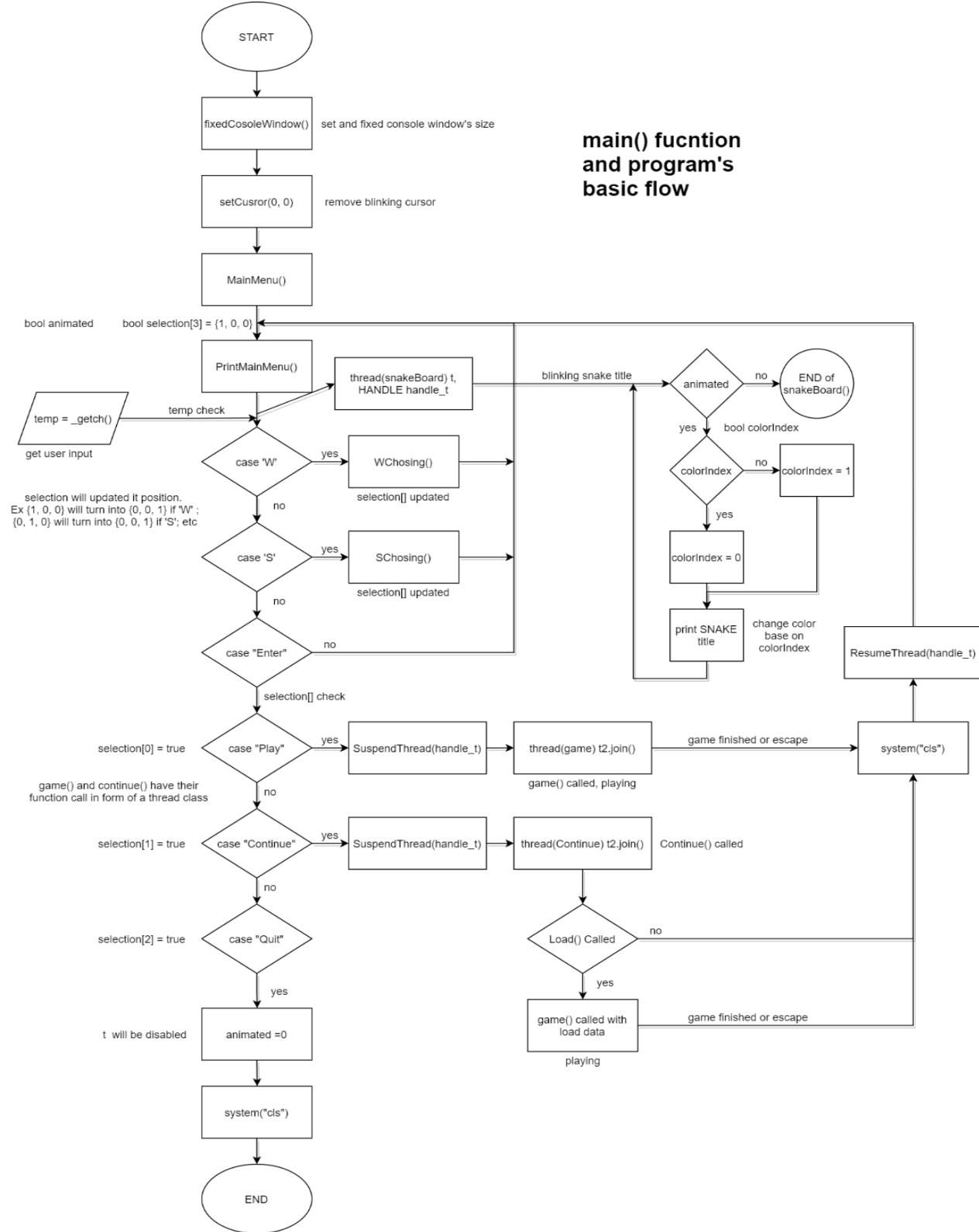
Đánh giá tổng thể: 100%

Chi tiết từng yêu cầu:

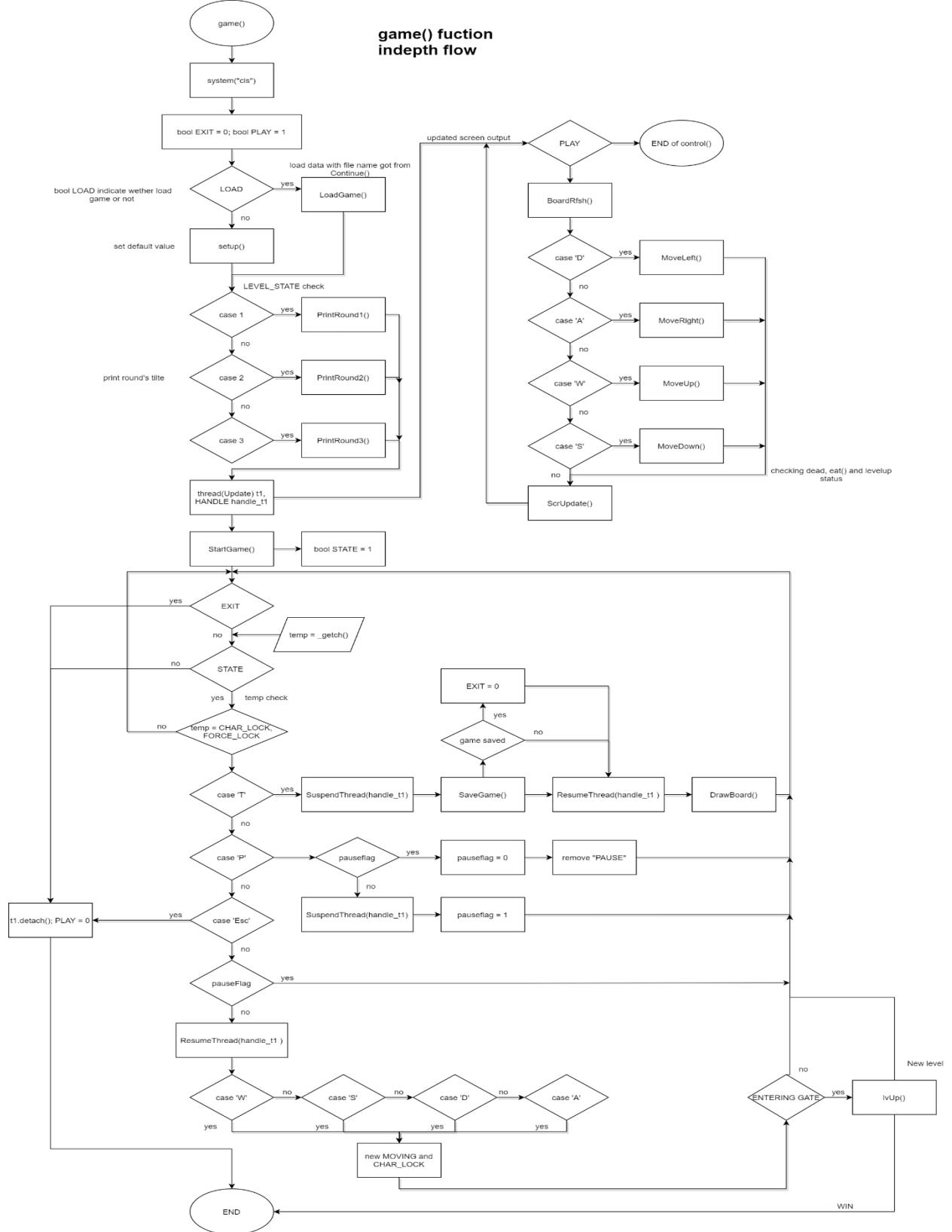
Yêu cầu	Hoàn thành	Người thực hiện
Xử lý khi đầu snake chạm vào thân snake	100%	Nguyên
Xử lý lưu trò chơi/tải trò chơi đã lưu	90%	Nguyên, Thịnh
Xử lý độ dài snake	100%	Thịnh
Xử lý khi ăn xong food ở một cấp	100%	Dương, Khanh
Xử lý hiệu ứng khi va chạm	100%	Khanh
Xử lý màu cho trò chơi	90%	Khôi, Khanh
Main Menu	100%	Khôi
Thiết kế màn chơi và qua màn	90%	Dương

2. LUU ĐO (FLOW CHART)

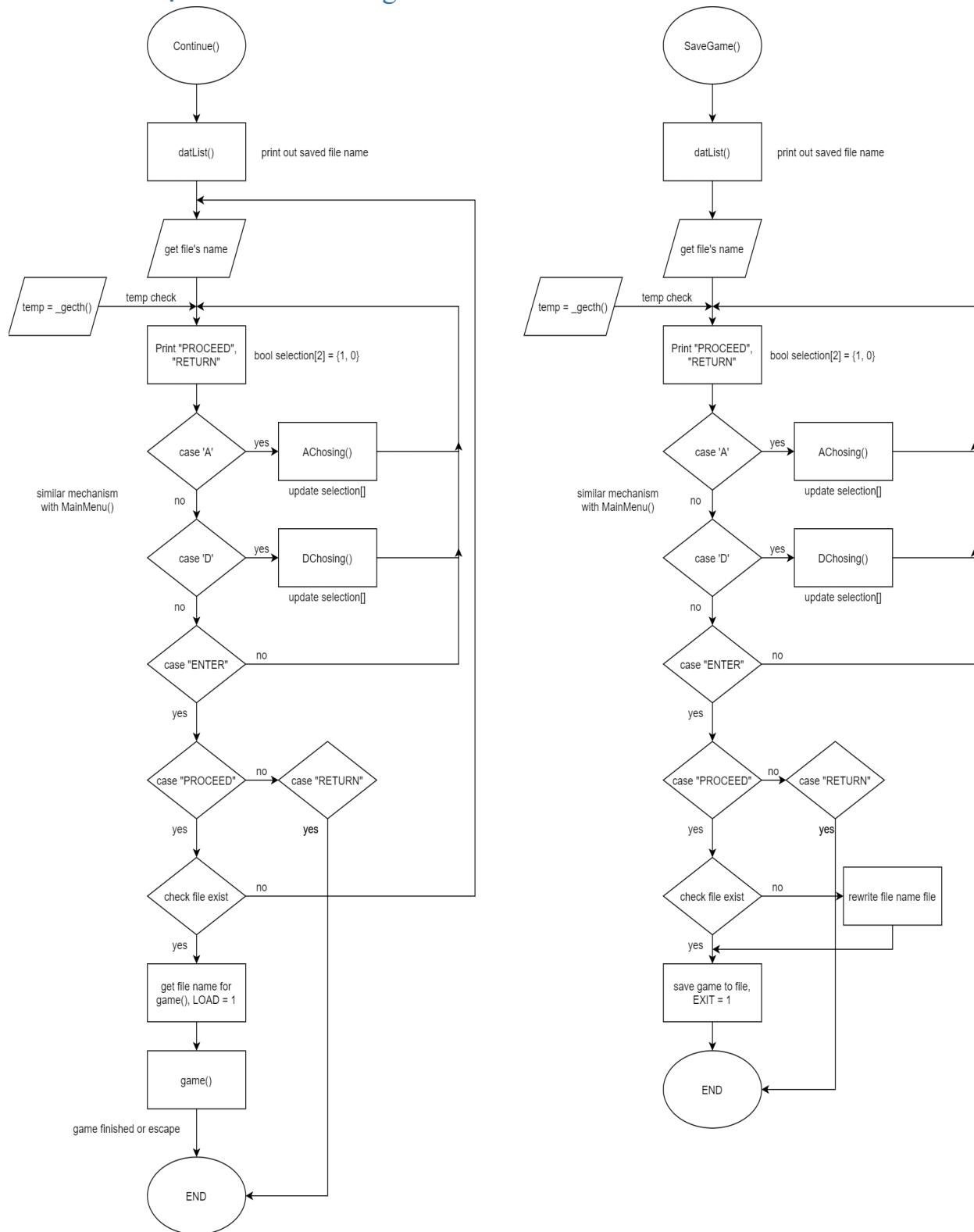
2.1. Lưu đồ vận hành menu chính



2.2. Lưu đồ vận hành game



2.3. Lưu đồ vận hành save/load game



3. CÁC HÀM CHÍNH TRONG GAME

3.1. KHỐI HÀM MENU

3.1.1. void snakeBoard()

- Mục tiêu của hàm này là xuất ra màn hình console logo của trò chơi với hiệu ứng nhấp nháy đổi màu (bảng màu ở đây được define trong header **snake.h**).

```
10 //Color
11 #define Black 0
12 #define Blue 1
13 #define Green 2
14 #define Aqua 3
15 #define Red 4
16 #define Purple 5
17 #define Yellow 6
18 #define White 7
19 #define Gray 8
20 #define LBlue 9
21 #define LGreen 10
22 #define LAqua 11
23 #define LRed 12
24 #define LPurple 13
25 #define LYellow 14
26 #define BWhite 15
```

3.1.2. void PrintMainMenu()

```
78 //menu
79 void PrintMainMenu() {
80     GoToXY(WIDTH_CONSOLE / 2 + 15, HEIGH_CONSOLE / 2 + 1);setOutputColor(selection[0] ? LRed : Black, White);
81     cout << "START";
82     GoToXY(WIDTH_CONSOLE / 2 + 13, HEIGH_CONSOLE / 2 + 3);setOutputColor(selection[1]? LRed : Black, White);
83     cout << "CONTINUE";
84     GoToXY(WIDTH_CONSOLE / 2 + 15, HEIGH_CONSOLE / 2 + 5);setOutputColor(selection[2] ? LRed : Black, White);
85     cout << "QUIT";
86     GoToXY(WIDTH_CONSOLE / 2 , HEIGH_CONSOLE / 2 + 10); setOutputColor(selection[2] ? LRed : Black, White);
87     setOutputColor(Black, White);
88     cout << "S to up, W to down, Enter to select.";
89 }
```

- Hàm đảm nhận việc in ra màn hình ba sự lựa chọn của trò chơi là “START”, “CONTINUE” và “QUIT” bên dưới phần logo của trò chơi. Hàm **setOutputColor(int text, int BG)** có chức năng thay đổi, hiển thị màu sắc cho ba sự lựa chọn, được khai báo ở file *console_and_output.cpp*, hàm nhận vào hai giá trị màu của chữ (*text*) và nền console (*BG - background*), xử lý màu theo công thức $BG*16 + text$ biểu thức này có được do cơ chế màu của hệ thống.

```
43 void setOutputColor(int text, int BG) {
44     //textColor+ BGColor*16
45     HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
46     SetConsoleTextAttribute(hConsole, text + BG*16);
47 }
```

3.1.3. void SaveMenu()

```
273 void SaveMenu() {
274     DrawBoard(WIDTH_CONSOLE / 2 - 8, HEIGH_CONSOLE - HEIGH_CONSOLE, 50, 15, Red);
275     datList();
276     setOutputColor(Red, White);
277     string a;
278     GoToXY(WIDTH_CONSOLE / 2 - 5, HEIGH_CONSOLE / 2 - 1);
279     cout << "enter for next step, then A,D to navigate";
280     setOutputColor(Blue, White);
281     GoToXY(WIDTH_CONSOLE / 2 + 25, HEIGH_CONSOLE / 2 - 3);
282     cout << ".txt";
283     GoToXY(WIDTH_CONSOLE / 2 - 5, HEIGH_CONSOLE / 2 - 3);
284     cout << "Enter name: "; getline(cin >> ws, a); a += ".txt";
285     Control3(a);
286 }
```

- In ra menu của mục save game (load game có cơ chế tương tự). Bao gồm những lệnh vẽ bảng, in ra danh sách file đã lưu, chỉnh màu,...

3.1.4. void MainMenu()

- Hàm mang tính chất điều hành hoạt động của menu. Mảng selection có chức năng như một biến cờ đánh dấu sự lựa chọn. Hàm WChosing() và SChosing() là 2 lệnh điều hướng để thay đổi vị trí của cờ đang đánh dấu. Hàm này cũng tận dụng thêm cơ chế của class thread để làm cho logo game chạy độc lập với lệnh điều khiển/hiện thị menu.

```
92 void MainMenu() {
93     setup();
94     int Index = 0; animated = 1;
95     selection[0] = 1; selection[1] = 0; selection[2] = 0;
96     PrintMainMenu();
97     thread t1(snakeBoard); HANDLE handle_t1 = t1.native_handle();
98     while (1) {
99         int key = _getch();
100        switch (key) {
101            case 119:
102                WChosing(Index, selection);
103                break;
104            case 115:
105                SChosing(Index, selection);
106                break;
107            case 13: // enter
108                if (selection[0]) {
109                    SuspendThread(handle_t1);
110                    Sleep(100);
111                    system("cls"); LOAD = 0;
112                    thread t2(game); t2.join();
113                    system("CLS");
114                }
115                if (selection[1]) {
116                    SuspendThread(handle_t1);
117                    Sleep(100);
118                    system("cls");
119                    thread t2(Continue); t2.join();
120                    system("CLS");
121                }
122                if (selection[2]) {
123                    t1.detach(); animated = 0; system("cls");
124                    return;
125                }
126                break;
127            default:
128                break;
129        }
130        ResumeThread(handle_t1);
131    }
132 }
```

3.2. KHÔI HÀM QUẢN LÝ CONSOLE VÀ OUTPUT

3.2.1. void DrawBoard(int x, int y, int width, int height, int color)

```
53 void DrawBoard(int x, int y, int width, int height, int color) {
54     setOutputColor(color, White);
55     GoToXY(x, y); cout << '\xC9';
56     GoToXY(x, y + height + y); cout << '\xC8';
57     GoToXY(x + width, y); cout << '\xBB';
58     GoToXY(x + width, y + height + y); cout << '\xBC';
59     GoToXY(x + 1, y); cout << '\xCD';
60     for (int i = x; i < x + width - 2; i++)
61         std::cout << '\xCD';
62     GoToXY(x + 1, y + height + y); std::cout << '\xCD';
63     for (int i = x; i < x + width - 2; i++)
64         std::cout << '\xCD';
65
66     for (int i = y + 1; i < y + height + y; i++) {
67         GoToXY(x, i); std::cout << '\xBA';
68         GoToXY(x + width, i); std::cout << '\xBA';
69     }
70     setOutputColor(BWhite, White);
71     GoToXY(0, 0);
72 }
```

- Chức năng chính của làm là vẽ ra khung của trò chơi mỗi lần được gọi đến.

3.2.2. void statusBoard()

```
70 void statusBoard() {
71     DrawBoard(WIDTH_CONSOLE + 3, 4, 35, 5, 12);
72     setOutputColor(Blue, White);
73     GoToXY(WIDTH_CONSOLE + 5, 5);
74     std::cout << "W, A, S, D to move";
75     GoToXY(WIDTH_CONSOLE + 5, 6);
76     std::cout << "P to pause or resume";
77     GoToXY(WIDTH_CONSOLE + 5, 7);
78     std::cout << "O to save and quit";
79     GoToXY(WIDTH_CONSOLE + 5, 8);
80     std::cout << "ESC to exit";
81     GoToXY(WIDTH_CONSOLE + 5, 9);
82     std::cout << "size: " << SIZE_SNAKE << " speed: " << SPEED;
83     GoToXY(WIDTH_CONSOLE + 5, 10);
84     std::cout << "score: " << SCORE;
85     setOutputColor(BWhite, White);
86     if (LEVEL_UP) {
87         GoToXY(WIDTH_CONSOLE + 14, HEIGH_CONSOLE - 5);
88         setOutputColor(Green, White);
89         cout << "<=_FINISHED_=>";
90         GoToXY(WIDTH_CONSOLE + 3, HEIGH_CONSOLE - 3);
91         setOutputColor(Red, White);
92         std::cout << "Enter gate and press any key to continue.";
93     }
94 }
```

- Đảm nhận việc vẽ khung cho bảng trạng thái của người chơi và chỉ dẫn điều khiển cho người chơi.

3.2.3. void foodOut(bool type)

```
96 void foodOut(bool type) {  
97     setOutputColor(LRed, White);  
98     char chr;  
99     if (type) chr = 3;  
100    else chr = ' ';  
101    GoToXY(food[FOOD_INDEX].x, food[FOOD_INDEX].y);  
102    std::cout << chr;  
103    setOutputColor(BWhite, White);  
104 }
```

- Chức năng chính của hàm này chỉ đơn giản là in khói thức ăn ra màn hình console. Biến bool type đóng vai trò báo cho hàm này xoá hay hiển thị food.

3.2.4. void snakeOut(bool type)

```
107 void snakeOut(bool type) {  
108     setOutputColor(LGreen, White);  
109     if (type) {  
110         char sChar[] = { '2', '0', '1', '2', '7', '4', '7', '4',  
111                         '2', '0', '1', '2', '7', '4', '0', '4',  
112                         '2', '0', '1', '2', '7', '2', '0', '4',  
113                         '2', '0', '1', '2', '7', '2', '1', '3',  
114                         '2', '0', '1', '2', '7', '3', '3', '5' };  
115         int index = 0;  
116         for (int i = 0; i < SIZE_SNAKE - 1; i++) {  
117             GoToXY(snake[i].x, snake[i].y);  
118             std::cout << sChar[index++];  
119         }  
120         GoToXY(snake[SIZE_SNAKE - 1].x, snake[SIZE_SNAKE - 1].y);  
121         std::cout << '0';  
122     }  
123     else {  
124         for (int i = 0; i < SIZE_SNAKE; i++) {  
125             GoToXY(snake[i].x, snake[i].y);  
126             std::cout << ' ';  
127         }  
128         setOutputColor(BWhite, White);  
129     }
```

- In ra con rắn trên màn hình console. Mảng sChar chứa những khúc của con rắn, mỗi khi ăn thành công thức ăn, hàm sẽ in ra thêm 1 phần tử trong mảng giúp con rắn dài ra. Đầu của con rắn được đánh dấu bằng kí tự 'O'. Biến type nhận vào có ý nghĩa như một biến điều kiện nếu điều kiện là đúng thì sẽ tiến hành in ra con rắn ngược lại sẽ in ra khoảng trắng (xoá thân rắn khi con rắn đã đi qua).

3.2.5. void gateOut()

```
133 void gateOut() {
134     for (int i = 0; i <= 2; i++)
135     {
136         GoToXY(GATE[i].x, GATE[i].y);
137         setOutputColor(Blue, White);
138         cout << (char)222;
139     }
140     setOutputColor(Blue, White);
141     GoToXY(GATE[3].x, GATE[3].y); cout << (char)220;
142     setOutputColor(Blue, White);
143     GoToXY(GATE[5].x, GATE[5].y); cout << (char)223;
144     setOutputColor(Blue, White);
145     GoToXY(GATE[4].x, GATE[4].y); cout << (char)176;
146     setOutputColor(Blue, White);
147     GoToXY(GATE[6].x, GATE[6].y); cout << (char)220;
148     setOutputColor(Blue, White);
149     GoToXY(GATE[7].x, GATE[7].y); cout << (char)223;
150 }
```

- Chức năng chính của hàm này in cổng qua màn khi đã ăn hết số food nhất định, từ dòng 134 đến dòng 149 ta lần lượt thực hiện các thao tác qui định màu (setOutputColor) và sau đó di chuyển đến toạ độ của từng phần tử trong mảng GATE đã được quy định trong hàm **void gateGeneration()** (được giải thích trong mục 3.3.1 – Khối hàm cơ chế game), sau đó in cổng ra.

3.2.6. void wallOut()

- Hàm có chức năng in ra các chướng ngại vật bắt đầu từ màn 2 trở đi. Ở đây ta sẽ dùng một chuỗi điều kiện switch case trong đó nhận vào biến LEVEL_STATE đại diện cho màn đang chơi, các hàm for trong mỗi trường hợp có chức năng khởi tạo và in tường ra màn hình console theo toạ độ cho trước.

```

149 void wallOut()
150 {
151     switch (LEVEL_STATE)
152     {
153     case 2:
154     {
155         for (int i = 3; i < HEIGH_CONSOLE - 2; i++)
156         {
157             GoToXY(WIDTH_CONSOLE / 2 - 3, i);
158             setOutputColor(LYellow, White);
159             cout << (char)219;
160             if (i == HEIGH_CONSOLE / 2 - 2) i = HEIGH_CONSOLE / 2 + 1;
161         }
162         for (int i = 3; i < HEIGH_CONSOLE - 2; i++)
163         {
164             GoToXY(WIDTH_CONSOLE / 2 + 3, i);
165             setOutputColor(LPurple, White);
166             cout << (char)219;
167             if (i == HEIGH_CONSOLE / 2 - 2) i = HEIGH_CONSOLE / 2 + 1;
168         }
169         for (int i = 4; i < WIDTH_CONSOLE - 3; i++)
170         {
171             GoToXY(i, HEIGH_CONSOLE / 2 - 2);
172             setOutputColor(LRed, White);
173             cout << (char)219;
174             if (i == WIDTH_CONSOLE / 2 - 4) i = WIDTH_CONSOLE / 2 + 3;
175         }
176         for (int i = 4; i < WIDTH_CONSOLE - 3; i++)
177         {
178             GoToXY(i, HEIGH_CONSOLE / 2 + 2);
179             setOutputColor(LAqua, White);
180             cout << (char)219;
181             if (i == WIDTH_CONSOLE / 2 - 4) i = WIDTH_CONSOLE / 2 + 3;
182         }
183         break;
184     }
185     case 3:
186     {
187         int i = 0;
188         int color = 1;
189         for (; i <= HEIGH_CONSOLE - 5; i++)
190         {
191             //Thanh doc thu 1 ben trai
192             GoToXY(4, 1 + i);
193             setOutputColor(color, White);
194             cout << (char)219;
195         }
196         color++;
197         for (i = 4; i < WIDTH_CONSOLE - 4; i++)
198         {
199             //Thanh ngang duoi thu 1
200             GoToXY(1, HEIGH_CONSOLE - 3);
201             setOutputColor(color, White);
202             cout << (char)219;
203         }
204         color++;
205         for (i = HEIGH_CONSOLE - 3; i > 3; i--)
206         {
207             //Thanh doc thu 1 ben phai
208             GoToXY(WIDTH_CONSOLE - 4, i);
209             setOutputColor(color, White);
210             cout << (char)219;
211         }
212         color++;
213         for (i = WIDTH_CONSOLE - 4; i > 8; i--)
214         {
215             //Thanh ngang tren thu 1
216             GoToXY(i, 3);
217             setOutputColor(color, White);
218             cout << (char)219;
219         }
220         color++;
221         for (i = 3; i < HEIGH_CONSOLE - 6; i++)
222         {
223             //Thanh doc ben trai thu 2
224             GoToXY(8, i);
225             setOutputColor(color, White);
226             cout << (char)219;
227         }
228         color++;
229         for (i = 8; i < WIDTH_CONSOLE - 8; i++)
230         {
231             //Thanh ngang duoi thu 2
232             GoToXY(i, HEIGH_CONSOLE - 6);
233             setOutputColor(color, White);
234             cout << (char)219;
235         }
236         color += 2;
237         for (i = HEIGH_CONSOLE - 6; i > 6; i--)
238         {
239             //Thanh doc ben phai lan 2
240             GoToXY(WIDTH_CONSOLE - 8, i);
241             setOutputColor(color, White);
242             cout << (char)219;
243         }
244         color++;
245         for (i = WIDTH_CONSOLE - 8; i > 10; i--)
246         {
247             //Thanh ngang tren lan 2
248             GoToXY(i, 6);
249             setOutputColor(color, White);
250             cout << (char)219;
251         }
252     }
253     break;
254 }
255 }
256 }
```

3.2.7. void ScrUpdate()

```
267 void ScrUpdate() {  
268     snakeOut(STATE);  
269     foodOut(!LEVEL_UP);  
270     statusBoard();  
271     Sleep(1000 / SPEED);  
272 }
```

- Chức năng chính của hàm này là cập nhật lại toạ độ mới nhất của rắn và toạ độ của thức ăn, đồng thời cập nhật lại bảng trạng thái của người chơi (điểm số, tốc độ di chuyển,...)

3.2.8. void BoardRfsh()

```
275 void BoardRfsh() {  
276     snakeOut(0);  
277     foodOut(0);  
278 }
```

- Hàm này có chức năng xoá toạ độ cũ của con rắn và của thức ăn đã được rắn ăn. Khi chạy hàm này, con rắn sẽ không dài ra một cách không kiểm soát khi di chuyển mà sẽ chỉ giãn ra khi ăn được thức ăn theo yêu cầu của trò chơi.

3.2.9. void deadAnimate()

```
281 void deadAnimate() {  
282     setOutputColor(Green, White);  
283     for (int i = SIZE_SNAKE - 1; i >= 0; i--) {  
284         GoToXY(snake[i].x, snake[i].y);  
285         std::cout << '\xB2';  
286         Sleep(100);  
287     }  
288     setOutputColor(LRed, White);  
289     for (int i = SIZE_SNAKE - 1; i >= 0; i--) {  
290         GoToXY(snake[i].x, snake[i].y);  
291         std::cout << '\xDB';  
292         Sleep(100);  
293     }  
294     setOutputColor(BWhite, White);  
295     for (int i = SIZE_SNAKE - 1; i >= 0; i--) {  
296         GoToXY(snake[i].x, snake[i].y);  
297         std::cout << ' ';  
298         Sleep(100);  
299     }  
300 }
```

- Khi con rắn chết cần phải có một hiệu ứng báo hiệu điều đó, hàm deadAnimate() có chức năng giúp ta làm điều đó. Ở đây nhóm em làm hiệu ứng khi con rắn chết sẽ nhấp nháy 3 lần trong thời gian ngắn bằng cách sử dụng lệnh Sleep() giữa 2 lệnh in và xoá rắn.

3.2.10. void printRound (int LEVEL_STATE)

```
288 void printRound(int LEVEL_STATE)
289 {
290     if (LEVEL_STATE == 1) {
291         system("cls");
292         setOutputColor(RED, WHITE);
293         GoToXY(WIDTH_CONSOLE - 20, HEIGHT_CONSOLE - 7);
294         cout << ">>ROUND 1<<";
295     }
296     if (LEVEL_STATE == 2) {
297     {
298         system("CLS");
299         setOutputColor(RED, WHITE);
300
301         GoToXY(WIDTH_CONSOLE - 20, HEIGHT_CONSOLE - 7);
302         cout << ">>ROUND 2<<";
303     }
304     if (LEVEL_STATE == 3) {
305         setOutputColor(RED, WHITE);
306         GoToXY(WIDTH_CONSOLE - 20, HEIGHT_CONSOLE - 7);
307         cout << ">>ROUND 3<<";
308     }
309     if (LEVEL_STATE == 4) {
310         setOutputColor(RED, WHITE);
311         GoToXY(WIDTH_CONSOLE / 2 - 23, HEIGHT_CONSOLE - 7);
312         cout << "YOU WIN!!!";
313     }
314 }
```

- Hàm này có chức năng chính là in ra tên màn chơi ra màn hình console. Như ta có thể thấy ở hình trên trước khi bắt đầu một màn chơi hàm sẽ giúp in ra thông tin trên màn hình console trong một khoảng thời gian nhất định.

3.3. KHÓI HÀM CƠ CHẾ GAME

3.3.1. void gateGeneration()

```
83 void gateGeneration()
84 {
85
86     GATE_SIZE = 8;
87     GATE = new POINT[GATE_SIZE];
88     srand((int)time(0));
89     do
90     {
91         GATE[4].x = rand() % (WIDTH_CONSOLE - 6) + 2;
92         GATE[4].y = rand() % (HEIGHT_CONSOLE - 5) + 3;
93         GATE[0].x = GATE[4].x - 1; GATE[0].y = GATE[4].y - 1;
94         GATE[1].x = GATE[4].x - 1; GATE[1].y = GATE[4].y;
95         GATE[2].x = GATE[4].x - 1; GATE[2].y = GATE[4].y + 1;
96         GATE[3].x = GATE[4].x; GATE[3].y = GATE[4].y - 1;
97         GATE[5].x = GATE[4].x; int GATE_SIZE
98         GATE[6].x = GATE[4].x + TE[4].y - 1;
99         GATE[7].x = GATE[4].x + global value(variable) TE[4].y + 1;
100    } while (!isValid(GATE, GATE_SIZE));
101 }
```

- Tạo cổng theo toạ độ ngẫu nhiên chọn bởi dòng 91 và 92 sau đó gán toạ độ cổng theo ý mà người lập trình muốn. Ở đây chúng ta sẽ xây dựng cái cổng theo mẫu sau:



3.3.2. void deleteGate()

```
106 void deleteGate()
107 {
108     for (int i = 0; i < GATE_SIZE; i++)
109     {
110         GoToXY(GATE[i].x, GATE[i].y);
111         std::cout << ' ';
112     }
113 }
114 delete[] GATE; GATE = NULL;
115 }
```

- Hàm deleteGate có chức năng xoá cửa khi được gọi. Vòng for ở dòng 108 có chức năng xoá công trên màn hình console, đồng thời xoá mảng GATE hiện hành và gán nó bằng NULL để có thể tạo lại khi cần thiết.

3.3.3. void wallGeneration()

- Bắt đầu mỗi màn chơi hàm này sẽ giúp in ra những chướng ngại vật của màn chơi đó. Như trong source code này nhóm em quy định màn 1 sẽ không có chướng ngại vật và không có tường chắn xung quanh, con rắn được di chuyển tự do nên khi gặp case 1 hàm sẽ return. Nhưng từ case 2 trở đi, hàm sẽ tạo những chướng ngại vật dạng chữ thập ở mảng 2 và vòng xoáy ở mảng 3 bằng cách gán toạ độ cho từng phần tử trong mảng động wall. Vòng lặp for chạy sẽ đảm nhận nhiệm vụ set giá trị cho từng phần tử trong mảng đó để đến khi gọi hàm **void wallOut()** (được giải thích ở mục 3.2.7 – khôi hàm quản lý console và output) các chướng ngại vật sẽ được in ra theo toạ độ đã được gán trước đó.

```

117     void wallGeneration()
118     {
119         int wIndex = 0;
120         switch (LEVEL_STATE)
121         {
122             case 1: return;
123             case 2:
124             {
125                 WALL_SIZE = 146;
126                 wall = new POINT[WALL_SIZE];
127                 for (int i = 3; i < HEIGH_CONSOLE - 2; i++)
128                 {
129                     wall[wIndex].x = WIDTH_CONSOLE / 2 - 3;
130                     wall[wIndex].y = i;
131                     if (i == HEIGH_CONSOLE / 2 - 2) i = HEIGH_CONSOLE / 2 + 1;
132                     wIndex++;
133                 }
134                 for (int i = 3; i < HEIGH_CONSOLE - 2; i++)
135                 {
136                     wall[wIndex].x = WIDTH_CONSOLE / 2 + 3;
137                     wall[wIndex].y = i;
138                     if (i == HEIGH_CONSOLE / 2 - 2) i = HEIGH_CONSOLE / 2 + 1;
139                     wIndex++;
140                 }
141                 for (int i = 4; i < WIDTH_CONSOLE - 3; i++)
142                 {
143                     wall[wIndex].x = i;
144                     wall[wIndex].y = HEIGH_CONSOLE / 2 - 2;
145                     if (i == WIDTH_CONSOLE / 2 - 4) i = WIDTH_CONSOLE / 2 + 3;
146                     wIndex++;
147                 }
148                 for (int i = 4; i < WIDTH_CONSOLE - 3; i++)
149                 {
150                     wall[wIndex].x = i;
151                     wall[wIndex].y = HEIGH_CONSOLE / 2 + 2;
152                     if (i == WIDTH_CONSOLE / 2 - 4) i = WIDTH_CONSOLE / 2 + 3;
153                     wIndex++;
154                 }
155                 break;
156             }
157         case 3:
158         {
159             WALL_SIZE = 275;
160             wall = new POINT[WALL_SIZE];
161             int i = 0;
162             for (; i <= HEIGH_CONSOLE - 5; i++)
163             {
164                 //Thanh doc thu 1 ben trai
165                 wall[wIndex].x = 4;
166                 wall[wIndex].y = 1 + i;// 1 + 20-5 = 16
167                 wIndex++;
168             }
169             for (i = 4; i < WIDTH_CONSOLE - 4; i++)
170             {
171                 //Thanh ngang duoi thu 1
172                 wall[wIndex].x = i;
173                 wall[wIndex].y = HEIGH_CONSOLE - 3; // 17
174                 wIndex++;
175             }
176             for (i = HEIGH_CONSOLE - 3; i > 3; i--)
177             {
178                 //Thanh doc thu 1 ben phai
179                 wall[wIndex].x = WIDTH_CONSOLE - 4;
180                 wall[wIndex].y = i;
181                 wIndex++;
182             }
183             for (i = WIDTH_CONSOLE - 4; i > 8; i--)
184             {
185                 //Thanh ngang tren thu 1
186                 wall[wIndex].x = i;
187                 wall[wIndex].y = 3;
188                 wIndex++;
189             }
190             for (i = 3; i < HEIGH_CONSOLE - 6; i++)
191             {
192                 //Thanh doc ben trai thu 2
193                 wall[wIndex].x = 8;
194                 wall[wIndex].y = i;
195                 wIndex++;
196             }
197             for (i = 8; i < WIDTH_CONSOLE - 8; i++)
198             {
199                 //Thanh ngang duoi thu 2
200                 wall[wIndex].x = i;
201                 wall[wIndex].y = HEIGH_CONSOLE - 6;
202                 wIndex++;
203             }
204             for (i = HEIGH_CONSOLE - 6; i > 6; i--)
205             {
206                 //Thanh doc ben phai lan 2
207                 wall[wIndex].x = WIDTH_CONSOLE - 8;
208                 wall[wIndex].y = i;
209                 wIndex++;
210             }
211             for (i = WIDTH_CONSOLE - 8; i > 10; i--)
212             {
213                 //Thanh ngang tren lan 2
214                 wall[wIndex].x = i;
215                 wall[wIndex].y = 6;
216                 wIndex++;
217             }
218         }
219     }
220 }

```

3.3.4. void ProgressDead()

```
441 void ProgressDead() { //dead initiate
442     FORCE_LOCK[0] = 27; FORCE_LOCK[1] = 'T';
443     deadAnimate();
444     STATE = 0; LEVEL_UP = 0;
445     system("cls");
446     if (LEVEL_STATE > 1) {
447         delete[] wall; //wall = NULL;
448     }
449     //reset snake and food data to NULL
450     for (int i = 0; i < SIZE_SNAKE; i++) {
451         snake[i] = { NULL, NULL };
452     }
453     for (int i = 0; i < MAX_SIZE_FOOD; i++) {
454         food[i] = { NULL, NULL };
455     }
456     deadMessage();
457 }
```

- Khi con rắn chết chúng ta cần một hàm có thể xử lý hiệu ứng, xoá tất cả toạ độ hiện thời của rắn, thức ăn và chướng ngại vật của màn chơi. Và hàm ProgressDead có chức năng như trên. Mảng FORCE_LOCK có chức năng giữ không cho người chơi thực hiện những chức năng như lưu game ở thời điểm đó (bấm phím ‘T’) và phím ESC (số 27 trong bảng mã ASCII), 2 vòng lặp for ở cuối hàm đóng vai trò xoá toạ độ của rắn và food tránh trường hợp cả rắn và food đè lên hàm deadMessage() (báo hiệu chết) do cơ chế chạy của class thread.

3.3.5. void setup()

```
460 void setup() {
461     //preset board and snake/food
462     //snake location lock
463     CHAR_LOCK = 'A', MOVING = 'D', SPEED = 4; FOOD_INDEX = 0;
464     WIDTH_CONSOLE = 70, HEIGHT_CONSOLE = 20, SIZE_SNAKE = 4; LEVEL_UP = false;
465     LEVEL_STATE = 1;
466     //snake location
467     snake[0] = { 10, 5 }; snake[1] = { 11, 5 };
468     snake[2] = { 12, 5 }; snake[3] = { 13, 5 };
469     //snake[4] = { 14, 5 }; snake[5] = { 15, 5 };
470     foodGeneration();
471     SCORE = 0;
472 }
```

- Hàm có chức năng đơn giản cài đặt các giá trị trong game vào trạng thái mặc định khi người chơi bắt đầu một màn chơi mới. Nó sẽ được thay thế bằng hàm LoadGame() nếu ta tải file game từ MainMenu().

3.3.6. void Update()

```
487 void Update() {
488     setcursor(0, 0);
489     while (PLAY) {
490         if (STATE == 1) {
491             BoardRfsh();
492             switch (MOVING) {
493                 case 'A': MoveLeft(); break;
494                 case 'D': MoveRight(); break;
495                 case 'W': MoveUp(); break;
496                 case 'S': MoveDown(); break;
497             }
498             ScrUpdate();
499         }
500     }
501 }
```

- Chức năng chính của hàm là cập nhật màn hình liên tục nếu rắn còn sống. Hàm này được đưa vào thread để chạy độc lập với hàm game(). Các hàm điều hướng (MoveLeft, MoveRight, MoveUp, MoveDown) sẽ kiểm tra rắn chết, rắn ăn, hướng rắn đi và cập nhật lại tọa độ mới cho rắn và food(nếu rắn ăn).

3.3.7. void lvUP()

```
578 void lvlUp()
579 {
580     SPEED += 2;
581     if (LEVEL_STATE > 1)
582     {
583         delete[] wall; wall = NULL;
584         LEVEL_STATE++;
585         system("cls");
586         printRound(LEVEL_STATE);
587         Sleep(3000);
588         StartGame();
589         wallGeneration();
590         foodGeneration();
591         deleteGate();
592         gateGeneration();
593         gateOut();
594         SIZE_SNAKE += 4;
595         for (int i = 0; i < SIZE_SNAKE; i++)
596         {
597             snake[i] = GATE[4];
598         }
599         wallOut();
600         FORCE_LOCK[0] = NULL; FORCE_LOCK[1] = NULL;
601 }
```

- Sau khi kết thúc một màn chơi hàm sẽ tự động xoá những chướng ngại vật, cập nhật LEVEL_STATE (trạng thái màn chơi) đồng thời chuyển qua màn tiếp theo với cài đặt mới. Hàm sẽ giữ độ dài rắn và tốc độ nhưng mỗi màn chơi lại có tốc độ khởi đầu lớn hơn màn trước đó 1 tí

3.3.8. bool samePoint (POINT a, POINT b)

Hàm này sẽ báo 2 toạ độ có trùng nhau hay không theo kiểu giá trị POINT

```
19  bool samePoint(POINT a, POINT b)
20  {
21      if (a.x == b.x && a.y == b.y)
22          return true;
23      return false;
24 }
```

3.3.9. bool pointValid(POINT target[], int nTarget)

Hàm kiểm tra xem toạ độ được kiểm tra có đè lên toạ độ cũ nào không.

```
45  bool pointValid(POINT target[], int nTarget)
46  {
47      for (int i = 0; i < nTarget; i++)
48      {
49          for (int j = 0; j < SIZE_SNAKE; j++)
50              if (samePoint(target[i], snake[j]))
51                  return false;
52          for (int j = 0; j < FOOD_INDEX; j++)
53          {
54              if (samePoint(target[i], food[i]))
55                  return false;
56          }
57          if (LEVEL_STATE > 1)
58          {
59              for (int j= 0; j < WALL_SIZE; j++)
60              {
61                  if (samePoint(target[i], wall[j]))
62                      return false;
63                  if (samePoint(target[i].x + 4, target[i].y, wall[j].x, wall[j].y))
64                      return false;
65              }
66          }
67      }
68      return true;
69 }
```

3.4. KHÓI HÀM SAVE/LOAD GAME

Trong đồ án này, nhóm em chọn phương thức load và save bằng cách yêu cầu người dùng nhập tên file muốn save hay tên file muốn load vào màn hình console sau khi xác nhận chương trình sẽ tiến hành mở hay ghi thông tin vào file có tên tương ứng.

3.4.1. string* loadRawDATA(int& n)

```
515  string* loadRawDATA(int& n) {
516      ifstream input("rawData.txt");
517      if (!input)
518          return NULL;
519      input >> n;
520      string* a = new string[n + 1];
521      for (int i = 0; i < n; i++) {
522          getline(input >> ws, a[i]);
523      }
524      input.close();
525      return a;
526 }
```

- Hàm này nhằm mục đích đọc nội dung từ file “rawData.txt” chứa tên của những file đã save trong hệ thống. Từ đó ta sẽ dùng dữ liệu lấy được xuất ra màn hình cho người dùng biết để nhập. file rawData.txt không thể thiếu trong hàm này.

3.4.2. void LoadGame (string a)

```
528  void LoadGame(string a) {
529      WIDTH_CONSOLE = 70, HEIGHT_CONSOLE = 20;
530      ifstream input(a.c_str());
531      //
532      input >> MOVING >> SPEED >> SCORE >> LEVEL_STATE;
533      //snake location
534      input >> SIZE_SNAKE;
535      for (int i = 0; i < SIZE_SNAKE; i++) {
536          input >> snake[i].x >> snake[i].y;
537      }
538      if (MOVING == 'D') CHAR_LOCK = 'A';
539      else if (MOVING == 'W') CHAR_LOCK = 'S';
540      else if (MOVING == 'S') CHAR_LOCK = 'W';
541      else CHAR_LOCK = 'D';
542      //
543      if (LEVEL_STATE > 1) {
544          wallGeneration();
545      }
546      FOOD_INDEX = 0;
547      foodGeneration();
548      input.close();
549      return;
550 }
```

- Khi được gọi, hàm sẽ tiến hành đọc file .txt theo địa chỉ nhập vào từ bàn phím từ đó lấy thông tin về điểm số, tốc độ, kích thước của con rắn,... đã được lưu trước đó. Nó sẽ thay thế hàm setup() do biến bool LOAD được đưa về true từ lệnh Continue().

3.4.3. void SaveGame(string a)

```
552 void SaveGame(string a) {
553     // rawData section
554     if (!FileExist(a)) {
555         int max;
556         string* b = loadRawDATA(max);
557         ofstream output2("rawData.txt");
558         output2 << max + 1 << endl;
559         for (int i = 0; i < max; i++) {
560             output2 << b[i] << endl;
561         }
562         output2 << a;
563         output2.close();
564         delete[] b;
565     }
566     // Save section
567     ofstream output(a.c_str());
568     output << MOVING << " " << SPEED << " " << SCORE << " " << LEVEL_STATE << endl;
569     output << SIZE_SNAKE << " ";
570     for (int i = 0; i < SIZE_SNAKE; i++) {
571         output << snake[i].x << " " << snake[i].y << " ";
572     }
573     output.close();
574 }
```

- Tương tự như hàm LoadGame(string a) nhưng lần này hàm SaveGame(string a) sẽ ghi những thông tin hiện có trên hệ thống của trò chơi vào file .txt có tên được người dùng nhập từ bàn phím. Hàm sẽ đồng thời đưa game về màn hình chính khi lưu thành công

3.4.4. void YesNoScreen()

```
145 //Save/Load game's display
146 void YesNoScreen() {
147     GoToXY(WIDTH_CONSOLE - 26, HEIGH_CONSOLE - HEIGH_CONSOLE + 12);
148     setOutputColor(selection2[0] ? Red : Blue, White);
149     cout << "PROCEED";
150     GoToXY(WIDTH_CONSOLE - 14, HEIGH_CONSOLE - HEIGH_CONSOLE + 12);
151     setOutputColor(selection2[1] ? Red : Blue, White);
152     cout << "RETURN";
153     setOutputColor(White, White);
154 }
```

- In ra bảng điều hướng sau khi đã nhập tên file xong(tương tự cơ chế của MainMenu())

3.4.5. void YesNoDisable()

```
156 void YesNoDisable() {
157     GoToXY(WIDTH_CONSOLE - 26, HEIGH_CONSOLE - HEIGH_CONSOLE + 12);
158     setOutputColor(White, White);
159     cout << "      ";
160     GoToXY(WIDTH_CONSOLE - 14, HEIGH_CONSOLE - HEIGH_CONSOLE + 12);
161     cout << "      ";
162 }
```

- Được sử dụng trong hàm Continue() để xoá YesNoScreen() nếu tải không thành công.

3.4.6. void datList()

```
165 void datList() {
166     int max;
167     string* b = loadRawDATA(max);
168     if (b != NULL) {
169         GoToXY(WIDTH_CONSOLE / 2 + 2, HEIGH_CONSOLE - HEIGH_CONSOLE + 1);
170         setOutputColor(Blue, White);
171         cout << "saved file's name: ";
172         setOutputColor(Green, White); int y = 0, x = 0;
173         for (int i = 0; i < max; i++) {
174             GoToXY(WIDTH_CONSOLE / 2 + 5 + x, HEIGH_CONSOLE - HEIGH_CONSOLE + 2 + y++);
175             cout << *(b + i);
176             if (y == 4) {
177                 x += 15; y = 0;
178             }
179         }
180         delete[] b;
181     }
182 }
```

- In ra màn hình những file đã lưu trước đó để người dùng lựa chọn. sử dụng thông tin lấy từ hàm loadRawData().

3.4.7. void Control 2(string a)

```
185 //save/load game's control
186 void Control2(string a) {
187     int Index = 0; selection2[0] = 1; selection2[1] = 0;
188     YesNoScreen();
189     while (1) {
190         int temp = toupper(_getch());
191         switch (temp) {
192             case 'A':
193                 AClosing(Index, selection2);
194                 break;
195             case 'D':
196                 DClosing(Index, selection2);
197                 break;
198             case 13:
199                 if (selection2[0]) {
200                     if (FileExist(a)) {
201                         FOUND = 1; LOAD = 1; file = a;
202                         Sleep(100);
203                         system("cls");
204                         thread t(game); t.join();
205                         system("cls");
206                     }
207                     else {
208                         GoToXY(WIDTH_CONSOLE / 2 + 3, HEIGH_CONSOLE / 2 - 2);
209                         setOutputColor(Red, White);
210                         cout << "find not found";
211                     }
212                     return;
213                 }
214                 else {
215                     FOUND = 1; return;
216                 }
217                 break;
218             }
219         YesNoScreen();
220     }
221 }
```

- Giống cơ chế điều hướng của hàm MainMenu(), Nếu khi không tìm thấy file sẽ in ra màn hình dòng chữ “find not found” và nhờ người chơi nhập lại tên. Ngược lại nếu kết quả là đúng thì tiến hành load game. Nếu chọn “RETURN” thì sẽ được đưa về màn hình chính.

3.4.8. void Control3(string a)

```

223 void Control3(string a) {
224     int Index = 0; selection2[0] = 1; selection2[1] = 0;
225     YesNoScreen();
226     while (1) {
227         int temp = toupper(_getch());
228         switch (temp) {
229             case 'A':
230                 AChosing(Index, selection2);
231                 break;
232             case 'D':
233                 DChosing(Index, selection2);
234                 break;
235             case 13:
236                 if (selection2[0]) {
237                     SaveGame(a);
238                     EXIT = 1;
239                     return;
240                 }
241                 else {
242                     return;
243                 }
244                 break;
245             }
246         YesNoScreen();
247     }
248 }
```

- Hàm này nhằm quản lý thao tác save game tương tự hàm Continue(), khi hiển thị thanh điều hướng nếu chọn PROCEED thì tiến hành save game vào file .txt , ngược lại sẽ thoát menu save game quay lại trò chơi.

3.4.9. void Continue()

```

250 //Save/Load menu's mechanic
251 void Continue() {
252     FOUND = 0;
253     DrawBoard(WIDTH_CONSOLE / 2 - 8, HEIGH_CONSOLE - HEIGH_CONSOLE, 50, 15, Red);
254     while (!FOUND) {
255         datList();
256         setOutputColor(White, Red);
257         string a;
258         GoToXY(WIDTH_CONSOLE / 2 - 3, HEIGH_CONSOLE / 2 - 1);
259         cout << "enter for next step, then A,D to navigate";
260         setOutputColor(Blue, White);
261         GoToXY(WIDTH_CONSOLE / 2 + 25, HEIGH_CONSOLE / 2 - 3);
262         cout << ".txt";
263         GoToXY(WIDTH_CONSOLE (const char [21]) " "
264         cout << " ";
265         GoToXY(WIDTH_CONSOLE / 2 - 5, HEIGH_CONSOLE / 2 - 3);
266         cout << "Enter name: "; getline(cin >> ws, a);
267         a += ".txt";
268         Control2(a);
269         YesNoDisable();
270     }
271 }
```

- Hàm sẽ chạy khi ta lựa chọn CONTINUE trong Main Menu, mục đích chính của hàm là hiển thị những chỉ dẫn người chơi nhập tên file và tiến hành load nó nếu tìm ra, nếu không thành công sẽ được để người dùng nhập lại tên

3.5. HÀM VÀ CÁC BIẾN TOÀN CỤC QUẢN LÝ TOÀN BỘ GAME

3.5.1. void game()

Đây là phần chơi của cả game, hàm Update() được đưa vào class thread để chạy độc lập cùng với game(), StartGame() sẽ khởi động các giá trị ảnh hưởng đến điều kiện thoát, chơi của game. ExitGame(std::thread &t) có hai lệnh là t.detach() và PLAY = 0, được dùng để đưa thread chạy ngầm sau đó tắt hàm thông qua biến PLAY, nếu không làm 1 trong hai lệnh trên đều có thể làm game bị lỗi ở các thao tác thoát. PauseGame() chỉ hiển thị “PAUSE” và xoá nó đi nếu bấm lại thông qua biến pauseflag đồng thời làm game buộc phải chờ người dùng bấm lại pause lần nữa mới chơi tiếp. SaveGame() sẽ lưu game và thoát (nếu thành công). Điều kiện qua màn sẽ được thực hiện nếu như rắn thỏa mãn điều kiện qua màn được cài ở hàm eat() (thực hiện ở các lệnh điều hướng trong Update())

```

605 void game() {
606     system("cls");
607     if (LOAD) //indicate wheater load file or first play
608     {
609         LoadGame(file);
610     }
611     else
612     {
613         setup();
614         printRound(LEVEL_STATE);
615         Sleep(3000);
616         StartGame();
617         std::thread t1(Update); //assign control to thread t1
618         HANDLE handle_t1 = t1.native_handle(); //handle thread t1
619         bool pauseflag = 0; PLAY = 1;
620         while (1)
621         {
622             if (EXIT) { //indicate file saved and exit
623                 ExitGame(t1); return;
624             }
625             if (STATE == 0)
626             {
627                 deadMessage();
628             }
629             //Sleep(150);
630             int temp = toupper(_getch());
631             if (STATE == 1)
632             {
633                 if (temp == CHAR_LOCK || temp == FORCE_LOCK[0] || temp == FORCE_LOCK[1])
634                     continue;
635                 if (temp == 'O')
636                     SuspendThread(handle_t1);
637                     Sleep(500);
638                     system("cls");
639                     SaveMenu();
640                     system("cls");
641                     DrawBoard(0, 0, WIDTH_CONSOLE, HEIGH_CONSOLE, Blue);
642                     ResumeThread(handle_t1);
643                     continue;
644             }
645             else
646             switch (temp)
647             {
648                 case 'P':
649                     pauseflag = (pauseflag ? 0 : 1);
650                     if (pauseflag)
651                         Pause(handle_t1);
652                     else
653                         GoToXY(WIDTH_CONSOLE / 2 - 2, HEIGH_CONSOLE / 2);
654                         setOutputColor(Black, White); cout << " ";
655                         ResumeThread(handle_t1);
656                     break;
657                 case 27:
658                     ExitGame(t1); return;
659                     break;
660             default:
661                 if (!pauseflag)
662                     ResumeThread(handle_t1);
663                     if (temp == 'D' || temp == 'A' || temp == 'W' || temp == 'S')
664                     {
665                         if (temp == 'D') CHAR_LOCK = 'A';
666                         else if (temp == 'W') CHAR_LOCK = 'S';
667                         else if (temp == 'S') CHAR_LOCK = 'W';
668                         else CHAR_LOCK = 'D';
669                         MOVING = temp;
670                     }
671                     // Through round process
672                     if (snake[0].x == snake[SIZE_SNAKE - 1].x && snake[0].y == snake[SIZE_SNAKE - 1].y && LEVEL_UP)
673                     {
674                         SuspendThread(handle_t1);
675                         lvUp();
676                         ResumeThread(handle_t1);
677                         MOVING = 'D';
678                     }
679                     if (GATE != NULL && !LEVEL_UP)
680                     {
681                         if (snake[0].x != GATE[4].x)
682                             {
683                                 deleteGate();
684                             }
685                     }
686             }
687         }
688     }
689     else
690     {
691         ExitGame(t1); return;
692     }
693 }

```

3.5.2. Global Variables

a) Khai báo ở header *snake.h*

```
29 //global value(variable)
30 extern int GATE_SIZE;
31 extern int WALL_SIZE;
32 extern POINT snake[MAX_SIZE_SNAKE]; //snake and it's body coordinate
33 extern POINT food[MAX_SIZE_FOOD]; //food coordinate
34 extern POINT* wall;
35 extern int MAX_SIZE_ROUND[];
36 extern int CHAR_LOCK, FORCE_LOCK[2];//unmoveable side
37 extern int MOVING;//define current side when moving
38 extern int SPEED;//speed, level
39 extern int HEIGH_CONSOLE, WIDTH_CONSOLE; //console size
40 extern int FOOD_INDEX;//currently showned up food
41 extern int SIZE_SNAKE; //snake size
42 extern int STATE; //snake status(die/alive)
43 extern int LEVEL_STATE;
44 extern long int SCORE; //score
45 extern bool LEVEL_UP;
46 extern POINT* GATE;
47 extern bool PLAY;
48 extern bool LOAD;
49 extern std::string file;
50 extern bool EXIT;
51 #endif
```

- Ở mục header ta thêm phần keyword `extern` trước mỗi biến để giải quyết lỗi linker bằng cách bắt linker tìm khai báo ở tệp .cpp thay vì hiểu nó là 1 biến.

b) Khai báo ở file *snake.cpp*(tệp chứa hàm main())

```
6 //global value(variable)
7 int GATE_SIZE;
8 int WALL_SIZE;
9 POINT snake[MAX_SIZE_SNAKE]; //snake and it's body coordinate
10 POINT food[MAX_SIZE_FOOD]; //food coordinate
11 int MAX_SIZE_ROUND[] = { 16, 32, 40 };
12 POINT* wall;
13 int CHAR_LOCK, FORCE_LOCK[2];//unmoveable side
14 int MOVING;//define current side when moving
15 int SPEED;//speed, level
16 int HEIGH_CONSOLE, WIDTH_CONSOLE; //console size
17 int FOOD_INDEX;//currently showned up food
18 int SIZE_SNAKE; //snake size
19 int STATE; //snake status(die/alive)
20 int LEVEL_STATE;
21 long int SCORE;
22 bool LEVEL_UP; //level up flag
23 POINT* GATE; //gate way
24 bool PLAY; //game status
25 bool LOAD;
26 std::string file;
27 bool EXIT;
```

4. TÀI LIỆU THAM KHẢO

Tài liệu “HƯỚNG DẪN ĐỒ ÁN RĂN SĂN MỎI”, giảng viên Trương Toàn Thịnh, ngày 20 tháng 03 năm 2020.

Website:

<http://stackoverflow.com/>

<https://www.cplusplus.com/>

<https://en.cppreference.com/w/>

<https://daynhauhoc.com/>