TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

--- ⁂ ---

ĐỒ ÁN CUỐI HỌC KÌ 2

Môn: Kỹ thuật lập trình

Lớp: 19CTT3

Năm học: 2019 – 2020

GAME RẮN SĂN MỒI

Nhóm 22:

Tạ Võ Anh Khuê – 19120551

Trần Tấn Lộc – 19120564

Nguyễn Minh Long – 19120568

Phạm Văn Nam – 19120597

Giáo viên hướng dẫn: *Thầy Trương Toàn Thịnh*

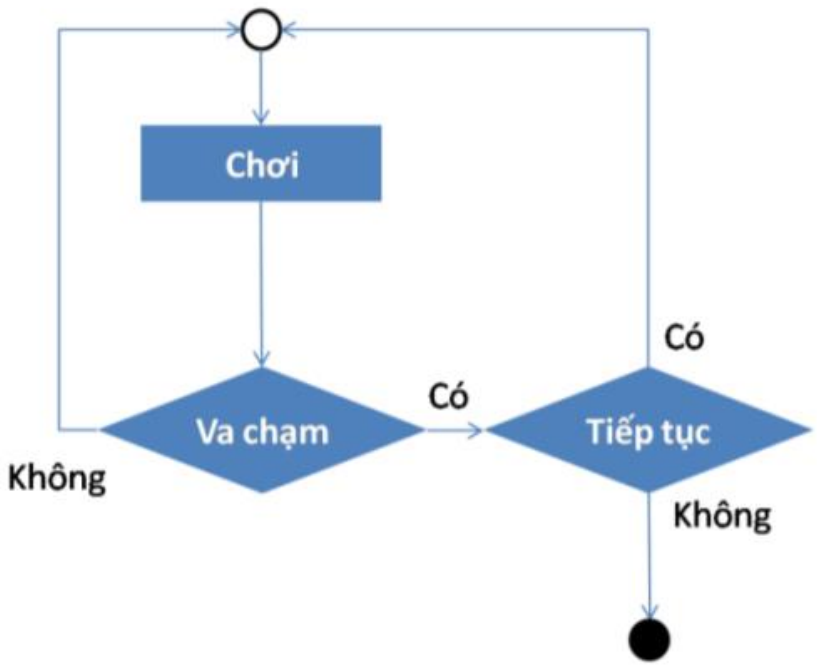
MỤC LỤC

I. Giới thiệu trò chơi

- Trò chơi với giao diện đơn giản, cách chơi quen thuộc như những game rắn săn mồi trước. Người chơi điều khiển rắn di chuyển để ăn thức ăn, khi ăn rắn dài ra và sau số lượng thức ăn nhất định thì tốc độ di chuyển của rắn sẽ tăng lên, khiến cho trò chơi khó hơn.

- Các chức năng chính của trò chơi:
    - Bắt đầu game
    - Lưu game
    - Load game đã chơi
    - Bảng hướng dẫn khi chơi

- Người chơi dùng các phím W,A,S,D đi di chuyển đi các hướng khi ăn đồ ăn. Khi ăn đủ số lượng thức ăn nhất định sẽ xuất hiện một cánh cổng để rắn chui vào, và khi chui ra khỏi cổng rắn sẽ tăng tốc độ di chuyển lên. Cứ thế tiếp tục cho đến khi qua hết các màn thì người chơi sẽ chiến thắng.



II. Quá trình xây dựng trò chơi

- Các công cụ, thông số, nhóm hàm chính đã sử dụng để tạo nên trò chơi gồm:

1) Các thông số, thư viện, biến toàn cục định nghĩa các dữ liệu cần thiết

```
1  #include <iostream>
2  #include <conio.h>
3  #include <Windows.h>
4  #include <time.h>
5  #include <stdlib.h>
6  #include <string.h>
7  #include <thread>
8  #include <fstream>
9  #include <malloc.h>
10 #define MAX_SIZE_SNAKE 33
11 #define MAX_SIZE_FOOD 4
12 #define MAX_SIZE_SPEED
13 using namespace std;
14 POINT* snakey; // con rắn
15 POINT muiten; // mũi tên ở menu
16 POINT* food = NULL; // thức ăn
17 POINT gate[6] = { 0 }; // cổng khi qua màn
18 int ROUND = 1; // màn chơi
19 int count_SIZE = 0; // cập nhật độ dài rắn khi vào cổng
20 int CHAR_LOCK; // khóa hướng di chueyern
21 int check = 1; // kiểm tra trạng thái ở main
22 int check_gate = 0; // kiểm tra có cổng nói chung
23 int check_outgate = 0; // kiểm tra cổng ra
24 int z = 1; // làm hiệu ứng chết
25 int MOVING; // xác định hướng đi của rắn
26 int SPEED; // tốc độ rắn
```

```
27  int HEIGH_CONSOLE = 20, WIDTH_CONSOLE = 70; // kích thước bảng chơi
28  int FOOD_INDEX; // chỉ số food hiện hành
29  int SIZE_SNAKE; // kích thước rắn
30  int STATE; // trạng thái sống / chết
31  char p[34] = "01912056419120597191205681912055 1"; // hình dạng thân rắn
32  char q[34] = "                                 "; // dùng để xóa rắn cũ
33  HANDLE consolehandle = GetStdHandle(STD_OUTPUT_HANDLE);
```

- Ý nghĩa các thông số đã có như chú thích, đây là các biến cần thiết, sử dụng xuyên suốt trong chương trình.

2) Các hàm liên quan đến giao diện console

```
1   void doimau(int color)// đổi màu chữ
2   {
3       SetConsoleTextAttribute(consolehandle, color);// chỉnh màu con trỏ trên console
4   }
5   void HideCursor() // xóa con trỏ xuất trên console
6   {
7       CONSOLE_CURSOR_INFO lpCursor;
8       lpCursor.bVisible = false;
9       lpCursor.dwSize = 20;
10      SetConsoleCursorInfo(consolehandle, &lpCursor);
11  }
12  void FixConsoleWindow()// xóa nút maximize và cố định console
13  {
14      HWND consoleWindow = GetConsoleWindow();
15      LONG style = GetWindowLong(consoleWindow, GWL_STYLE);
16      style = style & ~(WS_MAXIMIZEBOX) & ~(WS_THICKFRAME);
17      SetWindowLong(consoleWindow, GWL_STYLE, style);
18  }
19  void GotoXY(int x, int y)// di chuyển đến tọa độ (x, y) trên màn hình
20  {
21      COORD coord;
22      coord.X = x;
```

```
23      coord.Y = y;
24      SetConsoleCursorPosition(consolehandle, coord);
25  }
```

- **Hàm doimau** dùng để thay đổi màu sắc của con trỏ vẽ ký tự trên màn hình console.

- **Hàm HideCursor** dùng để xóa đi con trỏ vẽ ký tự trong giao diện console.

- **Hàm FixConsoleWindow** dùng để xóa đi nút maximize và cố định màn hình.

- **Hàm GotoXY** để di chuyển đến tọa độ (x,y) trong màn hình, hàm này được dùng xuyên suốt chương trình khi muốn vẽ đồ ăn, vẽ rắn, cổng,… hay nói chung là bất kỳ thao tác nào cần xuất ra màn hình của người chơi.

3) Các hàm kiểm tra hợp lệ

```
1   bool IsValid(int x, int y)// kiểm tra có trùng tọa độ thân rắn không
2   {
3       for (int i = 0; i < SIZE_SNAKE; i++)
4       {
5           if (i == 0)
6               continue;
7           if (snakey[i].x == x && snakey[i].y == y)
8               return false;
9       }
10      return true;
11  }
12  bool Isvalidgate(int x, int y)//kiem tra toa do co trung voi cong khong
13  {
14      for (int i = 0; i < 5; i++)
15          if (x == gate[i].x && y == gate[i].y)
16              return false;
17      return true;
18  }
```

- **Hàm IsValid** kiểm tra tọa độ truyền vào có bị trùng với con rắn không, thuật toán đơn giản chỉ là chạy vòng for cho kiểm tra từng tọa độ của rắn.

- **Hàm Isvalidgate** cũng tương tự, chỉ khác là kiểm tra tọa độ có bị trùng với cổng hay không thông qua duyệt vòng for.

4) Các hàm liên quan đến việc tạo dữ liệu và vẽ ra màn hình

```
1  void GenerateFood()// tao mang thuc an
2  {
3        int x, y;
4        srand(time(NULL));
5        for (int i = 0; i < MAX_SIZE_FOOD; i++) {
6              do {
7                    x = rand() % (WIDTH_CONSOLE - 1) + 2;
8                    y = rand() % (HEIGH_CONSOLE - 1) + 1;
9              } while (!IsValid(x, y));
10             food[i] = { x, y };
11       }
12 }
13 void generateGate()//tao cong de qua man
14 {
15       int x, y;
16       srand(time(NULL));
17       do {
18             x = rand() % ((WIDTH_CONSOLE - 5) - 5) + 5;
19             y = rand() % ((HEIGH_CONSOLE - 5) - 5) + 5;
20       } while (!IsValid(x, y));
21       gate[0] = { x - 1, y + 1 };
22       gate[1] = { x - 1, y };
23       gate[2] = { x, y };
24       gate[3] = { x + 1, y };
25       gate[4] = { x + 1, y + 1 };
26       gate[5] = { x, y + 1 };
27       for (int i = 0; i < 6; i++)
28       {
29
30             if (!IsValid(gate[i].x, gate[i].y))
31                   return generateGate();
32       }
```

```
}
void drawGate(POINT gate[])//ve cong vao
{
	doimau(245);
	for (int i = 0; i < 5; i++)
	{
		GotoXY(gate[i].x, gate[i].y);
		cout << "o";
	}
}
void drawOutGate(POINT gate[])//ve cong di ra
{
	doimau(245);
	for (int i = 0; i <= 5; i++)
	{
		if (i == 2) continue;
		GotoXY(gate[i].x, gate[i].y);
		cout << "o";
	}
}
void cleanGate(POINT gate[])//xoa cong
{
	for (int i = 0; i <= 5; i++)
	{
		GotoXY(gate[i].x, gate[i].y);
		cout << " ";
		gate[i].x = 0; gate[i].y = 0;
	}
}
void generateSnake() // tao ran
{
	POINT* snake_temp = (POINT*)realloc(snakey, (SIZE_SNAKE + 1) * sizeof(POINT));
	if (snake_temp != NULL)
	{
		snake_temp[SIZE_SNAKE].x = food[FOOD_INDEX].x;
		snake_temp[SIZE_SNAKE].y = food[FOOD_INDEX].y;
		SIZE_SNAKE++;
		snakey = snake_temp;
	}
}
void DrawSnakeAndFood(char str[], char str2) // ve ran va thuc an
{
	if (check_gate != 1)
```

```
76      {
77          GotoXY(food[FOOD_INDEX].x, food[FOOD_INDEX].y);
78          cout << str2;
79      }
80      for (int i = 0; i < SIZE_SNAKE; i++)
81      {
82          if (i == 0 && str[i] != ' ')
83          {
84              GotoXY(snakey[0].x, snakey[0].y);
85              cout << a;
86              continue;
87          }
88          GotoXY(snakey[i].x, snakey[i].y);
89          cout << str[i];
90      }
91  }
```

- **Hàm GenerateFood** và **hàm generateGate** sử dụng việc random số ngẫu nhiên để tạo ra những tọa độ, và lần lượt gán vào các biến toàn cục ở trên thông qua vòng for.

- Các **hàm drawGate**, **hàm drawOutGate**, chủ yếu dùng vòng for để duyệt và dùng hàm GotoXY để tới tọa độ từng phần trong cổng và thực hành động xuất ra chữ "o" vào các tọa độ để tạo thành cổng.

- **Hàm cleanGate** dùng GotoXY để xuất ra khoảng trắng trám vào chỗ của cổng đang hiện trên màn mình đồng thời reset lại dữ liệu của mảng cổng về 0.

- **Hàm generateSnake** dùng kỹ thuật con trỏ và mảng động để thiết lập tọa độ rắn.

- **Hàm DrawSnakeAndFood** dùng để vẽ rắn và thức ăn ra màn hình theo tọa độ đã được lưu.

5) Các hàm xử lý việc di chuyển khi đang chơi

```
1  void MoveRight()
2  {
3       int n = snakey[0].x;
4       int m = snakey[0].y;
5       if (n == WIDTH_CONSOLE || !IsValid(n + 1, m))
6       {
7            ProcessDead();
8            return;
9       }
10      if (n == food[FOOD_INDEX].x && m == food[FOOD_INDEX].y)
11           Eat();
12      if (!Isvalidgate(n + 1, m))
13      {
14           ProcessDead();
15           return;
16      }
17      if (check_outgate == 1)
18           ProcessOutGate();
19      for (int i = SIZE_SNAKE - 1; i > 0; i--)
20      {
21           snakey[i].x = snakey[i - 1].x;
22           snakey[i].y = snakey[i - 1].y;
23      }
24      snakey[0].x++;
25 }
26 void MoveLeft()
27 {
28      . . .
29 }
30 void MoveDown()
31 {
32      . . .
33 }
34 void MoveUp()
35 {
36      . . .
37           if (n == gate[2].x && m - 1 == gate[2].y)
38           {
39                ProcessInGate();
40           }
41           else if (check_outgate == 1)
42           {
43                if (SIZE_SNAKE == 1)
44                {
```

```
45                          generateGate();
46                          drawOutGate(gate);
47                          ROUND++;
48                      }
49                      ProcessOutGate();
50          . . .
51  }
```

- Các **hàm MoveRight, MoveLeft, MoveUp, MoveDown** tương tự nhau trong cách xử lí kiểm tra điều kiện va chạm, xử lí ăn thức ăn, xử lí ra khỏi cổng và xử lí dời tọa độ của snakey, nhưng riêng hàm MoveUp, được xử lí thêm phần đi vào cổng và bắt đầu đi ra khỏi cổng.

- Khi đầu rắn (snakey[0]) có cùng tọa độ với vị trí ô trắng của cổng, ta sẽ chạy hàm ProcessIngate (ở mục 6).

- Khi chạy xong hàm **ProcessIngate**, tức là SIZE_SNAKE còn bằng 0 thì biến check cổng ra check_outgate được gán bằng 1 và chương trình tiếp tục chay qua hàm ProcessOutGate.

6) Các hàm xử lý thao tác trong game

```
1   void Eat()
2   {
3       generateSnake();
4       if (SIZE_SNAKE == 32)
5       {
6           ProcessWin();
7           return;
8       }
9       if (FOOD_INDEX == MAX_SIZE_FOOD - 1)
10      {
11          FOOD_INDEX = 0;
12          check_gate = 1;
13          generateGate();
14          drawGate(gate);
15          GenerateFood();
16      }
17      else FOOD_INDEX++;
18  }
19  void ProcessWin()
```

```
{
	char a = 1, b = 31, c = 16, d = 17, e = 30;
	STATE = 0;
	Sleep(100);
	. . . // vẽ bảng “YOU WIN”
}
void ProcessDead()
{
	char a = 1, b = 31, c = 16, d = 17, e = 30;
	STATE = 2;
	z = 1;
	. . . // vẽ bảng “YOU LOSE”
}
void ProcessInGate()
{
	for (int i = 0; i < SIZE_SNAKE - 1; i++)
	{
		snakey[i].x = snakey[i + 1].x;
		snakey[i].y = snakey[i + 1].y;
	}
	SIZE_SNAKE--;
	count_SIZE++;
	if (SIZE_SNAKE == 0)
	{
		cleanGate(gate);
		check_outgate = 1;
		SIZE_SNAKE = 1;
	}
}
void ProcessOutGate()
{
	for (int i = SIZE_SNAKE - 1; i < SIZE_SNAKE; i++)
	{
		snakey[i].x = gate[5].x;
		snakey[i].y = gate[5].y - 1;
		count_SIZE--;
		if (count_SIZE == 0)
		{
			SIZE_SNAKE--;
			cleanGate(gate);
			check_outgate = 0;
			check_gate = 0;
			SPEED += 3;
			break;
```

```
64              }
65          }
66          SIZE_SNAKE++;
67      }
68      void ThreadFunc()
69      {
70          while (1)
71          {
72              if (STATE == 1)
73              {
74                  DrawSnakeAndFood(q, ' ');
75                  switch (MOVING)
76                  {
77                  case 'A':
78                  {
79                      MoveLeft();
80                      break;
81                  }
82                  case 'D':
83                  {
84                      MoveRight();
85                      break;
86                  }
87                  case 'W':
88                  {
89                      MoveUp();
90                      break;
91                  }
92                  case 'S':
93                  {
94                      MoveDown();
95                      break;
96                  }
97                  doimau(247);
98                  DrawSnakeAndFood(p, p[SIZE_SNAKE]);
99                  GotoXY(40, HEIGH_CONSOLE + 2);
100                 if (check_gate != 0)  // vẽ số thức ăn còn lại
101                     cout << 0;
102                 else if (ROUND == 7 && STATE == 1)
103                 {
104                     cout << 3 - FOOD_INDEX;
105                 }
106                 else if (STATE == 1)
107                     cout << MAX_SIZE_FOOD - FOOD_INDEX;
```

```
108                 cout << ROUND;              // vẽ màn hiện tại
109                 Sleep(1000 / SPEED);// 'tốc độ rắn' khi chạy chương trình
110             }
111             else if (STATE == 2)
112             {
113                 . . . // vẽ hiệu ứng chết
114             }
115         }
116 }
```

- **Hàm Eat** thực hiện khi tọa độ snakey[0] trùng tọa độ của food[FOOD_INDEX]. Trong đó hàm generateSnake() được gọi để cấp phát thêm một ô nhớ cho con trỏ snakey đồng thời tăng SIZE_SNAKE thêm 1. Sau đó kiểm tra điều kiện về độ dài để thực hiện ProcessWin hay không, nếu không thỏa thì tiếp tục kiểm tra FOOD_INDEX đã đạt cao nhất chưa. Nếu có thì FOOD_INDEX sẽ về 0, sau đó gán biến check_gate = 1, tức là cổng đang hiện, để xử lí ngừng tạo thức ăn ở trong hàm DrawSnakeAndFood nhưng vẫn cấp tọa độ cho thức ăn tiếp theo và thực hiện cấp tọa độ và vẽ 'cổng vào'. Nếu chưa đạt cao nhất thì tăng FOOD_INDEX thêm 1.

- **Hàm ProcessWin** và ProcessDead thực hiện gán STATE = 0 để dừng vẽ rắn và sau đó xuất ra bảng "YOU WIN" hay "YOU LOSE".

- **Hàm ProcessInGate** thực hiện gán tọa độ snakey[i] cho snakey[i+1] nhằm cố định tọa độ rắn nhưng vẫn tiếp tục ThreadFunc() để vẽ rắn. Mỗi khi ProcessIngate được thực thi (vì snakey[0] liên tục trùng tọa độ food[FOOD_INDEX]) thì SIZE_SNAKE được giảm để giảm số lượng kích thước cần vẽ, đồng thời thực hiện tang biến count_SIZE để lưu lại kích thước hiện tại của rắn. Khi SIZE_SNAKE == 0 thì sẽ xóa 'cổng vào' bằng hàm cleanGate, rồi gán check_outgate = 1 và SIZE_SNAKE = 1 để sau đó thực hiện vẽ 'cổng ra' và thực thi hàm ProcessOutGate.
- **Hàm ProcessOutGate** lần lượt gán tọa độ một chiều dài thân vào vị trí cố định (ô trắng của 'cổng ra') qua mỗi lần thực thi hàm. Sau đó trừ count_SIZE và tăng SIZE_SNAKE ngoài vòng for (dòng 66) để trả về chiều dài rắn. Trong đó, khi count_SIZE về 0 thì trừ một lần kích thước

rắn để tránh tang dữ độ dài rắn sau đó thực hiện xóa cổng và gán các biến kiểm tra cổng về 0 để tiếp tục xuất food rồi tăng tốc độ rắn lên.

- **Hàm ThreadFunc** chạy trong luồng thread t1 liên tục nhận giá trị từ bàn phím để thực hiện các hàm hướng di chuyển, xóa rắn cũ (gán rắn cũ bằng khoảng trắng), vẽ tuần tự rắn mới, vẽ số màn, số thức ăn và hiệu ứng chết.

7) Các hàm chức năng của game

```
1  void StartGame()
2  {
3        system("cls");
4        ResetData();
5        DrawBoard(0, 0, WIDTH_CONSOLE, HEIGH_CONSOLE);
6        STATE = 1;//để kích hoạt vẽ rắn
7  }
8  void ExitGame(HANDLE t)
9  {
10       if (snakey != NULL)
11            free(snakey);
12       if (food != NULL)
13            free(food);
14       snakey = NULL;
15       food = NULL;
16       system("cls");
17       TerminateThread(t, 0);
18 }
19 void PauseGame(HANDLE t)
20 {
21       SuspendThread(t);
22 }
23 void SaveGame()
24 {
25             . . .
26       char* a = (char*)malloc(100); memset(a, '\0', 100);
27       GotoXY(50, 11);
28       cout << "                                ";
29       GotoXY(50, 11);
30       cin.getline(a, 100);
```

```
31      fstream fo(a, ios::out);
32      fo << SPEED << endl;
33      fo << FOOD_INDEX << endl;
34      fo << SIZE_SNAKE << endl;
35      fo << count_SIZE << endl;
36      fo << CHAR_LOCK << endl;
37      fo << MOVING << endl;
38      for (int i = 0; i < SIZE_SNAKE; i++)
39      {
40          fo << snakey[i].x << endl;
41          fo << snakey[i].y << endl;
42      }
43      for (int i = 0; i < MAX_SIZE_FOOD; i++)
44      {
45          fo << food[i].x << endl;
46          fo << food[i].y << endl;
47      }
48      fo << ROUND << endl;
49      for (int i = 0; i <= 5; i++)
50      {
51          fo << gate[i].x << endl;
52          fo << gate[i].y << endl;
53      }
54      fo << check_gate << endl;
55      fo << check_outgate << endl;
56      fo.close();
57      free(a);
58      GotoXY(50, 12);
59      cout << "luu thanh cong!";
60      int temp4 = toupper(_getch());
61  }
62  void LoadGame()
63  {
64      doimau(242);
65      GotoXY(50, 10);
66      cout << "                                ";
67      GotoXY(50, 10);
68      cout << "nhap ten file muon mo";
69      if (food == NULL)
70          food = (POINT*)malloc(MAX_SIZE_FOOD * sizeof(POINT));
71      char* a = (char*)malloc(100); memset(a, '\0', 100);
72      GotoXY(50, 11);
73      cout << "                                ";
```

```
74      GotoXY(50, 11);
75      cin.getline(a, 100);
76      fstream fi(a, ios::in);
77      if (!fi)
78      {
79            GotoXY(50, 12);
80            cout << "file khong ton tai";
81            check = 3;// check = 3 để báo không load được file
82            int temp3 = toupper(_getch());
83            system("cls");
84      }
85      else
86      {
87            fi >> SPEED >> FOOD_INDEX >> SIZE_SNAKE >> count_SIZE
      >> CHAR_LOCK >> MOVING;
88            POINT*   snakynew   =   (POINT*)malloc((SIZE_SNAKE   +
      count_SIZE) * sizeof(POINT));
89            if (snakynew != NULL)
90            {
91                  if (snakey != NULL)
92                        free(snakey);
93                  snakey = snakynew;
94            }
95            for (int i = 0; i < SIZE_SNAKE; i++)
96                  fi >> snakey[i].x >> snakey[i].y;
97            for (int i = 0; i < MAX_SIZE_FOOD; i++)
98                  fi >> food[i].x >> food[i].y;
99      }
100     fi >> ROUND;
101     for (int i = 0; i <= 5; i++)
102           fi >> gate[i].x >> gate[i].y;
103     fi >> check_gate >> check_outgate;
104     fi.close();
105     free(a);
106 }
```

- **Hàm StartGame** sẽ thực hiện xóa màn hình menu rồi cấp lại các biến toàn cục về cố định sau đó vẽ màn hình chơi rồi gán STATE = 1 để bắt đầu vẽ rắn

- **Hàm ExitGame** lần lượt kiểm tra các con trỏ được cấp phát để xóa và gán lại bằng NULL để xử lí các điều kiện ở hàm ResetData hay LoadGame(). Sau đó sẽ xóa màn hình và thoát luồng bằng hàm TerminateThread.

- **Hàm PauseGame** dùng để dừng thread và tiếp tục thực thi trong main.

- **Hàm SaveGame** thực hiện lưu một số biến toàn cục và tọa độ của rắn, food và cổng.

- **Hàm LoadGame** thực hiện load file đã lưu và cấp phát lại bộ nhớ theo kích thước đã lưu của các biến con trỏ.

8) Các hàm liên quan việc xử lý menu và hiển thị thông tin

```
1  void DrawBoard(int x, int y, int width, int height, int curPosX = 0, int curPosY = 0)//vẽ bảng chơi, hướng dẫn, các thông số...
2  {
3        char k = 205, a = 186, b = 187, c = 188, d = 201, e = 200;
4        doimau(241);
5        for (int i = 0; i <= width + 2; i++)
6        {
7              Sleep(10);//làm chậm để tạo hiệu ứng
8              GotoXY(x + i, y);
9              cout << (char)219;
10             GotoXY(x + i, height + y);
11             cout << (char)219;
12       }
13       for (int i = y + 1; i < height + y; i++)
14       {
15             Sleep(20);//lam cham lai de co hieu ung ve
16             GotoXY(x, i);
17             cout << (char)219 << (char)219;
18             GotoXY(x + width + 1, i);
19             cout << (char)219 << (char)219;
20       }
21       GotoXY(x + width + 20, y + 1);
```

```
22      cout << "Huong dan";
23      GotoXY(x + width + 5, y + 2);
24      cout << "'W': Di Len";
25      GotoXY(x + width + 5, y + 4);
26      cout << "'A': Qua Phai";
27      GotoXY(x + width + 5, y + 6);
28      cout << "'S': Di xuong";
29      GotoXY(x + width + 5, y + 8);
30      cout << "'D': Qua Phai";
31      GotoXY(x + width + 5, y + 10);
32      cout << "(P): Pause game";
33      GotoXY(x + width + 5, y + 12);
34      cout << "'Bam nut bat ki de tiep tuc'";
35      GotoXY(x + width + 5, y + 14);
36      cout << "(L): Luu game";
37      GotoXY(x + width + 5, y + 16);
38      cout << "(T): load game";
39      GotoXY(WIDTH_CONSOLE + 4, 0);
40            cout << d;
41      GotoXY(WIDTH_CONSOLE + 45, 0);
42            cout << b;
43      GotoXY(WIDTH_CONSOLE + 4, 18);
44            cout << e;
45      GotoXY(WIDTH_CONSOLE + 45, 18);
46            cout << c;
47      for (int i = 0; i < 40; i++)
48      {
49            GotoXY(WIDTH_CONSOLE + 5 + i, 0);
50            cout << k;
51            GotoXY(WIDTH_CONSOLE + 5 + i, 18);
52            cout << k;
53      }
54      for (int i = 0; i < 17; i++)
55      {
56            GotoXY(WIDTH_CONSOLE + 4, 1 + i);
57            cout << a;
58            GotoXY(WIDTH_CONSOLE + 45, 1 + i);
59            cout << a;
60      }
61      GotoXY(curPosX, curPosY);
62      GotoXY(15, HEIGH_CONSOLE + 2); cout << "thuc an con lai la:
";
63      GotoXY(15, HEIGH_CONSOLE + 4); cout << "man: ";
```

```
64        GotoXY(11, HEIGH_CONSOLE + 1);
65        cout << d;
66        GotoXY(61, HEIGH_CONSOLE + 1);
67        cout << b;
68        for (int i = 1; i < 50; i++)
69        {
70              GotoXY(11 + i, HEIGH_CONSOLE + 1);
71              cout << k;
72              GotoXY(11 + i, HEIGH_CONSOLE + 5);
73              cout << k;
74        }
75        for (int i = 2; i < 5; i++)
76        {
77              GotoXY(11, i + HEIGH_CONSOLE);
78              cout << a;
79              GotoXY(61, i + HEIGH_CONSOLE);
80              cout << a;
81        }
82        GotoXY(11, HEIGH_CONSOLE + 5);
83        cout << e;
84        GotoXY(61, HEIGH_CONSOLE + 5);
85        cout << c;
86  }
87  void menu()
88  {
89        int k;
90        muiten.x = 42;
91        muiten.y = 20;
92        while (1)
93        {
94              if (check == 1)
95              {
96                    do{
                            k = rand() % 14;
                      }while(k == 7 && k == 8 && k == 15);
97                    doimau(k + 240);
98                    Sleep(150);
99                    if (check != 1)
100                         continue;
                      . . . //vẽ hình “SNAKE GAME”
128                   GotoXY(muiten.x, muiten.y);
129                   doimau(250);
130                   cout << "-->";
```

```
131                     GotoXY(55, 20);
132                     cout << "__START GAME__";
133                     GotoXY(55, 22);
134                     cout << "__OPEN FILE__";
135                     GotoXY(55, 24);
136                     cout << "__EXIT GAME__";
137             }
138     }
139     return;
140 }
```

- **Hàm DrawBoard** gồm 3 chức năng chính là: vẽ tường khi chơi, vẽ hướng dẫn cho người chơi và vẽ các thông số của game như thức ăn còn lại hay màn. Trong hàm này, chúng ta cũng sử dụng các vòng lặp for và hàm GotoXY để in các kí tự trong bảng mã ASCII để vẽ khung. Việc gọi hàm Sleep ở dòng 7 và 15 để tạo hiệu ứng vẽ khung lúc chơi bằng cách làm chậm tốc độ vẽ của vòng lặp for.

- **Hàm menu** chứa một vòng lặp while có tác dụng để vẽ menu chính của game. Việc vẽ menu cũng dựa trên các kí tự trong bảng mã ASCII kết hợp với vòng lặp for và hàm GotoXY. Khi chạy chương trình, hàm menu sẽ được phân luồng trong hàm main tương tự như hàm Threadfunc. Giá trị của biến k có được thông qua hàm rand để lấy ngẫu nhiên giá trị từ 1 đến 14 biểu diễn cho nhiều màu khác nhau và mỗi vòng while sẽ tương ứng với 1 màu tạo nên hiệu ứng đổi màu của chữ SNAKE GAME trong menu. Hàm Sleep để làm chậm tốc độ đổi màu giúp người chơi không cảm thấy rối mắt. Khi người dùng nhập các phím di chuyển sẽ được bắt phím vào biến temp2 trong hàm main và hàm menu sẽ sử dụng hàm GotoXY để xóa các mũi tên cũ mà vẽ mũi tên mới.

9) Hàm main

```
1   int main()
2   {
3         int temp{};
4         int temp2;
5         FixConsoleWindow();
6         HideCursor();
7         STATE = 0;
8         thread t2(menu);
9         HANDLE handle_t2 = t2.native_handle();
10        thread t1(ThreadFunc);
11        HANDLE handle_t1 = t1.native_handle();
12        while (1)
13        {
14              if (check == 100)//check = 100 để chạy con rắn
15              {
16                    temp = toupper(_getch());
17                    if (STATE == 2)
18                          STATE = 0;
19              }
20              if (check == 1)//check = 1 chạy menu;
21              {
22                    while (1)
23                    {
24                          temp2 = toupper(_getch());
25                          if (temp2 == 'S' && muiten.y < 24)
26                                muiten.y += 2;
27                          else if (temp2 == 'S' && muiten.y == 24)
28                                muiten.y = 20;
29                          else if (temp2 == 'W' && muiten.y > 20)
30                                muiten.y -= 2;
31                          else if (temp2 == 'W' && muiten.y == 20)
32                                muiten.y = 24;
33                          else if (temp2 == 13 && muiten.y == 20)
34                          {
35                                check = 0;
36                                system("cls");
37                                break;
38                          }
```

```
39                          else if (temp2 == 13 && muiten.y == 24)
40                          {
41                                ExitGame(handle_t2);
42                                ExitGame(handle_t1);
43                                t2.join();
44                                t1.join();
45                                return 0;
46                          }
47                          else if (temp2 == 13 && muiten.y == 22)
48                          {
49                                check = 0;
50                                system("cls");
51                                LoadGame();
52                                if (check != 3)
53                                      check = 2;
54                                temp = 0;
55                                break;
56                          }
57                    }
58              }
59              if (check == 2)//xử lí sau khi load file
60              {
61                    system("cls");
62                    DrawBoard(0, 0, WIDTH_CONSOLE, HEIGH_CONSOLE);
63                    STATE = 1;
64                    PauseGame(handle_t1);
65                    DrawSnakeAndFood(p, p[SIZE_SNAKE]);
66                    if (check_outgate == 1)
67                          drawOutGate(gate);
68                    else if (check_gate == 1)
69                          drawGate(gate);
70                    check = 100;
71                    temp = toupper(_getch());
72              }
73              if (check == 0)//xử lí khi tắt menu
74              {
75                    StartGame();
76                    check = 100;
77              }
78              if (STATE == 1)
79                    if (temp == 'P')
80                          PauseGame(handle_t1);
81                    else if (temp == 27)
```

```
82                  {
83                        ExitGame(handle_t2);
84                        ExitGame(handle_t1);
85                        t1.join();
86                        t2.join();
87                        system("cls");
88                        return 0;
89                  }
90                  else if (temp == 'L')
91                  {
92                        PauseGame(handle_t1);
93                        SaveGame();
94                        system("cls");
95                        temp = ' ';
96                        check = 1;
97                        continue;
98                  }
99                  else if (temp == 'T')
100                 {
101                       STATE = 0;
102                       system("cls");
103                       LoadGame();
104                       if (check != 3)
105                             check = 2;
106                 }
107                 else
108                 {
109                       ResumeThread(handle_t1);
110                       if ((temp != CHAR_LOCK) && (temp == 'D' ||
    temp == 'A' || temp == 'W' || temp == 'S'))
111                       {
112                             if (temp == 'D')
113                                   CHAR_LOCK = 'A';
114                             else if (temp == 'W')
115                                   CHAR_LOCK = 'S';
116                             else if (temp == 'S')
117                                   CHAR_LOCK = 'W';
118                             else
119                                   CHAR_LOCK = 'D';
120                             MOVING = temp;
121                       }
122                 }
123           else
```

```
124             {
125                     if (temp == 'Y')
126                     {
127                             Sleep(250);
128                             doimau(241);
129                             StartGame();
130                     }
131                     else
132                     {
133                             Sleep(200);
134                             doimau(241);
135                             system("cls");
136                             check = 1;
137                     }
138             }
139     }
140 }
```

- **Hàm main** sẽ tổng hợp các hàm lại và chạy chương trình. Đầu tiên, chúng ta sẽ gọi hàm FixConsoleWindow() và HideCursor() để cố định màn hình chơi và xóa con trỏ vẽ kí tự.
- Tiếp theo, 2 thread là **Threadfunc** và **menu** sẽ được khởi động bằng cách tạo 2 biến kiểu thread là t1 và t2 ứng với Threadfunc và menu. Sau khi phân luồng cho các hàm trên, chúng ta sẽ chạy vòng lặp while. Vòng lặp while này có chức năng bắt phím di chuyển khi người chơi điều khiển con rắn và xử lí chạy lại menu sau khi thua hoặc lưu **FILE.**
- Biến toàn cục **check** nhằm kiểm soát tiến trình của game trong quá trình chơi bằng cách điều khiển thread t2:

Check = 0: Start Game
Check = 1:chạy menu
Check = 2: load file mới
Check = 3: không load được FILE (do FILE không tồn tại hoặc tên không hợp lệ)
Check = 100: đang chơi

- Tương tự như biến check, biến **STATE** sẽ xử lí các trạng thái của con rắn bằng cách điều khiển thread t1:

STATE = 0: tạm dừng việc in con rắn
STATE = 1: in con rắn

STATE = 2: chạy hiệu ứng chết cho con rắn

III. Lời kết

- Quá trình thực hiện đồ án có chút khó khăn, do nhóm lúc thực hiện cũng chưa nắm rõ hoàn toàn hướng đi cũng như chưa có đủ kiến thức để hoàn thiện.

- Tuy nhiên nhóm đã cố gắng, nỗ lực, tìm tòi thêm qua sách, internet để bổ sung kiến thức. Nhóm đã sắp xếp thời gian, mỗi tuần dành thời cho buổi họp có tất cả thành viên, cùng thảo luận và thực hiện đồ án.

- Trong lúc thực hiện đồ án, nhóm đã cố gắng bám sát yêu cầu được đưa ra, tuy không có nhiều cải tiến, đột phá cho trò chơi, nhưng nhóm tự tin đã đạt được yêu cầu được đưa ra, những yêu cầu cơ bản nhất của một trò Rắn săn mồi.

- Qua việc thực hiện đồ án lần này, nhóm đã rút ra được nhiều bài học kinh nghiệm về tổ chức nhóm, làm việc nhóm sao cho hiệu quả. Và nhóm sẽ cố gắng khắc phục những điểm yếu trong khâu làm việc nhóm cho những môn học sau.

- Xin chân thành cảm ơn thầy Trương Toàn Thịnh đã hướng dẫn, chỉ bảo những bước cơ bản nhất để xây dựng nên trò chơi để chúng em hoàn thành đồ án lần này.

IV. Các nguồn tham khảo

- File pdf hướng dẫn đồ án của thầy Trương Toàn Thịnh.

- Những nguồn tài nguyên trên internet, Stack Overflow, những nội dung tham khảo khi chúng em bị bế tắc trong 1 vấn đề nào đó.