#include <stdio.h>

#include <Windows.h>

#include <stdlib.h>

#include <time.h>

#include <conio.h>

#define ROW 22 //游戏区行数

#define COL 42 //游戏区列数

#define KONG 0 //标记空（什么也没有）

#define WALL 1 //标记墙

#define FOOD 2 //标记食物

#define HEAD 3 //标记蛇头

#define BODY 4 //标记蛇身

#define UP 72 //方向键：上

#define DOWN 80 //方向键：下

#define LEFT 75 //方向键：左

#define RIGHT 77 //方向键：右

#define SPACE 32 //暂停

#define ESC 27 //退出

//蛇头

struct Snake

{

      int len; //记录蛇身长度

      int x; //蛇头横坐标

      int y; //蛇头纵坐标

}snake;

//蛇身

struct Body

{

      int x; //蛇身横坐标

      int y; //蛇身纵坐标

}body[ROW\*COL]; //开辟足以存储蛇身的结构体数组

int face[ROW][COL]; //标记游戏区各个位置的状态

//隐藏光标

void HideCursor();

//光标跳转

void CursorJump(int x, int y);

//初始化界面

void InitInterface();

//颜色设置

void color(int c);

//从文件读取最高分

void ReadGrade();

//更新最高分到文件

void WriteGrade();

//初始化蛇

void InitSnake();

//随机生成食物

void RandFood();

//判断得分与结束

void JudgeFunc(int x, int y);

//打印蛇与覆盖蛇

void DrawSnake(int flag);

//移动蛇

void MoveSnake(int x, int y);

//执行按键

void run(int x, int y);

//游戏主体逻辑函数

void Game();

int max, grade; //全局变量

int main()

{

#pragma warning (disable:4996) //消除警告

      max = 0, grade = 0; //初始化变量

      system("title 贪吃蛇"); //设置cmd窗口的名字

      system("mode con cols=84 lines=23"); //设置cmd窗口的大小

      HideCursor(); //隐藏光标

      ReadGrade(); //从文件读取最高分到max变量

      InitInterface(); //初始化界面

      InitSnake(); //初始化蛇

      srand((unsigned int)time(NULL)); //设置随机数生成起点

      RandFood(); //随机生成食物

      DrawSnake(1); //打印蛇

      Game(); //开始游戏

      return 0;

}

//隐藏光标

void HideCursor()

{

      CONSOLE\_CURSOR\_INFO curInfo; //定义光标信息的结构体变量

      curInfo.dwSize = 1; //如果没赋值的话，光标隐藏无效

      curInfo.bVisible = FALSE; //将光标设置为不可见

      HANDLE handle = GetStdHandle(STD\_OUTPUT\_HANDLE); //获取控制台句柄

      SetConsoleCursorInfo(handle, &curInfo); //设置光标信息

}

//光标跳转

void CursorJump(int x, int y)

{

      COORD pos; //定义光标位置的结构体变量

      pos.X = x; //横坐标

      pos.Y = y; //纵坐标

      HANDLE handle = GetStdHandle(STD\_OUTPUT\_HANDLE); //获取控制台句柄

      SetConsoleCursorPosition(handle, pos); //设置光标位置

}

//初始化界面

void InitInterface()

{

      color(6); //颜色设置为土黄色

      for (int i = 0; i < ROW; i++)

      {

            for (int j = 0; j < COL; j++)

            {

                  if (j == 0 || j == COL - 1)

                  {

                        face[i][j] = WALL; //标记该位置为墙

                        CursorJump(2 \* j, i);

                        printf("■");

                  }

                  else if (i == 0 || i == ROW - 1)

                  {

                        face[i][j] = WALL; //标记该位置为墙

                        printf("■");

                  }

                  else

                  {

                        face[i][j] = KONG; //标记该位置为空

                  }

            }

      }

      color(7); //颜色设置为白色

      CursorJump(0, ROW);

      printf("当前得分:%d", grade);

      CursorJump(COL, ROW);

      printf("历史最高得分:%d", max);

}

//颜色设置

void color(int c)

{

      SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), c); //颜色设置

      //注：SetConsoleTextAttribute是一个API（应用程序编程接口）

}

//从文件读取最高分

void ReadGrade()

{

      FILE\* pf = fopen("贪吃蛇最高得分记录.txt", "r"); //以只读的方式打开文件

      if (pf == NULL) //打开文件失败

      {

            pf = fopen("贪吃蛇最高得分记录.txt", "w"); //以只写的方式打开文件

            fwrite(&max, sizeof(int), 1, pf); //将max写入文件（此时max为0），即将最高得分初始化为0

      }

      fseek(pf, 0, SEEK\_SET); //使文件指针pf指向文件开头

      fread(&max, sizeof(int), 1, pf); //读取文件当中的最高得分到max当中

      fclose(pf); //关闭文件

      pf = NULL; //文件指针及时置空

}

//更新最高分到文件

void WriteGrade()

{

      FILE\* pf = fopen("贪吃蛇最高得分记录.txt", "w"); //以只写的方式打开文件

      if (pf == NULL) //打开文件失败

      {

            printf("保存最高得分记录失败\n");

            exit(0);

      }

      fwrite(&grade, sizeof(int), 1, pf); //将本局游戏得分写入文件当中

      fclose(pf); //关闭文件

      pf = NULL; //文件指针及时置空

}

//初始化蛇

void InitSnake()

{

      snake.len = 2; //蛇的身体长度初始化为2

      snake.x = COL / 2; //蛇头位置的横坐标

      snake.y = ROW / 2; //蛇头位置的纵坐标

      //蛇身坐标的初始化

      body[0].x = COL / 2 - 1;

      body[0].y = ROW / 2;

      body[1].x = COL / 2 - 2;

      body[1].y = ROW / 2;

      //将蛇头和蛇身位置进行标记

      face[snake.y][snake.x] = HEAD;

      face[body[0].y][body[0].x] = BODY;

      face[body[1].y][body[1].x] = BODY;

}

//随机生成食物

void RandFood()

{

      int i, j;

      do

      {

            //随机生成食物的横纵坐标

            i = rand() % ROW;

            j = rand() % COL;

      } while (face[i][j] != KONG); //确保生成食物的位置为空，若不为空则重新生成

      face[i][j] = FOOD; //将食物位置进行标记

      color(12); //颜色设置为红色

      CursorJump(2 \* j, i); //光标跳转到生成的随机位置处

      printf("●"); //打印食物

}

//判断得分与结束

void JudgeFunc(int x, int y)

{

      //若蛇头即将到达的位置是食物，则得分

      if (face[snake.y + y][snake.x + x] == FOOD)

      {

            snake.len++; //蛇身加长

            grade += 10; //更新当前得分

            color(7); //颜色设置为白色

            CursorJump(0, ROW);

            printf("当前得分:%d", grade); //重新打印当前得分

            RandFood(); //重新随机生成食物

      }

      //若蛇头即将到达的位置是墙或者蛇身，则游戏结束

      else if (face[snake.y + y][snake.x + x] == WALL || face[snake.y + y][snake.x + x] == BODY)

      {

            Sleep(1000); //留给玩家反应时间

            system("cls"); //清空屏幕

            color(7); //颜色设置为白色

            CursorJump(2 \* (COL / 3), ROW / 2 - 3);

            if (grade > max)

            {

                  printf("恭喜你打破最高记录，最高记录更新为%d", grade);

                  WriteGrade();

            }

            else if (grade == max)

            {

                  printf("与最高记录持平，加油再创佳绩", grade);

            }

            else

            {

                  printf("请继续加油，当前与最高记录相差%d", max - grade);

            }

            CursorJump(2 \* (COL / 3), ROW / 2);

            printf("GAME OVER");

            while (1) //询问玩家是否再来一局

            {

                  char ch;

                  CursorJump(2 \* (COL / 3), ROW / 2 + 3);

                  printf("再来一局?(y/n):");

                  scanf("%c", &ch);

                  if (ch == 'y' || ch == 'Y')

                  {

                        system("cls");

                        main();

                  }

                  else if (ch == 'n' || ch == 'N')

                  {

                        CursorJump(2 \* (COL / 3), ROW / 2 + 5);

                        exit(0);

                  }

                  else

                  {

                        CursorJump(2 \* (COL / 3), ROW / 2 + 5);

                        printf("选择错误，请再次选择");

                  }

            }

      }

}

//打印蛇与覆盖蛇

void DrawSnake(int flag)

{

      if (flag == 1) //打印蛇

      {

            color(10); //颜色设置为绿色

            CursorJump(2 \* snake.x, snake.y);

            printf("■"); //打印蛇头

            for (int i = 0; i < snake.len; i++)

            {

                  CursorJump(2 \* body[i].x, body[i].y);

                  printf("□"); //打印蛇身

            }

      }

      else //覆盖蛇

      {

            if (body[snake.len - 1].x != 0) //防止len++后将(0, 0)位置的墙覆盖

            {

                  //将蛇尾覆盖为空格即可

                  CursorJump(2 \* body[snake.len - 1].x, body[snake.len - 1].y);

                  printf(" ");

            }

      }

}

//移动蛇

void MoveSnake(int x, int y)

{

      DrawSnake(0); //先覆盖当前所显示的蛇

      face[body[snake.len - 1].y][body[snake.len - 1].x] = KONG; //蛇移动后蛇尾重新标记为空

      face[snake.y][snake.x] = BODY; //蛇移动后蛇头的位置变为蛇身

      //蛇移动后各个蛇身位置坐标需要更新

      for (int i = snake.len - 1; i > 0; i--)

      {

            body[i].x = body[i - 1].x;

            body[i].y = body[i - 1].y;

      }

      //蛇移动后蛇头位置信息变为第0个蛇身的位置信息

      body[0].x = snake.x;

      body[0].y = snake.y;

      //蛇头的位置更改

      snake.x = snake.x + x;

      snake.y = snake.y + y;

      DrawSnake(1); //打印移动后的蛇

}

//执行按键

void run(int x, int y)

{

      int t = 0;

      while (1)

      {

            if (t == 0)

                  t = 3000; //这里t越小，蛇移动速度越快（可以根据次设置游戏难度）

            while (--t)

            {

                  if (kbhit() != 0) //若键盘被敲击，则退出循环

                        break;

            }

            if (t == 0) //键盘未被敲击

            {

                  JudgeFunc(x, y); //判断到达该位置后，是否得分与游戏结束

                  MoveSnake(x, y); //移动蛇

            }

            else //键盘被敲击

            {

                  break; //返回Game函数读取键值

            }

      }

}

//游戏主体逻辑函数

void Game()

{

      int n = RIGHT; //开始游戏时，默认向后移动

      int tmp = 0; //记录蛇的移动方向

      goto first; //第一次进入循环先向默认方向前进

      while (1)

      {

            n = getch(); //读取键值

            //在执行前，需要对所读取的按键进行调整

            switch (n)

            {

            case UP:

            case DOWN: //如果敲击的是“上”或“下”

                  if (tmp != LEFT&&tmp != RIGHT) //并且上一次蛇的移动方向不是“左”或“右”

                  {

                        n = tmp; //那么下一次蛇的移动方向设置为上一次蛇的移动方向

                  }

                  break;

            case LEFT:

            case RIGHT: //如果敲击的是“左”或“右”

                  if (tmp != UP&&tmp != DOWN) //并且上一次蛇的移动方向不是“上”或“下”

                  {

                        n = tmp; //那么下一次蛇的移动方向设置为上一次蛇的移动方向

                  }

            case SPACE:

            case ESC:

            case 'r':

            case 'R':

                  break; //这四个无需调整

            default:

                  n = tmp; //其他键无效，默认为上一次蛇移动的方向

                  break;

            }

      first: //第一次进入循环先向默认方向前进

            switch (n)

            {

            case UP: //方向键：上

                  run(0, -1); //向上移动（横坐标偏移为0，纵坐标偏移为-1）

                  tmp = UP; //记录当前蛇的移动方向

                  break;

            case DOWN: //方向键：下

                  run(0, 1); //向下移动（横坐标偏移为0，纵坐标偏移为1）

                  tmp = DOWN; //记录当前蛇的移动方向

                  break;

            case LEFT: //方向键：左

                  run(-1, 0); //向左移动（横坐标偏移为-1，纵坐标偏移为0）

                  tmp = LEFT; //记录当前蛇的移动方向

                  break;

            case RIGHT: //方向键：右

                  run(1, 0); //向右移动（横坐标偏移为1，纵坐标偏移为0）

                  tmp = RIGHT; //记录当前蛇的移动方向

                  break;

            case SPACE: //暂停

                  system("pause>nul"); //暂停后按任意键继续

                  break;

            case ESC: //退出

                  system("cls"); //清空屏幕

                  color(7); //颜色设置为白色

                  CursorJump(COL - 8, ROW / 2);

                  printf(" 游戏结束 ");

                  CursorJump(COL - 8, ROW / 2 + 2);

                  exit(0);

            case 'r':

            case 'R': //重新开始

                  system("cls"); //清空屏幕

                  main(); //重新执行主函数

            }

      }

}