# 贪吃蛇游戏设计报告

##### 贪吃蛇游戏背景

贪吃蛇最初为人们所知的是诺基亚手机附带的一个小游戏，它伴随着诺基亚手机走向世界。现在的贪吃蛇出现了许多衍生版本，并被移植到各种平台上。

##### 游戏设计涉及知识

C语言中的三大结构和函数

C语言中动态内存申请和释放

C语言中文件基本操作

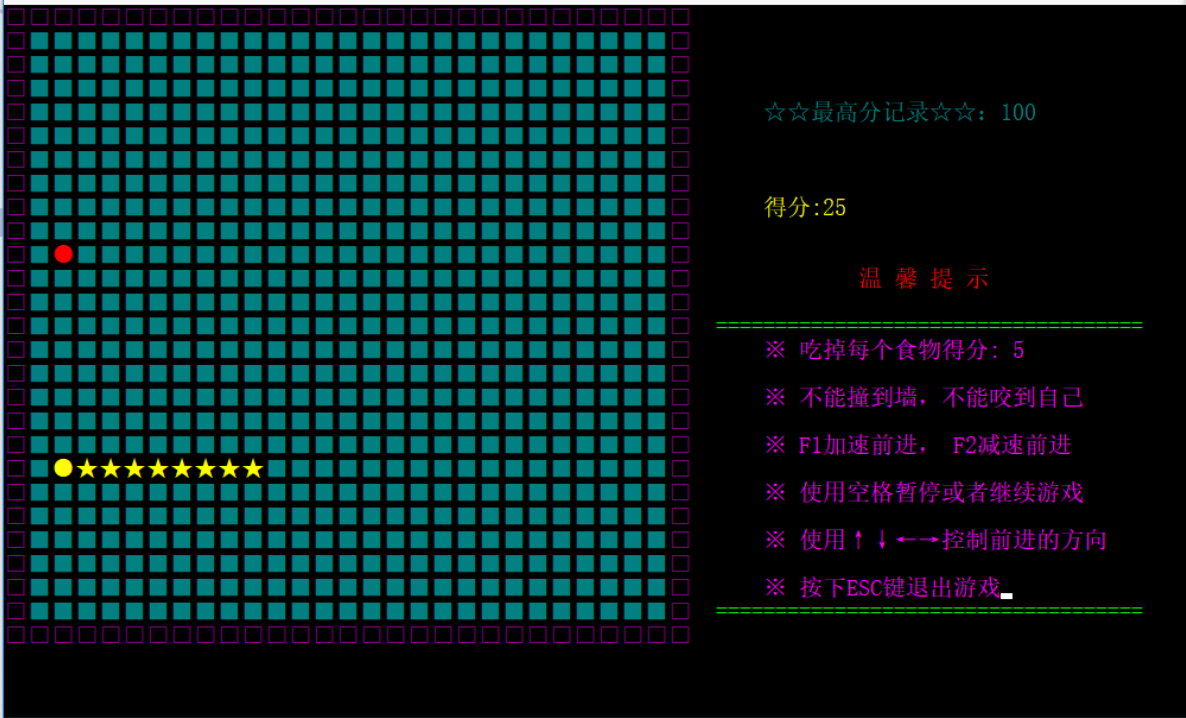
C语言中的简单单向链表

控制终端基本设置和监听按键

C语言简单游戏开发流程

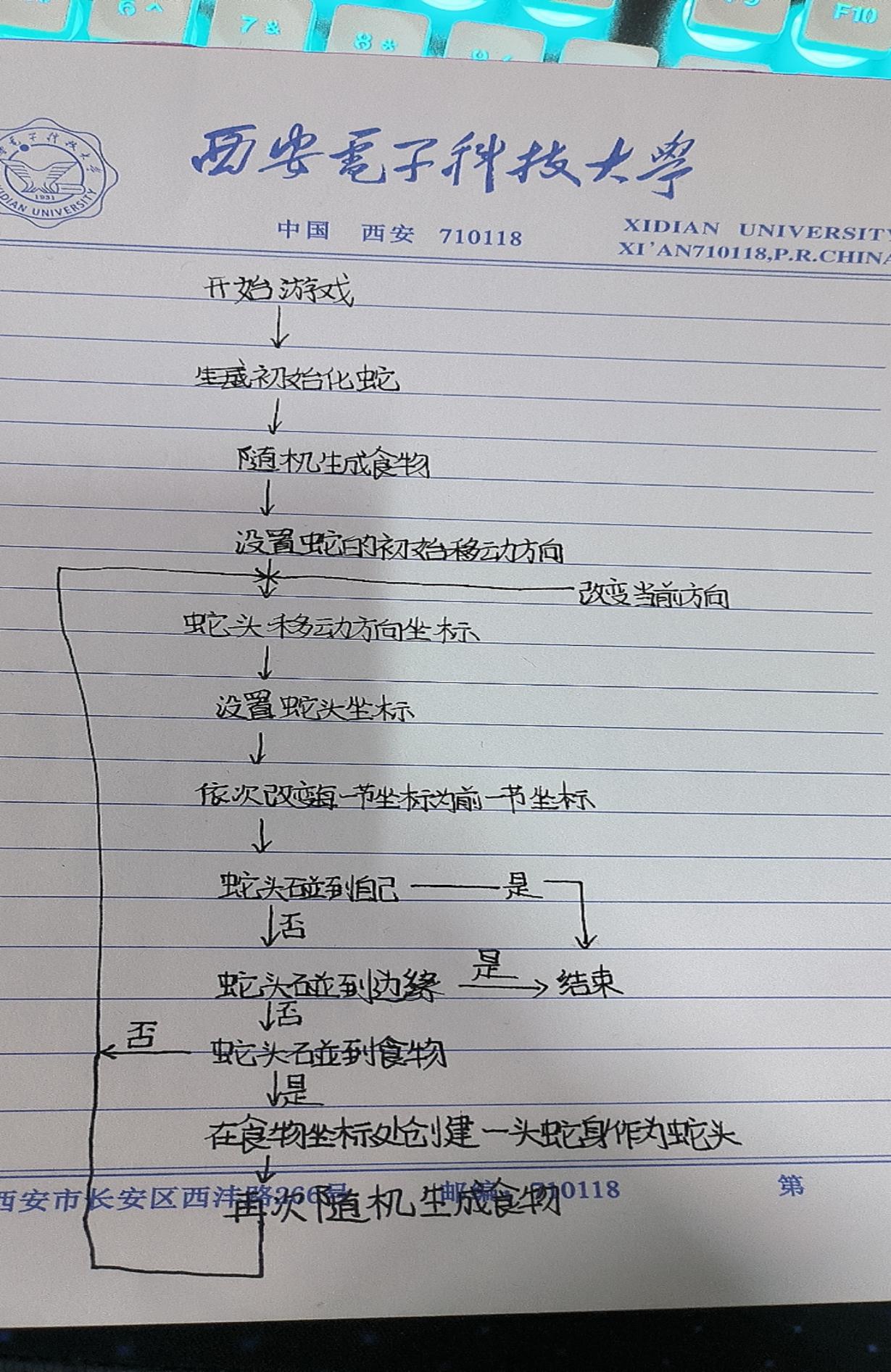
C语言简单游戏开发调试技巧

3.网上现有的贪吃蛇效果



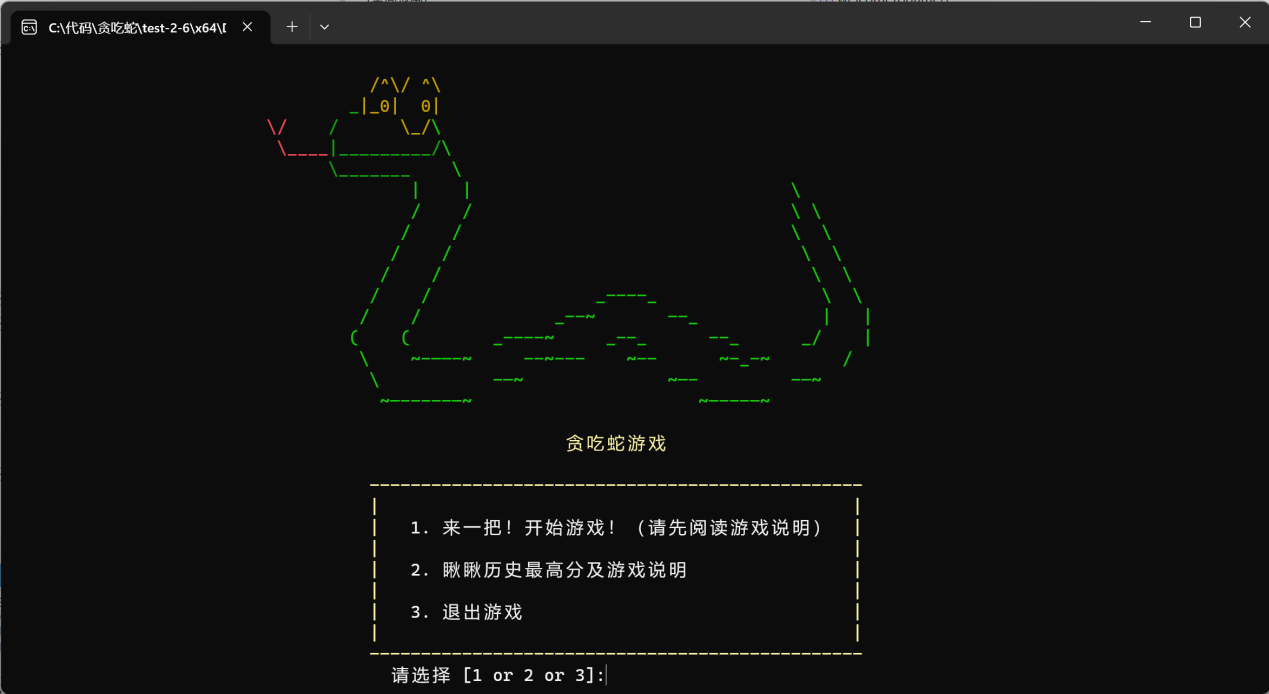
给我们起到了思维导向的作用，确定大致设计外框

游戏流程图



###### 一游戏欢迎界面

* 1. 常用终端控制函数
  2. 设置文本颜色函数
  3. 设置光标位置函数
  4. 绘制蛇的字符画函数
  5. 欢迎界面函数



结束

打印游戏选项

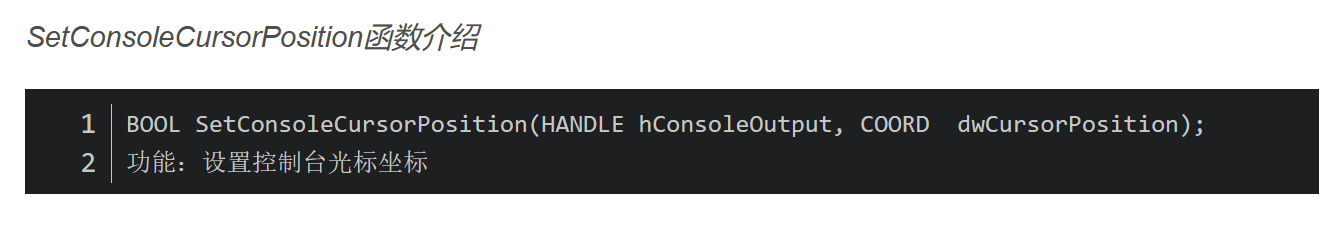
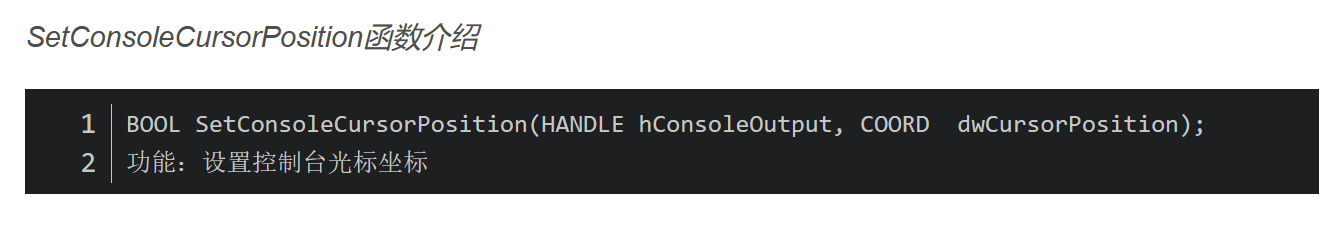
打印游戏边框

printSnake和标题

开始

* 1. 常用终端控制函数



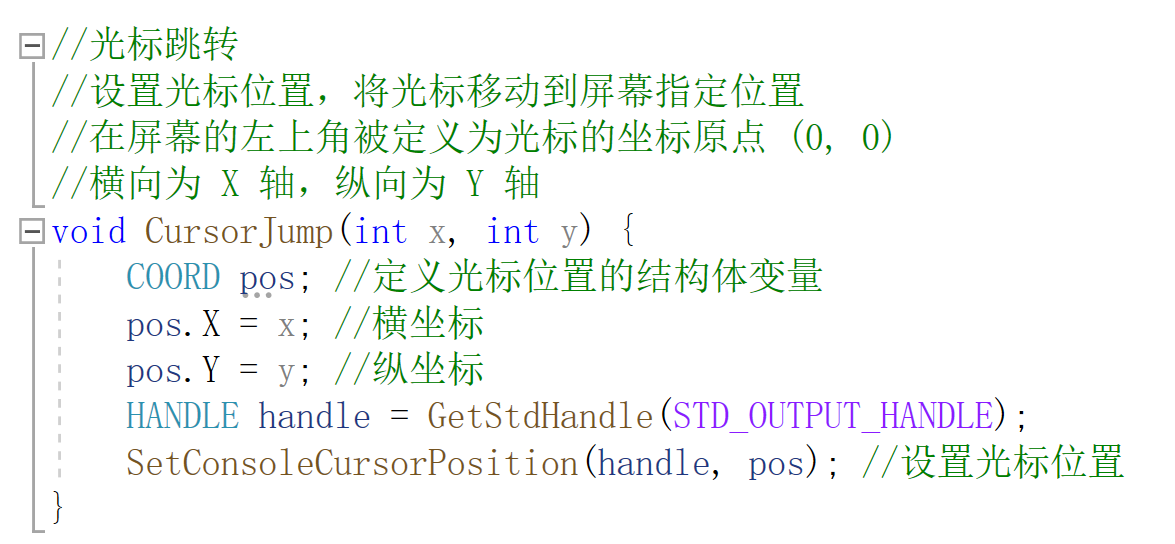


1.2设置文本颜色函数

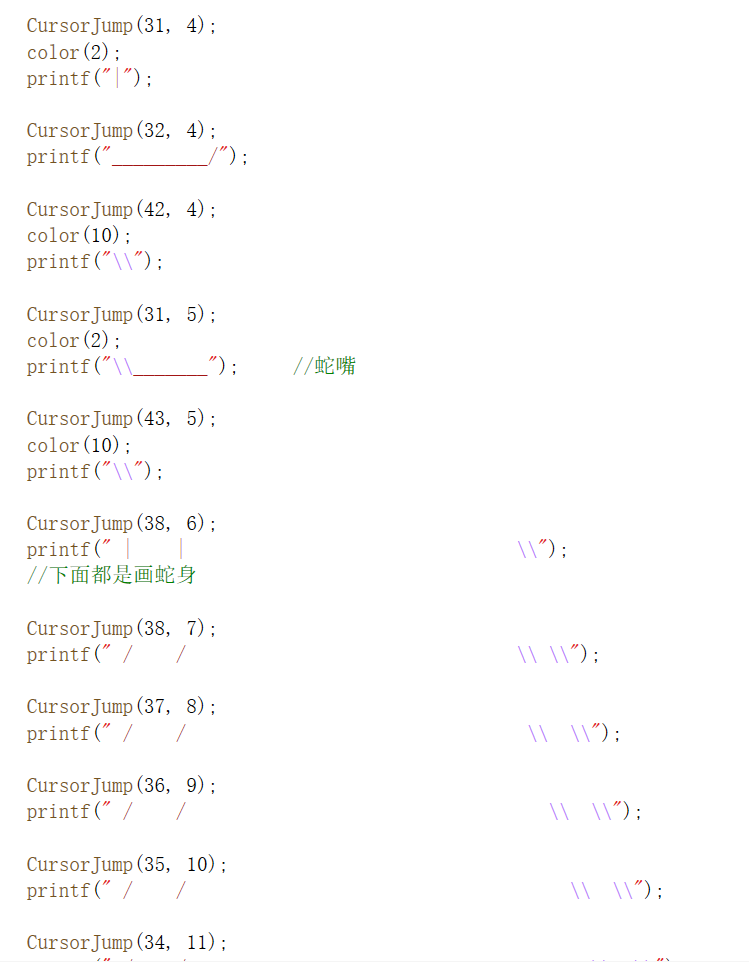




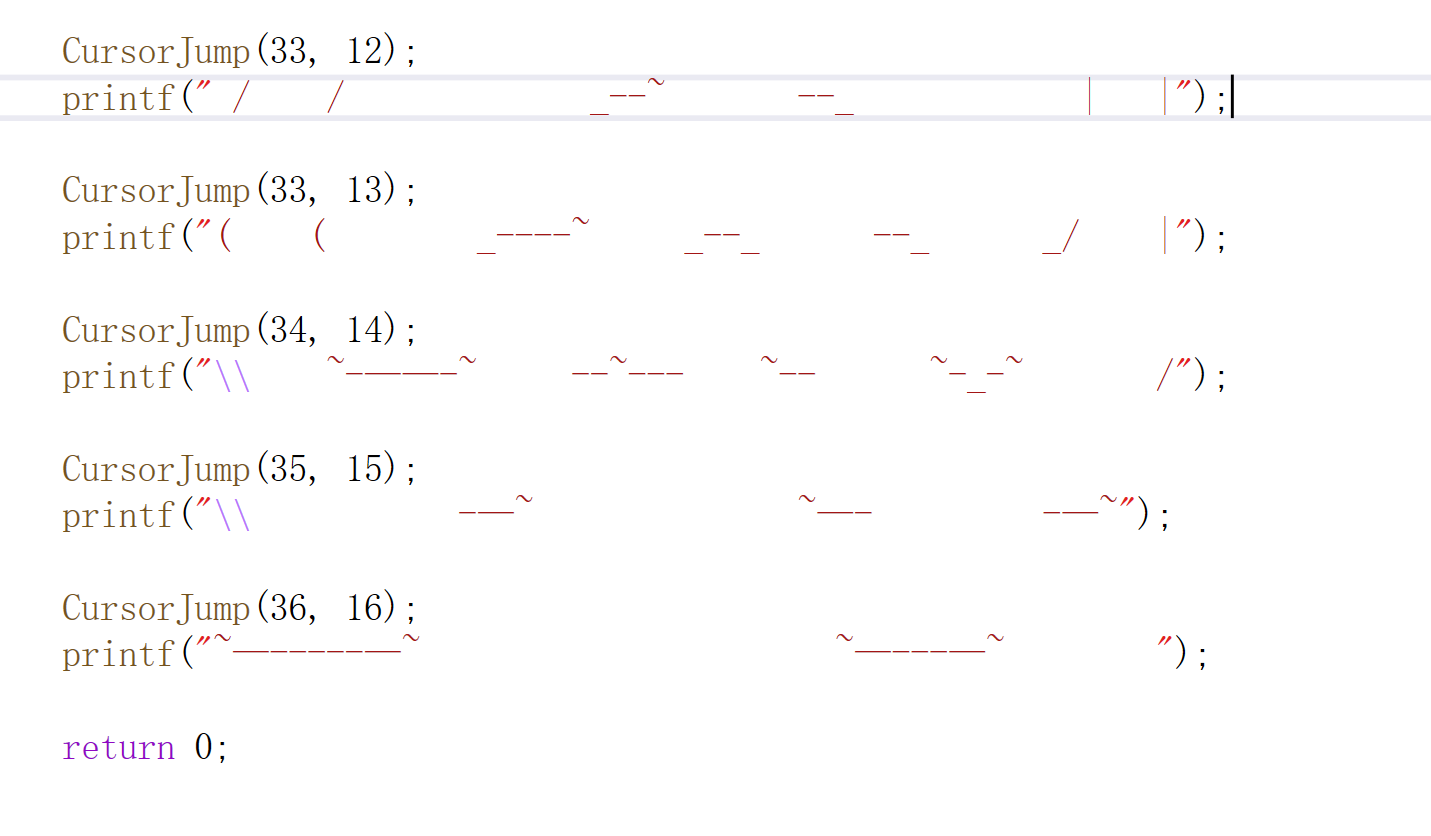
1.3设置光标位置函数



1.4绘制蛇的字符画函数



1 2



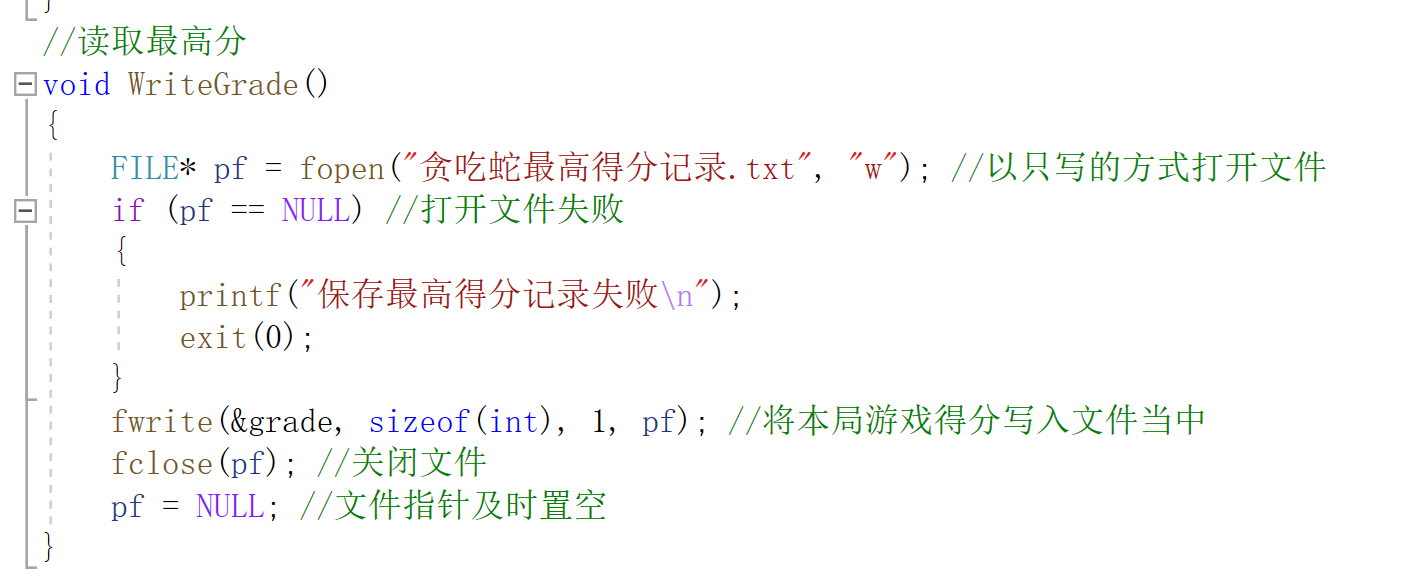
3

1.5欢迎界面函数

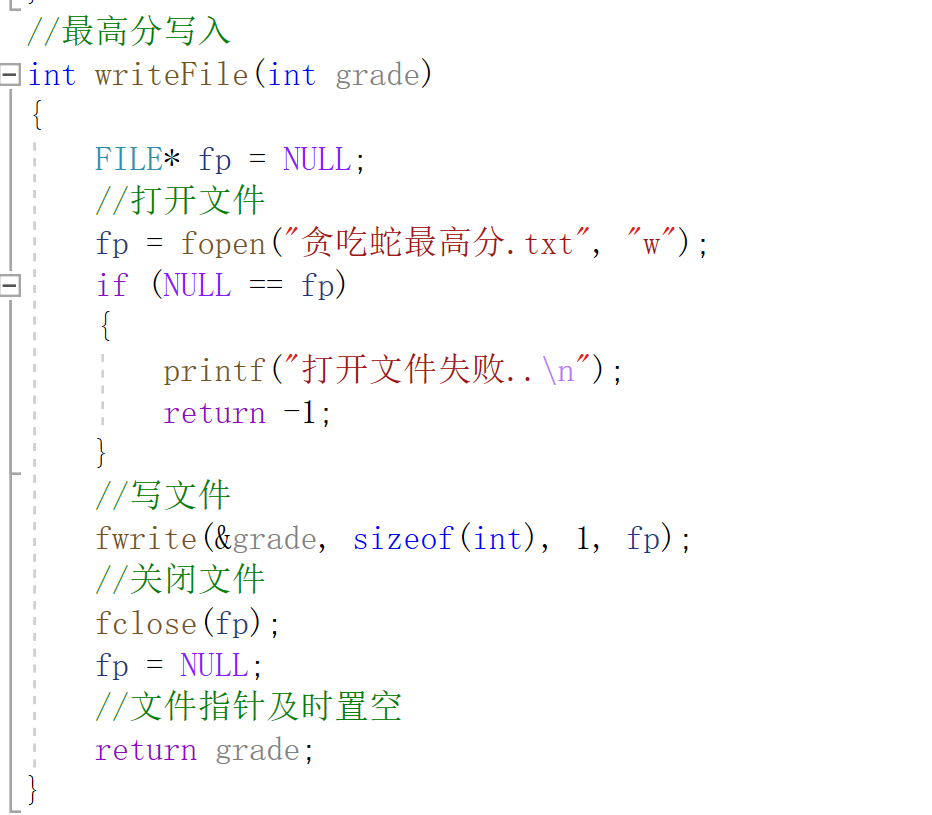


###### 文件写入读出

* 1. 读取最高分



**2.2**最高分写入



###### 二游戏进行设计过程

1.绘制地图、蛇和食物

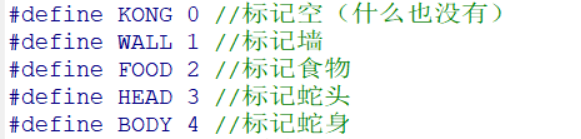
1.1首先定义游戏界面的大小，定义游戏区行数和列数，创建一个长\*宽大小的二维数组，用来存放游戏区每个格子的情况

代码展示



1.2定义图表中各个状态

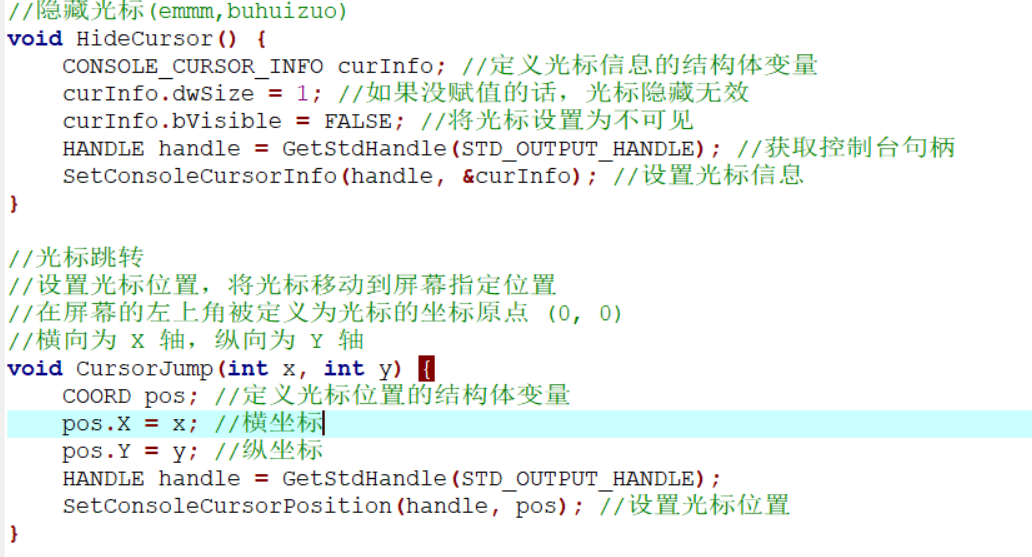
代码展示



## 1.3光标跳转和隐藏光标

光标跳转可以让我们实现在任意位置的打印，调用函数，光标跳转到指定地点，我们就可以实现任意位置的打印，除此之外我们还需要隐藏光标。（该部分需要查阅资料）

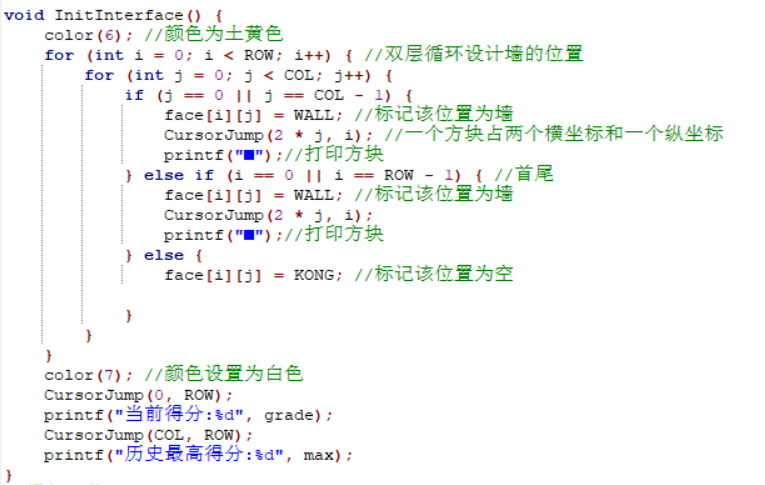
代码展示



1.4游戏界面的文字，图像的颜色设置

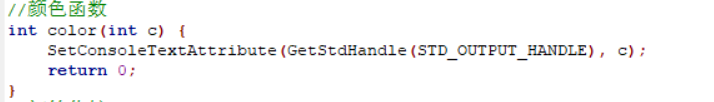
游戏界面的文字，图像的颜色设置，即初始化界面。首先打印游戏区格子，并设置颜色

代码展示



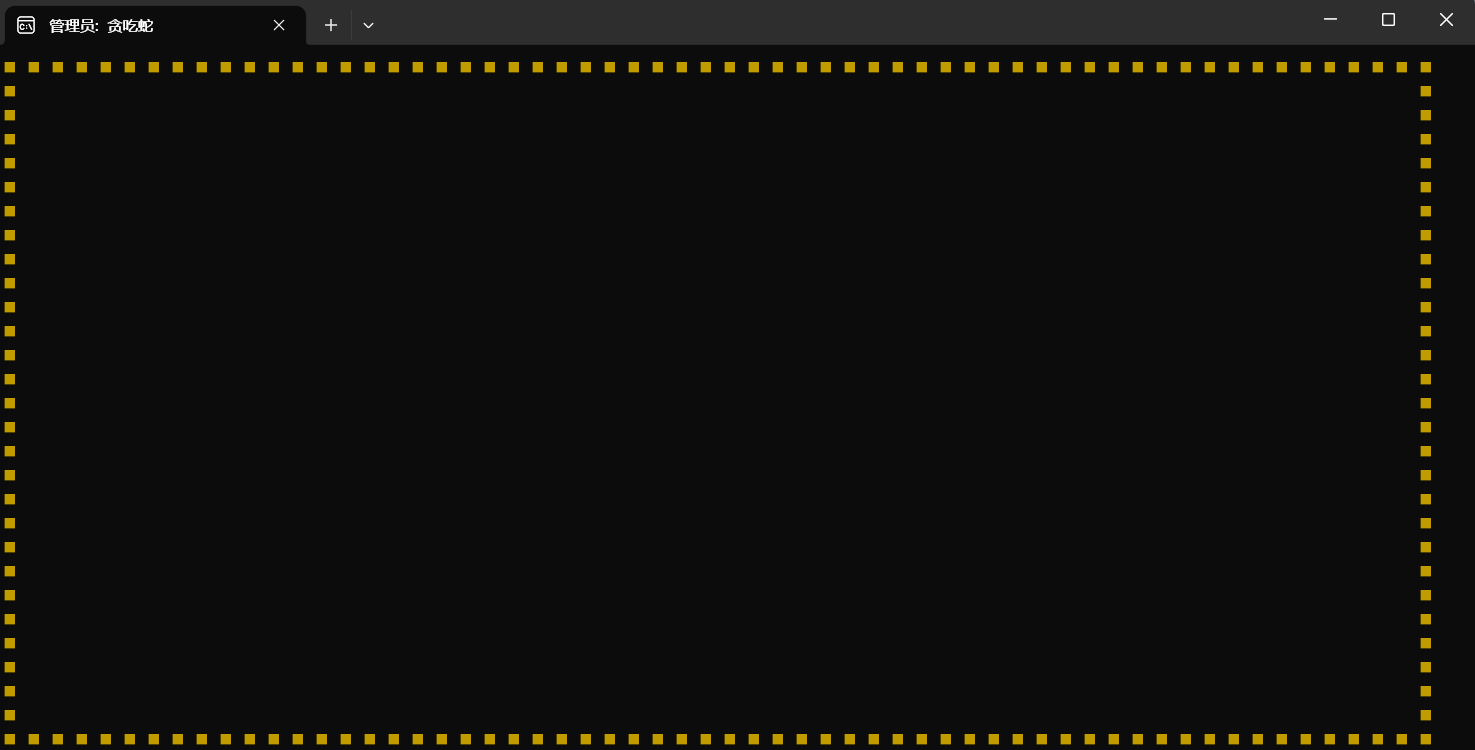
设计颜色

需要注意的是数字的颜色才是其所代表的颜色而不是后方背景





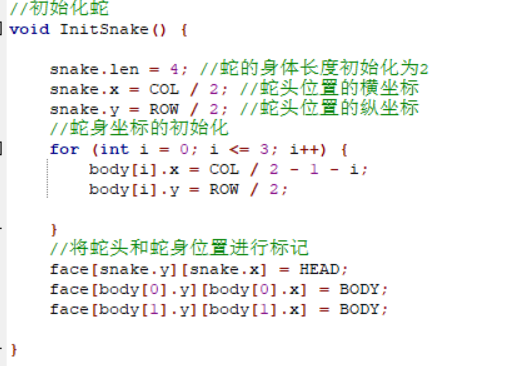
效果展示



1. 初始化蛇

绘制游戏中的蛇身体可用一定数量的特殊符号来 绘制蛇身，我们这里用了 5个黄色的五角星。在设计蛇身体时，我们首先定义蛇头，设置蛇头的初始位置，坐标为(行数的一半，列数的一半),然后设置蛇身，用了循环来设定其位置

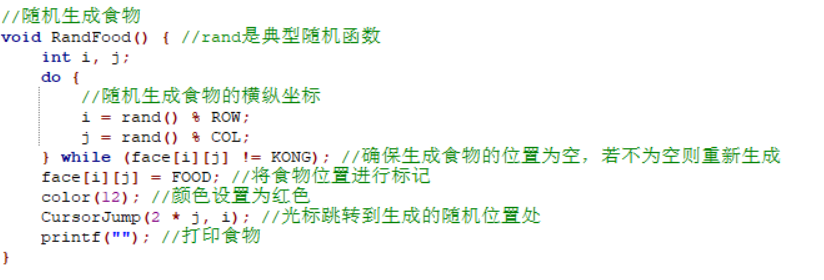
代码展示



## 随机生成食物

随机在游戏区生成食物，需要注意的是要对生成后的坐标进行判断，只有该位置为空才能在此生成食物，否则需要重新生成坐标。食物坐标确定后，需要对游戏区该位置的状态进行标记。

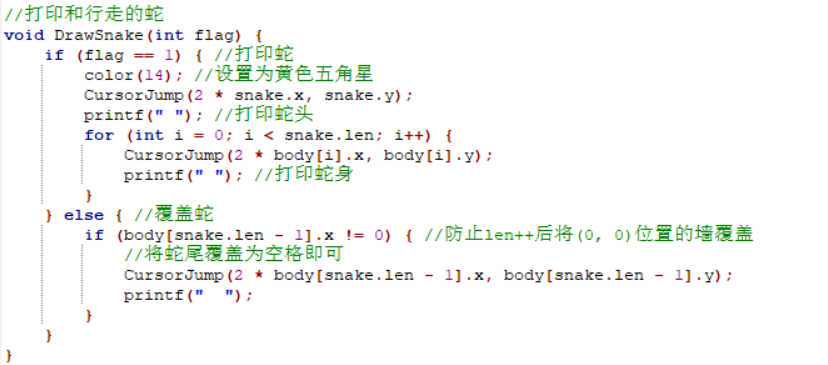
代码展示

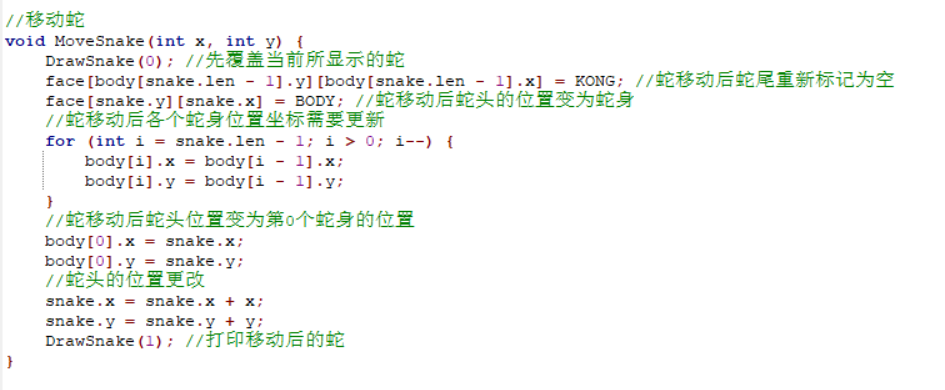


## 打印蛇与覆盖蛇

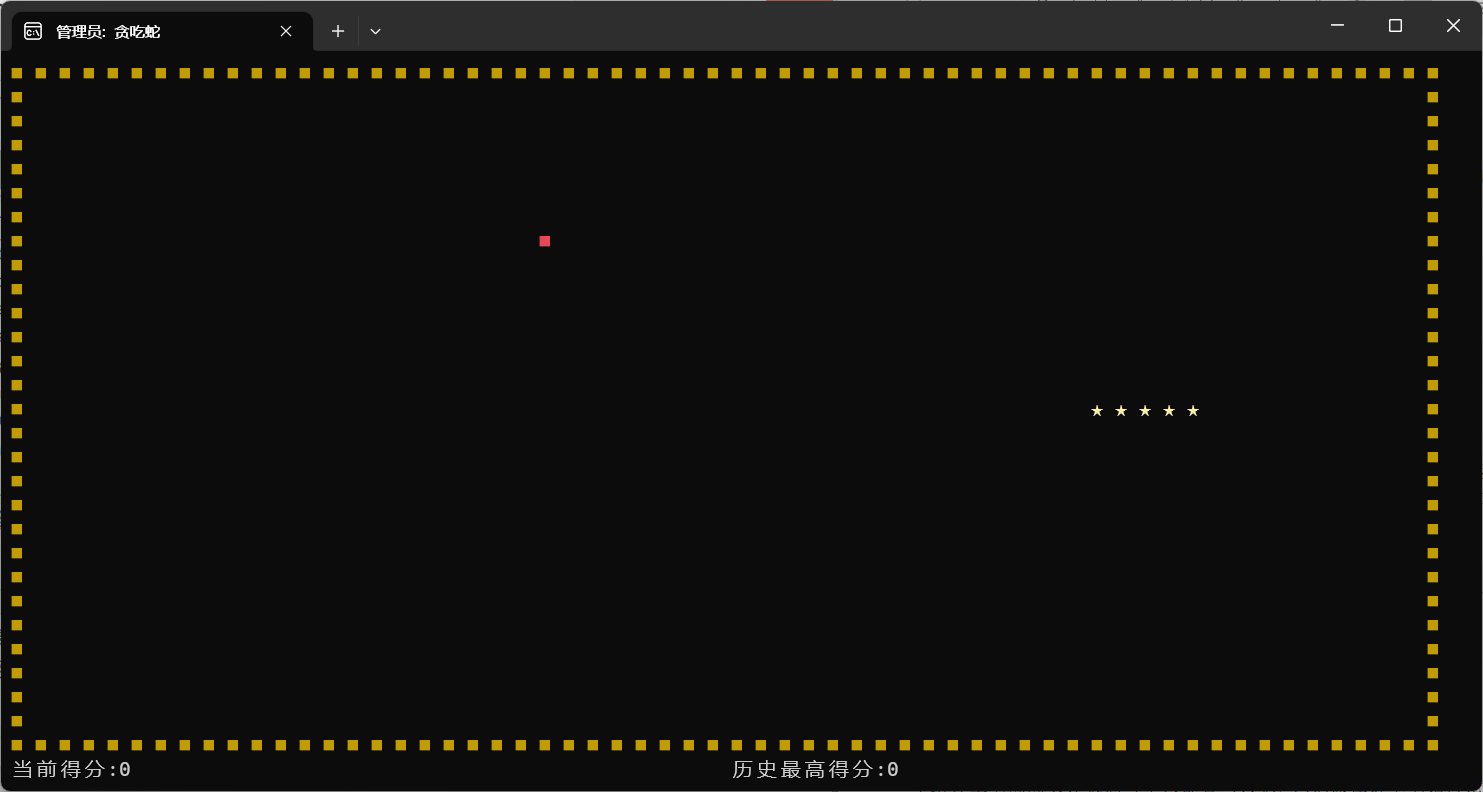
这里实现蛇移动的方法是利用空格覆盖来实现蛇的移动，将原来的蛇覆盖后打印新蛇

代码展示





效果展示



**5.控制蛇的函数**

为方便读取键值，将键盘的一些按键定义为数字

（使用getch来读取键盘输入的键值，再用switch case来判断各个键值对应的动作）

//游戏主体逻辑函数

void Game()

{

int n = RIGHT; //开始游戏时，默认向右移动

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 'S':

//case 's':

// /\*fen[1] = fen[0] + 2;

// fen[0] = fen[1];\*/

//case 'd':

//case 'D':

/\*if (fen[1] - 2 <= 0)

{

fen[1] = 1;

fen[0] = fen[1];

}

else

{

fen[1] = fen[0] - 2;

fen[0] = fen[1];

}\*/

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

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

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

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

tmp = RIGHT; //记录当前蛇的移动方向

break;

case 'S':

case 's':

t = shijian - (fen[1] - fen[0]) \* 100;

break;

case 'd':

case 'D':

t = shijian + (fen[0] - fen[1]) \* 100;

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

}

}

}

###### 游戏结束设计

### 1判断得分：

### 若蛇头即将到达的位置是食物，则得分。得分后需要将蛇身加长，并且更新当前得分，除此之外，还需要重新生成食物。

### 2判断结束：

### 若蛇头即将到达的位置是墙或者蛇身，则游戏结束。游戏结束后比较本局得分和历史最高得分，给出相应的提示语句，并且询问玩家是否再来一局，可自由发挥。

//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("创记录啦！最高分被你刷新啦，真棒");  
WriteGrade();  
}  
else if (grade == max)  
{  
printf("与最高记录持平，加油再创佳绩", grade);  
}  
else  
{  
printf("继续加油努力吧，你离最高分还差%d", max - grade);  
}

##### 四 游戏设计过程存在的问题

欢迎界面置放行列的部署不当

游戏边框循环中行列的大小与控制不佳

文字边框的颜色不够醒目

Getchar玩家输入后的多出来的回车键

文件指针的即时置空以及特殊情况无法打开的考虑

游戏界面不够延展

蛇移动速度过快

游戏结束当前得分未清空

历史最高分未存储与显示在游戏界面

缺少f1f2加速减速功能

游戏界面的适时清屏

开始设计时没有注意到初始界面的长宽，导致有有部分排版缺失，影响了界面的完整性

如何进行颜色的设计和相应界面的转换有点陌生，但是通过查资料后基本上能解决

打印方块时，方块重叠，查阅资料后发现一个方块占一个纵坐标和两个横坐标，

如何在按要求地打印方块，成了我的难题，调试中学习了隐藏光标和光标跳转的程序。

初始化蛇是最麻烦和重要的因为它直接关系到下面蛇运动的原理。定义蛇头或者蛇尾再设计蛇身

随机生成食物要考虑食物不在蛇已占取的位置

打印蛇和蛇移动后的位置转换，要考虑坐标的位置关系

##### 五 实验设计总结

难度大，培养了清晰缜密的编程逻辑思维，找bug的耐心的能力以及与队友的协调能力。

小组分工明确确实让我们每个人都各司其职，也积极在群里电话讨论，这种氛围着实令人羡慕

好多知识都需要自己复习还要对尚未涉足的新知识进行理解和运用，显著提高的不只是代码能力还有应对复杂问题和新知识的能力

从没写过如此之多的代码量，知道自己能力还有很大的发展空间，学习分析和拆解问题的能力

游戏代码前后部分的衔接需要组员的沟通交流，避免衔接的不流畅甚至是错误

通过查阅资料合力完成了贪吃蛇游戏，意识到大型游戏制作的复杂和难度之高，对大型游戏的设计更加感兴趣了。