

第30讲-项目实践-贪吃蛇

目录：

1. 游戏背景
2. 游戏效果演示
3. 课程目标
4. 课程定位
5. 技术要点
6. 贪吃蛇游戏设计与分析
7. 贪吃蛇游戏数据结构设计
8. 相关Win32API介绍
9. 参考代码

正文开始

1. 游戏背景

贪吃蛇是久负盛名的游戏，它也和俄罗斯方块，扫雷等游戏位列经典游戏的行列。

在编程语言的教学，我们以贪吃蛇为例，从设计到代码实现来提升学生的编程能力和逻辑能力。

2. 游戏效果演示



3. 课程目标

使用C语言在Windows环境的控制台中模拟实现经典小游戏贪吃蛇

实现基本的功能：

- 贪吃蛇地图绘制
- 蛇吃食物的功能（上、下、左、右方向键控制蛇的动作）
- 蛇撞墙死亡
- 蛇撞自身死亡
- 计算得分
- 蛇身加速、减速
- 暂停游戏

4. 课程定位

- 提高学生对编程的兴趣
- 对C语言语法做一个基本的巩固。
- 对游戏开发有兴趣的同学做一个启发。
- 项目适合：C语言学完的同学，有一定的代码能力，初步解除数据结构链表。

5. 技术要点

6. Win32 API介绍

本次实现贪吃蛇会使用到的一些Win32 API知识，那么就学习一下

6.1 Win32 API

Windows 这个多作业系统除了协调应用程序的执行、分配内存、管理资源之外，它同时也是一个很大的服务中心，调用这个服务中心的各种服务（每一种服务就是一个函数），可以帮应用程式达到开启视窗、描绘图形、使用周边设备等目的，由于这些函数服务的对象是应用程序(Application)，所以便称之为 Application Programming Interface，简称 API 函数。**WIN32 API**也就是Microsoft Windows 32位平台的应用程序编程接口。

6.2 控制台程序

平常我们运行起来的黑框程序其实就是控制台程序

我们可以使用cmd命令来设置控制台窗口的长宽：设置控制台窗口的大小，30行，100列

```
1 mode con cols=100 lines=30
```

也可以通过命令设置控制台窗口的名字：

```
1 title 贪吃蛇
```



6.3 控制台屏幕上的坐标COORD

COORD 是Windows API中定义的一种结构，表示一个字符在控制台屏幕上的坐标

```
1 typedef struct _COORD {
2     SHORT X;
3     SHORT Y;
4 } COORD, *PCOORD;
```

```
1 COORD pos = { 10, 15 };
```

6.4 GetStdHandle

GetStdHandle是一个Windows API函数。它用于从一个特定的标准设备（标准输入、标准输出或标准错误）中取得一个句柄（用来标识不同设备的数值），使用这个句柄可以操作设备。

```
1 HANDLE GetStdHandle(DWORD nStdHandle);
```

实例：

```
1 HANDLE hOutput = NULL;
2
3 //获取标准输出的句柄(用来标识不同设备的数值)
4 hOutput = GetStdHandle(STD_OUTPUT_HANDLE);
```

6.5 GetConsoleCursorInfo

检索有关指定控制台屏幕缓冲区的光标大小和可见性的信息

```
1 BOOL WINAPI GetConsoleCursorInfo(
2     HANDLE hConsoleOutput,
3     PCONSOLE_CURSOR_INFO lpConsoleCursorInfo
4 );
```

实例：

```
1 HANDLE hOutput = NULL;
2 //获取标准输出的句柄(用来标识不同设备的数值)
3 hOutput = GetStdHandle(STD_OUTPUT_HANDLE);
4
5 CONSOLE_CURSOR_INFO CursorInfo;
6 GetConsoleCursorInfo(hOutput, &CursorInfo); //获取控制台光标信息
```

6.5.1 CONSOLE_CURSOR_INFO

比特就业课主页: <https://m.cctalk.com/inst/s9yewhfr>

这个结构体，包含有关控制台游标的信息

```
1 typedef struct _CONSOLE_CURSOR_INFO {
2     DWORD dwSize;
3     BOOL bVisible;
4 } CONSOLE_CURSOR_INFO, *PCONSOLE_CURSOR_INFO;
```

- dwSize，由光标填充的字符单元格的百分比。此值介于1到100之间。光标外观会变化，范围从完全填充单元格到单元底部的水平线条。
- bVisible，游标的可见性。如果光标可见，则此成员为 TRUE。

```
1 CursorInfo.bVisible = false; //隐藏控制台光标
```

6.6 SetConsoleCursorInfo

设置指定控制台屏幕缓冲区的光标的大小和可见性。

```
1 BOOL WINAPI SetConsoleCursorInfo(
2     HANDLE hConsoleOutput,
3     const CONSOLE_CURSOR_INFO *lpConsoleCursorInfo
4 );
```

实例：

```
1 HANDLE hOutput = GetStdHandle(STD_OUTPUT_HANDLE);
2
3 //隐藏光标操作
4 CONSOLE_CURSOR_INFO CursorInfo;
5 GetConsoleCursorInfo(hOutput, &CursorInfo); //获取控制台光标信息
6 CursorInfo.bVisible = false; //隐藏控制台光标
7 SetConsoleCursorInfo(hOutput, &CursorInfo); //设置控制台光标状态
```

6.7 SetConsoleCursorPosition

比特就业课-专注IT大学生就业的精品课程

设置指定控制台屏幕缓冲区中的光标位置，我们将想要设置的坐标信息放在COORD类型的pos中，调用SetConsoleCursorPosition函数将光标位置设置到指定的位置。

```
1 BOOL WINAPI SetConsoleCursorPosition(
2     HANDLE hConsoleOutput,
3     COORD pos
4 );
```

实例：

```
1     COORD pos = { 10, 5};
2     HANDLE hOutput = NULL;
3     //获取标准输出的句柄(用来标识不同设备的数值)
4     hOutput = GetStdHandle(STD_OUTPUT_HANDLE);
5     //设置标准输出上光标的位置为pos
6     SetConsoleCursorPosition(hOutput, pos);
```

SetPos：封装一个设置光标位置的函数

```
1 //设置光标的坐标
2 void SetPos(short x, short y)
3 {
4     COORD pos = { x, y };
5     HANDLE hOutput = NULL;
6     //获取标准输出的句柄(用来标识不同设备的数值)
7     hOutput = GetStdHandle(STD_OUTPUT_HANDLE);
8     //设置标准输出上光标的位置为pos
9     SetConsoleCursorPosition(hOutput, pos);
10 }
```

6.8 GetAsyncKeyState

获取按键情况，GetAsyncKeyState的函数原型如下：

```
1 SHORT GetAsyncKeyState(
2     int vKey
3 );
```

将键盘上每个键的虚拟键值传递给函数，函数通过返回值来分辨按键的状态。

`GetAsyncKeyState` 的返回值是`short`类型，在上一次调用 `GetAsyncKeyState` 函数后，如果返回的16位的`short`数据中，最高位是1，说明按键的状态是按下，如果最高是0，说明按键的状态是抬起；如果最低位被置为1则说明，该按键被按过，否则为0。

如果我们要判断一个键是否被按过，可以检测`GetAsyncKeyState`返回值的最低值是否为1。

```
1 #define KEY_PRESS(VK) ( (GetAsyncKeyState(VK) & 0x1) ? 1 : 0 )
```

7. 贪吃蛇游戏设计与分析

7.1 地图

我们最终的贪吃蛇大纲要是这个样子，那我们的地图如何布置呢？





这里不得不讲一下控制台窗口的一些知识，如果想在控制台的窗口中指定位置输出信息，我们得知道该位置的坐标，所以首先介绍一下控制台窗口的坐标知识。

控制台窗口的坐标如下所示，横向的是X轴，从左向右依次增长，纵向是Y轴，从上到下依次增长。



在游戏地图上，我们打印墙体使用宽字符：□，打印蛇使用宽字符●，打印食物使用宽字符★

普通的字符是占一个字节的，这类宽字符是占用2个字节。

这里再简单的讲一下C语言的国际化特性相关的知识，过去C语言并不适合非英语国家（地区）使用。C语言最初假定字符都是自己的。但是这些假定并不是在世界的任何地方都适用。

C语言字符默认是采用ASCII编码的，ASCII字符集采用的是单字节编码，且只使用了单字节中的低7位，最高位是没有使用的，可表示为0xxxxxxx；可以看到，ASCII字符集共包含128个字符，在英语国家中，128个字符是基本够用的，但是，在其他国家语言中，比如，在法语中，字母上方有注音符，它就无法用ASCII码表示。于是，一些欧洲国家就决定，利用字节中闲置的最高位编入新的符号。比如，法语中的é的编码为130（二进制10000010）。这样一来，这些欧洲国家使用的编码体系，可以表示最多256个符号。但是，这里又出现了新的问题。不同的国家有不同的字母，因此，哪怕它们都使用256个符号的编码方式，代表的字母却不一样。比如，130在法语编码中代表了é，在希伯来语编码中却代表了字母Gimel (ג)，在俄语编码中又会代表另一个符号。但是不管怎样，所有这些编码方式中，0--127表示的符号是一样的，不一样的只是128--255的这一段。

至于亚洲国家的文字，使用的符号就更多了，汉字就多达10万左右。一个字节只能表示256种符号，肯定是不够的，就必须使用多个字节表达一个符号。比如，简体中文常见的编码方式是GB2312，使用两个字节表示一个汉字，所以理论上最多可以表示 $256 \times 256 = 65536$ 个符号。

后来为了使C语言适应国家化，C语言的标准中不断加入了国际化的支持。比如：加入和宽字符的类型 `wchar_t` 和宽字符的输入和输出函数，加入和 `<locale.h>` 头文件，其中提供了允许程序员针对特定地区（通常是国家或者说某种特定语言的地理区域）调整程序行为的函数。

7.1.1 <locale.h>本地化

比特就业课主页：<https://m.cctalk.com/inst/s9yewhfr>

<locale.h>提供的函数用于控制C标准库中对于不同的地区会产生不一样行为的部分。

在标准可以中，依赖地区的部分有以下几项：

- 数字量的格式
- 货币量的格式
- 字符集
- 日期和时间的表示形式

7.1.2 类项

通过修改地区，程序可以改变它的行为来适应世界的不同区域。但地区的改变可能会影响库的许多部分，其中一部分可能是我们不希望修改的。所以C语言支持针对不同的类项进行修改，下面的一个宏，指定一个类项：

- LC_COLLATE
- LC_CTYPE
- LC_MONETARY
- LC_NUMERIC
- LC_TIME
- LC_ALL - 针对所有类项修改

每个类项的详细说明，请参考

7.1.3 setlocale函数

```
1 char* setlocale (int category, const char* locale);
```

setlocale 函数用于修改当前地区，可以针对一个类项修改，也可以针对所有类项。

setlocale 的第一个参数可以是前面说明的类项中的一个，那么每次只会影响一个类项，如果第一个参数是LC_ALL，就会影响所有的类项。

C标准给第二个参数仅定义了2种可能取值："C"和""。

在任意程序执行开始，都会隐藏式执行调用：

```
1 setlocale(LC_ALL, "C");
```

当地区设置为"C"时，库函数按正常方式执行，小数点是一个点。
比特就业课主页: <https://www.cctalk.com/inst/s9yewhfr>

当程序运行起来后想改变地区，就只能显示调用setlocale函数。用" "作为第2个参数，调用setlocale函数就可以切换到本地模式，这种模式下程序会适应本地环境。比如：切换到我们的本地模式后就支持宽字符（汉字）的输出等。

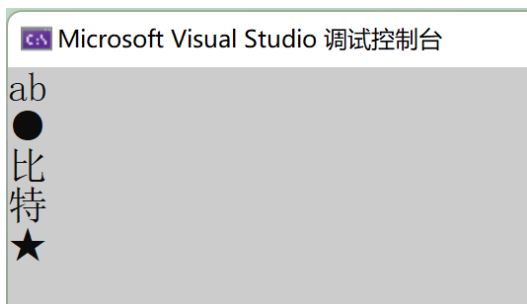
```
1 setlocale(LC_ALL, " "); //切换到本地环境
```

7.1.4 宽字符的打印

那如果想在屏幕上打印宽字符，怎么打印呢？

```
1 #include <stdio.h>
2 #include <locale.h>
3
4 int main() {
5     setlocale(LC_ALL, "");
6     wchar_t ch1 = L'●';
7     wchar_t ch2 = L'比';
8     wchar_t ch3 = L'特';
9     wchar_t ch4 = L'★';
10    printf("%c%c\n", 'a', 'b');
11
12    wprintf(L"%c\n", ch1);
13    wprintf(L"%c\n", ch2);
14    wprintf(L"%c\n", ch3);
15    wprintf(L"%c\n", ch4);
16    return 0;
17 }
```

输出结果：

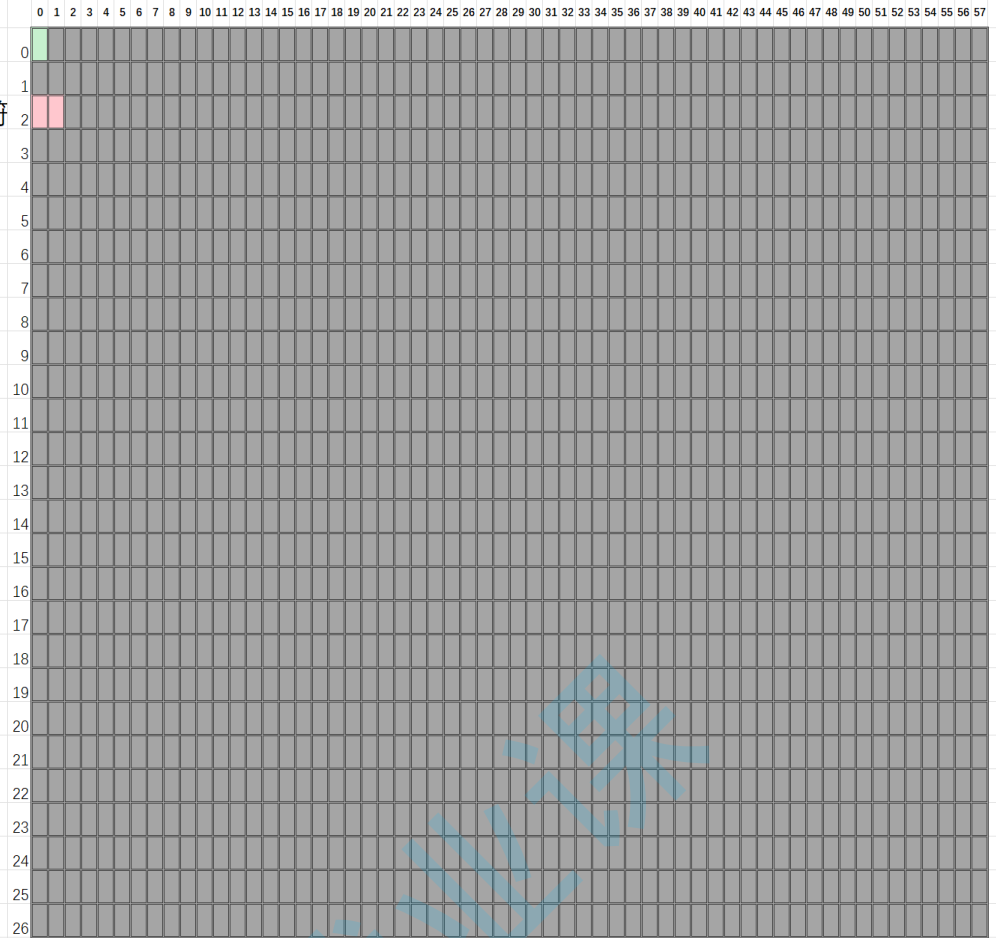


从输出的结果来看，我们发现一个普通字符占一个字符的位置，但是打印一个汉字字符，占用2个字符的位置，那么我们如果要在贪吃蛇中使用宽字符，就得处理好地图上坐标的计算。

普通字符和宽字符打印出宽度的展示如下：

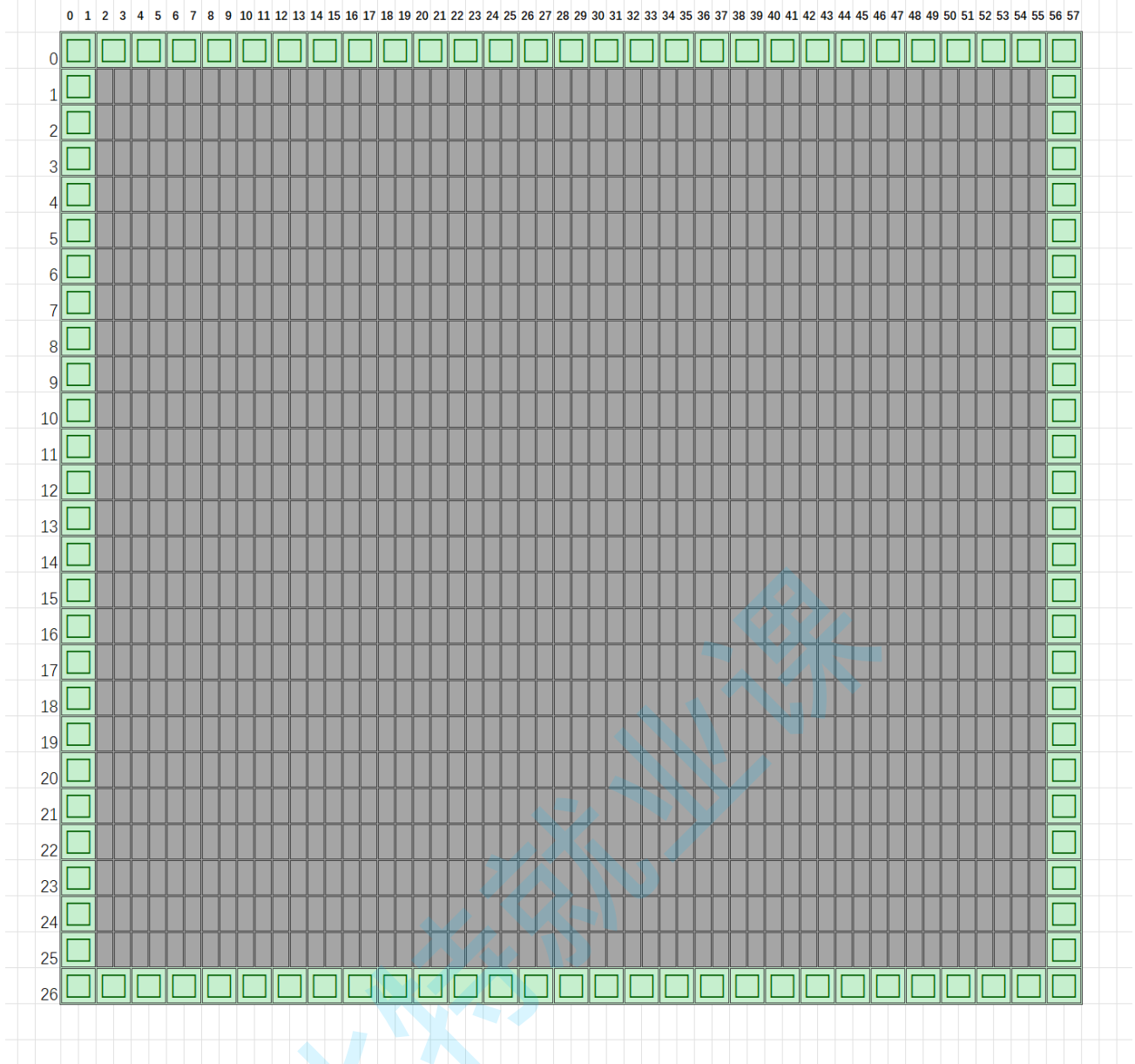
1个坐标1个正常字符

2个坐标可以存放1个宽字符



7.1.5 地图坐标

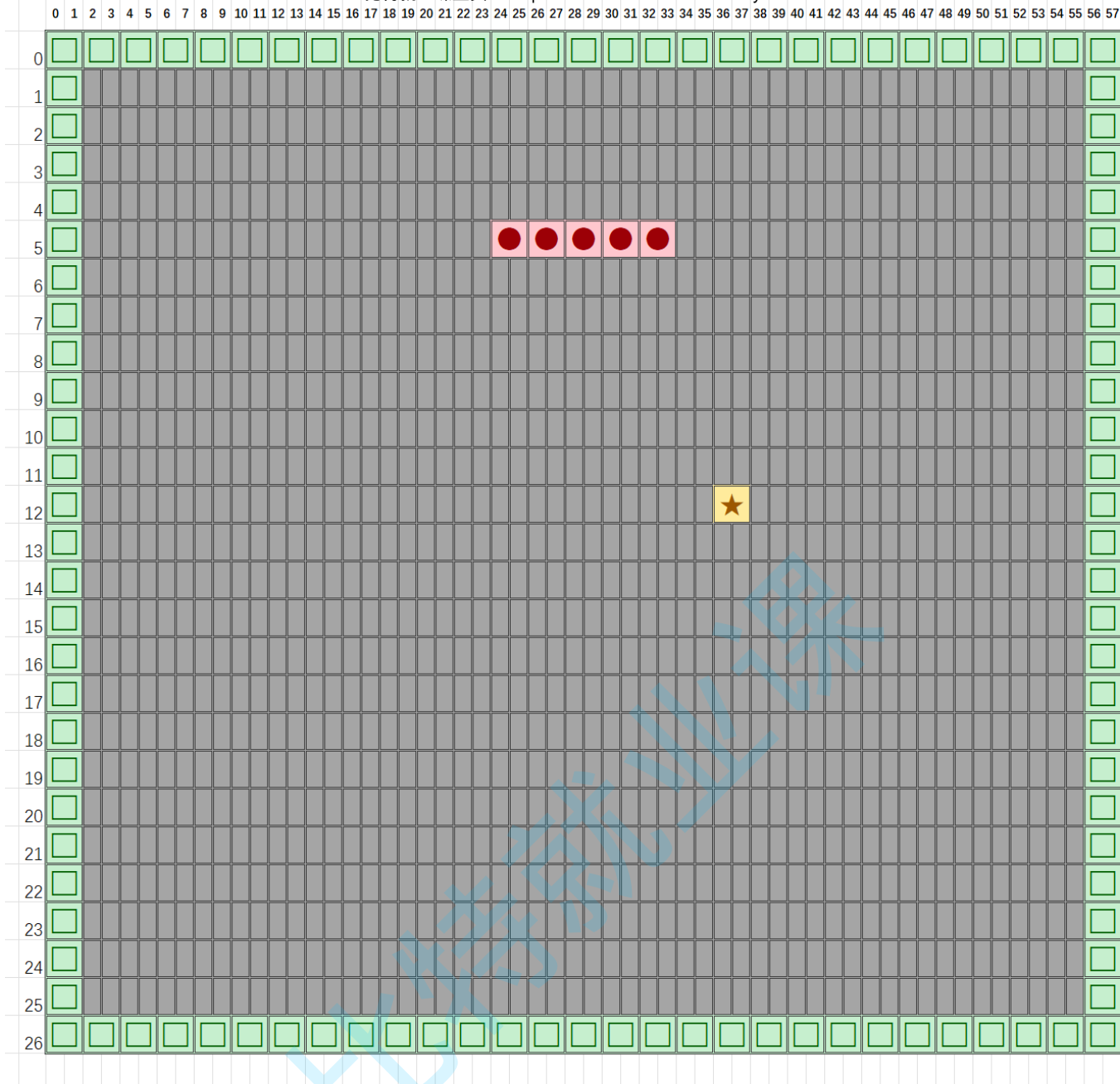
我们假设实现一个棋盘27行，58列的棋盘（行和列可以根据自己的情况修改），再围绕地图画出墙，如下：



7.2 蛇身和食物

初始化状态，假设蛇的长度是5，蛇身的每个节点是●，在固定的一个坐标处，比如(24, 5)处开始出现蛇，连续5个节点。**注意：**蛇的每个节点的x坐标必须是2个倍数，否则可能会出现蛇的一个节点有一半儿出现在墙体中，另外一般在墙外的现象，坐标不好对齐。

关于食物，就是在墙体内随机生成一个坐标（x坐标必须是2的倍数），坐标不能和蛇的身体重合，然后打印★。



7.3 数据结构设计

在游戏运行的过程中，蛇每次吃一个食物，蛇的身体就会变长一节，如果我们使用链表存储蛇的信息，那么蛇的每一节其实就是链表的每个节点。每个节点只要记录好舍身节点在地图上的坐标就行，所以舍蛇节点结构如下：

```
1 typedef struct SnakeNode
2 {
3     int x;
4     int y;
5     struct SnakeNode* next;
6 }SnakeNode, * pSnakeNode;
```

要管理整条贪吃蛇，我们再封装一个Snake的结构来维护整条贪吃蛇：

```
1 typedef struct Snake
2 {
3     pSnakeNode _pSnake; //维护整条蛇的指针
4     pSnakeNode _pFood; //维护食物的指针
5     enum DIRECTION _Dir; //蛇头的方向默认是向右
6     enum GAME_STATUS _Status; //游戏状态
7     int _Score; //当前获得分数
8     int _Add; //默认每个食物10分
9     int _SleepTime; //每走一步休眠时间
10 }Snake, * pSnake;
```

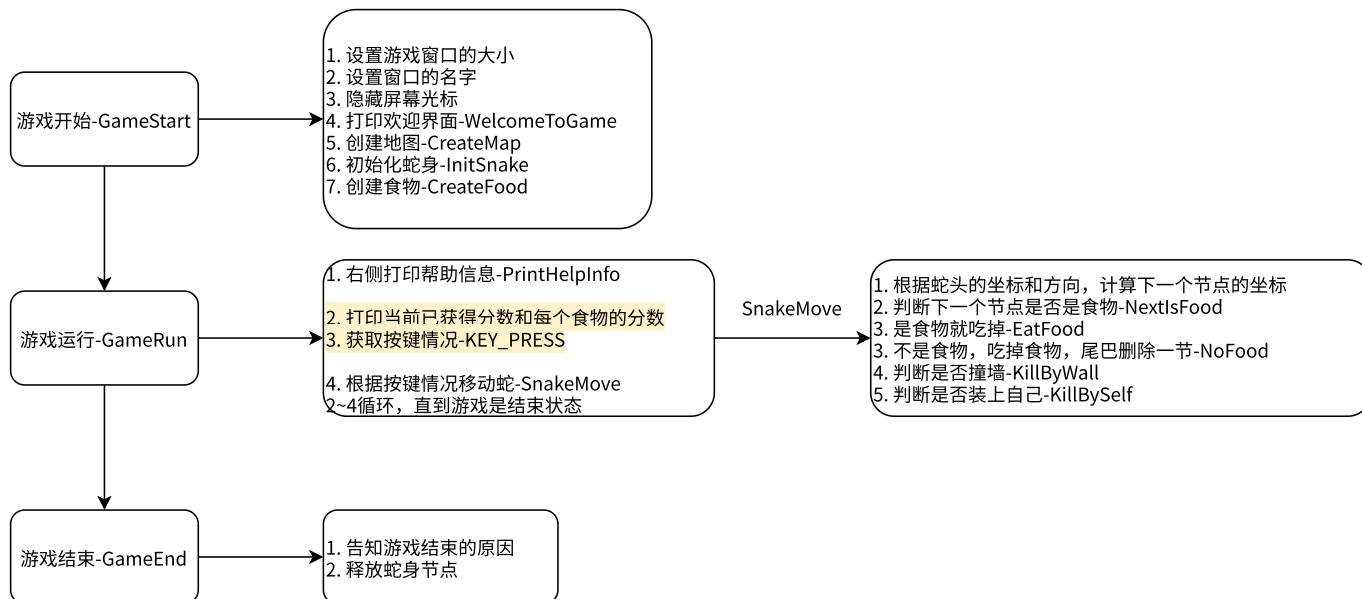
蛇的方向，可以一一列举，使用枚举

```
1 //方向
2 enum DIRECTION
3 {
4     UP = 1,
5     DOWN,
6     LEFT,
7     RIGHT
8 };
```

游戏状态，可以一一列举，使用枚举

```
1 //游戏状态
2 enum GAME_STATUS
3 {
4     OK, //正常运行
5     KILL_BY_WALL, //撞墙
6     KILL_BY_SELF, //咬到自己
7     END_NOMAL //正常结束
8 };
```

7.4 游戏流程设计



8. 核心逻辑实现分析

8.1 游戏主逻辑

```

1  #include <locale.h>
2
3  void test()
4  {
5      int ch = 0;
6      srand((unsigned int)time(NULL));
7
8      do
9      {
10         Snake snake = { 0 };
11         GameStart(&snake);
12         GameRun(&snake);
13         GameEnd(&snake);
14         SetPos(20, 15);
15         printf("再来一局吗? (Y/N):");
16         ch = getchar();
17         getchar(); //清理\n
18
19     } while (ch == 'Y');
20     SetPos(0, 27);
21 }
22
23 int main()
24 {

```



```

25 //修改当前地区为本地模式，为了支持中文宽字体的打印
26 setlocale(LC_ALL, "");
27 //测试逻辑
28 test();
29 return 0;
30 }

```

8.2 游戏开始

```

1 void GameStart(pSnake ps)
2 {
3     //设置控制台窗口的大小，30行，100列
4     //mode 为DOS命令
5     system("mode con cols=100 lines=30");
6     //设置cmd窗口名称
7     system("title 贪吃蛇");
8
9     //获取标准输出的句柄(用来标识不同设备的数值)
10    HANDLE hOutput = GetStdHandle(STD_OUTPUT_HANDLE);
11
12    //隐藏光标操作
13    CONSOLE_CURSOR_INFO CursorInfo;
14    GetConsoleCursorInfo(hOutput, &CursorInfo); //获取控制台光标信息
15    CursorInfo.bVisible = false; //隐藏控制台光标
16    SetConsoleCursorInfo(hOutput, &CursorInfo); //设置控制台光标状态
17
18    //打印欢迎界面
19    WelcomeToGame();
20    //打印地图
21    CreateMap();
22    //初始化蛇
23    InitSnake(ps);
24    //创造第一个食物
25    CreateFood(ps);
26 }

```

8.2.1 打印欢迎界面

在游戏正式开始之前，做一些功能提醒

```

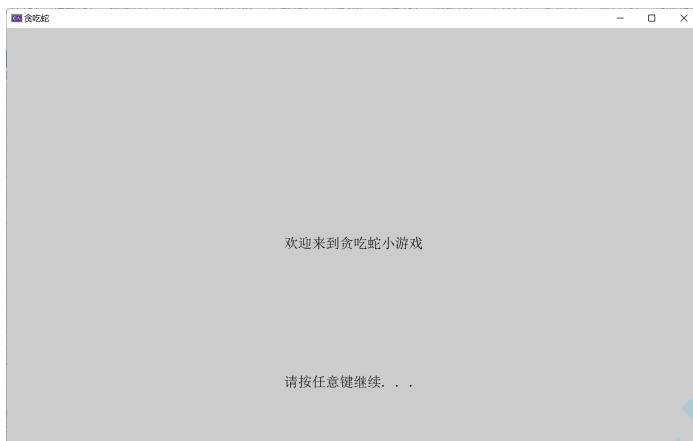
1 void WelcomeToGame()
2 {
3     SetPos(40, 15);
4     printf("欢迎来到贪吃蛇小游戏");

```

```

5  SetPos(40, 25); //让按任意键继续的出现的位点好看
6  system("pause");
7  system("cls");
8  SetPos(25, 12);
9  printf("用 ↑ ↓ ← → 分别控制蛇的移动, F3为加速, F4为减速\n");
10 SetPos(25, 13);
11 printf("加速将能得到更高的分数.\n");
12 SetPos(40, 25); //让按任意键继续的出现的位点好看
13 system("pause");
14 system("cls");
15 }

```



8.2.2 创建地图

创建地图就是将强打印出来, 因为是宽字符打印, 所有使用wprintf函数, 打印格式串前使用L
打印地图的关键是要算好坐标, 才能在想要的位置打印墙体。

墙体打印的宽字符:

```
1 #define WALL L'□'
```

创建地图函数CreateMap

```

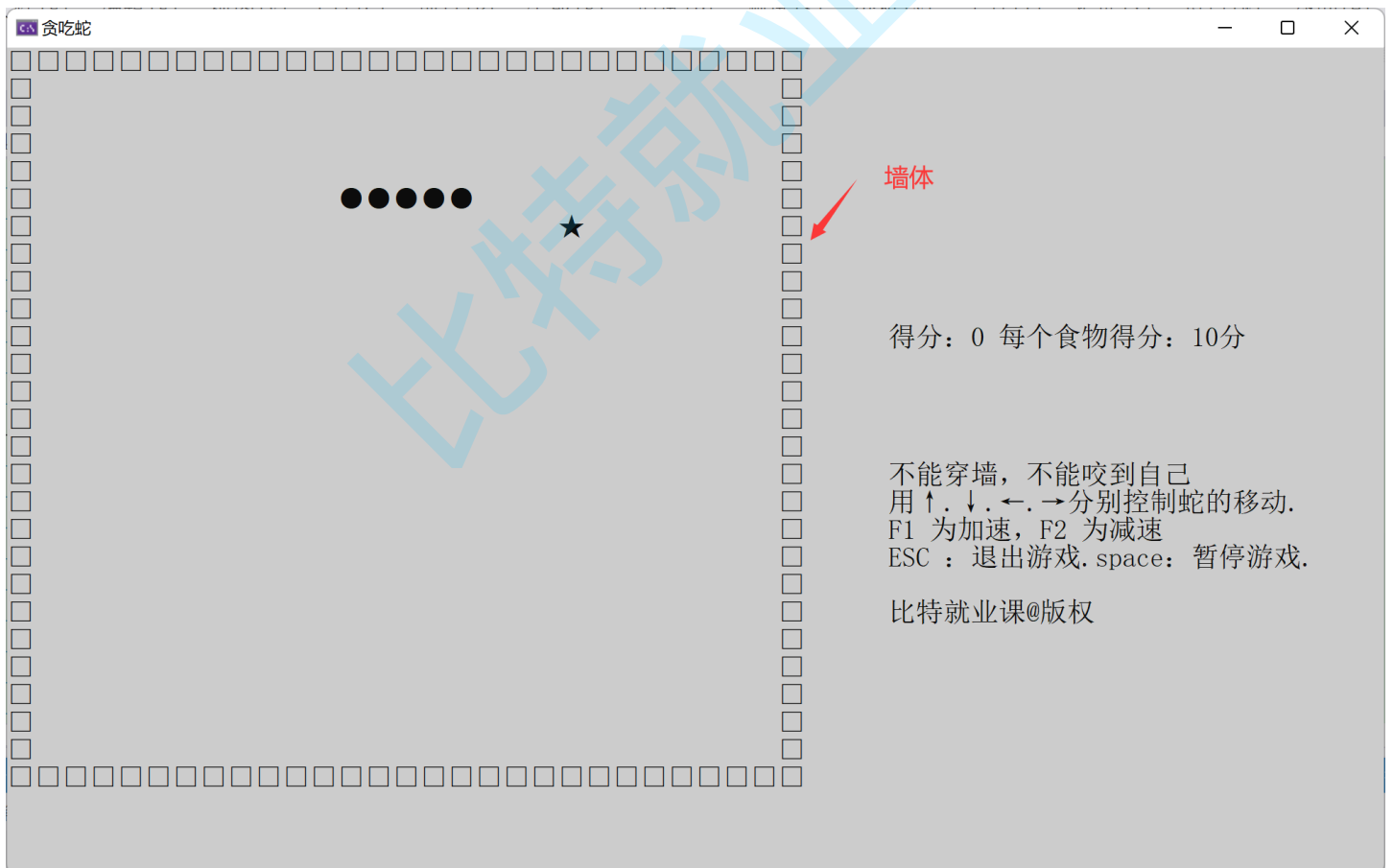
1 void CreateMap()
2 {
3     int i = 0;
4     //上(0,0)-(56, 0)
5     SetPos(0, 0);
6     for (i = 0; i < 58; i += 2)
7     {
8         wprintf(L"%c", WALL);
9     }
10    //下(0,26)-(56, 26)

```

```

11     SetPos(0, 26);
12     for (i = 0; i < 58; i += 2)
13     {
14         wprintf(L"%c", WALL);
15     }
16     //左
17     //x是0, y从1开始增长
18     for (i = 1; i < 26; i++)
19     {
20         SetPos(0, i);
21         wprintf(L"%c", WALL);
22     }
23     //x是56, y从1开始增长
24     for (i = 1; i < 26; i++)
25     {
26         SetPos(56, i);
27         wprintf(L"%c", WALL);
28     }
29 }

```



8.2.3 初始化蛇身

蛇最开始长度为5节，每节对应链表的一个节点，蛇身的每一个节点都有自己的坐标。

创建5个节点，然后将每个节点存放在链表中进行管理。创建完蛇身后，将蛇的每一节打印在屏幕上。

再设置当前游戏的状态，蛇移动的速度，默认的方向，初始成绩，蛇的状态，每个食物的分数。

蛇身打印的宽字符：

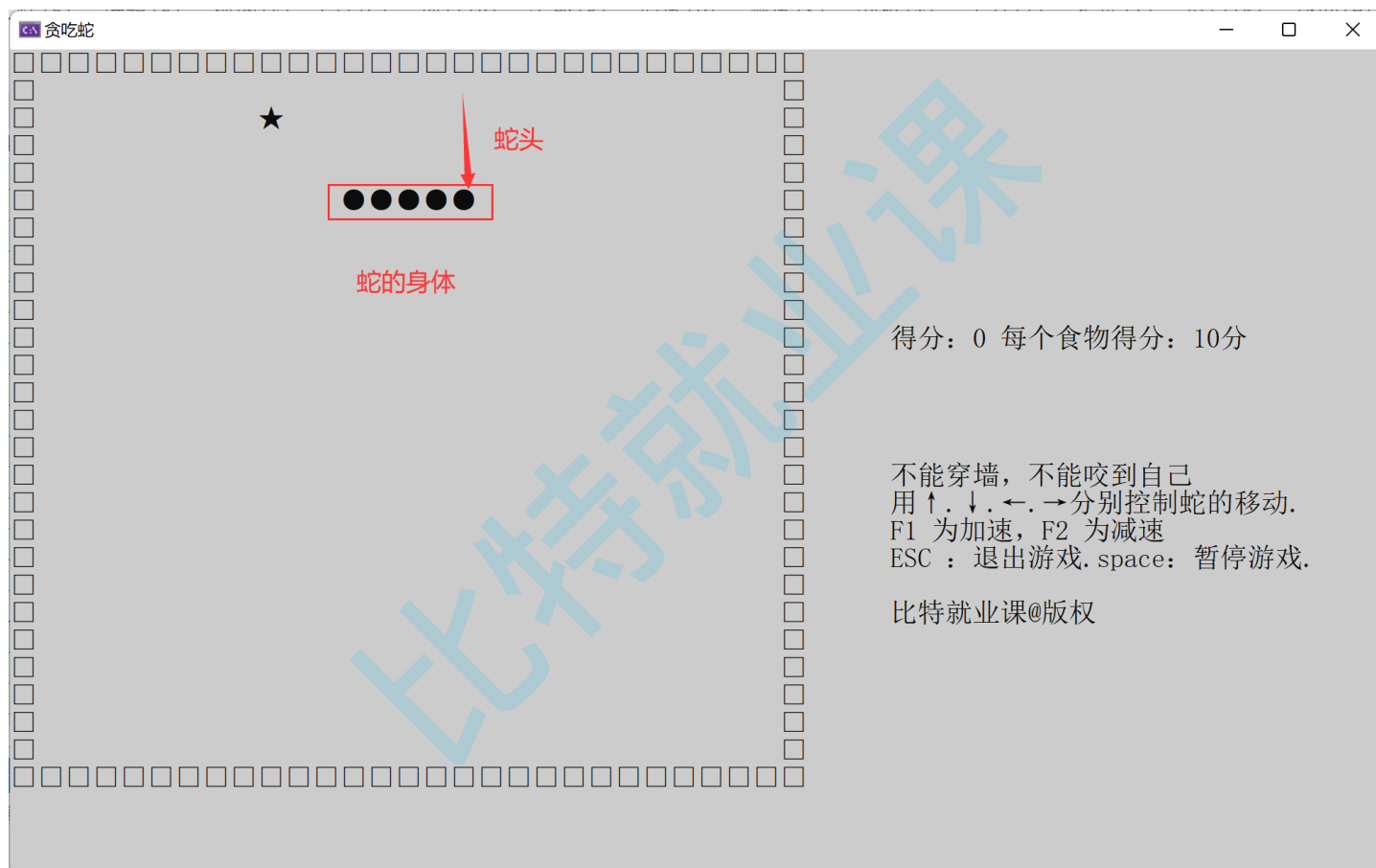
```
1 #define BODY L'●'
```

初始化蛇身函数：InitSnake

```
1 void InitSnake(pSnake ps)
2 {
3     pSnakeNode cur = NULL;
4     int i = 0;
5     //创建蛇身节点，并初始化坐标
6     //头插法
7     for (i = 0; i < 5; i++)
8     {
9         //创建蛇身的节点
10        cur = (pSnakeNode)malloc(sizeof(SnakeNode));
11        if (cur == NULL)
12        {
13            perror("InitSnake():malloc()");
14            return;
15        }
16        //设置坐标
17        cur->next = NULL;
18        cur->x = POS_X + i * 2;
19        cur->y = POS_Y;
20
21        //头插法
22        if (ps->pSnake == NULL)
23        {
24            ps->pSnake = cur;
25        }
26        else
27        {
28            cur->next = ps->pSnake;
29            ps->pSnake = cur;
30        }
31    }
32
33    //打印蛇的身体
34    cur = ps->pSnake;
35    while (cur)
36    {
```

```

37     SetPos(cur->x, cur->y);
38     wprintf(L"%c", BODY);
39     cur = cur->next;
40 }
41
42 //初始化贪吃蛇数据
43 ps->_SleepTime = 200;
44 ps->_Socre = 0;
45 ps->_Status = OK;
46 ps->_Dir = RIGHT;
47 ps->_Add = 10;
48 }
    
```



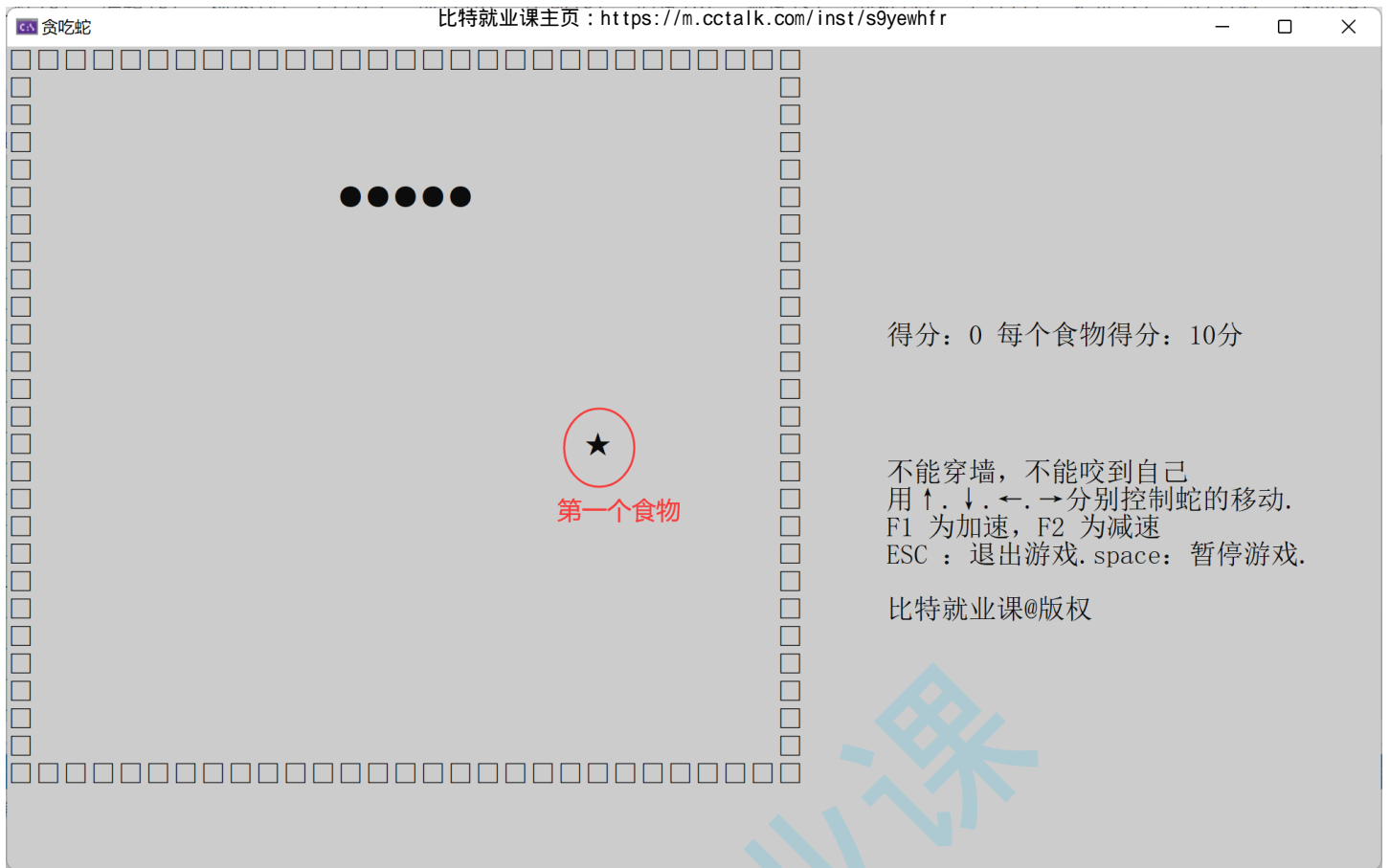
8.2.4 创建第一个食物

- 先随机生成食物的坐标
 - x坐标必须是2的倍数
 - 食物的坐标不能和蛇身每个节点的坐标重复
- 创建食物节点，打印食物

食物打印的宽字符:

创建食物的函数: CreateFood

```
1 void CreateFood(pSnake ps)
2 {
3     int x = 0;
4     int y = 0;
5
6     again:
7         //产生的x坐标应该是2的倍数, 这样才可能和蛇头坐标对齐。
8         do
9         {
10             x = rand() % 53 + 2;
11             y = rand() % 25 + 1;
12         } while (x % 2 != 0);
13
14     pSnakeNode cur = ps->_pSnake; //获取指向蛇头的指针
15     //食物不能和蛇身冲突
16     while (cur)
17     {
18         if (cur->x == x && cur->y == y)
19         {
20             goto again;
21         }
22         cur = cur->next;
23     }
24
25     pSnakeNode pFood = (pSnakeNode)malloc(sizeof(SnakeNode)); //创建食物
26     if (pFood == NULL)
27     {
28         perror("CreateFood:malloc()");
29         return;
30     }
31     else
32     {
33         pFood->x = x;
34         pFood->y = y;
35         SetPos(pFood->x, pFood->y);
36         wprintf(L"%c", FOOD);
37         ps->_pFood = pFood;
38     }
39 }
```



8.3 游戏运行

游戏运行期间，右侧打印帮助信息，提示玩家

根据游戏状态检查游戏是否继续，如果是状态是OK，游戏继续，否则游戏结束。

如果游戏继续，就是检测按键情况，确定蛇下一步的方向，或者是否加速减速，是否暂停或者退出游戏。

确定了蛇的方向和速度，蛇就可以移动了。

```
1 void GameRun(pSnake ps)
2 {
3     //打印右侧帮助信息
4     PrintHelpInfo();
5     do
6     {
7         SetPos(64, 10);
8         printf("得分: %d ", ps->_Score);
9         printf("每个食物得分: %d分", ps->_Add);
10        if (KEY_PRESS(VK_UP) && ps->_Dir != DOWN)
11        {
12            ps->_Dir = UP;
13        }
14        else if (KEY_PRESS(VK_DOWN) && ps->_Dir != UP)
15        {
```

```
16     ps->_Dir = DOWN;
17 }
18 else if (KEY_PRESS(VK_LEFT) && ps->_Dir != RIGHT)
19 {
20     ps->_Dir = LEFT;
21 }
22 else if (KEY_PRESS(VK_RIGHT) && ps->_Dir != LEFT)
23 {
24     ps->_Dir = RIGHT;
25 }
26 else if (KEY_PRESS(VK_SPACE))
27 {
28     pause();
29 }
30 else if (KEY_PRESS(VK_ESCAPE))
31 {
32     ps->_Status = END_NOMAL;
33     break;
34 }
35 else if (KEY_PRESS(VK_F3))
36 {
37     if (ps->_SleepTime >= 50)
38     {
39         ps->_SleepTime -= 30;
40         ps->_Add += 2;
41     }
42 }
43 else if (KEY_PRESS(VK_F4))
44 {
45     if (ps->_SleepTime < 350)
46     {
47         ps->_SleepTime += 30;
48         ps->_Add -= 2;
49         if (ps->_SleepTime == 350)
50         {
51             ps->_Add = 1;
52         }
53     }
54 }
55 //蛇每次一定之间要休眠的时间,时间短,蛇移动速度就快
56 Sleep(ps->_SleepTime);
57 SnakeMove(ps);
58
59 } while (ps->_Status == OK);
60 }
```


8.3.1 KEY_PRESS

比特就业课主页：<https://m.cctalk.com/inst/s9yewhfr>

检测按键状态，我们封装了一个宏

```
1 #define KEY_PRESS(VK) ((GetAsyncKeyState(VK)&0x1) ? 1 : 0)
```

8.3.2 PrintHelpInfo

```
1 void PrintHelpInfo()
2 {
3     //打印提示信息
4     SetPos(64, 15);
5     printf("不能穿墙，不能咬到自己\n");
6     SetPos(64, 16);
7     printf("用↑.↓.←.→分别控制蛇的移动.");
8     SetPos(64, 17);
9     printf("F1 为加速，F2 为减速\n");
10    SetPos(64, 18);
11    printf("ESC : 退出游戏.space: 暂停游戏.");
12    SetPos(64, 20);
13    printf("比特就业课@版权");
14 }
```



8.3.3 蛇身移动

先创建下一个节点, 根据移动方向和蛇头的坐标, 蛇移动到下一个位置的坐标。

确定了下一个位置后, 看下一个位置是否是食物 (NextIsFood), 是食物就做吃食物处理 (EatFood), 如果不是食物则做前进一步的处理 (NoFood)。

蛇身移动后, 判断此次移动是否会造成撞墙 (KillByWall) 或者撞上自己蛇身 (KillBySelf), 从而影响游戏的状态。

```

1 void SnakeMove(pSnake ps)
2 {
3     //创建下一个节点
4     pSnakeNode pNextNode = (pSnakeNode)malloc(sizeof(SnakeNode));
5     if (pNextNode == NULL)
6     {
7         perror("SnakeMove()::malloc()");
8         return;
9     }
10    //确定下一个节点的坐标, 下一个节点的坐标根据, 蛇头的坐标和方向确定
11    switch (ps->_Dir)
12    {
13        case UP:
14        {
15            pNextNode->x = ps->_pSnake->x;
16            pNextNode->y = ps->_pSnake->y - 1;

```

```

17     }
18     break;
19     case DOWN:
20     {
21         pNextNode->x = ps->_pSnake->x;
22         pNextNode->y = ps->_pSnake->y + 1;
23     }
24     break;
25     case LEFT:
26     {
27         pNextNode->x = ps->_pSnake->x - 2;
28         pNextNode->y = ps->_pSnake->y;
29     }
30     break;
31     case RIGHT:
32     {
33         pNextNode->x = ps->_pSnake->x + 2;
34         pNextNode->y = ps->_pSnake->y;
35     }
36     break;
37 }
38
39 //如果下一个位置就是食物
40 if (NextIsFood(pNextNode, ps))
41 {
42     EatFood(pNextNode, ps);
43 }
44 else//如果没有食物
45 {
46     NoFood(pNextNode, ps);
47 }
48
49 KillByWall(ps);
50 KillBySelf(ps);
51 }

```

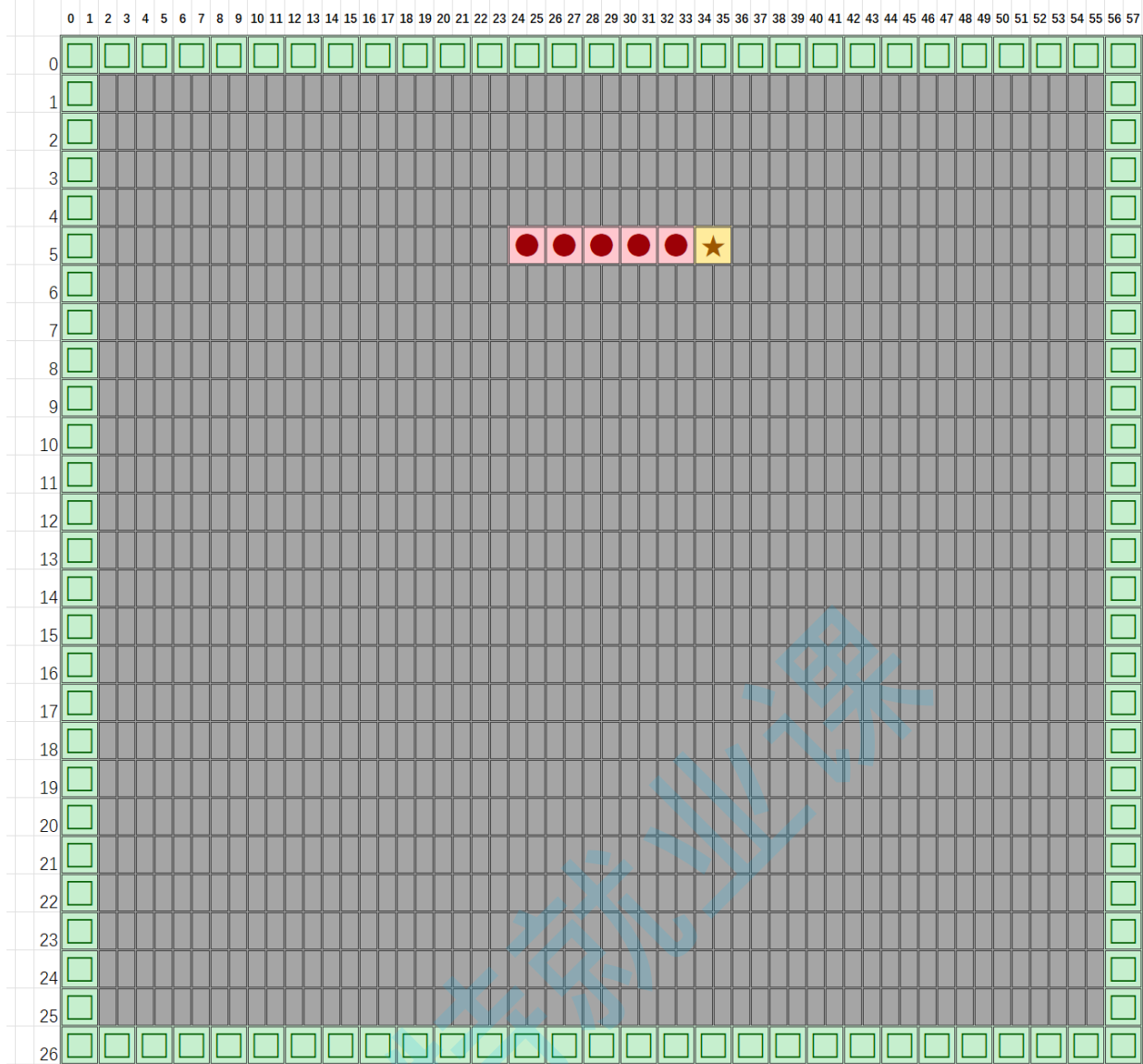
8.3.3.1 NextIsFood

```

1 //pSnakeNode psn 是下一个节点的地址
2 //pSnake ps 维护蛇的指针
3 int NextIsFood(pSnakeNode psn, pSnake ps)
4 {
5     return (psn->x == ps->_pFood->x) && (psn->y == ps->_pFood->y);
6 }

```

```
1 //pSnakeNode psn 是下一个节点的地址
2 //pSnake ps 维护蛇的指针
3 void EatFood(pSnakeNode psn, pSnake ps)
4 {
5     //头插法
6     psn->next = ps->_pSnake;
7     ps->_pSnake = psn;
8     pSnakeNode cur = ps->_pSnake;
9     //打印蛇
10    while (cur)
11    {
12        SetPos(cur->x, cur->y);
13        wprintf(L"%c", BODY);
14        cur = cur->next;
15    }
16    ps->_Score += ps->_Add;
17
18    //释放食物节点
19    free(ps->_pFood);
20    //创建新的食物
21    CreateFood(ps);
22 }
```



8.3.3.3 NoFood

将下一个节点头插入蛇的身体，并将之前蛇身最后一个节点打印为空格，放弃掉蛇身的最后一个节点

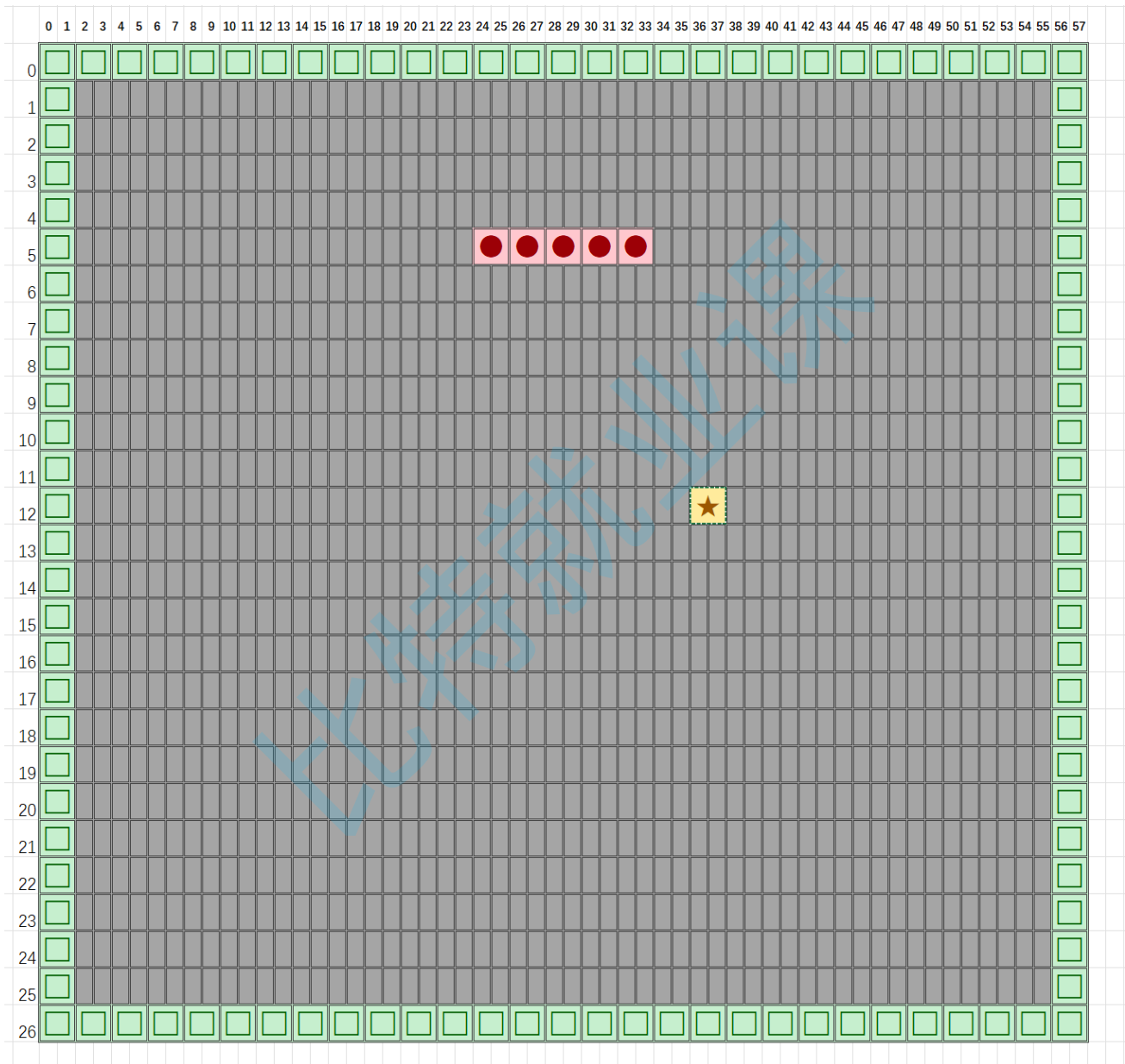
```

1 //pSnakeNode psn 是下一个节点的地址
2 //pSnake ps 维护蛇的指针
3 void NoFood(pSnakeNode psn, pSnake ps)
4 {
5     //头插法
6     psn->next = ps->_pSnake;
7     ps->_pSnake = psn;
8     pSnakeNode cur = ps->_pSnake;
9     //打印蛇
10    while (cur->next->next)
11    {
12        SetPos(cur->x, cur->y);
13        wprintf(L"%c", BODY);
14        cur = cur->next;
    
```

```

15     }
16
17     //最后一个位置打印空格，然后释放节点
18     SetPos(cur->next->x, cur->next->y);
19     printf(" ");
20     free(cur->next);
21     cur->next = NULL;
22 }

```



8.3.3.4 KillByWall

判断蛇头的坐标是否和强的坐标冲突

```

1 //pSnake ps 维护蛇的指针
2 int KillByWall(pSnake ps)
3 {
4     if ((ps->pSnake->x == 0)

```

```

5      || (ps->_pSnake->x == 56)
6      || (ps->_pSnake->y == 0)
7      || (ps->_pSnake->y == 26))
8      {
9          ps->_Status = KILL_BY_WALL;
10         return 1;
11     }
12     return 0;
13 }

```



8.3.3.5 KillBySelf

判断蛇头的坐标是否和蛇身体的坐标冲突

```

1  //pSnake ps 维护蛇的指针
2  int KillBySelf(pSnake ps)
3  {
4      pSnakeNode cur = ps->_pSnake->next;
5      while (cur)
6      {
7          if ((ps->_pSnake->x == cur->x)
8              && (ps->_pSnake->y == cur->y))
9          {
10             ps->_Status = KILL_BY_SELF;
11             return 1;

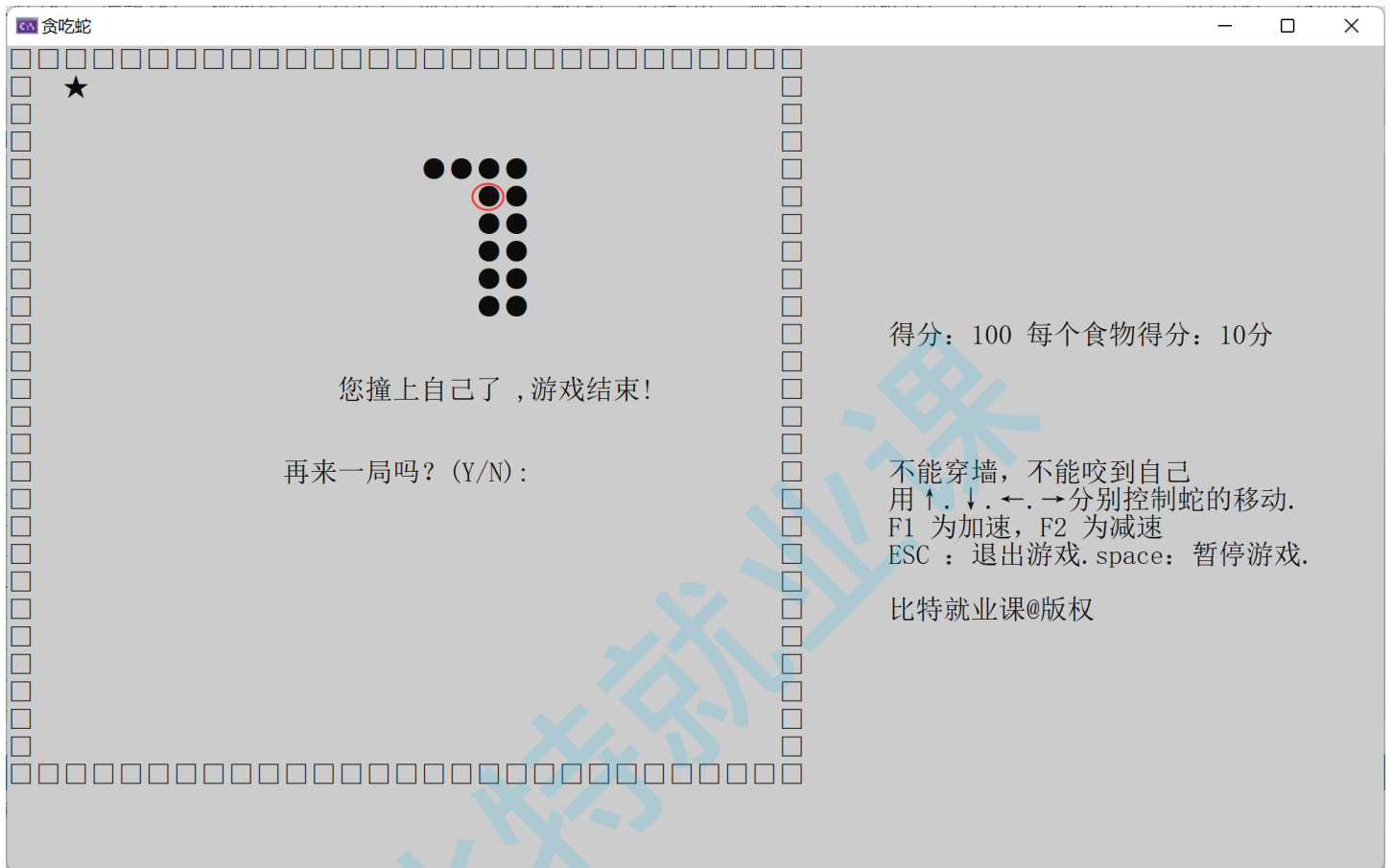
```

比特就业课-专注IT大学生就业的精品课程

```

12     }
13     cur = cur->next;
14 }
15 return 0;
16 }
17

```



8.4 游戏结束

游戏状态不再是OK（游戏继续）的时候，要告知游戏技术的原因，并且释放蛇身节点。

```

1 void GameEnd(pSnake ps)
2 {
3     pSnakeNode cur = ps->_pSnake;
4     SetPos(24, 12);
5     switch (ps->_Status)
6     {
7     case END_NOMAL:
8         printf("您主动退出游戏\n");
9         break;
10    case KILL_BY_SELF:
11        printf("您撞上自己了，游戏结束!\n");
12        break;
13    case KILL_BY_WALL:

```


比特就业课主页: <https://m.cctalk.com/inst/s9yewhfr>

```

14     printf("您撞墙了,游戏结束!\n");
15     break;
16 }
17
18 //释放蛇身的节点
19 while (cur)
20 {
21     pSnakeNode del = cur;
22     cur = cur->next;
23     free(del);
24 }
25 }

```

9. 参考代码

完整代码实现，分3个文件实现

test.cpp

```

1  #include "Snake.h"
2  #include <locale.h>
3
4  void test()
5  {
6      int ch = 0;
7      srand((unsigned int)time(NULL));
8
9      do
10     {
11         Snake snake = { 0 };
12         GameStart(&snake);
13         GameRun(&snake);
14         GameEnd(&snake);
15         SetPos(20, 15);
16         printf("再来一局吗? (Y/N):");
17         ch = getchar();
18         getchar(); //清理\n
19
20     } while (ch == 'Y');
21     SetPos(0, 27);
22 }
23
24 int main()
25 {
26     //修改当前地区为本地模式，为了支持中文宽字符的打印

```

```

27     setlocale(LC_ALL, "");
28     //测试逻辑
29     test();
30     return 0;
31 }

```

snake.h

```

1  #pragma once
2
3  #include <windows.h>
4  #include <time.h>
5  #include <stdio.h>
6
7  #define KEY_PRESS(VK) ((GetAsyncKeyState(VK)&0x1) ? 1 : 0)
8
9  //方向
10 enum DIRECTION
11 {
12     UP = 1,
13     DOWN,
14     LEFT,
15     RIGHT
16 };
17
18 //游戏状态
19 enum GAME_STATUS
20 {
21     OK, //正常运行
22     KILL_BY_WALL, //撞墙
23     KILL_BY_SELF, //咬到自己
24     END_NOMAL //正常结束
25 };
26
27 #define WALL L'□'
28 #define BODY L'●' //★○◇◆■
29 #define FOOD L'★' //★○◇◆■
30
31 //蛇的初始位置
32 #define POS_X 24
33 #define POS_Y 5
34
35 //蛇身节点

```

```
36 typedef struct SnakeNode
37 {
38     int x;
39     int y;
40     struct SnakeNode* next;
41 }SnakeNode, * pSnakeNode;
42
43 typedef struct Snake
44 {
45     pSnakeNode _pSnake; //维护整条蛇的指针
46     pSnakeNode _pFood; //维护食物的指针
47     enum DIRECTION _Dir; //蛇头的方向默认是向右
48     enum GAME_STATUS _Status; //游戏状态
49     int _Score; //当前获得分数
50     int _Add; //默认每个食物10分
51     int _SleepTime; //每走一步休眠时间
52 }Snake, * pSnake;
53
54
55
56 //游戏开始前的初始化
57 void GameStart(pSnake ps);
58
59 //游戏运行过程
60 void GameRun(pSnake ps);
61
62 //游戏结束
63 void GameEnd(pSnake ps);
64
65 //设置光标的坐标
66 void SetPos(short x, short y);
67
68 //欢迎界面
69 void WelcomeToGame();
70
71 //打印帮助信息
72 void PrintHelpInfo();
73
74 //创建地图
75 void CreateMap();
76
77 //初始化蛇
78 void InitSnake(pSnake ps);
79
80 //创建食物
81 void CreateFood(pSnake ps);
82
```

```
83 //暂停响应
84 void pause();
85
86 //下一个节点是食物
87 int NextIsFood(pSnakeNode psn, pSnake ps);
88
89 //吃食物
90 void EatFood(pSnakeNode psn, pSnake ps);
91
92 //不吃食物
93 void NoFood(pSnakeNode psn, pSnake ps);
94
95 //撞墙检测
96 int KillByWall(pSnake ps);
97
98 //撞自身检测
99 int KillBySelf(pSnake ps);
100
101 //蛇的移动
102 void SnakeMove(pSnake ps);
103
104 //游戏初始化
105 void GameStart(pSnake ps);
106
107 //游戏运行
108 void GameRun(pSnake ps);
109
110 //游戏结束
111 void GameEnd(pSnake ps);
```

snake.cpp

```
1 #include "Snake.h"
2
3 //设置光标的坐标
4 void SetPos(short x, short y)
5 {
6     COORD pos = { x, y };
7     HANDLE hOutput = NULL;
8     //获取标准输出的句柄(用来标识不同设备的数值)
9     hOutput = GetStdHandle(STD_OUTPUT_HANDLE);
10    //设置标准输出上光标的位置为pos
11    SetConsoleCursorPosition(hOutput, pos);
```

```
12 }
13
14 void WelcomeToGame()
15 {
16     SetPos(40, 15);
17     printf("欢迎来到贪吃蛇小游戏");
18     SetPos(40, 25); //让按任意键继续的出现的位置好看点
19     system("pause");
20     system("cls");
21     SetPos(25, 12);
22     printf("用 ↑ . ↓ . ← . → 分别控制蛇的移动， F3为加速， F4为减速\n");
23     SetPos(25, 13);
24     printf("加速将能得到更高的分数。 \n");
25     SetPos(40, 25); //让按任意键继续的出现的位置好看点
26     system("pause");
27     system("cls");
28 }
29
30 void CreateMap()
31 {
32     int i = 0;
33     //上(0,0)-(56, 0)
34     SetPos(0, 0);
35     for (i = 0; i < 58; i += 2)
36     {
37         wprintf(L"%c", WALL);
38     }
39     //下(0,26)-(56, 26)
40     SetPos(0, 26);
41     for (i = 0; i < 58; i += 2)
42     {
43         wprintf(L"%c", WALL);
44     }
45     //左
46     //x是0, y从1开始增长
47     for (i = 1; i < 26; i++)
48     {
49         SetPos(0, i);
50         wprintf(L"%c", WALL);
51     }
52     //x是56, y从1开始增长
53     for (i = 1; i < 26; i++)
54     {
55         SetPos(56, i);
56         wprintf(L"%c", WALL);
57     }
58 }
```

```
59
60
61 void InitSnake(pSnake ps)
62 {
63     pSnakeNode cur = NULL;
64     int i = 0;
65     //创建蛇身节点，并初始化坐标
66     //头插法
67     for (i = 0; i < 5; i++)
68     {
69         //创建蛇身的节点
70         cur = (pSnakeNode)malloc(sizeof(SnakeNode));
71         if (cur == NULL)
72         {
73             perror("InitSnake()::malloc()");
74             return;
75         }
76         //设置坐标
77         cur->next = NULL;
78         cur->x = POS_X + i * 2;
79         cur->y = POS_Y;
80
81         //头插法
82         if (ps->_pSnake == NULL)
83         {
84             ps->_pSnake = cur;
85         }
86         else
87         {
88             cur->next = ps->_pSnake;
89             ps->_pSnake = cur;
90         }
91     }
92
93     //打印蛇的身体
94     cur = ps->_pSnake;
95     while (cur)
96     {
97         SetPos(cur->x, cur->y);
98         wprintf(L"%c", BODY);
99         cur = cur->next;
100     }
101
102     //初始化贪吃蛇数据
103     ps->_SleepTime = 200;
104     ps->_Socre = 0;
105     ps->_Status = OK;
```

```
106     ps->_Dir = RIGHT;
107     ps->_Add = 10;
108 }
109
110
111
112 void CreateFood(pSnake ps)
113 {
114     int x = 0;
115     int y = 0;
116
117     again:
118         //产生的x坐标应该是2的倍数, 这样才可能和蛇头坐标对齐。
119         do
120         {
121             x = rand() % 53 + 2;
122             y = rand() % 25 + 1;
123         } while (x % 2 != 0);
124
125         pSnakeNode cur = ps->_pSnake; //获取指向蛇头的指针
126         //食物不能和蛇身冲突
127         while (cur)
128         {
129             if (cur->x == x && cur->y == y)
130             {
131                 goto again;
132             }
133             cur = cur->next;
134         }
135
136         pSnakeNode pFood = (pSnakeNode)malloc(sizeof(SnakeNode)); //创建食物
137         if (pFood == NULL)
138         {
139             perror("CreateFood::malloc()");
140             return;
141         }
142         else
143         {
144             pFood->x = x;
145             pFood->y = y;
146             SetPos(pFood->x, pFood->y);
147             wprintf(L"%c", FOOD);
148             ps->_pFood = pFood;
149         }
150 }
151
152 void PrintHelpInfo()
```

```
153 {
154     //打印提示信息
155     SetPos(64, 15);
156     printf("不能穿墙, 不能咬到自己\n");
157     SetPos(64, 16);
158     printf("用↑.↓.←.→分别控制蛇的移动.");
159     SetPos(64, 17);
160     printf("F1 为加速, F2 为减速\n");
161     SetPos(64, 18);
162     printf("ESC : 退出游戏.space: 暂停游戏.");
163     SetPos(64, 20);
164     printf("比特就业课@版权");
165 }
166
167 void pause() //暂停
168 {
169     while (1)
170     {
171         Sleep(300);
172         if (KEY_PRESS(VK_SPACE))
173         {
174             break;
175         }
176     }
177 }
178
179 //pSnakeNode psn 是下一个节点的地址
180 //pSnake ps 维护蛇的指针
181 int NextIsFood(pSnakeNode psn, pSnake ps)
182 {
183     return (psn->x == ps->_pFood->x) && (psn->y == ps->_pFood->y);
184 }
185
186 //pSnakeNode psn 是下一个节点的地址
187 //pSnake ps 维护蛇的指针
188 void EatFood(pSnakeNode psn, pSnake ps)
189 {
190     //头插法
191     psn->next = ps->_pSnake;
192     ps->_pSnake = psn;
193     pSnakeNode cur = ps->_pSnake;
194     //打印蛇
195     while (cur)
196     {
197         SetPos(cur->x, cur->y);
198         wprintf(L"%c", BODY);
199         cur = cur->next;
```



```
200     }
201     ps->_Socre += ps->_Add;
202
203     free(ps->_pFood);
204     CreateFood(ps);
205 }
206
207 //pSnakeNode psn 是下一个节点的地址
208 //pSnake ps 维护蛇的指针
209 void NoFood(pSnakeNode psn, pSnake ps)
210 {
211     //头插法
212     psn->next = ps->_pSnake;
213     ps->_pSnake = psn;
214     pSnakeNode cur = ps->_pSnake;
215     //打印蛇
216     while (cur->next->next)
217     {
218         SetPos(cur->x, cur->y);
219         wprintf(L"%c", BODY);
220         cur = cur->next;
221     }
222
223     //最后一个位置打印空格, 然后释放节点
224     SetPos(cur->next->x, cur->next->y);
225     printf(" ");
226     free(cur->next);
227     cur->next = NULL;
228 }
229
230 //pSnake ps 维护蛇的指针
231 int KillByWall(pSnake ps)
232 {
233     if ((ps->_pSnake->x == 0)
234         || (ps->_pSnake->x == 56)
235         || (ps->_pSnake->y == 0)
236         || (ps->_pSnake->y == 26))
237     {
238         ps->_Status = KILL_BY_WALL;
239         return 1;
240     }
241     return 0;
242 }
243
244 //pSnake ps 维护蛇的指针
245 int KillBySelf(pSnake ps)
246 {
```

```

247     pSnakeNode cur = ps->pSnake->next;
248     while (cur)
249     {
250         if ((ps->pSnake->x == cur->x)
251             && (ps->pSnake->y == cur->y))
252         {
253             ps->_Status = KILL_BY_SELF;
254             return 1;
255         }
256         cur = cur->next;
257     }
258     return 0;
259 }
260
261
262 void SnakeMove(pSnake ps)
263 {
264     //创建下一个节点
265     pSnakeNode pNextNode = (pSnakeNode)malloc(sizeof(SnakeNode));
266     if (pNextNode == NULL)
267     {
268         perror("SnakeMove():malloc()");
269         return;
270     }
271     //确定下一个节点的坐标, 下一个节点的坐标根据, 蛇头的坐标和方向确定
272     switch (ps->_Dir)
273     {
274         case UP:
275         {
276             pNextNode->x = ps->pSnake->x;
277             pNextNode->y = ps->pSnake->y - 1;
278         }
279         break;
280         case DOWN:
281         {
282             pNextNode->x = ps->pSnake->x;
283             pNextNode->y = ps->pSnake->y + 1;
284         }
285         break;
286         case LEFT:
287         {
288             pNextNode->x = ps->pSnake->x - 1;
289             pNextNode->y = ps->pSnake->y;
290         }
291         break;
292         case RIGHT:
293         {

```

```
294     pNextNode->x = ps->_pSnake->x + 2;  
295     pNextNode->y = ps->_pSnake->y;  
296 }  
297 break;  
298 }  
299  
300 //如果下一个位置就是食物  
301 if (NextIsFood(pNextNode, ps))  
302 {  
303     EatFood(pNextNode, ps);  
304 }  
305 else//如果没有食物  
306 {  
307     NoFood(pNextNode, ps);  
308 }  
309  
310 KillByWall(ps);  
311 KillBySelf(ps);  
312 }  
313  
314  
315  
316 void GameStart(pSnake ps)  
317 {  
318     //设置控制台窗口的大小, 30行, 100列  
319     //mode 为DOS命令  
320     system("mode con cols=100 lines=30");  
321     //设置cmd窗口名称  
322     system("title 贪吃蛇");  
323  
324     //获取标准输出的句柄(用来标识不同设备的数值)  
325     HANDLE hOutput = GetStdHandle(STD_OUTPUT_HANDLE);  
326  
327     //隐藏光标操作  
328     CONSOLE_CURSOR_INFO CursorInfo;  
329     GetConsoleCursorInfo(hOutput, &CursorInfo); //获取控制台光标信息  
330     CursorInfo.bVisible = false; //隐藏控制台光标  
331     SetConsoleCursorInfo(hOutput, &CursorInfo); //设置控制台光标状态  
332  
333     //打印欢迎界面  
334     WelcomeToGame();  
335     //打印地图  
336     CreateMap();  
337     //初始化蛇  
338     InitSnake(ps);  
339     //创造第一个食物  
340     CreateFood(ps);
```

```
341 }
342
343
344 void GameRun(pSnake ps)
345 {
346     //打印右侧帮助信息
347     PrintHelpInfo();
348     do
349     {
350         SetPos(64, 10);
351         printf("得分: %d ", ps->_Score);
352         printf("每个食物得分: %d分", ps->_Add);
353         if (KEY_PRESS(VK_UP) && ps->_Dir != DOWN)
354         {
355             ps->_Dir = UP;
356         }
357         else if (KEY_PRESS(VK_DOWN) && ps->_Dir != UP)
358         {
359             ps->_Dir = DOWN;
360         }
361         else if (KEY_PRESS(VK_LEFT) && ps->_Dir != RIGHT)
362         {
363             ps->_Dir = LEFT;
364         }
365         else if (KEY_PRESS(VK_RIGHT) && ps->_Dir != LEFT)
366         {
367             ps->_Dir = RIGHT;
368         }
369         else if (KEY_PRESS(VK_SPACE))
370         {
371             pause();
372         }
373         else if (KEY_PRESS(VK_ESCAPE))
374         {
375             ps->_Status = END_NOMAL;
376             break;
377         }
378         else if (KEY_PRESS(VK_F3))
379         {
380             if (ps->_SleepTime >= 50)
381             {
382                 ps->_SleepTime -= 30;
383                 ps->_Add += 2;
384             }
385         }
386         else if (KEY_PRESS(VK_F4))
387         {
```

```
388     if (ps->_SleepTime < 350)
389     {
390         ps->_SleepTime += 30;
391         ps->_Add -= 2;
392         if (ps->_SleepTime == 350)
393         {
394             ps->_Add = 1;
395         }
396     }
397 }
398 //蛇每次一定之间要休眠的时间,时间短,蛇移动速度就快
399 Sleep(ps->_SleepTime);
400 SnakeMove(ps);
401
402 } while (ps->_Status == OK);
403 }
404
405 void GameEnd(pSnake ps)
406 {
407     pSnakeNode cur = ps->_pSnake;
408     SetPos(24, 12);
409     switch (ps->_Status)
410     {
411     case END_NOMAL:
412         printf("您主动退出游戏\n");
413         break;
414     case KILL_BY_SELF:
415         printf("您撞上自己了,游戏结束!\n");
416         break;
417     case KILL_BY_WALL:
418         printf("您撞墙了,游戏结束!\n");
419         break;
420     }
421
422     //释放蛇身的节点
423     while (cur)
424     {
425         pSnakeNode del = cur;
426         cur = cur->next;
427         free(del);
428     }
429 }
```

参考：[汉字字符集编码查询](#); [中文字符集编码](#): GB2312、BIG5、GBK、GB18030、Unicode

比特就业课