

《创新训练课B》课程设计报告

学院： 机电工程与自动化学院

题目： **贪吃蛇的设计**

班级： **自动化五班**

姓名： **葛旭 黄弋斌**

学号： **190320517 190320514**

教师： **吴晓军**

上交日期： 2020年10月31日

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
|  | v1.0 | 第一版仅运用不断printf在控制台进行操作，较卡顿。 | 黄弋斌 |
|  | V2.0 | 第二版用了easyx库，相对界面友好，流畅 | 葛旭 |
|  |  |  |  |
|  |  |  |  |

**目 录**

[1 引言 1](#_Toc519846913)

[1.1编写目的 1](#_Toc519846914)

[1.2背景 1](#_Toc519846915)

[1.3定义 1](#_Toc519846916)

[1.4参考资料 1](#_Toc519846917)

[2 任务概述 1](#_Toc519846918)

[3 需求分析 1](#_Toc519846919)

[3.1 用户需求分析 1](#_Toc519846920)

[3.2 运行环境 1](#_Toc519846921)

[4 功能及操作介绍 1](#_Toc519846922)

[4.1 操作 1](#_Toc519846923)

[4.2 功能 1](#_Toc519846924)

[5 系统设计 2](#_Toc519846925)

[5.1 总体架构设计 2](#_Toc519846926)

[5.2 模块分析与设计 2](#_Toc519846927)

[5.3软件结构（流程图） 2](#_Toc519846928)

[6 调试与测试 2](#_Toc519846929)

[6.1调试过程 2](#_Toc519846930)

[6.2测试结果 2](#_Toc519846931)

[7 编程中遇到的问题 2](#_Toc519846932)

[7.1 问题1 2](#_Toc519846933)

[7.2 问题2 2](#_Toc519846934)

[7.3 问题3 2](#_Toc519846935)

[8分析总结与心得体会 2](#_Toc519846936)

# 1 引言

## 1.1编写目的

市面上有很多种贪吃蛇游戏，但是最原始的版本仍然是吃食物变长的版本，既然学了c语言，就充分发挥它的用处，不再仅仅局限于控制台应用，而是创造一个GUI界面，让自己发挥创造力，编写自己想要的贪吃蛇游戏，创造独有的模式，尽量做到界面友好，运行流畅，规则简明。

## 1.2背景

学习了C语言一个学期的情况下，重新拾起来编写一个贪吃蛇程序。以前没有接触过图形界面，要自己到网上查找一些库，了解相关的使用操作。在python这种简洁的语言盛行的时代，用c语言写小游戏虽然略显臃肿但是运行效率依然很高。

## 1.3定义

一个依靠键盘选择模式并控制小蛇移动的游戏程序。

## 1.4参考资料

《C高级语言程序设计》《C pimer》

# 2 任务概述

设计一个贪吃蛇游戏，用连续的格子代替蛇，独立的格子代替食物或者障碍物。碰到食物蛇长加一，碰到墙壁或者障碍物即死亡。障碍物和蛇的颜色可以自己设定。

# 3 需求分析

## 3.1 用户需求分析

需要一些代表物：蛇，食物，砖块等。

需要设定模式：改变速度。

需要键盘操控。

需要满足贪吃蛇基本规则。

## 3.2 运行环境

Windows10+安装有easyx库的visual studio。

# 4 功能及操作介绍

## 4.1 操作

按住WASD即可实现转向。

## 4.2 功能

开始界面按1，2或者3可以调整蛇的移动速度。

吃到食物蛇可以变长，蛇不能朝与头相反的方向移动。

蛇撞到墙会死，撞到自己会死。

# 5 系统设计

## 5.1 总体架构设计

1，我们需要一张地图，图形库初始化的窗口是自带坐标的（左上角为0，0）。  
2，我们需要一条蛇，这条蛇由蛇头和蛇身组成（这里我们初始化三节蛇）。  
3，我们需要食物，并且在蛇吃掉食物之后将蛇的身体变长，而且重新生成一个食物。  
4，蛇需要移动，这应该是最难实现的（实现蛇身坐标的传值，除蛇头以外，每一节蛇的坐标都是上节蛇的上一次运动之前的坐标）。  
5，蛇撞到墙或者撞到自己的身体就会死亡（判断坐标是否重合就可以）。  
6，我们需要能用键盘控制蛇的运动方向，这个会和蛇的移动有一些联系。

## 5.2 模块分析与设计

1.初始化游戏数据：init()，用枚举变量定义空地，蛇，墙，食物等

1. **void** init()
2. {
3. srand((unsigned)time(NULL));    //随机种子
4. setbkcolor(WHITE);              //设置背景颜色
6. memset(map, SPACE, **sizeof**(map));//初始化map数组为0即SPACE
7. //每一行的 第一个 和 最后一个 是墙
8. **for** (**int** i = 0; i < ROW; i++)
9. {
10. map[i][0] = map[i][COL - 1] = WALL;
11. }
12. //每一列的 第二个 和 倒数第二 个是墙
13. **for** (**int** j = 1; j < COL - 1; j++)
14. {
15. map[0][j] = map[ROW - 1][j] = WALL;
16. }
17. //定义蛇头和蛇的身体
18. map[3][5] = HEAD;
19. map[3][4] = map[3][3] = SNAKE;
20. //初始化蛇
21. SnakeSize = 3;  //蛇 长
22. SnakeDir = 'D'; //蛇方向向右
23. snake[0].X = 3;
24. snake[0].Y = 5;
25. snake[1].X = 3;
26. snake[1].Y = 4;
27. snake[2].X = 3;
28. snake[2].Y = 3;
29. addwall();
30. addfood();
31. }
32. 开始界面：start()，打印三个速度模式
33. **void** start()
34. {
35. setbkcolor(WHITE);      //设置窗口背景色为白色
36. cleardevice();          //刷新屏幕
38. setbkmode(TRANSPARENT); //设置字体背景色为透明
39. settextcolor(BLACK);        //设置字体颜色为黑色
40. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*游戏规则\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
41. outtextxy(290, 80, L"功能选择:");
42. outtextxy(280, 150, L"1.低速模式");
43. outtextxy(280, 200, L"2.中速模式");
44. outtextxy(280, 250, L"3.高速模式");
45. outtextxy(200, 280, L"数字键 1,2,3 选择模式，Enter键进入游戏");
46. outtextxy(200, 300, L"字母键 W,S,A,D 方向键 上下左右 控制方向");
47. outtextxy(130, 350, L"相关内容：");
48. outtextxy(160, 380, L"作者：葛旭，黄弋斌");
49. }
50. 选择：chose(),选择其中一个速度模式
51. **void** chose()
52. {
53. **switch** (\_getch())
54. {
55. **case** '1':
56. start();
57. outtextxy(260, 150, L"->");
58. level = 1;
59. **break**;
60. **case** '2':
61. start();
62. outtextxy(260, 200, L"->");
63. level = 2;
64. **break**;
65. **case** '3':
66. start();
67. outtextxy(260, 250, L"->");
68. level = 3;
69. **break**;
70. **case** 13:
71. **return**;
72. **break**;
73. }
74. }
75. 循环检索并填充:DrawMap()，根据每一个位置的值进行绘图，给不同的模块填充不同的颜色。
76. **void** DrawMap()
77. {
78. BeginBatchDraw();   //开始绘图
79. setbkcolor(RGB(247, 238, 214)); //设置背景颜色为白色
80. settextcolor(RGB(238, 0, 0));
81. cleardevice();      //清屏
82. **WCHAR** arr[10];      //保存成绩
83. //WCHAR level[10];
84. wsprintf(arr, L"总分：%d", score); //将成绩格式化输出到字符串arr中
85. //wsprintf(level, L"速度等级：%d", level);
86. outtextxy(0, 0, arr);               //显示成绩
87. //outtextxy(150, 0, level);
88. **for** (**int** y = 0; y < ROW; y++)        //y轴方向向下
89. {
90. **for** (**int** x = 0; x < COL; x++)    //x轴方向向右
91. {
92. **switch** (map[y][x])
93. {
94. **case** SPACE:
95. **break**;
96. **case** WALL:
97. setlinecolor(BLACK);
98. setfillcolor(RGB(238, 233, 233));   //灰色
99. fillrectangle(x \* 10, y \* 10 + 20, x \* 10 + 10, y \* 10 + 30);
100. **break**;
101. **case** SNAKE:
102. setlinecolor(RGB(255, 165, 0));     //绿色
103. setfillcolor(RGB(255, 165, 0));
104. fillrectangle(x \* 10, y \* 10 + 20, x \* 10 + 10, y \* 10 + 30);
105. **break**;
106. **case** HEAD:
107. //画蛇头
108. setfillcolor(RGB(0, 0, 255));       //蓝  0 0 255
109. solidrectangle(x \* 10, y \* 10 + 20, x \* 10 + 10, y \* 10 + 30);
110. **break**;
111. **case** FOOD:
112. setfillcolor(RGB(255, 0, 0));           //红色
113. solidrectangle(x \* 10, y \* 10 + 20, x \* 10 + 10, y \* 10 + 30);
114. **break**;
115. **default**:
116. **break**;
117. }
118. }
119. }
120. EndBatchDraw();
121. }
122. **void** addfood()
123. {
124. **int** row, col;
125. **do**
126. {
127. row = rand() % (ROW - 1) + 1;
128. col = rand() % (COL - 1) + 1;
129. } **while** (map[row][col] != SPACE && map[row][col] != WALL);
130. map[row][col] = FOOD;
131. }
132. **void** addwall()
133. {
134. **int** row, col;
135. **int** i = 0;
136. **for** (i = 0; i < BLOCK; i++)
137. {
138. **do**
139. {
140. row = rand() % (ROW - 1) + 1;
141. col = rand() % (COL - 1) + 1;
142. } **while** (map[row][col] != SPACE && map[row][col] != WALL);
143. map[row][col] = WALL;
144. }
145. }
146. 添加食物：addfood()
147. **void** addfood()
148. {
149. **int** row, col;
150. **do**
151. {
152. row = rand() % (ROW - 1) + 1;
153. col = rand() % (COL - 1) + 1;
154. } **while** (map[row][col] != SPACE && map[row][col] != WALL);
155. map[row][col] = FOOD;
156. }
157. 添加墙体：addwall()
158. **void** addwall()
159. {
160. **int** row, col;
161. **int** i = 0;
162. **for** (i = 0; i < BLOCK; i++)
163. {
164. **do**
165. {
166. row = rand() % (ROW - 1) + 1;
167. col = rand() % (COL - 1) + 1;
168. } **while** (map[row][col] != SPACE && map[row][col] != WALL);
169. map[row][col] = WALL;
170. }
171. }

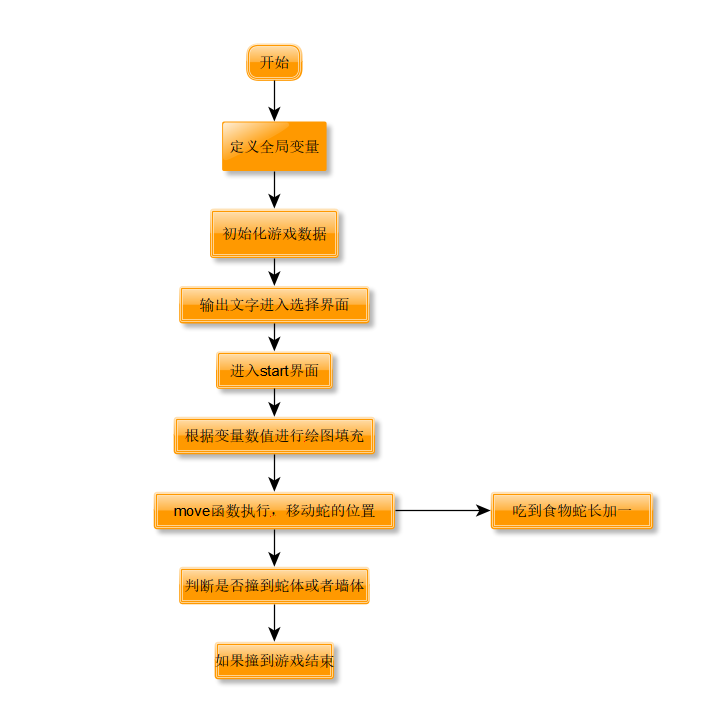
7.改变方向：ChangeDir(),根据按键字母改变方向定义

1. **void** ChangeDir()
2. {
3. **switch** (\_getch())
4. {
5. **case**'A':
6. **case**'a':
7. **case** 75:
8. **if** (SnakeDir != 'D') SnakeDir = 'A';    //蛇不能后退
9. **break**;
10. **case**'D':
11. **case**'d':
12. **case** 77:
13. **if** (SnakeDir != 'A') SnakeDir = 'D';
14. **break**;
15. **case**'W':
16. **case**'w':
17. **case** 72:
18. **if** (SnakeDir != 'S') SnakeDir = 'W';
19. **break**;
20. **case**'S':
21. **case**'s':
22. **case** 80:
23. **if** (SnakeDir != 'W') SnakeDir = 'S';
24. **break**;
25. **case** 32:
26. \_getch();
27. **break**;
28. **default**:
29. **break**;
30. }
31. }

8.根据方向移动蛇体并判断下一个是否撞到蛇体或墙体：move(),

1. **void** move()
2. {
3. COORD next;     //蛇头的下一个位置
4. **switch** (SnakeDir)
5. {
6. **case**'A':
7. next.X = snake[0].X;
8. next.Y = snake[0].Y - 1;
9. **break**;
10. **case**'W':
11. next.X = snake[0].X - 1;
12. next.Y = snake[0].Y;
13. **break**;
14. **case**'D':
15. next.X = snake[0].X;
16. next.Y = snake[0].Y + 1;
17. **break**;
18. **case**'S':
19. next.X = snake[0].X + 1;
20. next.Y = snake[0].Y;
21. **break**;
22. **default**:
23. **break**;
24. }
26. **switch** (map[next.X][next.Y])
27. {
28. **case** SPACE://直接移动
29. map[snake[SnakeSize - 1].X][snake[SnakeSize - 1].Y] = SPACE;//地图蛇尾所在地置空
30. **for** (**int** i = SnakeSize - 1; i > 0; i--)                      //蛇尾到蛇头整体移动一位
31. {
32. snake[i] = snake[i - 1];
33. }
34. map[snake[0].X][snake[0].Y] = SNAKE;                        //蛇头置 蛇
35. snake[0] = next;                                            //将下一个位置赋值给蛇头
36. map[snake[0].X][snake[0].Y] = HEAD;                         //设置头
37. **break**;
38. **case** WALL:
39. MessageBox(GetHWnd(), L"游戏结束", L"SORRY", MB\_OK);
40. outtextxy(320, 200, L"是否要结束？");
41. outtextxy(320, 280, L"结束按N/n");
42. **if** (\_getch() == 'N' || \_getch() == 'n')
43. {
44. exit(0);
45. }
46. **break**;
47. **case** SNAKE:
48. MessageBox(GetHWnd(), L"游戏结束", L"SORRY", MB\_OK);
49. outtextxy(320, 200, L"是否要结束？");
50. outtextxy(320, 280, L"结束按N/n");
51. **if** (\_getch() == 'N' || \_getch() == 'n')
52. {
53. exit(0);
54. }
55. **break**;
56. **case** FOOD://食物              蛇尾不变
57. **for** (**int** i = SnakeSize; i > 0; i--)                          //蛇尾到蛇头整体移动一位
58. {
59. snake[i] = snake[i - 1];
60. }
61. map[snake[0].X][snake[0].Y] = SNAKE;                        //蛇头 置 蛇
62. snake[0] = next;                                            //将下一个位置赋值给蛇头
63. score++;            //分数加一
64. (SnakeSize)++;      //蛇尺度加一
65. map[snake[0].X][snake[0].Y] = HEAD;                         //地图上重置蛇头
66. addfood();
67. **break**;
68. **default**:**break**;
69. }
70. }

## 5.3软件结构（流程图）



# 6 调试与测试

## 6.1调试过程

将主框架与各部分模块单独分写出来，例如生成食物，生成墙体，生成蛇，填充方块等，各个模块确认无误后再组合到一起，用全局变量进行交互，逻辑更清晰，减少了工作量。

## 6.2测试结果

各代码块独立运行正常，组合到一起联合运行正常。

# 7 编程中遇到的问题

## 7.1 问题1

界面显示不流畅，经常卡顿，并且不美观

解决办法：换用easyx库，用库函数以及RGB调色，使得界面尽量简洁美观。

## 7.2 问题2

定义墙体，蛇体，食物等元素繁琐而不容易记忆。

解决办法：用枚举变量依次定义，并用字母代替，省去了宏定义的繁琐。

## 7.3 问题3

蛇体的移动逻辑错误。

解决办法：将移动分为两个模块，一个用来后端定义方向，一个用于前端根据定义依次移动蛇体(蛇体的索引为二维数组)。

## 7.4 问题4

使用链表经常出现各种bug

解决办法：放弃单向链表改用二维数组，大大减少了出现错误的可能性。

# 8分析总结与心得体会

在设计这个程序中我主要学会了如何运用以下有关C语言的知识:  
1)函数定义是要做到顾名思义是很重要的，它对读程序的人正确认识程序  
十分重要，在修改这个程序的过程中也能很快找到程序各模块的作用，大大增  
加了程序的可读性。  
3)在做程序的时候先列框架，将这个程序所要达到的目的(功能)分析出  
来，选择正确的数据结构然后在将程序模块化，按照模块编写函数更加简单合理。  
4)我还了解了很多的库函数的作用，如字符串函数中有很多对字符串进行  
处理的函数，起功能我都有所了解。  
5)学会了有关头文件的使用方法，及系统调配问题的解决方法等。  
6)在调试过程中，遇到了很多的问题，比如，因为使用的刷屏函数，所以屏幕经常出  
现闪烁的情况,但是通过我和我的伙伴们的共同努力和学习，通过网上和书籍的查阅，我们也得到了很多的收获，我们的小组也在中真正的学会了很多的东西，这次实训给我们带来了很大的收获。可以按照程序运行的错误提示对原程序进行修改,在调试过程中有时也会遇到不懂的问题,我去图书馆或上网查阅- -些资料或者是向老师请教也解决了对源程序一一修改直到运行成功。