**《数据结构与算法设计》大作业**

**题目名称： 餐厅点餐系统**

**班级： 信安11班**

**姓名： 学号：**

**合作者姓名： 无 学号：**

**日期： 2022，12，17**

**餐厅点餐系统**

**一、选题**

背景：传统餐饮行业，点餐过程都是由餐厅工作人员手动记录完成，即客人进入餐厅后，选好座位后服务员等待客人点餐完成，并将客人点餐的内容送到厨房。此过程消耗大量人力和财力成本，且在忙碌时期，客人的体验感也不佳。

为了解决以上问题，使用C语言实现了一个餐厅点餐系统的雏形，这减轻了服务员的工作量，便于餐厅后厨管理出菜，提高了服务效率，有效减少了因为人工点菜而会产生的错误，提高了用户体验和服务质量，也为餐厅减少了很大的人工成本支出。

**二、需求分析**

软件功能：

1. 主菜单可以进入顾客选择菜单和商家管理员菜单
2. 管理员菜单，输入登入密码，实现今日菜品的录入（编号、菜名、价格）和今日菜品的删除
3. 管理员可以查看本餐厅有多少餐桌在使用，以及各桌的点餐情况
4. 在顾客结账后可以清除掉餐桌信息，方便之后来的客人使用
5. 顾客可以看到菜单的显示，选择菜品
6. 每桌客人记账功能，结算功能
7. 选择菜品后可看自己餐桌已经选定的菜品，可考虑加菜和减菜
8. 数据的保存和读取

此处采用预先设置好的“菜单.txt”和“餐桌.txt”文件来读取初始信息，并对所有功能进行调试。

**三、概要设计（总体设计）**

**设计方案论证：**

根据作业要求及语言限制，此处**采用C语言**来实现该点餐系统；使用C语言的**文档读写功能**来对信息进行储存和读取。

**数据容器选择：**

1. 使用数组。如果使用数组，可以更加方便的索引。但代价便是需要在定义数组时就开一个非常大的内存，大多数情况下并不能完全利用，内存利用率低，并且这个“非常大”的连续内存也可能无法满足需求；由于餐厅点餐系统是一个实时更新的系统，需要实现数据的增删，这需要移动大量元素的内存地址，造成比较多的时间消耗；如果进行系统升级，扩充容量，使用数组进行储存难以扩展。
2. 使用链表。链表在索引方面具有一定缺点，但其基本符合该系统的要求。链表方便对菜单和客户点菜的增删操作，可以很好利用内存空间，并具有可扩展性。

因此链表作为数据容器，进行数据储存与处理。

**数据结构的设计：**

此处针对菜单上的菜品和客户的餐桌设计两个链表数据结构，分别为：

菜品菜单链表：

1. **typedef** **struct** Menu
2. {
3. **int** code;           // 菜品编码
4. **char** name[100];     // 菜品名称
5. **int** price;          // 菜品价格
6. **struct** Menu \*next;
7. }MENU;

此链表节点用来储存菜品信息。

餐桌链表：

1. **typedef** **struct** Desk
2. {
3. **int** deskCode;                   // 桌子的编号
4. **int** peopleNum;                  // 一桌的人数
5. **int** purchasedDishNum;           // 购买菜品的数量
6. MENU \*purchasedDish;            // 本桌购买菜品
7. **int** totalPrice;                 // 购买菜品总价
8. **struct** Desk \*next;
9. }DESK;

此链表用来储存点餐桌的信息。

且此处定义以下全局变量：

1. MENU \*menu;     // 菜单信息
2. DESK \*desk;     // 菜桌信息
3. **int** count;      // 总菜品数

**各模块描述：**

* **文件操作部分**

1. 菜单操作
2. **void SaveMenu();**

将链表中的菜单信息保存到文件“菜单.txt”中

1. **void ReadMenu();**

将文件“菜单.txt”中的菜单信息读取到链表中

1. **void MenuOutput();**

打印当前链表中的菜品信息

1. 餐桌操作
2. **void SaveDesk();**

将链表中的餐桌信息保存到文件“餐桌.txt”中

1. **void ReadDesk();**

将文件“餐桌.txt”中的餐桌信息读取到链表中

* 点餐与管理员操作界面

1. 用户点餐

**void ClientMenu();**

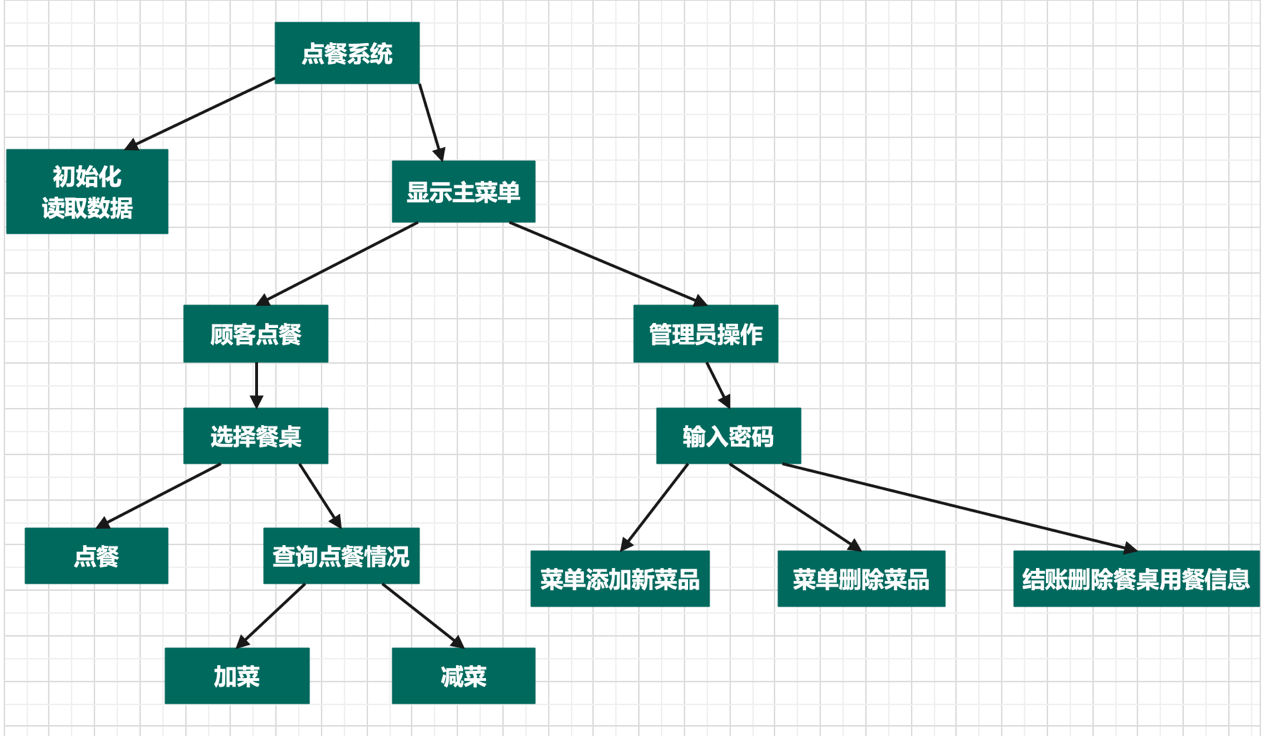
该函数下面包含用户可执行的**点餐**和**查询**操作。点餐即根据给出的菜单选择想要购买的菜品；查询即用户查看自己现在拥有的菜品，并可在此基础上对现有菜品进行添加和删除操作。

1. 管理员操作

**void AdminMenu();**

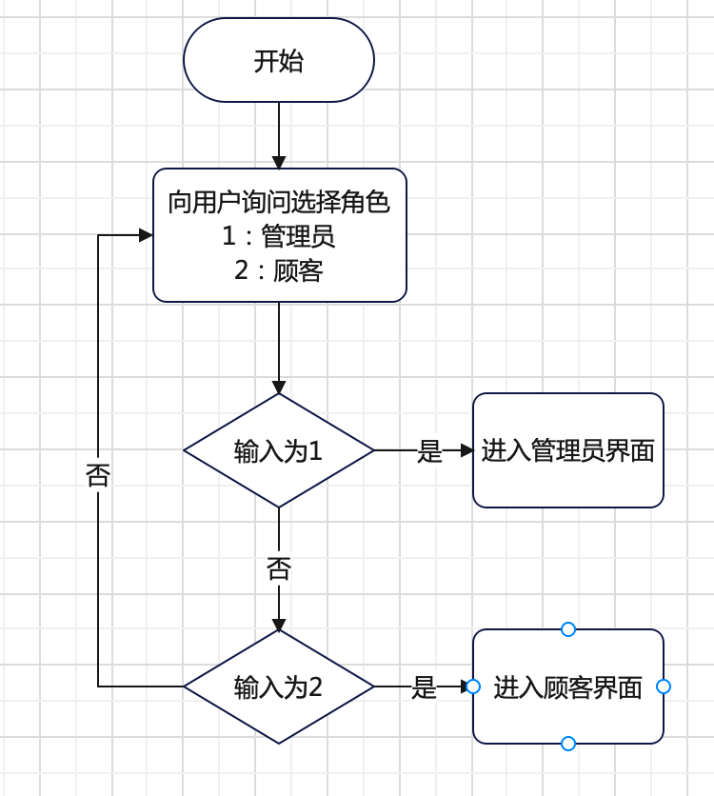
该函数下面包含管理员可执行的**添加菜品**、**删除菜品**、**结账**和**查看订单**操作。添加菜品，即对菜单中售卖的菜品进行添加；删除菜品，即对菜单中售卖的菜品进行删除；结账，即用户用餐完毕，支付账单以后，删除用户用餐所在桌的信息；查看订单，即对当前各个餐桌的用餐情况进行查看。

**软件结构图：**

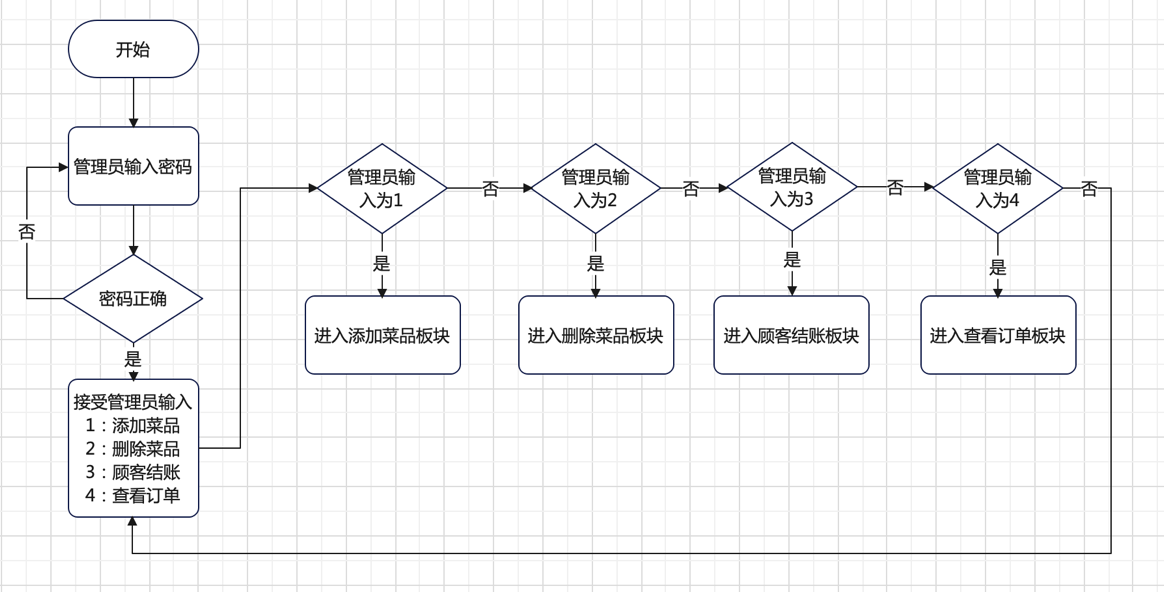


**四、详细设计**

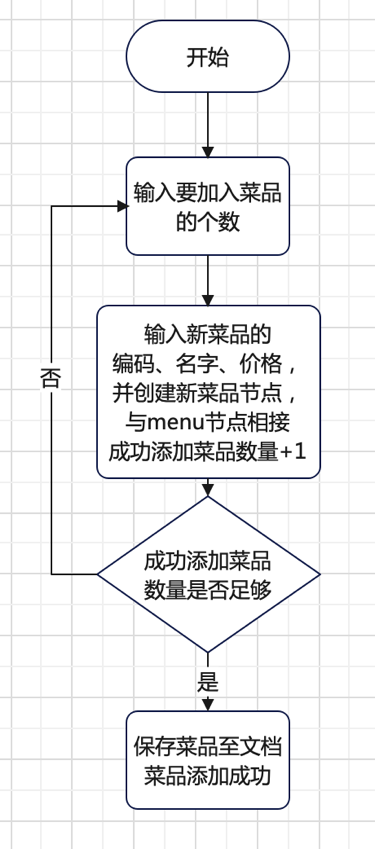
1. 主菜单板块



1. 管理员界面板块

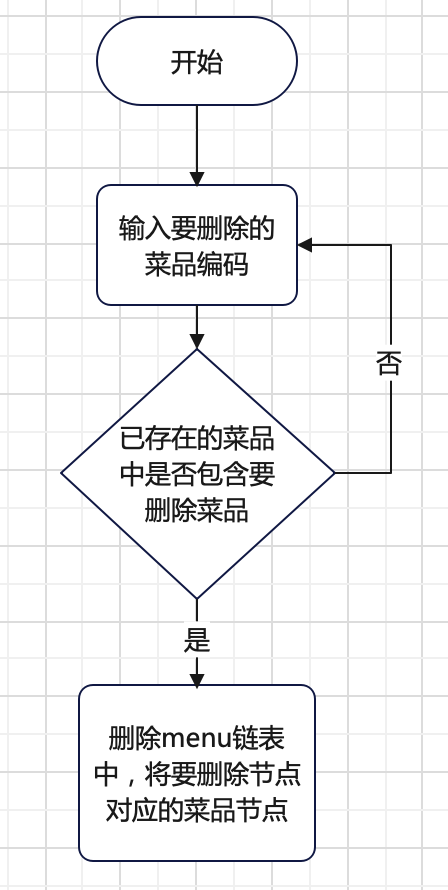


* + 添加菜品板块



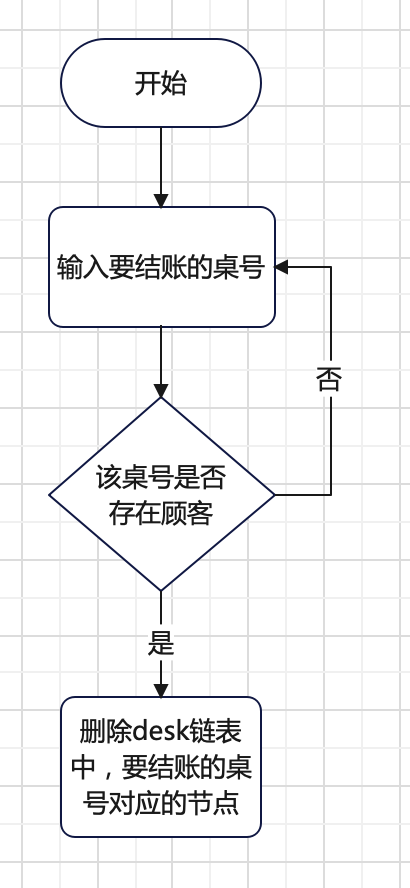
添加部分代码：

1. printf("请输入添加菜品数:\n");
2. scanf("%d", &n);
4. MENU \*s;
5. **for**(**int** i = 0; i < n; i ++ )    // 增加链表节点
6. {
7. s = (MENU\*)malloc(**sizeof**(MENU));
9. printf("----------------------------------------\n");
10. 请输入第%d个菜品编码:\n", i + 1);
11. scanf("%d", &s -> code);
12. printf("请输入第%d个菜品名:\n",i+1);
13. scanf("%s", s -> name);
14. printf("请输入价格\n");
15. scanf("%d", &s -> price);
16. count ++ ;      // 总菜品数+1
18. s -> next = menu;   // 将新节点放入链表尾
19. menu = s;
20. }
21. SaveMenu();  // 保存菜单
    * 删除菜品板块



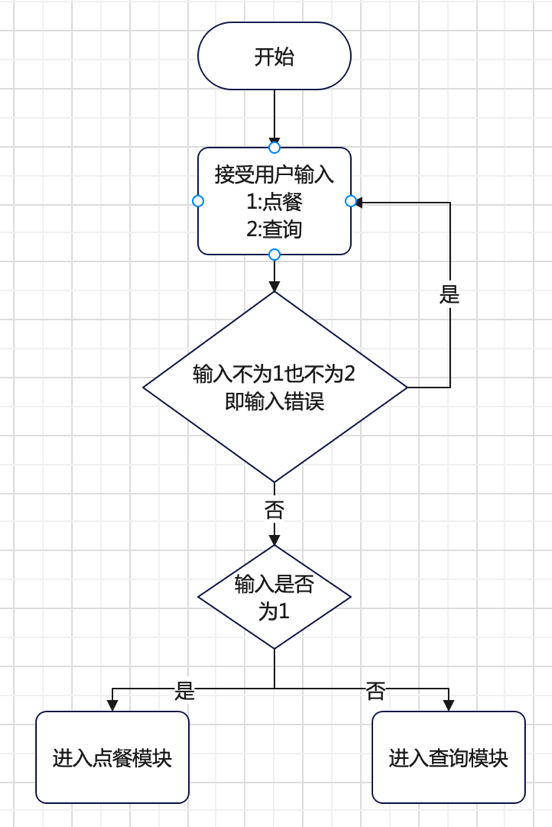
删除部分代码：

1. printf("输入要删除的菜品编码:\n");
2. scanf("%d", &code);
4. **int** flag = 0;
5. **if**(menu -> code == code)    // 若为首节点，直接删除首节点
6. {
7. menu = menu -> next;
8. flag = 1;
9. count -- ;
10. }
11. **else**    // 否则使用一个辅助节点，对菜品链表节点进行删除
12. **for**(MENU \*i = menu; i; i = i -> next)
13. {
14. MENU \*j = NULL;
15. **if**(i -> next)
16. j = i -> next;
17. **if**(j -> code == code)
18. {
19. count -- ;
20. i -> next = j -> next;
21. j -> next = NULL;
22. free(j);     // 释放内存
24. flag = 1;
25. **break**;
26. }
27. }
    * 顾客结账板块

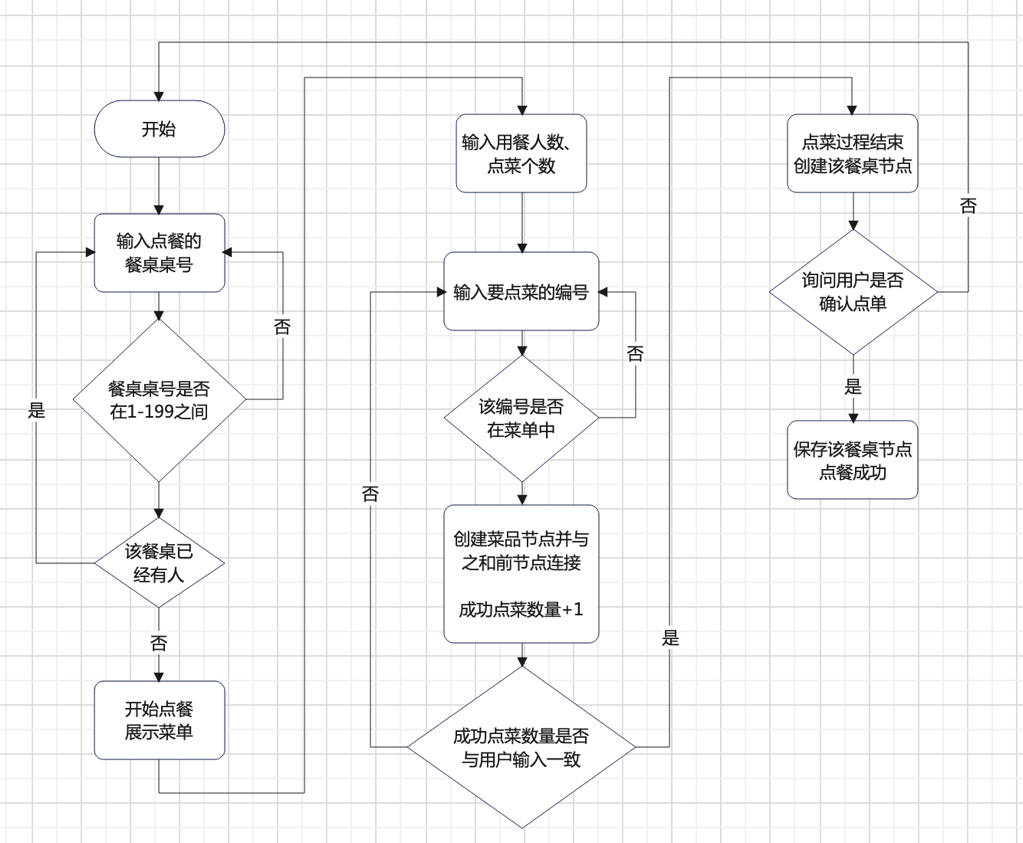


结账部分代码：

1. printf("请输入结账的桌号：\n");
2. scanf("%d", &code);
4. **int** flag = 0;
5. **if**(desk -> deskCode == code)    // 首节点直接将其置后
6. {
7. desk = desk -> next;
8. flag = 1;
9. }
10. **else**    // 否则使用一个辅助节点对节点进行删除
11. **for**(DESK \*i = desk; i; i = i -> next)
12. {
13. DESK \*j = NULL;
14. **if**(i -> next)
15. j = i -> next;
16. **if**(j -> deskCode == code)
17. {
18. i -> next = j -> next;
19. j -> next = NULL;
21. free(j);
23. flag = 1;
24. **break**;
25. }
26. }
27. 顾客界面

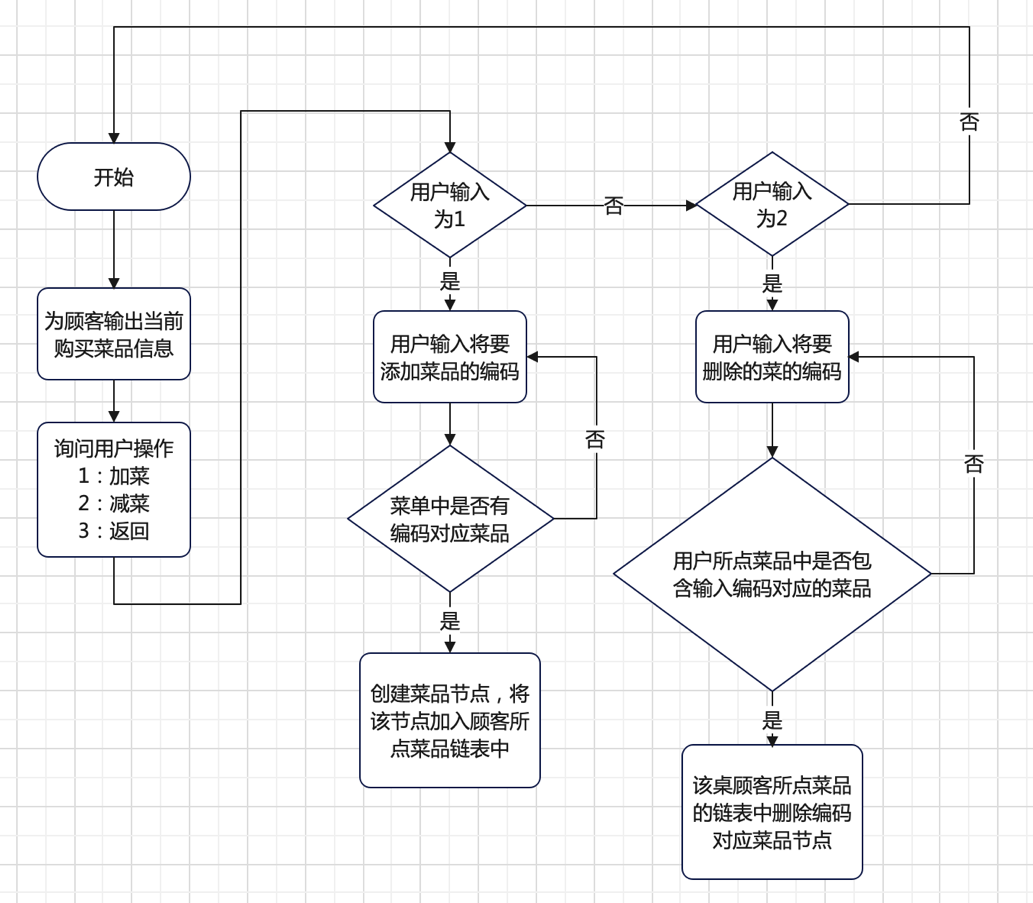


* + 点餐模块



点餐模块主要操作代码：

1. DESK \*s;        // 为点餐信息创建desk节点
2. s = (DESK\*)malloc(**sizeof**(DESK));
3. // 更新s节点信息
4. s -> deskCode = deskcode;
5. s -> purchasedDish = NULL;  // 用于复制菜品信息
6. printf("请你输入用餐人数：\n");
7. scanf("%d", &s -> peopleNum);
8. printf("请你输入点菜的个数：\n");
9. scanf("%d", &s -> purchasedDishNum);
10. s -> totalPrice = 0;
12. **for**(**int** i = 0; i < s -> purchasedDishNum; i ++ )
13. {
14. **int** dishcode;
15. printf("----------------------------------------\n");
16. printf("请输入您要购买的第%d个菜的编码:\n", i + 1);
17. scanf("%d", &dishcode);
18. MENU \*t;  // 创建链表使用
19. MENU \*k = (MENU\*)malloc(**sizeof**(MENU));  // 用于复制菜品信息
20. **for**(MENU \*j = menu; j; j = j -> next)
21. **if**(j -> code == dishcode)
22. {
23. flag2 = 1;
24. k -> code = j -> code;
25. strcpy(k -> name, j -> name);
26. k -> price = j -> price;
27. **break**;
28. }
29. s -> totalPrice += k -> price;
31. **if**(s -> purchasedDish == NULL)  // 构造s节点purchasedDish链表
32. s -> purchasedDish = k;
33. **else**
34. t -> next = k;
35. t = k;
36. }
37. // 将菜节点置首
38. s -> next = desk;
39. desk = s;
    * 查询模块



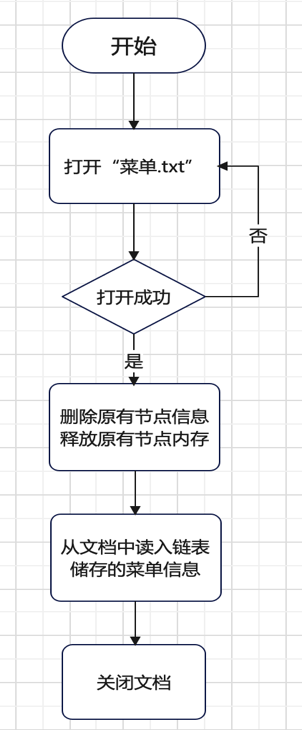
加菜部分：

1. printf("请输入您要添加的菜的编码:\n");
2. scanf("%d", &dishcode);
4. MENU \*k = (MENU\*)malloc(**sizeof**(MENU));

7. **for**(MENU \*j = menu; j; j = j -> next)   // 查找并复制菜节点
8. **if**(j -> code == dishcode)
9. {
10. flag2 = 1;
11. k -> code = j -> code;
12. strcpy(k -> name, j -> name);
13. k -> price = j -> price;
14. **break**;
15. }
17. // 更新desknow节点信息
18. desknow -> totalPrice += k -> price;
19. desknow -> purchasedDishNum ++ ;
20. k -> next = desknow -> purchasedDish;
21. desknow -> purchasedDish = k;
23. SaveDesk();     // 保存

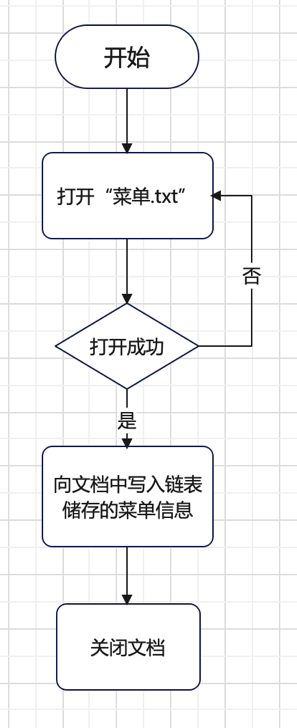
删除菜部分：

1. printf("请输入您要删除的菜的编码:\n");
2. scanf("%d", &dishcode);
4. MENU \*k = (MENU\*)malloc(**sizeof**(MENU));
6. **if**(desknow -> purchasedDish -> code == dishcode)    // 若链表头为要删除节点，直接让链表头等于next
7. {
8. // 更新节点信息
9. desknow -> totalPrice -= desknow -> purchasedDish -> price;
10. desknow -> purchasedDishNum -- ;
12. desknow -> purchasedDish = desknow -> purchasedDish -> next;
14. }
15. **else**    // 否则采用两个节点协助删除
16. **for**(MENU \*i = desknow -> purchasedDish; i; i = i -> next)
17. {
18. MENU \*j = NULL;
19. **if**(i -> next)
20. j = i -> next;
22. **if**(j -> code == dishcode)
23. {
24. // 更新节点信息
25. desknow -> totalPrice -= j -> price;
26. desknow -> purchasedDishNum --；
27. i -> next = j -> next;
28. j -> next = NULL;
29. free(j);    // 释放内存
31. **break**;
32. }
33. }
34. 文件读写板块
    * 菜单读



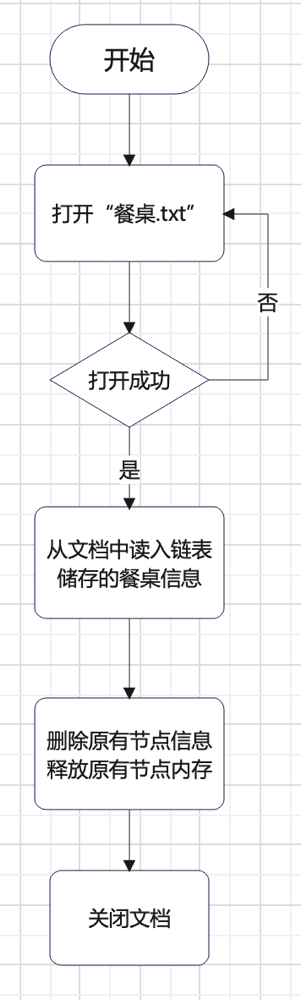
菜单读主要代码：

1. fscanf(fp, "%d", &count);       // 读取总菜品数
2. MENU \*s, \*r;
3. **for**(**int** i = 0; i < count; i ++ )        // 读取菜品编码、菜名和价格，并创建链表
4. {
5. s = (MENU\*)malloc(**sizeof**(MENU));
6. fscanf(fp, "%d%s%d", &s -> code, s -> name, &s -> price);
7. **if**(menu == NULL)
8. menu = s;
9. **else**
10. {
11. r -> next = s;
12. }
13. r = s;
14. }
15. **if**(r != NULL)
16. r -> next = NULL;
    * 菜单写



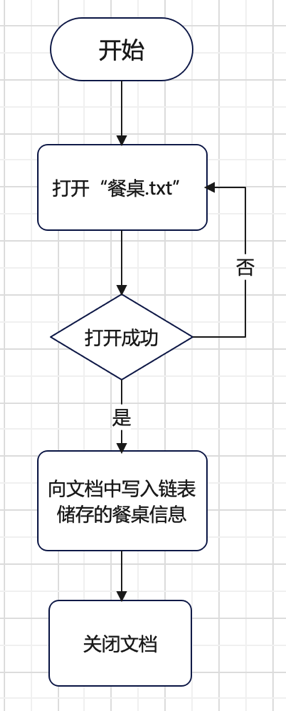
菜单写主要代码：

1. fprintf(fp, "%d\n", count);     // 保存总菜品数
3. **for**(MENU \*i = menu; i; i = i -> next)
4. fprintf(fp, "%d %s %d\n", i -> code, i -> name, i -> price);    // 分别保存编码、菜名和价格
   * 餐桌读



餐桌读主要代码：

1. DESK \*s = NULL, \*r = NULL;
2. **while**(!feof(fp))        // 当文件未结束时，读取文件中的餐桌信息，并构建链表
3. {
4. MENU \*s1 = NULL, \*r1 = NULL;
5. s = (DESK\*)malloc(**sizeof**(DESK));        // 创建desk节点
6. s -> purchasedDish = NULL;
7. fscanf(fp, "%d%d%d%d", &s -> deskCode, &s -> peopleNum, &s -> purchasedDishNum, &s -> totalPrice); // 读入信息
9. **for**(**int** i = 0; i < s -> purchasedDishNum; i ++ )    // 每个desk节点中还要创建menu链表
10. {
11. s1 = (MENU\*)malloc(**sizeof**(MENU));
12. fscanf(fp, "%d %s %d ", &s1 -> code, s1 -> name, &s1 -> price);
14. **if**(s -> purchasedDish == NULL)
15. s -> purchasedDish = s1;
16. **else**
17. r1 -> next = s1;
18. r1 = s1;
19. }
20. **if**(r1 != NULL)
21. r1 -> next = NULL;
23. **if**(desk == NULL)
24. desk = s;
25. **else**
26. {
27. r -> next = s;
28. }
29. r = s;
30. }
31. **if**(r != NULL)
32. r -> next = NULL;
    * 餐桌写



餐桌写主要代码：

1. **for**(DESK \*i = desk; i; i = i -> next)   // 保存餐桌的编号、用餐人数、购买菜品数目、总价格和所点的菜品存入文件
2. {
3. fprintf(fp, "%d %d %d %d ", i -> deskCode, i -> peopleNum, i -> purchasedDishNum, i -> totalPrice);
5. **for**(MENU \*j = i -> purchasedDish; j; j = j -> next)
6. fprintf(fp, "%d %s %d ", j -> code, j -> name, j -> price);
8. fprintf(fp, "\n");
9. }

**五、使用说明**

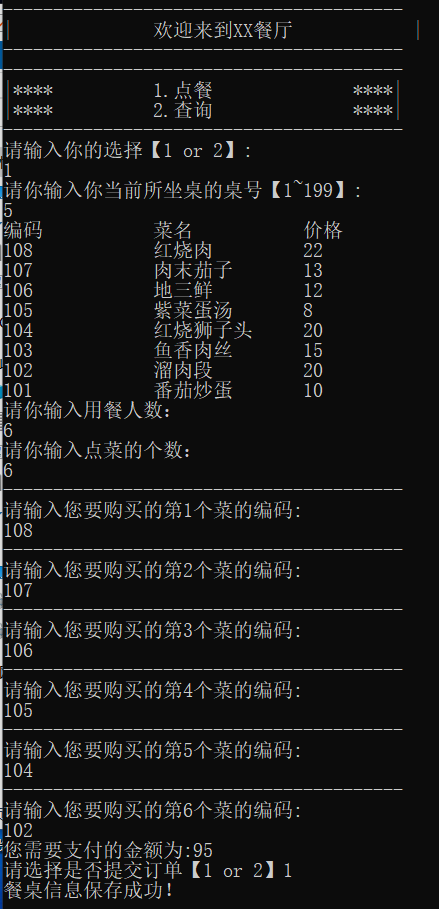
运行环境：各平台均可，本代码在调试阶段在windows 10和macos Ventura环境中均成功运行。

使用方法：在所给附件中启用“餐厅点餐系统.exe”，根据提示操作即可

**六、测试结果**

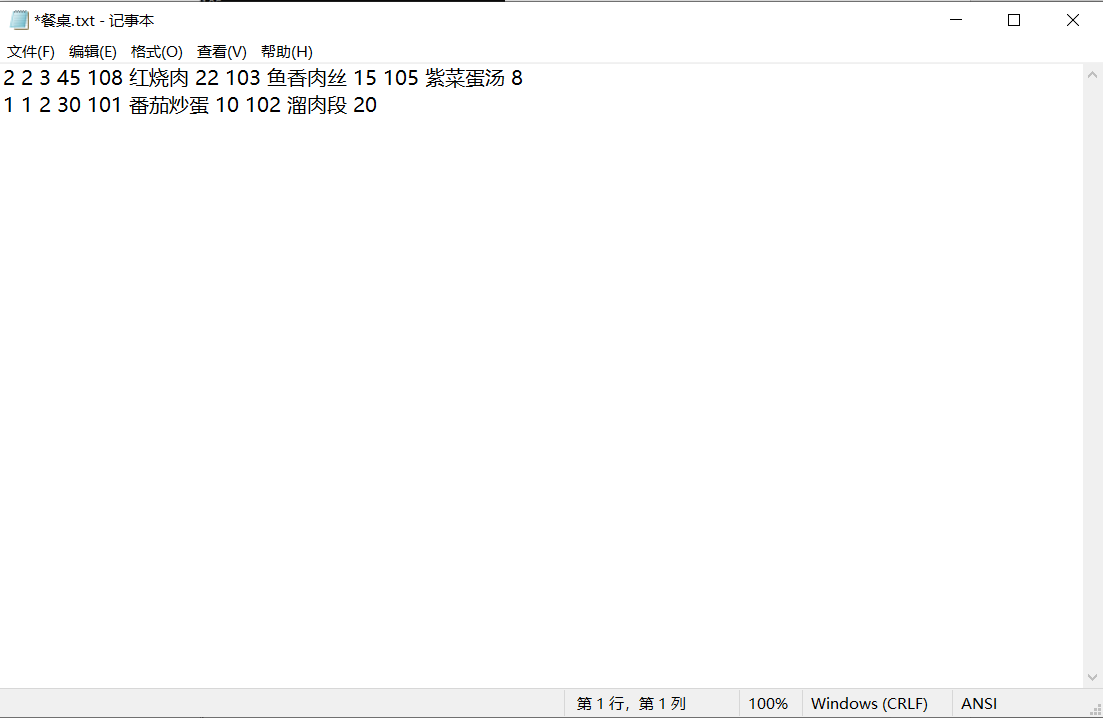
此处将对该系统的**每个功能**进行测试。

* 顾客操作
  + - 1. 点餐

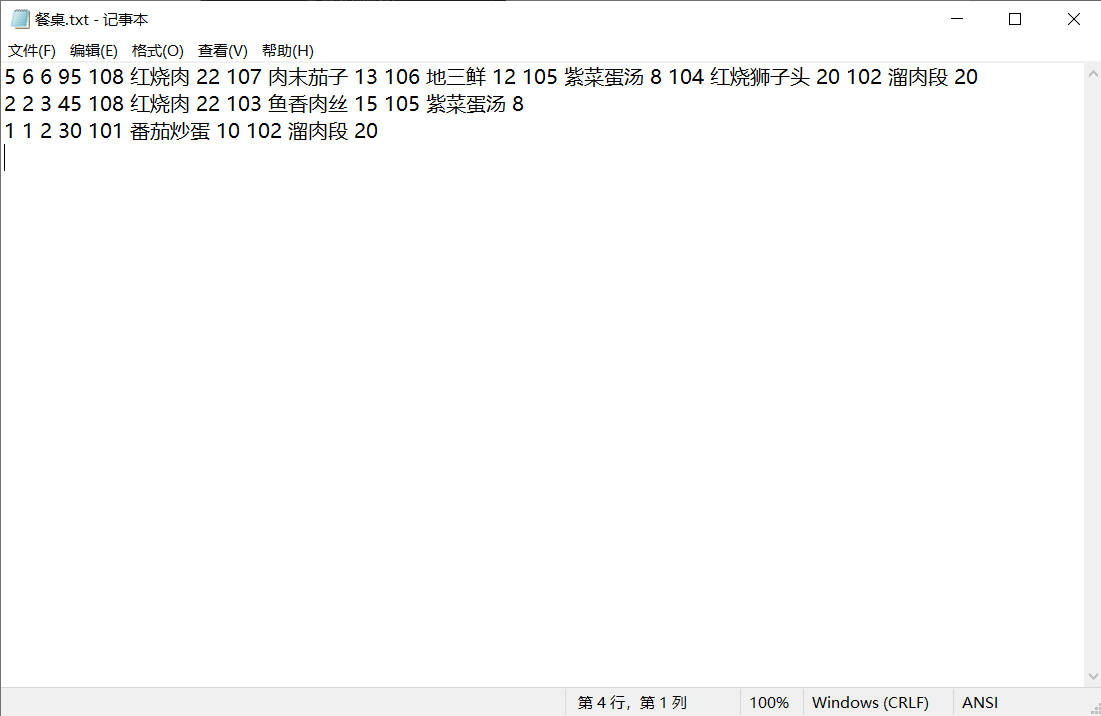


而点餐前后，“餐桌.txt”的变化为：

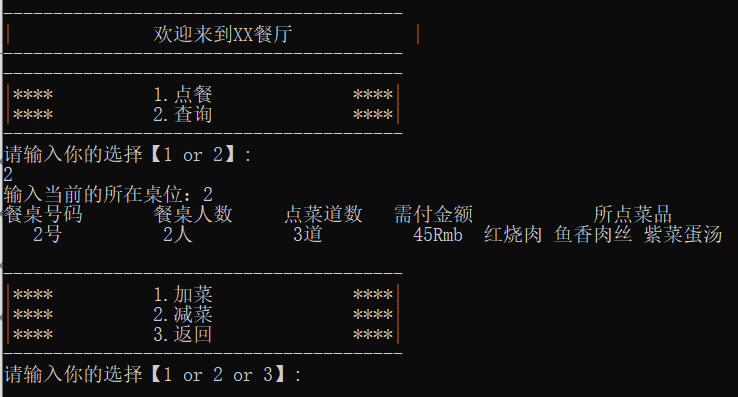
前：



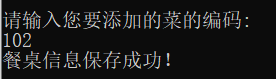
后：



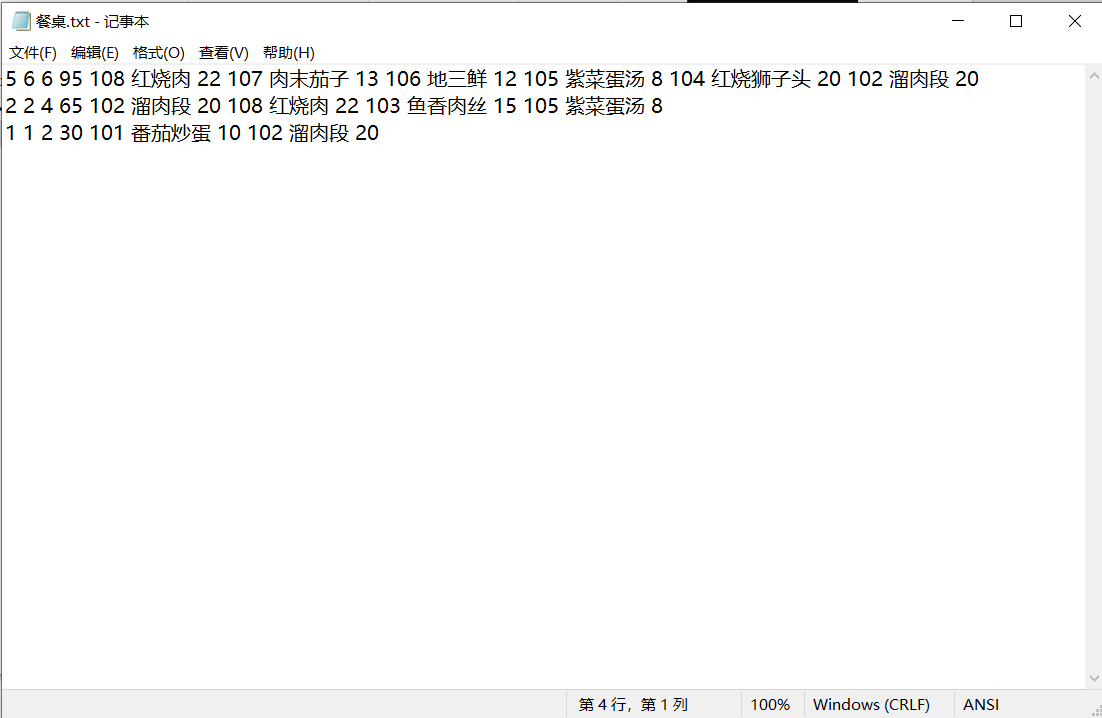
* + - 1. 查询



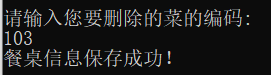
对此桌加“溜肉段”：



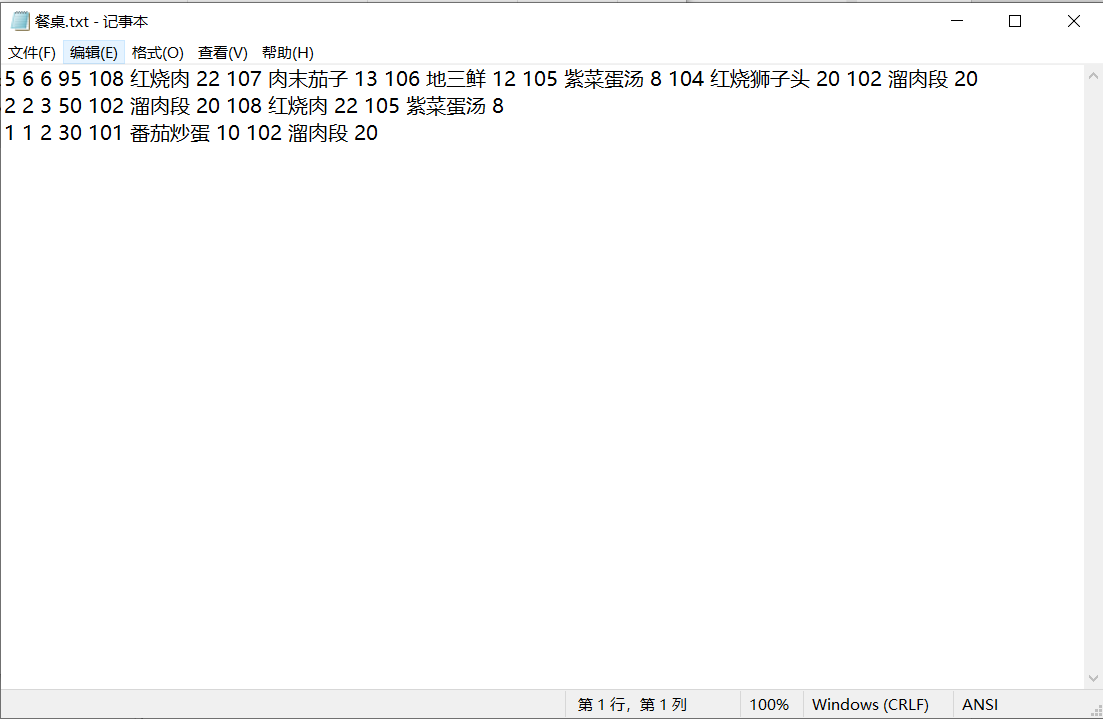
此时文件变为：



对此桌减“鱼香肉丝”：



此时文档变为：

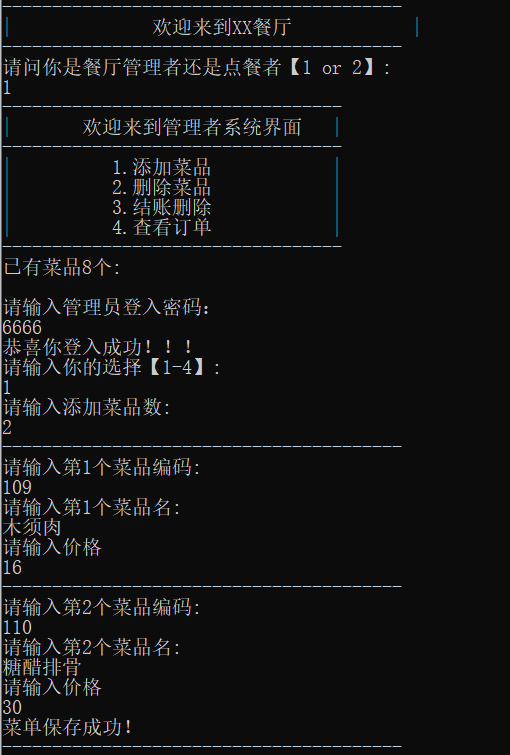


顾客板块调试完毕，代码运行正常、正确。

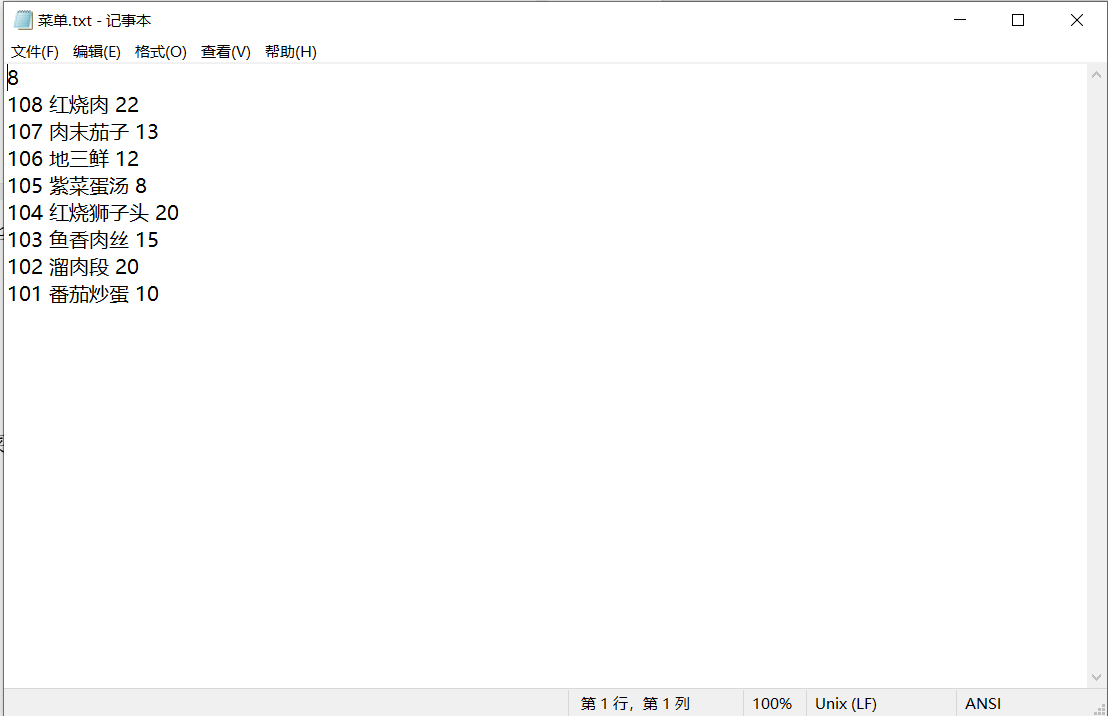
* 管理员操作
  + - 1. 添加菜品

添加两道菜品，分别为“109 木须肉 16”和“110 糖醋排骨 30”

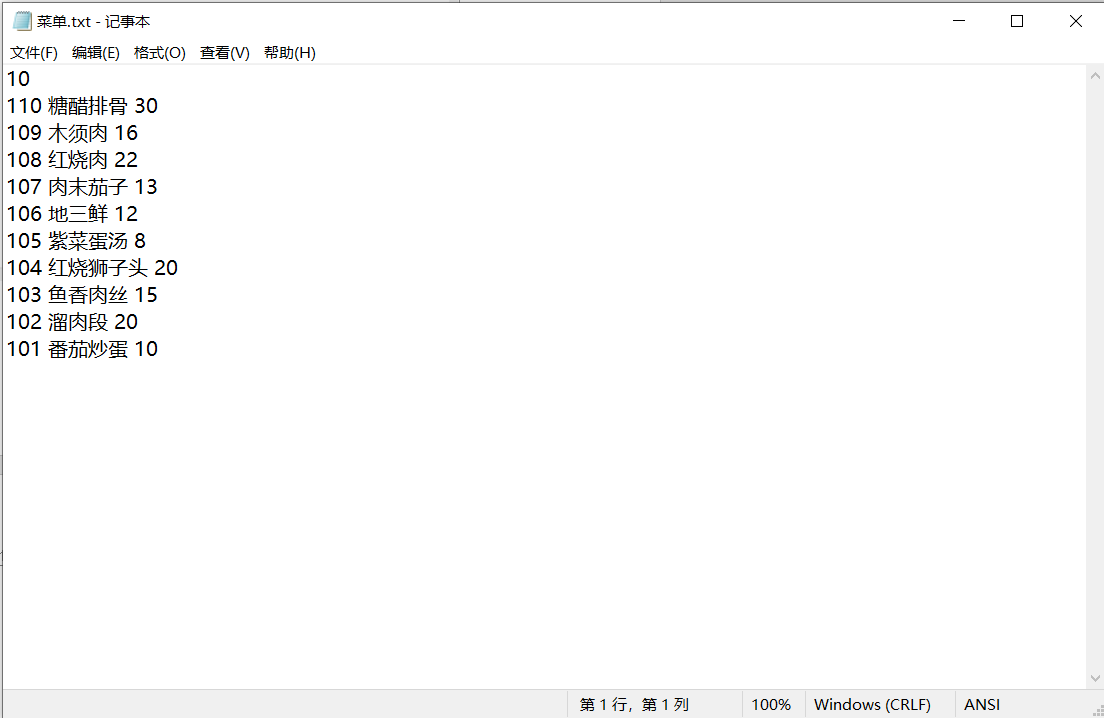
代码运行过程：



添加前：



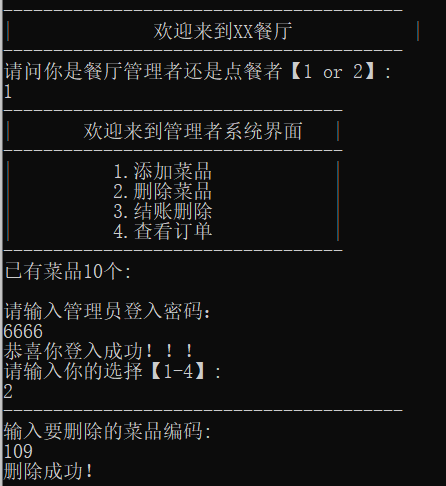
添加后：



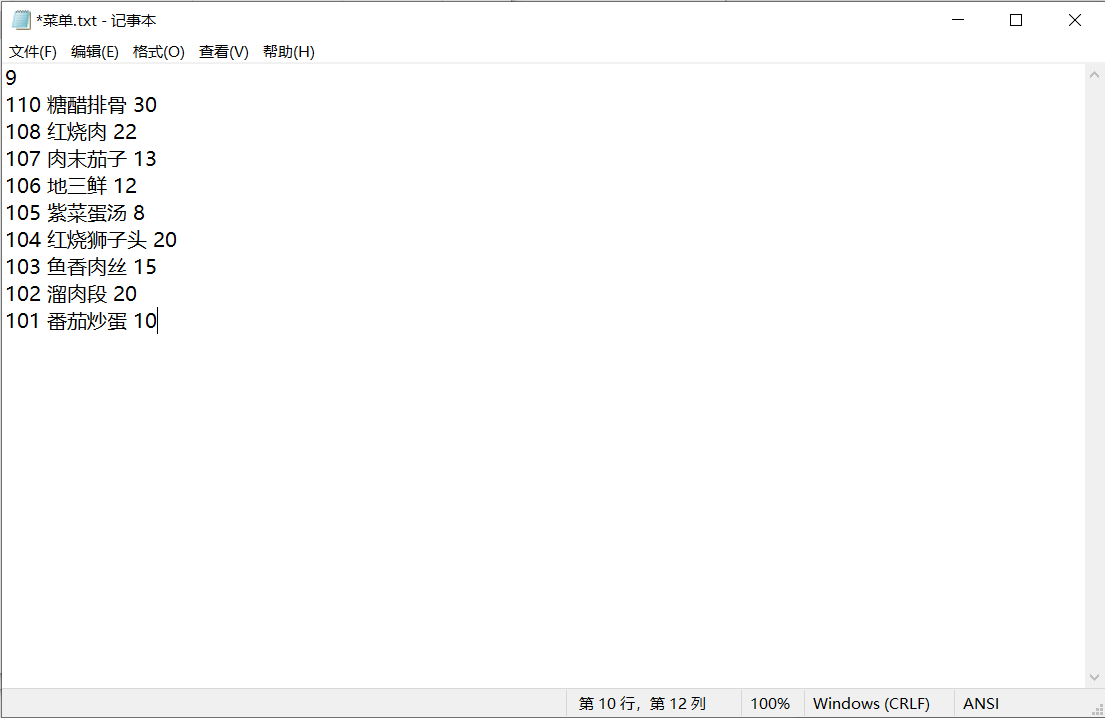
* + - 1. 删除菜品

此处删除“109 木须肉”这道菜

代码运行过程：



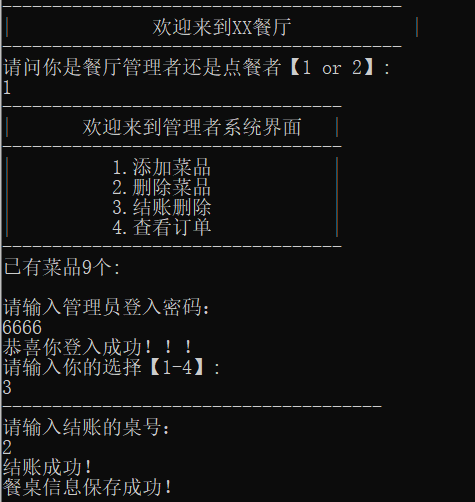
删除前如添加菜品后所示，删除后：



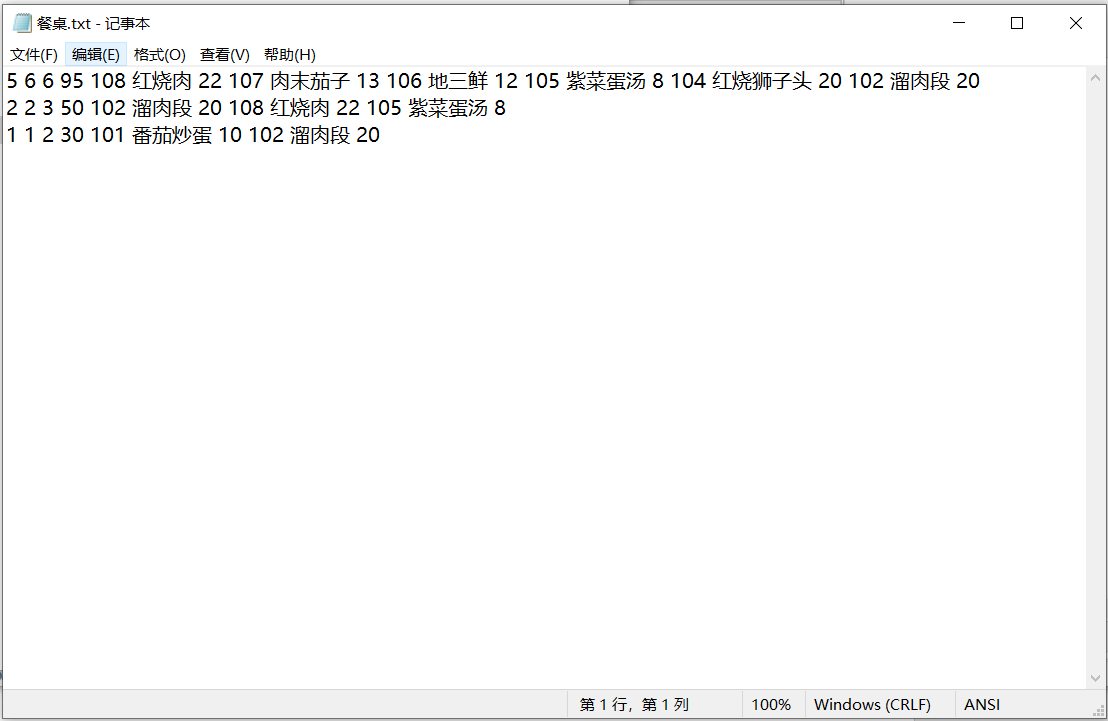
* + - 1. 顾客结账

为桌号为2的顾客结账。

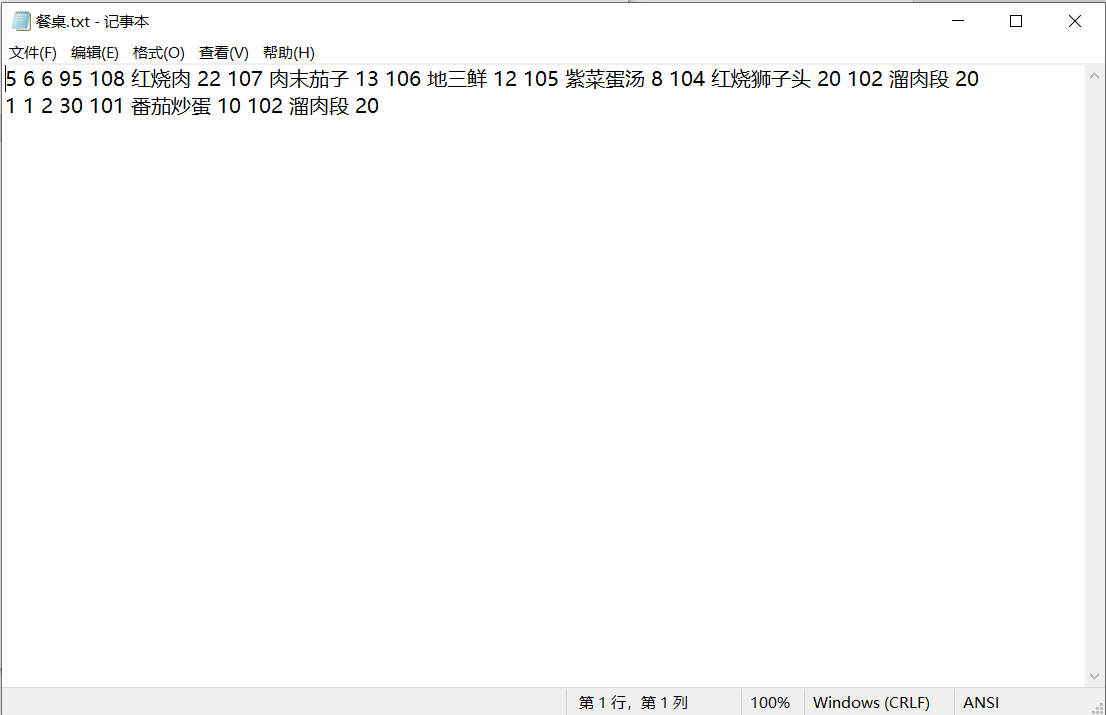
代码运行过程：



结账前：

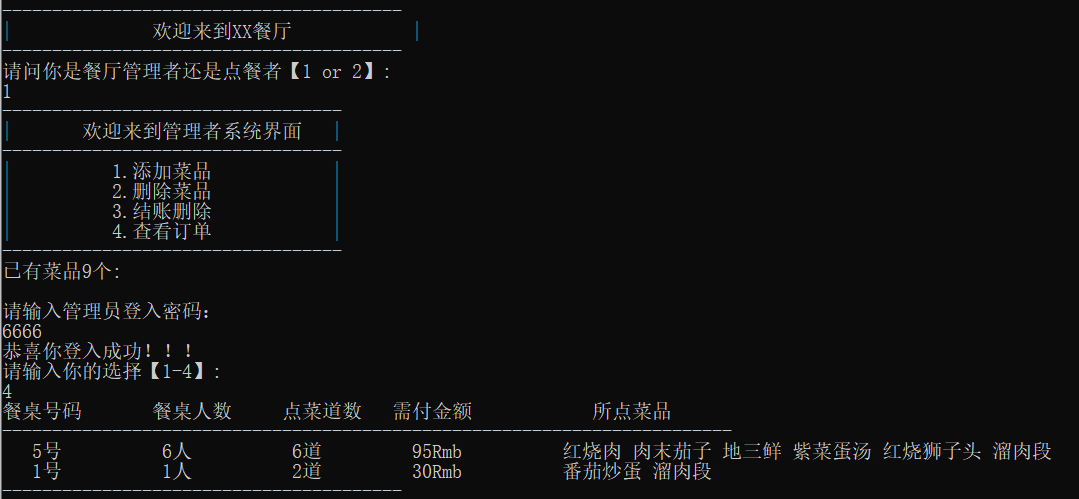


结账后：



* + - 1. 查看订单

直接查看当前的用餐情况。



管理员板块调试结束，代码运行正常、正确。

**七、附录**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct Menu

{

int code; // 菜品编码

char name[100]; // 菜品名称

int price; // 菜品价格

struct Menu \*next;

}MENU;

typedef struct Desk

{

int deskCode; // 桌子的编号

int peopleNum; // 一桌的人数

int purchasedDishNum; // 购买菜品的数量

MENU \*purchasedDish; // 本桌购买菜品

int totalPrice; // 购买菜品总价

struct Desk \*next;

}DESK;

MENU \*menu; // 菜单信息

DESK \*desk; // 菜桌信息

int count; // 总菜品数

//===========================================================================

// 菜单部分

//===========================================================================

void SaveMenu() // 菜单保存

{

FILE \*fp = NULL;

fp = fopen("菜单.txt", "w");

if(fp == NULL)

{

printf("菜单保存失败！\n");

return;

}

fprintf(fp, "%d\n", count); // 保存总菜品数

for(MENU \*i = menu; i; i = i -> next)

fprintf(fp, "%d %s %d\n", i -> code, i -> name, i -> price); // 分别保存编码、菜名和价格

fclose(fp);

printf("菜单保存成功！\n");

}

void ReadMenu()

{

FILE \*fp;

fp = fopen("菜单.txt", "r");

if(fp == NULL)

{

printf("菜单读取失败！\n");

return;

}

fscanf(fp, "%d", &count); // 读取总菜品数

for(MENU \*i = menu; i;) // 删除原有节点，并释放内存

{

MENU \*j = i -> next;

free(i);

if(j)

i = j;

}

MENU \*s, \*r;

for(int i = 0; i < count; i ++ ) // 读取菜品编码、菜名和价格，并创建链表

{

s = (MENU\*)malloc(sizeof(MENU));

fscanf(fp, "%d%s%d", &s -> code, s -> name, &s -> price);

if(menu == NULL)

menu = s;

else

{

r -> next = s;

}

r = s;

}

if(r != NULL)

r -> next = NULL;

fclose(fp);

printf("菜单读取成功！\n");

}

void MenuOutput() // 菜单输出

{

printf("%-15s","编码");

printf("%-15s","菜名");

printf("%-5s","价格");

printf("\n");

for(MENU \*i = menu; i; i = i -> next)

{

printf("%-15d", i -> code);

printf("%-15s", i -> name);

printf("%-5d", i -> price);

printf("\n");

}

}

//===========================================================================

// 餐桌部分

//===========================================================================

void SaveDesk() // 保存餐桌信息

{

FILE \*fp = NULL;

fp = fopen("餐桌.txt", "w");

if(fp == NULL)

{

printf("餐桌信息保存失败！\n");

return;

}

for(DESK \*i = desk; i; i = i -> next) // 保存餐桌的编号、用餐人数、购买菜品数目、总价格和所点的菜品存入文件

{

fprintf(fp, "%d %d %d %d ", i -> deskCode, i -> peopleNum, i -> purchasedDishNum, i -> totalPrice);

for(MENU \*j = i -> purchasedDish; j; j = j -> next)

fprintf(fp, "%d %s %d ", j -> code, j -> name, j -> price);

fprintf(fp, "\n");

}

fclose(fp);

printf("餐桌信息保存成功！\n");

}

void ReadDesk() // 读取餐桌信息

{

FILE \*fp = NULL;

fp = fopen("餐桌.txt", "r");

if(fp == NULL)

{

printf("餐桌信息打开失败！\n");

return;

}

for(DESK \*i = desk; i;) // 删除原有的餐桌信息，并释放内存，注意里面菜品链表内存的释放

{

for(MENU \*k = i -> purchasedDish; k;)

{

MENU \*q = k -> next;

free(k);

if(q)

k = q;

}

DESK \*j = i -> next;

free(i);

if(j)

i = j;

}

DESK \*s = NULL, \*r = NULL;

while(!feof(fp)) // 当文件未结束时，读取文件中的餐桌信息，并构建链表

{

MENU \*s1 = NULL, \*r1 = NULL;

s = (DESK\*)malloc(sizeof(DESK)); // 创建desk节点

s -> purchasedDish = NULL;

fscanf(fp, "%d%d%d%d", &s -> deskCode, &s -> peopleNum, &s -> purchasedDishNum, &s -> totalPrice); // 读入信息

for(int i = 0; i < s -> purchasedDishNum; i ++ ) // 每个desk节点中还要创建menu链表

{

s1 = (MENU\*)malloc(sizeof(MENU));

fscanf(fp, "%d %s %d ", &s1 -> code, s1 -> name, &s1 -> price);

if(s -> purchasedDish == NULL)

s -> purchasedDish = s1;

else

r1 -> next = s1;

r1 = s1;

}

if(r1 != NULL)

r1 -> next = NULL;

if(desk == NULL)

desk = s;

else

{

r -> next = s;

}

r = s;

}

if(r != NULL)

r -> next = NULL;

fclose(fp);

printf("餐桌信息读取成功！\n");

}

//===========================================================================

// 顾客界面

//===========================================================================

void ClientMenu() // 顾客菜单列表

{

int choice, deskcode;

printf("----------------------------------------\n");

printf("| 欢迎来到XX餐厅 |\n");

printf("----------------------------------------\n");

printf("----------------------------------------\n");

printf("|\*\*\*\* 1.点餐 \*\*\*\*|\n");

printf("|\*\*\*\* 2.查询 \*\*\*\*|\n");

printf("----------------------------------------\n");

printf("请输入你的选择【1 or 2】:\n");

scanf("%d", &choice);

while(choice!=1 && choice!=2) // 若输入不为1或2，则重新输入

{

printf("输入错误，请重新再输入：\n");

scanf("%d", &choice);

}

if(choice == 1) // 点餐时

{

int flag = 0; // 判断用户输入是否正确，若不正确则循环输入直到正确为止

while(!flag)

{

printf("请你输入你当前所坐桌的桌号【1~199】:\n");

scanf("%d", &deskcode);

while(deskcode < 1 || deskcode > 199)

{

printf("输入错误,请输入1-199之间的值\n");

scanf("%d", &deskcode);

}

int flag1 = 1;

for(DESK \*i = desk; i; i = i -> next) // 判断用户输入的桌号是否有人，若找到已有桌号和输入相同，则错误

if(i -> deskCode == deskcode)

{

flag1 = 0;

break;

}

if(flag1) // 输入都正确则开始点餐

{

MenuOutput(); // 输出菜单

flag = 1; // 点餐成功，flag变为1，停止循环

DESK \*s; // 为点餐信息创建desk节点

s = (DESK\*)malloc(sizeof(DESK));

// 更新s节点信息

s -> deskCode = deskcode;

s -> purchasedDish = NULL; // 置空，防止报错

printf("请你输入用餐人数：\n");

scanf("%d", &s -> peopleNum);

printf("请你输入点菜的个数：\n");

scanf("%d", &s -> purchasedDishNum);

s -> totalPrice = 0;

for(int i = 0; i < s -> purchasedDishNum; i ++ )

{

int dishcode;

printf("----------------------------------------\n");

printf("请输入您要购买的第%d个菜的编码:\n", i + 1);

scanf("%d", &dishcode);

int flag2 = 0; // 判断输入菜品是否在菜单表中

MENU \*t;

MENU \*k = (MENU\*)malloc(sizeof(MENU)); // 用于复制菜品信息

while(!flag2)

{

for(MENU \*j = menu; j; j = j -> next) // 查找并复制菜节点

if(j -> code == dishcode)

{

flag2 = 1;

k -> code = j -> code;

strcpy(k -> name, j -> name);

k -> price = j -> price;

break;

}

if(!flag2)

{

printf("非常抱歉,您需要的菜品今日不供应,请重新输入:\n");

printf("请输入您要购买的第%d个菜的编码:\n", i + 1);

scanf("%d", &dishcode);

}

}

// 菜品输入正确，则创建新的menu节点

s -> totalPrice += k -> price;

if(s -> purchasedDish == NULL) // 构造s节点purchasedDish链表

s -> purchasedDish = k;

else

t -> next = k;

t = k;

}

// 将菜节点置首

s -> next = desk;

desk = s;

printf("您需要支付的金额为:%d\n", s -> totalPrice);

printf("请选择是否提交订单【1 or 2】");

int issubmit;

scanf("%d", &issubmit);

while(issubmit != 1 && issubmit != 2)

{

printf("输入错误，请重新再输入：\n");

scanf("%d",&issubmit);

}

if(issubmit == 1) // 如果确认点单则保存此数据

{

SaveDesk();

}

else

{

ClientMenu(); // 否则重新运行该函数

}

}

else

printf("该桌已经有顾客,请输入自己当前所在的桌号!\n");

}

}

else if(choice == 2) // 展示该桌所点单

{

int deskcode, c;

DESK \*desknow;

printf("输入当前的所在桌位：");

scanf("%d",&deskcode);

printf("%5s","餐桌号码");

printf("%15s","餐桌人数");

printf("%13s","点菜道数");

printf("%11s","需付金额");

printf("%20s","所点菜品");

printf("\n");

for(DESK \*i = desk; i; i = i -> next)

{

if(i -> deskCode == deskcode)

{

desknow = i;

printf("%4d号", i -> deskCode);

printf("%11d人", i -> peopleNum);

printf("%11d道", i -> purchasedDishNum);

printf("%11dRmb", i -> totalPrice);

printf("\t");

for(MENU \*j = i -> purchasedDish; j; j = j -> next)

printf("%s ", j -> name);

printf("\n");

break;

}

}

printf("\n");

//顾客点单完查看后，根据需要加餐和减餐

printf("----------------------------------------\n");

printf("|\*\*\*\* 1.加菜 \*\*\*\*|\n");

printf("|\*\*\*\* 2.减菜 \*\*\*\*|\n");

printf("|\*\*\*\* 3.返回 \*\*\*\*|\n");

printf("----------------------------------------\n");

printf("请输入你的选择【1 or 2 or 3】:\n");

scanf("%d", &c);

while(c != 1 && c != 2 && c != 3)

{

printf("输入错误，请重新再输入:\n");

scanf("%d", &c);

}

printf("----------------------------------------\n");

switch (c)

{

case 1:

{

int dishcode;

printf("----------------------------------------\n");

printf("请输入您要添加的菜的编码:\n");

scanf("%d", &dishcode);

int flag2 = 0; // 判断输入菜品是否在菜单表中

MENU \*k = (MENU\*)malloc(sizeof(MENU));

while(!flag2) // 当输入编码非菜单菜品时一直循环

{

for(MENU \*j = menu; j; j = j -> next) // 查找并复制菜节点

if(j -> code == dishcode)

{

flag2 = 1;

k -> code = j -> code;

strcpy(k -> name, j -> name);

k -> price = j -> price;

break;

}

if(!flag2)

{

printf("非常抱歉,您需要的菜品今日不供应,请重新输入:\n");

printf("请输入您要添加的菜的编码:\n");

scanf("%d", &dishcode);

}

}

// 更新desknow节点信息

desknow -> totalPrice += k -> price;

desknow -> purchasedDishNum ++ ;

k -> next = desknow -> purchasedDish;

desknow -> purchasedDish = k;

SaveDesk(); // 保存

break;

}

case 2:

{

int dishcode;

printf("----------------------------------------\n");

printf("请输入您要删除的菜的编码:\n");

scanf("%d", &dishcode);

int flag2 = 0; // 判断输入菜品是否在用户点单菜品中

MENU \*k = (MENU\*)malloc(sizeof(MENU));

while(!flag2)

{

if(desknow -> purchasedDish -> code == dishcode) // 若链表头为要删除节点，直接让链表头等于next

{

// 更新节点信息

desknow -> totalPrice -= desknow -> purchasedDish -> price;

desknow -> purchasedDishNum -- ;

desknow -> purchasedDish = desknow -> purchasedDish -> next;

flag2 = 1;

}

else // 否则采用两个节点协助删除

for(MENU \*i = desknow -> purchasedDish; i; i = i -> next)

{

MENU \*j = NULL;

if(i -> next)

j = i -> next;

if(j -> code == dishcode)

{

// 更新节点信息

desknow -> totalPrice -= j -> price;

desknow -> purchasedDishNum -- ;

i -> next = j -> next;

j -> next = NULL;

free(j); // 释放内存

flag2 = 1;

break;

}

}

if(!flag2)

{

printf("非常抱歉,您并未点该菜品,无法删除，请重新输入:\n");

printf("请输入您要删除的菜的编码:\n");

scanf("%d", &dishcode);

}

}

SaveDesk();

break;

}

case 3:

{

ClientMenu();

break;

}

}

}

}

//===========================================================================

// 管理员界面

//===========================================================================

void AdminMenu() // 管理员界面

{

int code, choice, n;

printf("----------------------------------\n");

printf("| 欢迎来到管理者系统界面 |\n");

printf("----------------------------------\n");

printf("| 1.添加菜品 |\n"); //ok

printf("| 2.删除菜品 |\n"); //ok

printf("| 3.结账删除 |\n");

printf("| 4.查看订单 |\n");

printf("----------------------------------\n");

printf("已有菜品%d个:\n\n",count);

printf("请输入管理员登入密码：\n");

scanf("%d",&code);

if(code == 6666) // 密码匹配则进入管理员界面

{

printf("恭喜你登入成功！！！\n");

printf("请输入你的选择【1-4】:\n");

scanf("%d", &choice);

if(choice == 1) // 添加菜品，即增加链表元素，并保存到文档中

{

printf("请输入添加菜品数:\n");

scanf("%d", &n);

MENU \*s;

for(int i = 0; i < n; i ++ ) // 增加链表节点

{

s = (MENU\*)malloc(sizeof(MENU));

printf("----------------------------------------\n");

printf("请输入第%d个菜品编码:\n", i + 1);

scanf("%d", &s -> code);

printf("请输入第%d个菜品名:\n",i+1);

scanf("%s", s -> name);

printf("请输入价格\n");

scanf("%d", &s -> price);

count ++ ; // 总菜品数+1

s -> next = menu; // 将新节点放入链表尾

menu = s;

}

SaveMenu(); // 保存菜单

}

else if(choice == 2) // 删除菜品，即删除链表节点

{

printf("----------------------------------------\n");

printf("输入要删除的菜品编码:\n");

scanf("%d", &code);

int flag = 0;

if(menu -> code == code) // 若为首节点，直接删除首节点

{

menu = menu -> next;

flag = 1;

count -- ;

}

else // 否则使用一个辅助节点，对菜品链表节点进行删除

for(MENU \*i = menu; i; i = i -> next)

{

MENU \*j = NULL;

if(i -> next)

j = i -> next;

if(j -> code == code)

{

count -- ;

i -> next = j -> next;

j -> next = NULL;

free(j); // 释放内存

flag = 1;

break;

}

}

if(flag == 1)

printf("删除成功！\n");

else if(flag == 0)

printf("不存在指定节点，删除失败！\n");

SaveMenu(); // 保存菜单

}

else if(choice == 3) // 结账，删除desk中的节点

{

printf("--------------------------------------\n");

printf("请输入结账的桌号：\n");

scanf("%d", &code);

int flag = 0;

if(desk -> deskCode == code) // 首节点直接将其置后

{

desk = desk -> next;

flag = 1;

}

else // 否则使用一个辅助节点对节点进行删除

for(DESK \*i = desk; i; i = i -> next)

{

DESK \*j = NULL;

if(i -> next)

j = i -> next;

if(j -> deskCode == code)

{

i -> next = j -> next;

j -> next = NULL;

free(j);

flag = 1;

break;

}

}

if(flag == 1)

{

printf("结账成功！\n");

SaveDesk();

}

else if(flag == 0)

printf("结账桌号不存在，请检查后重试！\n");

}

else if(choice == 4) // 查询餐桌状况

{

printf("%5s","餐桌号码");

printf("%15s","餐桌人数");

printf("%13s","点菜道数");

printf("%11s","需付金额");

printf("%20s","所点菜品");

printf("\n");

printf("-------------------------------------------------------------------------\n");

for(DESK \*i = desk; i; i = i -> next)

{

printf("%4d号", i -> deskCode);

printf("%11d人", i -> peopleNum);

printf("%11d道", i -> purchasedDishNum);

printf("%11dRmb", i -> totalPrice);

printf("\t\t");

for(MENU \*j = i -> purchasedDish; j; j = j -> next)

printf("%s ", j -> name);

printf("\n");

}

}

}

else

{

printf("密码错误!请重新输入\n");

}

}

void MainMenu() // 主菜单

{

int choice;

printf("----------------------------------------\n");

printf("| 欢迎来到XX餐厅 |\n");

printf("----------------------------------------\n");

printf("请问你是餐厅管理者还是点餐者【1 or 2】:\n");

scanf("%d", &choice);

while(choice != 1 && choice != 2)

{

printf("输入错误，请重新再输入:\n");

scanf("%d", &choice);

}

if(choice == 1)

AdminMenu();

else if(choice == 2)

ClientMenu();

}

int main()

{

ReadMenu();

ReadDesk();

//MenuOutput();

while(1)

MainMenu();

return 0;

}