**《计算导论与程序设计课程设计》**

**实验报告**

**实验名称： 《麦当劳订单点餐系统程序设计》概要设计<6.0>**

**班 级： 2023211314**

**学 号： 2023211304**

**姓 名： 唐术洋**

**1 用户交互界面设计**

**1.1输入输出交换方式**

键盘输入：

第一行包含一个整数n表示订单个数。

对于接下来的n行，用格式类似于11:11:11的方式，给出第i个订单的时间。然后输入一个字符串，表示套餐或食物的名称。

Example:

2

07:00:00 Croutons

07:01:02 CaesarSaladCombo

文件输入：

第一行输入和，其中表示食物的种类数()，表示套餐的种 类数()。

第二行输入个字符串，第i个字符串表示第i种食物的名称(每个字符串不超过50个字符)。

第三行包含个整数，值均在1到70之间，其中第i个整数表示第i种食物的制作时长。

第四行包含个整数，值均在1与54001之间，其中第i个整数表示第i种食物的最大存储容量。

第五行包含两个整数()。

接下来行，其中的第i行包含多个字符串，第一个字符串表示第i个套餐的名称(不超过50个字符)，后续的第j个字符串表示第i个套餐中包含的第j种食物的名称。一个套餐包含的食物种类不超过5，每种食物只有1个。

Example：

2 1

CaesarSalad Croutons

35 6

1 1

2 2

CaesarSaladCombo CaesarSalad GrilledChickenPieces Croutons

输出：

按照订单顺序输出订单完成时间。对于第i行，如果第i个订单不成功，则输出Fail；否则，输出这个订单完成的时间，时间格式与输入格式(11：11：11)一致。

Example:（与前文无关）

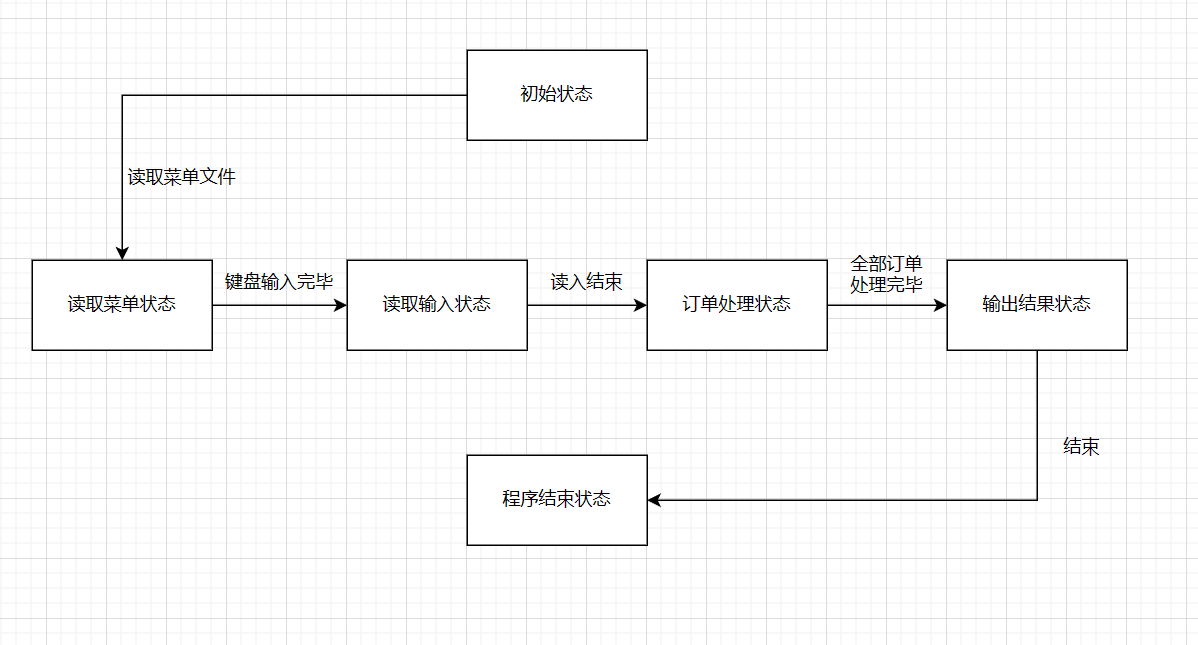
07:00:06

08:02:09

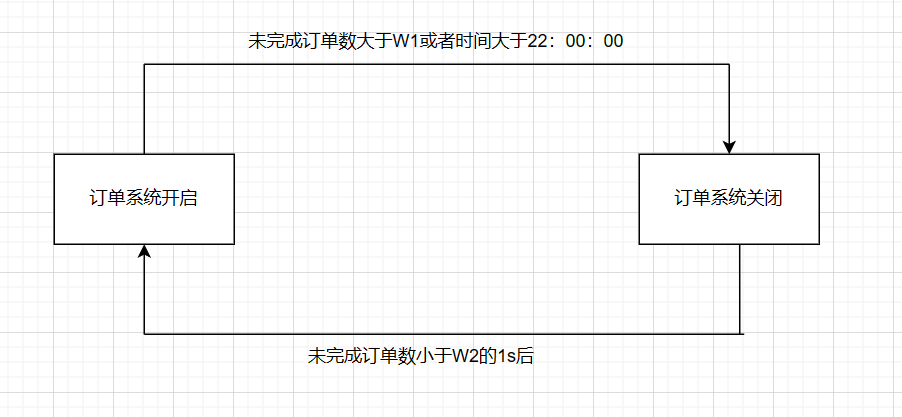
Fail

**2 有限状态自动机状态转换图**

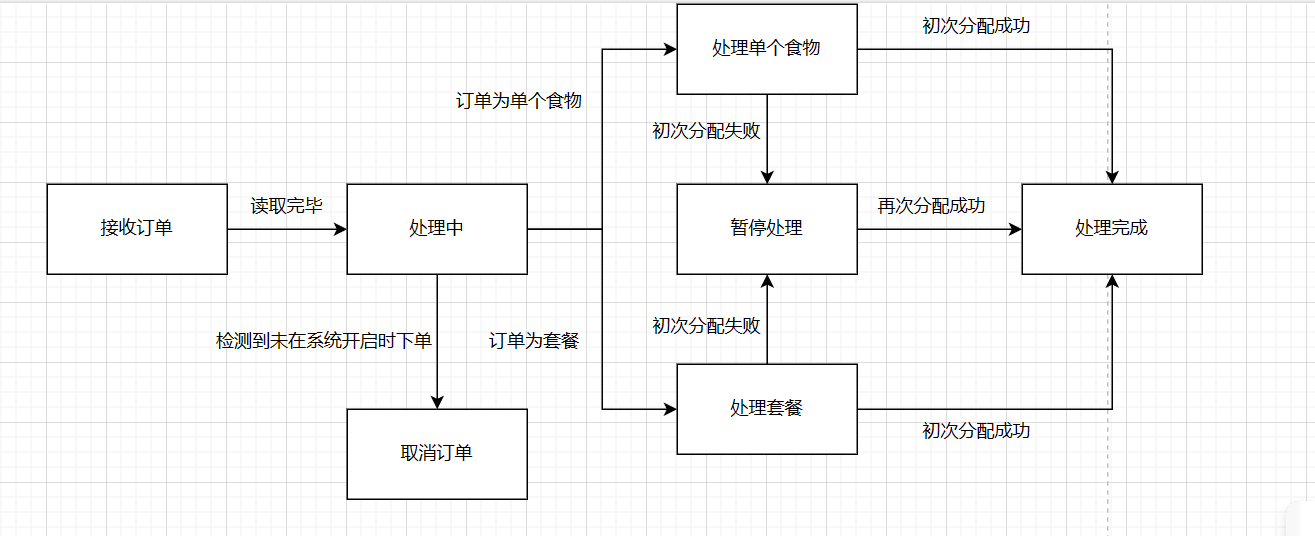
**A.程序整体状态图：**



**B.系统开闭状态图：**



1. **订单处理状态图：**



1. **食物制作状态图**



**2.2状态说明**

1. **程序整体状态图状态说明：**

对于一个点餐模拟系统，存在初始状态、读入菜单与订单、处理订单，并输出结果，最后进入结束状态的过程。如图所示，当程序开始运行，进入初始状态；当接收到读取菜单指令（程序内调用函数），则进入读取菜单状态；当键盘输入结束（最后敲下回车）后，进入读取键盘输入状态；读入结束后进入订单处理状态；当订单全部处理完毕后，进入输出结果状态；输出结果完毕后，程序运行结束。

1. **系统开闭状态图：**

当时间大于7:00:00时或者未完成订单数小于规定值W2的1s后，系统开启。

当未完成订单数大于规定值W1或者时间大于22:00:00时，系统关闭。

1. **订单处理状态图：**

“接收订单”，即初始状态，在读取完毕后进入“处理中”状态，此时对订单状态进行检测，若未在系统开启时下单，则进入“取消订单”状态（结束状态1）。若成功下单且为单个食物，则进入“处理单个食物”状态。若成功下单且为套餐（多个食物），则进入“处理套餐”状态。以上两种情况若初次分配失败，便进入“暂停处理”状态，若首次分配成功或再次分配成功便进入“处理完成”状态（结束状态2）。

1. **食物制作状态图**

每经过食物所需的制作时间t秒，对应食物数量加1，同时再进入食物的制作。

**3高层数据结构设计**

**3.1全局常量定义**

全局常量：

#define MAX\_NAME\_LEN 51

#define MAX\_FOOD\_NUMBER 100

#define MAX\_PACKAGE\_FOOD\_NUMBER 5

#define MAX\_PACKAGES\_NUMBER 100

#define MAX\_ORDER\_NUMBER 54001

**3.2全局变量定义**

int ordersNum; //订单数量

int orderIndex ; //订单序列号

int currentTimeInt; //用整形数据表示的当前时间

int endTimeInt;//用整形数据表示的结束时间

int stopTimeInt;//用整形数据表示的停止时间

int N; //N为食物数量

int M; //套餐数量

int W1; //系统关闭阈值

int W2; //系统重开阈值

int pending; //处理队列中的数量

int sysrun; //系统运行标志

int reopen; //系统重开标志

**3.3数据结构的定义：**

**（1）食物结构体**

typedef struct \_Food

{

char name[MAX\_NAME\_LEN];//食物名称

int store;//食物存在的数量

int capacity;//食物容量（最大数量）

int produce;//食物生产周期

int remain;//食物生产剩余时间

Queue queue;//队列节点

}Food;

**（2）套餐结构体**

typedef struct \_Package

{

char name[MAX\_NAME\_LEN];//套餐名字

Food\* food[10];//套餐中的食物

int num;//套餐中食物数量

}Package;

**（3）订单结构体**

typedef struct \_Order

{

char name[MAX\_NAME\_LEN];

int time;//订单接受时间

int endt;//订单结束时间

OrderStatus status;//订单状态

Food\* food[MAX\_PACKAGE\_FOOD\_NUMBER];//订单中包含的食物

int foodsNum;//订单中食物数量

}Order;

**（4）订单队列结点**

typedef struct \_Element

{

Order\* order;//队列节点中的订单数据

struct \_Element\* next;//处理队列的指针域

}Element;

**（5）订单处理队列节点定义**

typedef struct \_Queue

{

Element\* front;

Element\* rear;

int size;

}Queue;

**（6）定义订单状态的枚举类型**

typedef enum

{

unarrive,//订单未到达

arrive,//订单到达

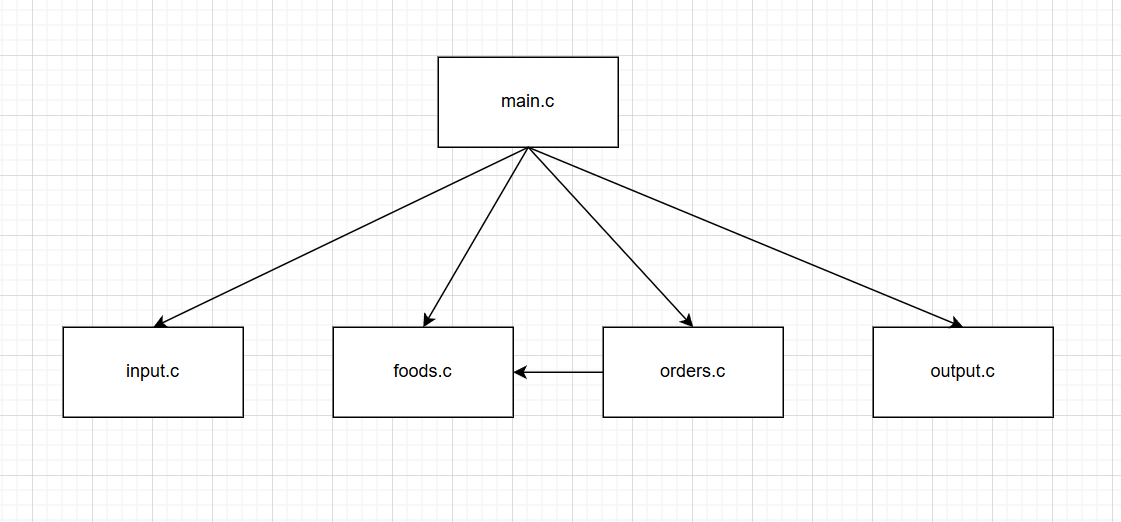
complete,//订单完成

Fail//订单下单失败

}OrderStatus;

**4 系统模块划分**

**4.1系统模块划分**



**1.模块名称main.c**

模块功能为模拟系统时钟控制各个模块。

**2.模块名称input.c**

模块功能为读入文件中的数据以及读入键盘输入的数据或鼠标指令。

**3.模块名称foods.c**

模块功能为模拟食物的制作并更新数据。

**4.模块名称order.c**

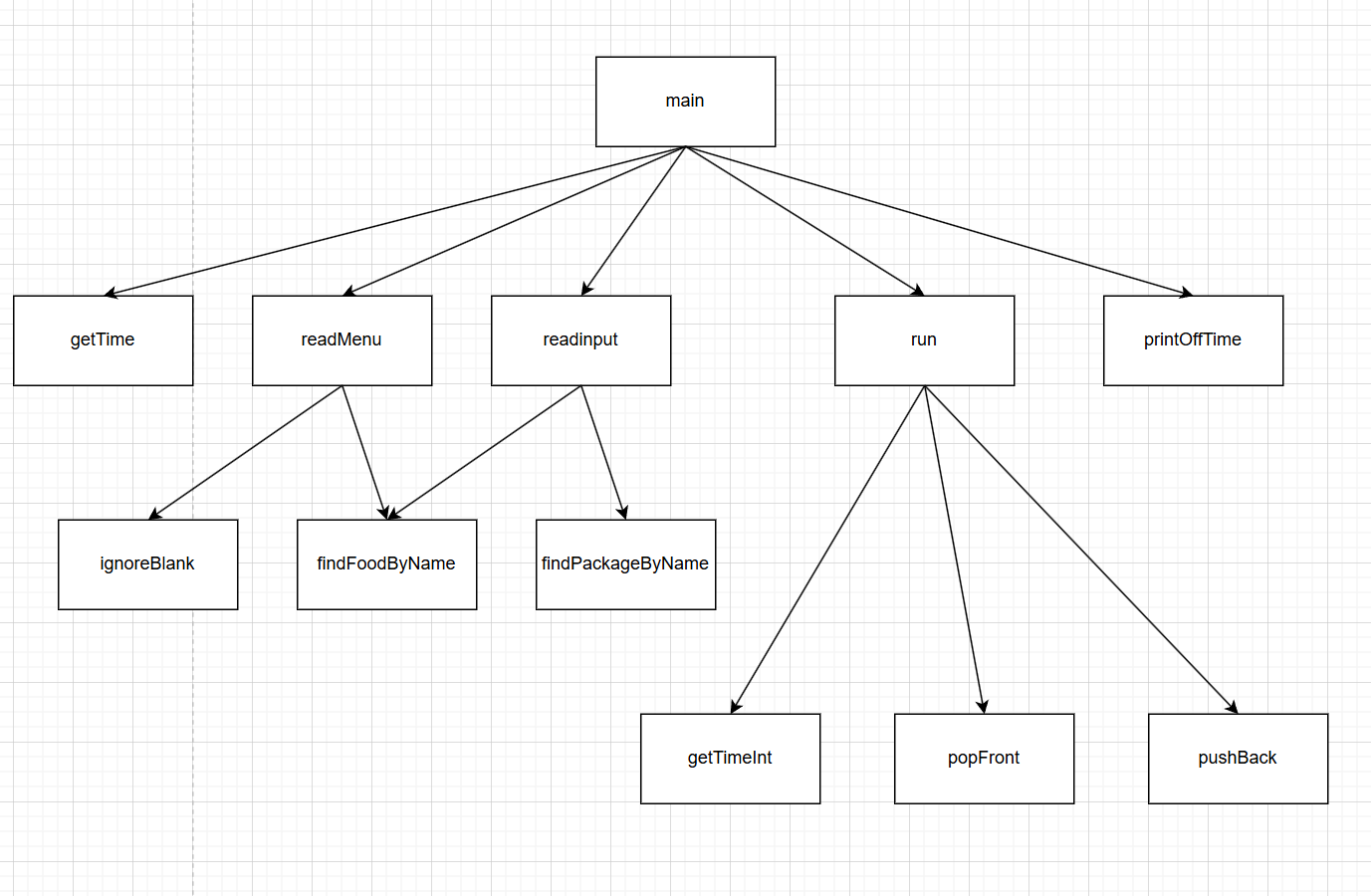
模块功能为处理订单序列

**5.模块名称output.c**

模块功能为输出已处理的订单结果

**4.2各模块函数说明**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **函数原型** | **功能** | **参数** | **返回值** |
| **1** | char\* ignoreBlank(char\* p) | 忽略字符串开头的空格、制表符和换行符，返回第一个非空字符的位置。在读入菜单文件时被input模块调用。 | char\* p - 字符串指针。 | char\* |
| **2** | int getTimeInt(int h, int m, int s) | 将小时、分钟和秒转换为总秒数。在订单处理时被order模块调用。 | int h - 小时，int m - 分钟，int s - 秒。 | int |
| **3** | void getTime(int t, int\* h, int\* m, int\* s) | 将总秒数转换为小时、分钟和秒。程序开始时被main模块调用。 | int t - 总秒数，int\* h - 指向存储小时的指针，int\* m - 指向存储分钟的指针，int\* s - 指向存储秒的指针。 | void |
| **4** | int findPackageByName(const char\* name) | 根据名称查找包装食品的索引号。在读取键盘输入时被input模块调用。 | const char\* name - 套餐的名称。 | int |
| **5** | void popFront(Queue\* q, Order\*\* order) | 从队列的头部移除订单，并将其返回。在订单处理时被order模块调用。 | Queue\* q - 指向队列结构的指针，Order\*\* order - 指向订单指针的指针。 | void |
| **6** | Food\* findFoodByName(const char\* name) | 根据名称查找食物的指针。在读取菜单与键盘输入时被input模块调用。 | const char\* name - 食物的名称。 | Food\* |
| **7** | void readMenu() | 从文件中读取菜单信息。  在菜单读取时被input模块调用。 | 无 | void |
| **8** | void readinput() | 从标准输入读取订单信息，并初始化订单结构。在键盘读取时被input模块调用。 | 无 | void |
| **9** | void run() | 核心运行函数，处理食物生产和订单的满足情况 | 无 | void |
| **10** | void pushBack(Queue\* q, Order\* order) | 将订单添加到队列的尾部。  在订单处理时被order模块调用。 | Queue\* q - 指向队列结构的指针，Order\* order - 指向订单结构的指针。 | void |
| **11** | void printOffTime() | 打印每个订单的完成时间或失败状态.在订单状态输出时被output模块调用。 | 无 | void |

**4.3函数调用图示及说明**

**main函数：**

通过调用getTimeInt函数，初始化当前时间 (currentTimeInt)、结束时间 (endTimeInt)、停止时间 (stopTimeInt)。

通过调用 readMenu 函数读取菜单信息。

通过调用 readinput 函数读取订单信息。

通过调用 run 函数处理订单。

通过调用 printOffTime 函数输出订单完成时间或失败信息。

**readMenu函数：**

通过对ignoreBlank函数的调用，过滤掉文件中的杂乱字符。

通过对findFoodByName函数的调用，完成读入数据与食物对象的匹配。

**readinput函数：**

通过对findFoodByName，完成读入数据与食物对象的匹配。

通过对findPackageByName，完成读入数据与套餐对象的匹配。

**run函数：**

通过调用getTimeInt函数，换算当前的时间为整形数据。

通过调用popFront函数，将订单从处理队列头部取出。

通过调用pushBack函数，将订单添加到处理队列末尾。

**5 高层算法设计**

**核心算法设计：**

程序的核心算法主要分为两个部分：食物库存管理和订单处理。

**食物库存管理**

在食物库存管理部分，程序首先检查每种食物的库存量是否低于其容量。如果库存不足，程序会根据食物的生产时间是否达到所需时间判断是否增加库存。如果食物的生产时间等于所需时间，则该食物对应的库存量加1，否则不变。

**订单处理**

订单处理部分首先检查每种食物是否有库存。如果有库存，程序会尝试满足处理队列中的订单需求。程序从队列中取出订单，减少食物库存，并减少订单所需的食物数量。如果订单所需的食物数量减少到0，表示订单已完成，程序将订单状态设置为complete，并记录完成时间。

**控制策略算法**

**系统运行状态控制**

程序通过监控订单的待处理数量来控制系统的运行状态。如果待处理订单数量超过阈值W1，系统将关闭，不再接受新订单。如果当前时间未超过晚上10点，且待处理订单数量低于阈值W2，系统将重新开放。

**输入输出的时间控制逻辑**

**输入**

程序的输入包括订单信息、食物信息、当前时间、系统运行状态等。订单信息包括订单到达时间、所需食物种类和数量。食物信息包括食物的库存量、容量和生产时间。

**输出**

程序的输出主要是订单的完成时间。对于已完成的订单，程序会输出其完成的详细时间（小时、分钟、秒）。

**时间控制**

程序通过一个循环来模拟时间的流逝。每次循环，当前时间currentTimeInt增加1，并调用run()函数来处理订单和更新库存。如果当前时间超过晚上10点，系统将关闭。

**教师评语：**