**概要设计报告**

**一、全局变量定义**

int w1 ;// 未完成订单数最大值

int w2;// 未完成订单数最小值

int time = 0;//初始化时间为0

int ordernum = 0;//初始化当前未完成订单数为0

int ordercon = 2;//初始化总订单状态（ordernum>w1为 1， ordernum<=w1为2）

int porder = -1;//初始化未完成的最早的订单的编号，初值为-1

food \*foods = NULL;//创建一个结构体数组，用来储存食物的信息

int foodsize = 0;//将数组的大小初始化为0

set \*sets = NULL;//创建一个结构体数组，用来储存套餐的信息

int setsize = 0;//将数组的大小初始化为0

order orders = [100];//创建一个结构体数组，用来储存订单的信息

int ordersize = 0;//将数组的大小初始化为 0

int TIME = 0;//初始化图形化界面显示的时间为0

int check;//初始化当前时刻是否点单的标记

int ac = 0;//初始化时间加速倍数

int shine;//初始化订单显示时间

char timept[10];//将时间表示成时分秒的形式

ExMessage a;//初始化鼠标结构体信息

IMAGE img;//初始化图片结构体信息

**二、数据结构的定义**

（1）食物的结构：

typedef struct Food

{

char foodname [51]; 食物的名称

int foodtime; 食物制作所需时间

int foodmax; 食物的最大储存容量

int remain; 食物在当前秒的存量

int ordernum; 食物被订单分配的个数

int orderfinnum; 食物被分配已经完成的个数

int timer; 距离上一个食物制作周期的时间

} food;

（2）套餐的结构：

typedef struct Set

{

char setname [51]; 套餐的名称

int foodid [5]; 套餐中所含的食物的代码

int size; 套餐所含食物的种类

} set;

（3）订单的结构：

typedef struct Order

{

int ordertime; 订单下单的时间

int foodid [5]; 订单所含的食物的代码

int foodcon [5]; 订单所含食物的状态（0表示未完成，1表示已完成）

int size;订单所含食物的种类

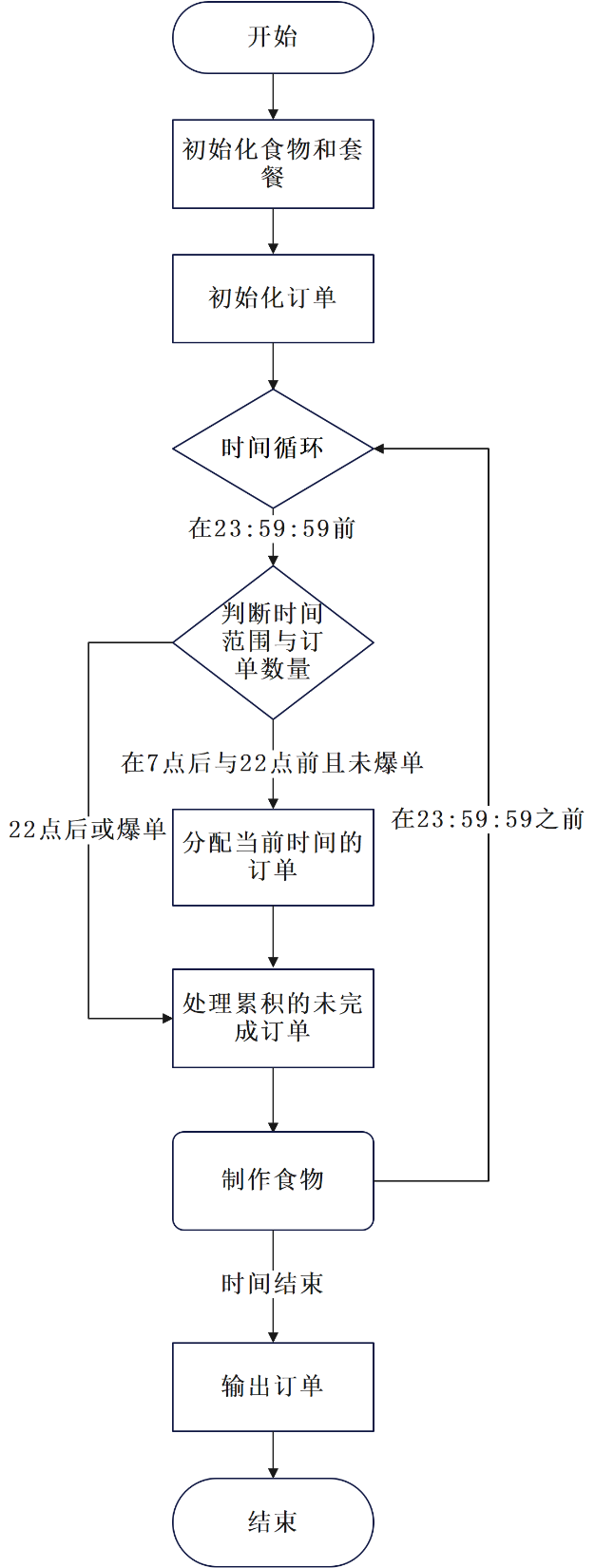
int finishtime; 订单完成的时间

int con; 订单的状态（con）(0表示未分配，1表示已分配未完成，2表示已完成，3表示失败)

} order;

**三、函数声明**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序 号 | 函数原型 | 功能 | 参数 | 返回值 |
| 1 | void initializefoodset () | 初始化食物和套餐的结构体数组，并设定w1,w2 | void | void |
| 2 | void initializeorder () | 初始化订单的结构体数组 | void | void |
| 3 | void makefood () | 进行每秒食品的制作 | void | void |
| 4 | void allocateorder (int a) | 将一个订单分配给食物 | a是一个订单的id | void |
| 5 | int checkorder () | 检查目前未完成订单的总数量并判断订单状态 | void | 若可以继续接单则返回1，若不能继续接单则返回0 |
| 6 | void processorder () | 处理当前未完成订单 | void | void |
| 7 | void printorder () | 将订单按照题目要求进行打印 | void | void |
| 8 | int findfood (char a []) | 根据食物名称查找食物的id | a是食物名称的字符串 | 查找食物的id，并返回食物的id |
| 9 | int findset (char a []) | 根据套餐名称查找套餐的id | a是套餐名称的字符串 | 查找套餐的id，并返回套餐的id |
| 10 | int timechange1 (int a, int b, int c) | 将时：分：秒的时间格式转换为秒数 | a是时的数值，b是分的数值，c是秒的数值 | 返回转换之后的秒 |
| 11 | int findorder () | 查找指定时间中的订单 | void | 返回找到的订单的id，没有相应的订单则返回-1 |
| 12 | int judgeorder (int a) | 判断给出的订单是否完成 | a是一个订单的id | 若给出的订单达成完成的条件返回2，若未达成则返回1 |

**四、流程图示**

**五、第一版源代码**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct Food

{

char foodname[51];

int foodtime;

int foodmax;

int remain;

int ordernum;

int orderfinnum;

int timer;

} food;

typedef struct Set

{

char setname[51];

int foodid[5];

int size;

} set;

typedef struct Order

{

int ordertime;

int foodid[5];

int foodcon[5];

int size;

int finishtime;

int con;

} order;

int w1;

int w2;

int time = 0;

int ordernum = 0;

int ordercon = 2;

food\* foods = NULL;

int foodsize = 0;

set\* sets = NULL;

int setsize = 0;

order\* orders = NULL;

int ordersize = 0;

int timechange1(int a, int b, int c)

{

return (a \* 3600 + b \* 60 + c);

}

int findfood(char a[])

{

int b;

for (b = 0; b < foodsize; b++)

if (strcmp(a, foods[b].foodname) == 0)

return b;

return -1;

}

int findset(char a[])

{

int b;

for (b = 0; b < setsize; b++)

if (strcmp(a, sets[b].setname) == 0)

return b;

return -1;

}

int judgeorder(int a)

{

int b;

for (b = 0; b < orders[a].size; b++)

if (orders[a].foodcon[b] == 0)

return 1;

return 2;

}

int checkorder()

{

if (ordernum > w1 && ordercon == 2)

{

ordercon = 1;

return 0;

}

else if (ordernum < w2 && ordercon == 1)

{

ordercon = 2;

return 1;

}

else if (ordernum <= w1&& ordercon==2)

{

ordercon = 2;

return 1;

}

else

return 0;

}

int findorder()

{

int a;

for (a=0; a<ordersize; a++)

if (orders[a].ordertime == time)

return a;

return -1;

}

void initializefoodset()

{

int a,c,d;

char b[51];

FILE\* fptr;

fptr = fopen("1.txt", "r");

fscanf(fptr, "%d %d", &foodsize,&setsize);

foods = (food\*)malloc(foodsize \* sizeof(food));

for (a = 0; a < foodsize; a++)

{

fscanf(fptr, "%s", foods[a].foodname);

}

for (a = 0; a < foodsize; a++)

{

fscanf(fptr, "%d", &foods[a].foodtime);

}

for (a = 0; a < foodsize; a++)

{

fscanf(fptr, "%d", &foods[a].foodmax);

foods[a].orderfinnum = foods[a].ordernum = foods[a].remain = foods[a].timer = 0;

}

fscanf(fptr, "%d %d", &w1, &w2);

sets = (set\*)malloc(setsize \* sizeof(set));

fscanf(fptr, "%s", sets[0].setname);

a = c=0;

while(!feof(fptr))

{

fscanf(fptr, "%s", b);

d = findfood(b);

if ( d== -1&&a<setsize)

{

sets[a].size = c ;

a++;

strcpy(sets[a].setname, b);

c = 0;

}

else if(sets[a].foodid[c-1] != d)

{

sets[a].foodid[c] = d;

c++;

}

}

sets[a].size = c;

fclose(fptr);

}

void initialzeorder()

{

int x;

int a, b, c;

char d[51];

scanf("%d", &ordersize);

orders = (order\*)malloc(ordersize \* sizeof(order));

for (x = 0; x < ordersize; x++)

{

scanf("%d:%d:%d", &a, &b, &c);

orders[x].ordertime = timechange1(a, b, c);

scanf("%s", d);

a = findfood(d);

if (a != -1)

{

orders[x].foodid[0] = a;

orders[x].size = 1;

}

else

{

a = findset(d);

for (b = 0; b < sets[a].size; b++)

orders[x].foodid[b] = sets[a].foodid[b];

orders[x].size = sets[a].size;

}

}

for (x = 0; x < ordersize; x++)

orders[x].con = 0;

}

void makefood()

{

int a;

for (a = 0; a < foodsize; a++)

{

if(foods[a].remain<foods[a].foodmax)

foods[a].timer++;

if (foods[a].timer == foods[a].foodtime)

{

if (foods[a].ordernum == 0)

foods[a].remain++;

else

{

foods[a].ordernum--;

foods[a].orderfinnum++;

}

foods[a].timer = 0;

}

}

}

void allocateorder(int a)

{

int x;

for (x = 0; x < orders[a].size; x++)

{

if(foods[orders[a].foodid[x]].remain==0)

foods[orders[a].foodid[x]].ordernum++;

orders[a].foodcon[x] = 0;

}

orders[a].con = 1;

ordernum++;

}

void processorder(int a)

{

int b;

for (b = 0; b < orders[a].size; b++)

{

if (foods[orders[a].foodid[b]].remain > 0 && orders[a].foodcon[b] == 0)

{

foods[orders[a].foodid[b]].remain--;

orders[a].foodcon[b] = 1;

}

else if (foods[orders[a].foodid[b]].orderfinnum > 0 && orders[a].foodcon[b] == 0)

{

foods[orders[a].foodid[b]].orderfinnum--;

orders[a].foodcon[b] = 1;

}

}

if (judgeorder(a) == 2)

{

orders[a].finishtime = time;

orders[a].con = 2;

ordernum--;

}

}

void printorder()

{

int a;

int hour, minute, second;

for (a = 0; a < ordersize; a++)

{

if (orders[a].con == 2)

{

hour = orders[a].finishtime / 3600;

minute = (orders[a].finishtime % 3600) / 60;

second = orders[a].finishtime % 60;

printf("%02d:%02d:%02d\n", hour, minute, second);

}

else

printf("Fail\n");

}

}

int main()

{

int a;

initializefoodset();

initialzeorder();

for (time = 25200; time <= 86400; time++)

{

if (time <= 79200)

{

a = findorder();

if (a != -1&&checkorder()==1)

{

allocateorder(a);

if (ordernum > w1)

ordercon = 1;

}

}

if (ordernum!=0)

{

for ( a=0; a<ordersize; a++)

if (orders[a].con == 1)

processorder(a);

}

makefood();

}

printorder();

return 0;

}