<外卖订单派送系统>概要设计

版本号：-Design-05

编制时间：2019年4月28日星期日

编制人员：李一博 张千芊 胡晓雯

1. **用户界面设计**

1.1文件方式

1. [文件名称]sales
2. [文件格式].txt

[文件使用/产生说明]

文件输入方式：系统启动后检查当前目录下是否有标准文件名“sales.txt”,若有则自动进入订单强制文件输入方式。

输入格式：订单编号<空格>下单时间<空格>餐馆横坐标<空格>餐馆纵坐标<空格>食客横坐标<空格>食客纵坐标

1.2动画方式

[界面操作说明]

1. **高层数据结构设计**

2.1常量定义

2.2全局变量定义

extern struct order \*firstorder//用链表存储时第一个订单的指针

extern int money,time,accept,over,finish,ridernumber，temp;//在源文件中定义 ,钱数，时间，接单量，超时量，完成量 ,当前骑手数 ，程序结束辅助判断变量

extern struct rider allriders[50]//所有骑手的结构体数组

2.3结构体定义

struct order{

int num;//编号

int time;//订单发起的时间

int cx;

int cy;//餐馆横纵坐标

int sx;

int sy;//食客横纵坐标

int current=0;//该单状态 未派单0/待取餐1/待送餐2/已完成3

struct order \*orderptr;//用链表存储时下一个订单的位置

};

struct area{

int leftx;

int lefty;//子区域左上角坐标

int rightx;

int righty;//子区域右下角坐标

};

// divide.c中 用于存储划分好的模块的信息的结构体

struct rider{

int state,num;//骑手的状态，空闲0/取餐1/送餐2,骑手编号

int x;

int y;//骑手的坐标

int accept,over,finish;//该骑手的接单数，超时数，完成数；

int x1,y1,x2,y2,same;//该骑手派送区域 ,是否和其他骑手有相同区域即是否是跨区域骑手，不是0/是1

struct order task[200];//骑手的订单序列

};

1. **系统模块划分**

3.1系统模块划分

1. input.c

功能:读入输入文件，并将订单信息存储在结构体数组中

1. output.c

功能：将信息（当前账户金额数、每位骑手的位置、接单数、完成数、超时数）写在输出文件中

1. distribute.c

功能：将订单分配给现有骑手

1. path.c

功能：根据骑手所接订单，找到较优路径

1. add.c

功能：统计当前账户金额数、每位骑手的位置、接单数、完成数、超时数

1. divide.c

功能：根据骑手数量进行模块划分

main.c

output.c

input.c

distribute.c

divide.c

add.c

path.c

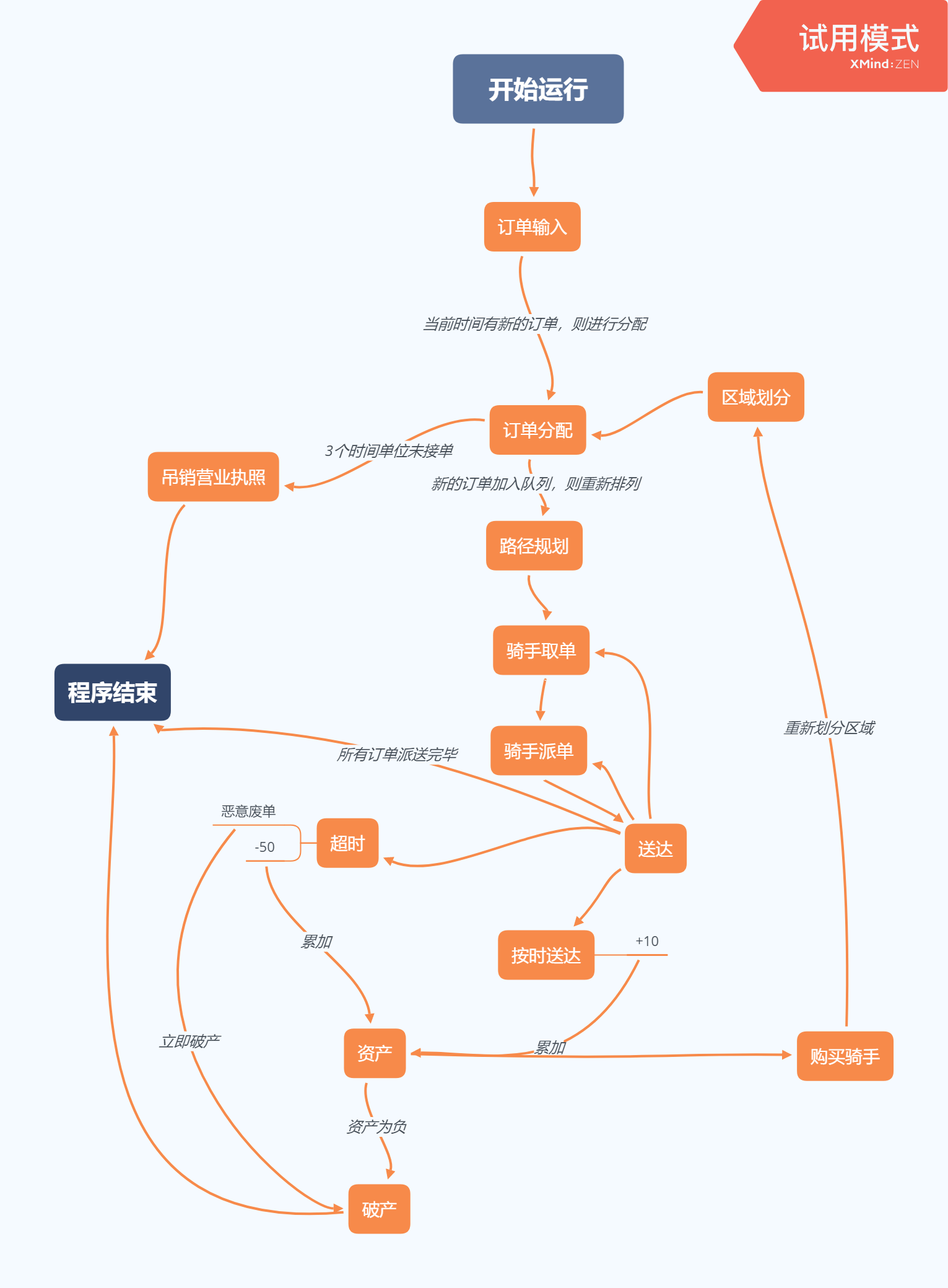
3.2各模块接口说明

Head.h

3.3各模块函数说明

|  |  |  |  |
| --- | --- | --- | --- |
| 模块文件 | 模块说明 | 模块包含的函数名 | 函数功能 |
| input.c | 读入输入文件，并将订单信息存储在结构体数组中 | extern void input(void) |  |
| output.c | 将信息（当前账户金额数、每位骑手的位置、接单数、完成数、超时数）写在输出文件中 | extern void txtoutput(void)  extern void pictureoutput(void) | 文本输出和符号化的图形输出 |
| divide.c | 根据骑手数量进行模块划分 | extern void divide(void) | 根据现有骑手数量修改每位骑手派送区域 |
| distribute.c | 给骑手分配订单 | extern void decide(void  ) | 判定某个订单属于哪个骑手并分配订单 |
| path.c | 规划路径 | extern void bestwaywhile(void)  extern void bestway(void) | 进行循环以确保为每位骑手都规化了路径  判断是否加入订单，若加入则重新规划，未加入则不执行任何操作 |
| add.c | 1. 根据任务单进行所有骑手的移动并且统计移动后的情况（各个骑手的完成数，超时数。钱数）。 | extern void addwhile(void)  extern void move(void) | 进行循环确保每位骑手都走了一步  根据骑手现在的坐标以及当前要去的坐标改变骑手坐标 |
|  | 2.设置预警判断是否结束程序。  3.判断是否购买骑手并完成购买骑手的操作。 | extern void current(void) | 根据骑手当前位置更新骑手订单完成情况和钱数，并根据更新后的状态判断是否结束程序（根据订单是否被恶意废单，是否在3个时间单位内没有派单，是否破产） |
|  |  | extern int buy(void) | 根据当前情况决定是否购买骑手,未购买则返回0，购买则返回1 |

4.核心算法设计

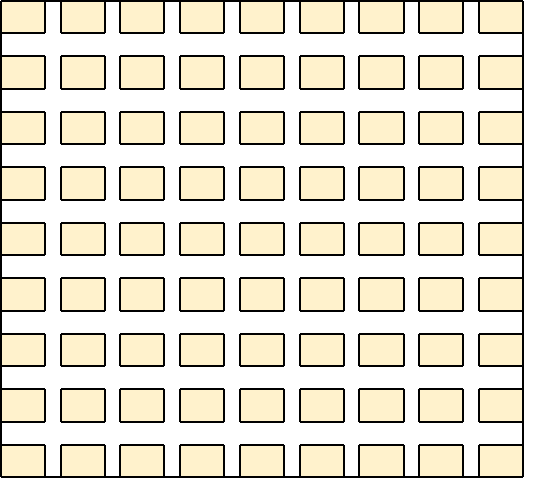
****

**【区域划分】**

骑手数量≤4时，则保留1个跨区域骑手，其它作为子区域骑手，均匀划分区域

当骑手数目≥5时，可以如图划分4个子区域，分配4个子区域骑手，其他的作为跨区域骑手

④



③

①

②

在4\*7的区域内，骑手绕圈，将有效避免超时情况

**【骑手订单排列】**

当前时间有新的订单发出，则进行订单分配

新的订单进入骑手的订单序列，则要重新进行订单序列的排序

1

2

2

1

3

3

计算各个任务点（取餐点或送餐点）与骑手的距离X 与 权重 weight 的乘积，作为参与排序的参考值，参考值小则排在序列前面

若未有新的订单进入订单序列，骑手订单序列不变，按顺序执行

**【骑手路径】**

“∟”形路径

先令 X 增加 再令 Y 增加