# 15\_<03>小组\_概要设计（13周版）

版本号：03小组—Design—002

编制时间：2019年5月23日

编制人员：吴静迪、王宇航、蒋明君

**1.1输入、输出设计**

**输入：(1) 文件方式：**1.程序启动，建立一个表示订单的结构数组。2.检查当前目录下是否有标准文件名“sales.txt”, 如果有，则以只读方式打开文件，逐行读入，从第一个订单开始，将当前行的六个数据分别赋给：表示当前订单编号、下单时间、餐馆横坐标、餐馆纵坐标、食客横坐标、食客纵坐标的变量。3.继续读取下一行数据，对应下一个订单，读到文件尾时，关闭文件。**(2) 命令行方式：**1.程序启动，建立一个表示订单的结构数组。2.检查当前目录，若无标准文件“sales.txt”，运行读取输入代码段：从第一个订单开始，将读入的六个数据分别赋给：表示当前订单编号、下单时间、餐馆横坐标、餐馆纵坐标、食客横坐标、食客纵坐标的变量。3.继续读取下一个订单的数据，直至订单数组初始化完毕。**输出：(1)文件方式：**1.以更新(r+)方式打开文件“output.txt”，该文件包含：当前账户金额数、每位骑手的位置（横纵坐标）、接单数、完成数、超时数。2.读取当前时间单位的信息，写入输出文件中。3.每隔一个时间单位，就对文件“output.txt”，进行一次更新，文件尾部追加更新后的内容。4.直到程序结束时停止更新并关闭文件**(2)命令行方式：** 程序每运行一个时间单位，就读取当前时间单位的信息，将当前账户金额数、每位骑手的位置（横纵坐标）、接单数、完成数、超时数目连同数据打印到终端。

**1.2算法设计**

1. 对于订单的分配：

设置一个全局变量num\_deliver\_man 标定快递员数组的结尾即现在快递员的数量，对于订单分配，假设快递员接到这个订单用dsf算出最小死亡和最短路径，再结合快递员当前的订单数量给出一个权值，对现在的每一个快递员做这个过程，选出权值最小的订单派给该骑手。

1. 对于快递员路径的优化：

设计一个外卖员的数组储存所有外卖员的信息，每次时间遍历这个数组。设计了一个深搜算法，利用这个深搜算法当某一个外卖员被分配新任务或者快递员的订单列表即将完成的时候，重新规划路线，选出在最小罚单数下最快遍历这个图的方法。为了防止深搜的时间复杂度过高，设置了当快递员订单数量多于8单（注：一个订单被拆成两个订单），按时间顺序搜索前八单，即寻找到以最短时间遍历这个图的办法。重置快递员的任务队列。

**1.3高层数据结构定义**

**1.3.1 常量定义：**

#define max\_dispose 30 //设置为一个外卖员处理订单的最大数量

#define max\_deliver\_man 100 //设置为外卖员数组的大小，即预计会招快递员的最大人数

#define inf 0x3f3f3f3f //定义可能为数组赋的最大值

#define dead\_line 30 //超过这个时间就会被认为超时

#define total\_dead\_line 60 //超过这个时间限制直接破产

#define initial\_x\_location 9 //骑手初始x坐标

#define initial\_y\_location 8 //骑手初始y坐标

#define warning 5 //跨区域骑手预警值

#define deep\_dsf 8 //限制dsf算法最大搜索深度，控制时间复杂度

**1.3.2 全局变量定义：**

//保存输出信息的变量

extern int time; //作为总的时间变量，记录时间

extern int num\_order; //记录接单数量

extern int num\_completed; //记录完成订单的变量

extern int num\_overtime; //记录超时的订单数量

extern int money; //记录钱的数量

extern int is\_break; //若出现破产条件将其置一，跳出循环

extern int num\_completing[20]; //储存当前时间结单编号的数组

extern int num\_overtiming[20]; //储存当前时间超时单编号的数组

extern int num\_iscompleting; //记录当前时间结单数

extern int num\_isovertiming; //记录当前时间超时数

//每一次处理的外卖员数组

extern int num\_deliver\_man; //用该变量标记快递员的人数

extern deliver deliver\_man[max\_deliver\_man];

//总的任务队列

extern Linkqueue mission\_queue;

**1.3.3 结构体定义：**

**1.//一个外卖员的结构体**

typedef struct

{

int x\_location = initial\_x\_location; //外卖员x坐标

int y\_location = initial\_y\_location; //外卖员y坐标

int num\_mission = 0; //外卖员的任务数量

int mission\_queue[max\_dispose \* 2] = { -1 }; //标记外卖员的任务队列，每完成一个任务index\_misson++

int index\_misson = 0; //标记任务进行到哪一个

int is\_allocate\_new = 0; //是否在一个时间段内被分配了新任务

mission task[max\_dispose]; //该外卖员处理订单列表

int stopflag1; //代表外卖员左边/上边的停靠情况

int stopflag2; //代表外卖员右边/下边的停靠情况

} deliver;

**2.//一个任务的信息记录**

typedef struct

{

int number = 0; //订单编号

int order\_time = 0; //下单时间

int x\_start = -1; //餐馆x坐标

int y\_start = -1; //餐馆y坐标

int x\_terminal = -1; //食客x坐标

int y\_terminal = -1; //食客y坐标

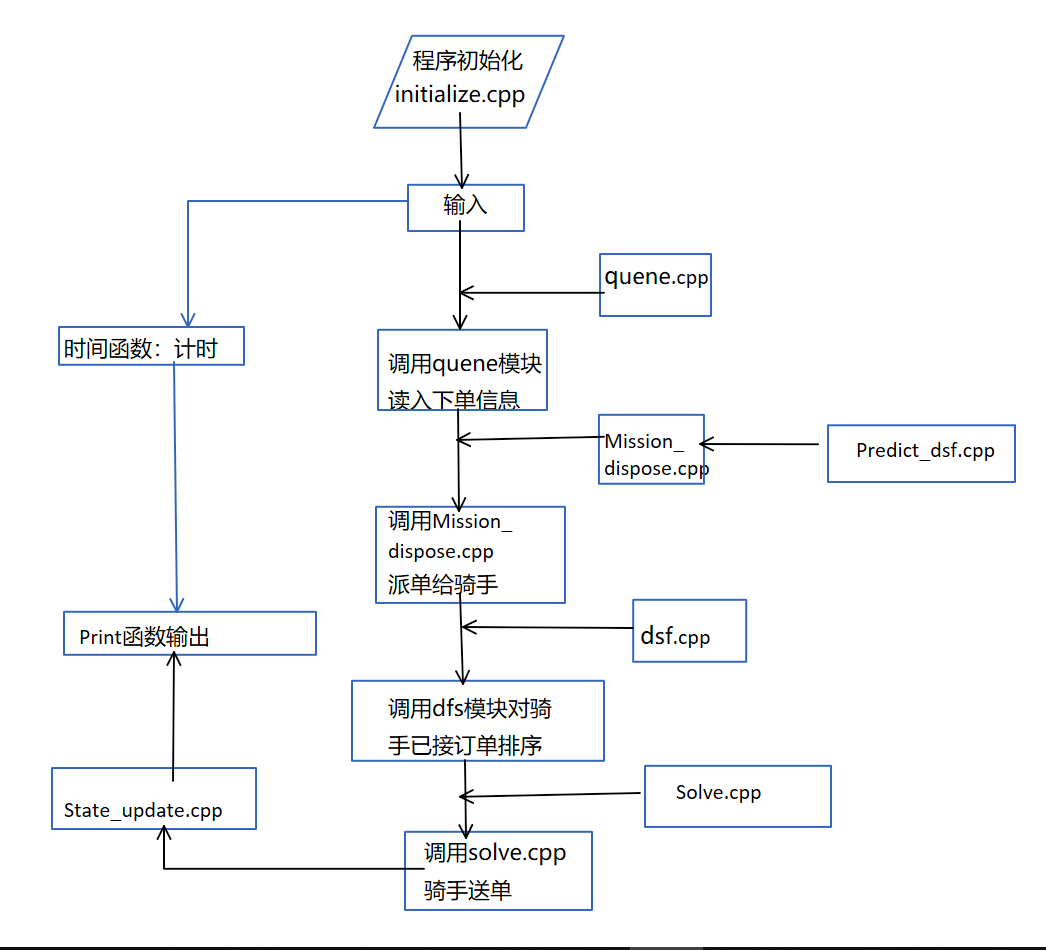
int fetch\_time = 0; //标记该订单被取走的时间，如果订单还未被取走fetch\_time==0

int is\_complete = 1; //如果该变量为1说明完成了任务，下一个插入外卖员任务队列直接插在这,一开始全初始化为一，标记为空订单

} mission;

**1.4 系统模块划分**

**模块关系图:**



**各模块功能：**

1.模块名称 quene.cpp

模块功能简要描述：队列函数操作集合

2.模块名称 solve.cpp

模块功能简要描述：完整表示骑手送单过程

3.模块名称 dsf.cpp和predict\_dsf.cpp

模块功能简要描述：路径优化算法模块

4.模块名称 mission\_dispose.cpp

模块功能简要描述：派单给相应骑手

5.模块名称 state\_update.cpp

模块功能简要描述：对状态进行实时更新