**模拟外卖派单详细设计**

**模块1<输入模块>**

* 局部数据结构设计

int number=0,x\_r=0,y\_r=0,x\_c=0,y\_c=0,t0\_0=0,i=0;//与订单结构体对应的临时变量，i为循环变量

struct orderList \*headptr=NULL,\*currentptr=NULL,\*lastptr=NULL;//订单链表的头节点、当前节点、下一节点

FILE \*fptr;//文件指针

* 算法设计

orderlistptr input\_File()

fclose(stdin);

fclose(fptr);

开辟数组orderlist\_array[num\_of\_orders+1];

currentptr=headptr;

return headptr;

将链表中订单变量对应存储到结构体数组中

orderlist\_array[i].rider=currentptr->rider;

currentptr=currentptr->nextOrder;

for (i=1;i<=num\_of\_orders;i++)

headptr==NULL

Yes No

给订单信息链表中变量赋值 给订单信息链表中变量赋值

指向下一节点 指向下一节点

num\_of\_orders++; num\_of\_orders++;

lastptr->nextOrder=NULL;

开辟currentptr空间

if 开辟成功

freopen(“文件名”，“r”,stdin)

while(scanf(读取一组订单数据)！=EOF

if(fptr=fopen(“文件名”，“r”)!=NULL

No Yes

**模块2<建图模块>**

* 1.

局部数据结构设计

int i,j;//循环变量

int id=1;//mark indexes of the road

算法设计

void initial\_Coor\_G(COOR\_G a[][17][THI])

for(i=0;i<=16;i++)

判断是否为十字路口

for()

for()//初始化周围节点的坐标

循环 将周围节点中超出坐标图的节点设为无效

初始化每一个index坐标

是否为道路

Yes No

是否为十字路口 index、tag标记为INV

Yes No

标记为十字路口 储存其坐标

for(i=0;i<=16;i++)//initialize tag

for(j=0;j<=16;j++)

周围道路节点初始化

index初始化

for(int t=1 ;t<=4 ;++t)

for(int k=0 ;k<THI ;++k)

for(j = 0; j <= 16; j++)

2.

局部数据结构设计

int i；//循环变量

int x\_cli,y\_cli,x\_res,y\_res;//临时变量

算法设计

void assign\_Ind()

for(i=1;i<=num\_of\_orders;i++)//遍历订单

x\_cli、y\_cli被赋值为对应订单中食客坐标

CG(坐标)图中该x、y、订单编号对应的index赋值为订单编号

CG(坐标)图中该x、y、订单编号对应的tag标记为CLI

x\_res、y\_res被赋值为对应订单中餐馆坐标

CG(坐标)图中该x、y、订单编号对应的index赋值为订单编号

CG(坐标)图中该x、y、订单编号对应的tag标记为RES

* 局部数据结构设计

int i,j,x;//循环变量

算法设计

void initial\_Rel\_G()

for(i=0;i<17;i++)

for(j=0;j<17;j++)//初始化

for(i=0;i<17;i++)

for(j=0;j<17;j++)

RG[][]=MAX;

x=CG图中index

路径

Yes No

RG[x][x]=0;

是否有房间（餐馆、食客可能所在）

Yes No

按行标记初始化RG 按列标记初始化RG

floyd()函数求各个顶点的最短路径

* 1. 局部数据结构设计

int i,j;//循环变量

算法设计

void initial\_Path\_G()

for(i=1;i<=144;i++)

for(j=1;j<=144;j++)

初始化PG图 i=j-->PG[][]=0;

2. 局部数据结构设计

函数无内部变量

算法设计

int add\_node(int x1 ,int y1 ,int x2 ,int y2 ,int num1 ,int num2 ,int tag1 ,int tag2)

else第二种：从餐馆到食客

Yes No

for(int i=1;i<=4;i++)

If第一种情况：从餐馆到餐馆

Yes No

for(int i=1;i<=4;i++)

CG.x,y,0周围节点valid!=RAN&&CG.x,y,num1

周围节点valid==1

Yes No

add\_node\_start(CG[x1][y1][0].arou[i].index, x2, y2, INV ,tag2);

CG.x,y,0周围节点valid!=RAN&&CG.x,y,num1周围节点valid==1

Yes No

add\_node\_start(CG[x1][y1][0].arou[i].index, x2, y2, num2 ,tag2);

else第三种：CLI to CLI

Yes No

for(int i=1;i<=4;i++)

CG.x,y,0周围节点valid!=RAN&&CG.x,y,0==1

Yes

No

add\_node\_start(CG[x1][y1][0].arou[i].index, x2, y2, INV ,tag2);

3. 局部数据结构设计

int min=MAX;

算法设计

int add\_node\_start(int index ,int x ,int y ,int num ,int tag)

tag==RES

Yes No(if(tag==CLI))

for(int i=1 ;i<=4 ;++i) for(int i=1 ;i<=4 ;++i)

for(int i=1;i<=4;i++) for（int i=1;i<=4;i++）

return（min）;

min==RG[index][CG[x][y][0].arou[i].index]

PG[index][CG[x][y][0].arou[i].index]=RG[index][CG[x][y][0].arou[i].index]

If min==下单时间t0-spent\_time和最短路径之间最大值

在PG图中将两个节点连接

min=min和RG[index][CG[x][y][0].arou[i].index]相比得到的最大值

min=通过比较得到下单时间-spent\_time和最短路径之间最大值

(引用了max函数)

**模块3<模拟模块>**

* 模拟开始寻找距离当前起点的最近餐馆、（找到合适的道路节点做起点后）模拟从起点开始的送单过程
* 1.局部数据结构设计

int temp；

算法设计

int simulate\_start()

int temp= find\_near\_Res(cur\_start);

temp!=INV

Yes No

在CG图中激活当前节点和订单，且add\_node\_pathlist

temp==INV

Yes No

return INV;

从此起点到最近餐馆所用时间+餐馆到对应食客所用

时间<订单失效的时间

Yes No

int min\_dist=此起点到最近餐馆所用时间

此订单餐馆横坐标储存到cur\_posi[0]；

纵坐标cur\_posi[1]；

cur\_order\_index=订单编号

调用函数add\_node\_start

spent\_time+=此起点到最近餐馆所用时间

将订单中vis\_res标记为VIS

return 1;

调用函数delete\_node\_pathlist()，将餐馆坐标从path路径中删除

调用函数inactivate\_Ro\_start

灭活可行起点

return INV;

* 2.局部数据结构设计

int temp;//临时变量：订单编号

int min;//临时变量

算法设计

void simulate()

temp=find\_near\_res\_and\_res;//时间、距离综合考虑后离第一个餐馆最近的餐馆所对应的订单编号；min=MAX;

找到了合适的订单编号

Yes No

调用函数active\_Ro （ 将CG图中后一个餐馆周围节点valid=1，且

add\_node\_pathlist

while(temp != INV)

add\_cli\_to\_PG();

走相应路线不超过t\_end

Yes No

将此节点变为last\_posi,后一个餐馆为cur\_posi

将两餐馆间花费的时间加到spent\_time

cur\_order\_index=temp;

OL(temp).vis\_res = VIS;

temp=find\_near\_res\_and\_res;//时间、距离综合考虑后离此餐馆最近的餐馆所对应的订单编号；

if(temp != INV) active\_Ro

delete\_node\_pathlist();

inactivate\_Ro

OL(temp).valid = INV;

**模块4<输出模块>**

* 局部数据结构设计

int i=0;

算法设计

void output\_File(outputlistptr currentptr,output\_riderptr current)

freopen("output.txt","a+",stdout)

printf:时间、钱、接单数、完成数、超时数

fclose(stdout)

输出骑手所在位置

链表节点指针指向下一个

for(i=0;i<骑手个数;i++)