同濟大學
TONGJI UNIVERSITY

项目说明文档

勇闯迷宫游戏

指导教师：张颖

1751984 王舸飞

1.分析

1.1 背景分析

已知有这样一个迷宫，它只存在一个入口和出口。一个骑士骑马从入口进入迷宫，他需要在迷宫中寻找一条通路从入口走到出口。

1.2 功能分析

本次设计对功能的要求较为具体，描述如下：

迷宫问题的求解过程可以采用回溯法即在一定的约束条件下试探地搜索前进，若前进中受阻，则及时回头纠正错误另择通路继续搜索的方法。从入口出发，按某一方向向前探索，若能走通，即某处可达，则到达新点，否则探索下一个方向；若所有的方向均没有通路，则沿原路返回前一点，换下一个方向再继续试探，直到所有可能的道路都探索到，或找到一条通路，或无路可走又返回入口点。在求解过程中，为了保证在达到某一个点后不能向前继续行走时，能正确返回前一个以便从下一个方向向前试探，则需要在试探过程中保存所能够达到的每个点的下标以及该点前进的方向，当找到出口时试探过程就结束了。

即要求用上文指定的方法实现路径的寻找和输出。

2.设计

2.1 数据结构设计

在对算法功能的要求中可以发现，算法要求从入口出发沿某一方向探索，若能走通则继续前进，否则要沿原路返回之前点，即要求有一个存储结构存储之前所经过的一系列点，这很容易让人联想到栈的结构，故本次主要采用的数据结构为一个栈。

此外，为了存储之前的坐标和链的指向，还需要一个表示节点的结构体。

2.2 类结构设计

本次课程设计的栈未采用标准库，而是采用了自己定义的链式栈WorkStack，用于存储之前定义的结构体position。

2.3 成员与操作设计

链式栈类(Workstack)

受保护的成员：

```
 private:
 position *top;//栈的头指针
```

公有操作：

```
public:
    WorkStack(){top=NULL;}//置头为空的构造函数

    void Push(position *x);//将x押入栈的函数

    void Pop(position *px);//出栈并赋值给px的函数

    bool IsEmpty(){return top==NULL?true:false;}//判断是否为空的函数

    void Display();//输出函数

    position* GetTop(){return top;}//获取当前头指针
```

代表位置的结构体节点(position)

```
 struct position{
     int x;
     int y;
     int dir=0;
     position *next=NULL;
 };
```

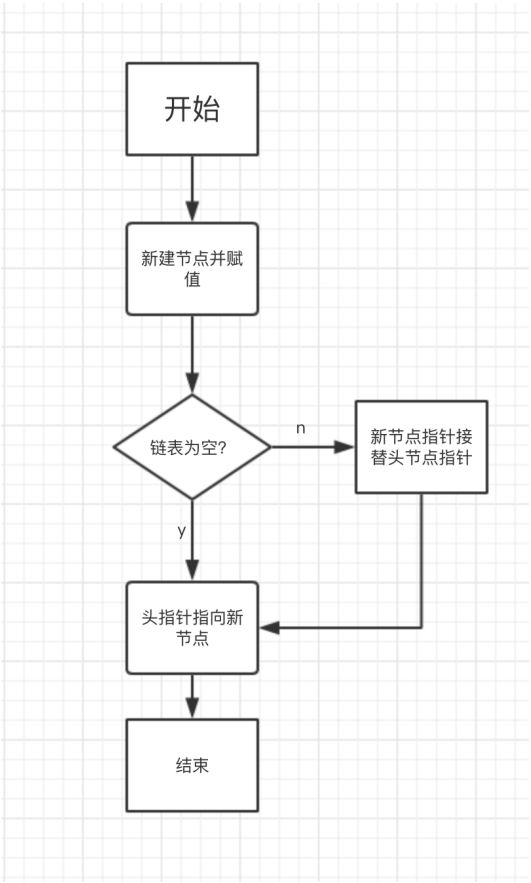
2.4 系统设计

程序运行之后，系统会给出一个预先设定好的迷宫并将它输出展示，之后调用solution函数给出路径并将其输出，完成题目要求。

3.实现

3.1 押入功能的实现

3.1.1 押入功能流程图

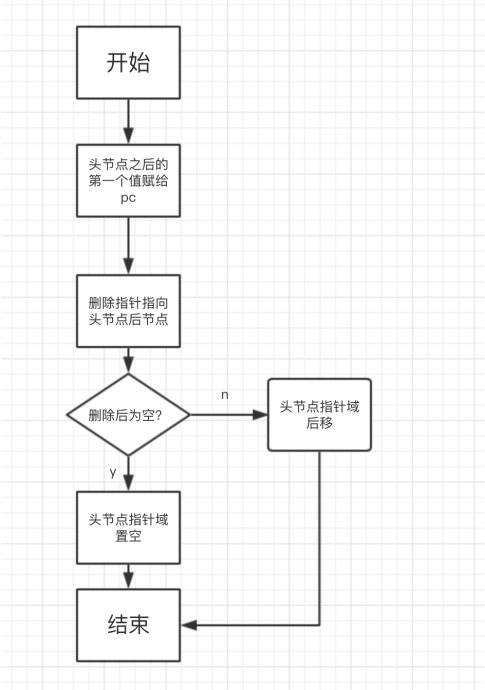


3.1.2 押入功能核心代码

```
void WorkStack::Push(position *px){if(top!=NULL){position *in=new position;
    in->x=px->x;in->y=px->y;in->dir=px->dir;
    in->next=top;top=in;
}
else{position *in=new position;
    in->x=px->x;in->y=px->y;in->dir=px->dir;
    top=in;}
}
```

3.2 弹出功能的实现

3.1.1 弹出功能流程图

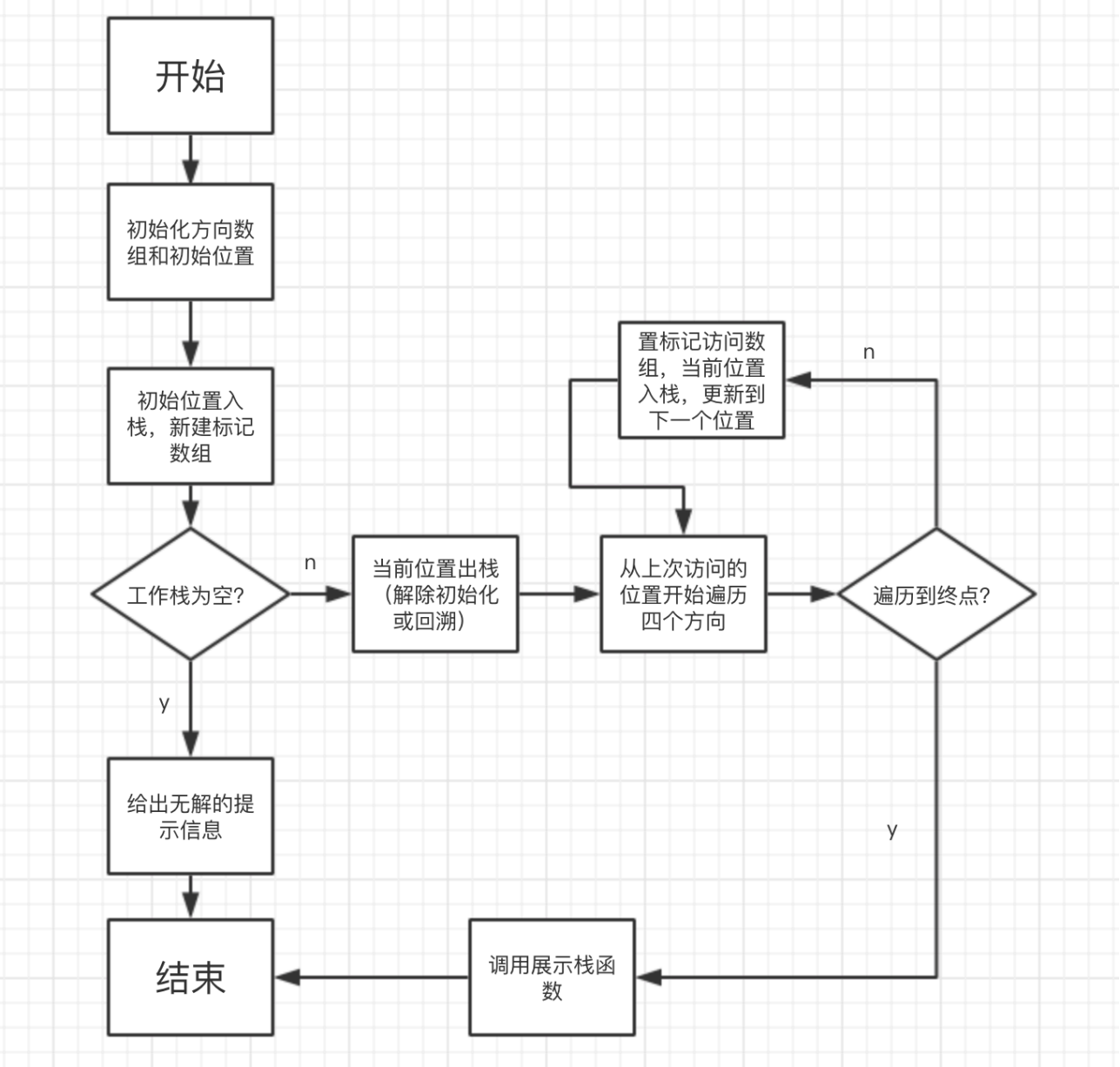


3.1.2 弹出功能核心代码

```
void WorkStack::Pop(position *px){position *t=new position;t=top;
    top=top->next;
    px->x=t->x;px->y=t->y;px->dir=t->dir;
    delete t;
}
```

3.3 寻找出口功能的实现

3.3.1 寻找出口功能流程图



3.3.2 solution函数核心代码

```
void solution(char map[7][7]){offsets move[4];
    move[0].l=-1;move[0].r=0;
    move[1].l=1;move[1].r=0;
    move[2].l=0;move[2].r=-1;
    move[3].l=0;move[3].r=1;

    int i,j,d(0),g,h;

    position *tmp=new position;tmp->x=1;tmp->y=1;
    WorkStack ws;ws.Push(tmp);

    char mark[7][7]={'0','0','0','0','0','0','0',
        '0','#','0','0','0','0','0',
        '0','0','0','0','0','0','0',
        '0','0','0','0','0','0','0',
        '0','0','0','0','0','0','0',
        '0','0','0','0','0','0','0'
    };

    while(ws.IsEmpty()==false){
        ws.Pop(tmp);
        i=tmp->x;j=tmp->y;d=tmp->dir;
        while(d<4){g=i+move[d].l;h=j+move[d].r;
            if(g==5&&h==5){cout<<"(5,5)<-"<<"("<<i<<","<<j<<")";ws.Display();return;}
            if(map[g][h]=='0'&&mark[g][h]=='0'){mark[g][h]='#';tmp->x=i;tmp->y=j;tmp->dir=d;ws.Push(tmp);i=g;j=h;d=0;}
            else d++;
        }
    }
    cout<<"no path in maze"<<endl;
}
```

4.测试

4.1 常规测试

4.1.1 指定测试用例

```
迷宫地图:
 # # # # # # #
 # 0 # 0 0 0 #
 # 0 # 0 # # #
 # 0 0 0 # 0 #
 # 0 # 0 0 0 #
 # 0 # 0 # 0 #
 # # # # # # #
迷宫路径:
(5,5)<-(4,5)<-(4,4)<-(4,3)<-(3,3)<-(3,2)<-(3,1)<-(2,1)<-(1,1)
Program ended with exit code: 0
```

4.1.2 无解的情况

```
迷宫地图:
 # # # # # # #
 # 0 # 0 0 0 #
 # 0 # 0 # # #
 # 0 0 0 # 0 #
 # 0 # 0 # 0 #
 # 0 # 0 # 0 #
 # # # # # # #
迷宫路径:
no path in maze
```