数据结构实验报告

<table>
<tr><td>学号-姓名</td><td>桑龙龙-20030540015</td><td>实验时间</td><td>2020 年 10 月 17 日</td></tr>
<tr><td>诚信声明</td><td colspan="3">本实验及实验报告所写内容为本人所作，没有抄袭。 桑龙龙</td></tr>
<tr><td>实验题目</td><td colspan="3">实验二 栈和队列的实现与应用题目
题目一 数制转换
题目二 括号匹配问题
题目三 停车场管理
题目四 迷宫问题</td></tr>
<tr><td>实验过程中遇到的主要问题</td><td colspan="3">在实现链栈的过程中，考虑如何使链栈的实现更为实用，最后通过将链栈封装到类中。</td></tr>
<tr><td>实验小结</td><td colspan="3">本次实验进行了队列和栈的应用，通过这次实验，对递归理解更为深刻，我们通过递归可以使代码看起来更为简洁，这是通过系统栈帮助我们实现的，当我自己将以前以递归方法写的代码改成非递归方法时，需要自己去注意很多细节，如何标记一个方法的运行过程，如何判断进栈出栈条件，学习到很多。</td></tr>
<tr><td>数据结构
（自定义数据类型）</td><td colspan="3">

```
struct node{
    int x, y , i;
    //x,y 表示坐标
    //i 表示要递归哪个方向，初始 i 为 1
    //当 i 为 5 的时候表示 4 个方向已经递归过
    node(int x, int y, int i){
        this->x = x;
        this->y = y;
        this->i = i;
    }
};
class linkstack{
    //链栈
    struct mynode{
        int x, y , i;
        mynode* next;
        mynode(int x, int y, int i){
            this->x = x;
            this->y = y;
            this->i = i;
            this->next = NULL;
        }
        mynode(const node& a){
            mynode(a.x, a.y, a.i);
        }
```

</td></tr>
</table>

```
    };
    private:
        int mysize;
        mynode* head;
    public:
        linkstack(){
            mysize = 0;
            head = NULL;
        }
        void push(node& a){
            mynode* temp = new mynode(a.x, a.y, a.i);
            mysize++;
            if(head == NULL){
                head = temp;
            }else{
                temp->next = head;
                head = temp;
            }
        }
        node top(){
            node ret(head->x, head->y, head->i);
            return ret;
        }
        void pop(){
            if(!head) return;
            mynode* temp = head->next;
            delete head;
            head = temp;
            mysize--;
        }
        int size(){
            return mysize;
        }
        bool empty(){
            return mysize == 0;
        }
};
```

主要算法 （或算法说明）	

```c
//1、数制转换
#include<stdlib.h>
#include <stdio.h>
int stack[30];
//栈
void d2b(long long ori);
//十进制转二进制

int main(){
    long long ori;
    scanf("%lld\n",&ori);
    d2b(ori);
    return 0;
}
void d2b(long long ori){
    //十进制转二进制
    if(ori == 0)
    {//如果 ori 为 0

        putchar('0');
        return;
    }else if(ori < 0)
    {//如果 ori 为负数

        putchar('-');
        ori = -ori;
    }
    int top=0;
    while(ori){
        stack[top++] = ori % 2;
        ori /= 2;
    }
    while(top){
        putchar('0' + stack[(top--) - 1]);
    }
}
//1、数制转换 结束
```

```
//4、迷宫问题
#include<stdlib.h>
#include <stdio.h>
#define N 256
/*
使用语言:C++
输入格式
第一行两个空格分隔的正整数 r，c
随后 r 行，每行 c 个由 01 组成的字符
例:
5 5
01000
00100
10000
11110
00000
注释：0 表示可以走，1 表示不可以走
*/
int dir[5][2]={0,0,1,0,0,1,-1,0,0,-1};
//方向数组 dir[1 2 3 4]表示向下右上左四个方向
int r,c;
//表示图的行数和列数
bool grid[N][N];
//表示一个 r*c 的 10 地图
bool vis[N][N];
//访问标记
char path[N][N];
//路径记录
class linkstack;
//链栈
struct node;
void pri(int i);


int main(){
    //输入部分
    scanf("%d %d\n", &r, &c);
    char a;
    for(int i = 0; i < r ; i++){
        for(int j = 0; j < c; j++){
            scanf("%c", &a);
            grid[i][j] = a == '1' ? false : true;
        }
```

```
        getchar();
    }
    //主程序部分
    //如果入口 grid[0][0]处就是 1，表示被堵死
    if(!grid[0][0]){
        printf("end\n");
        return 0;
    }
    linkstack S;
    //声明链栈
    node cur(0, 0, 1);
    S.push(cur);
    bool find_path = false;
    while(!S.empty()){
        //模拟递归
        cur = S.top();
        S.pop();
        if(cur.i == 5){
            //(cur.x,cur.y)坐标递归完成
            //并将其访问标记重新标记为未访问
            vis[cur.x][cur.y] = false;
            continue;
        }else{
            //标记(cur.x,cur.y)被访问标记
            vis[cur.x][cur.y] = true;
        }
        if(cur.x == r - 1 && cur.y == c -1 ){
            //到达终点（右下角），停止循环
            find_path = true;
            break;
        }
        cur.i++;
        S.push(cur);//重新入站
        cur.i--;
        //next_x 和 next_y 下一个要访问的点
        int next_x = dir[cur.i][0] + cur.x;
        int next_y = dir[cur.i][1] + cur.y;
        if(next_x >= r || next_x < 0 || next_y >= c || next_y
< 0) continue;
        if(vis[next_x][next_y] || !grid[next_x][next_y]) conti
nue;
        //如果 next_x,next_y 出了 grid 的范围
        //或者本身是一堵墙（即 grid[x][y]==false）或者被访问过
        path[next_x][next_y] = cur.i;
        //记录路径
```

```
88.         node next(next_x, next_y, 1);
89.         S.push(next);
90.     }
91.     while(!S.empty()) S.pop();
92.     cur = {r-1, c-1, path[r-1][c-1]};
93.     while(cur.x != 0 || cur.y != 0){
94.         S.push(cur);
95.         int x = cur.x, y = cur.y;
96.         cur.x = x - dir[cur.i][0];
97.         cur.y = y - dir[cur.i][1];
98.         cur.i = path[cur.x][cur.y];
99.     }
100.     S.push(cur);
101.     while(!S.empty()){
102.         cur = S.top();
103.         S.pop();
104.         if(!S.empty()){
105.             printf("%d %d ", cur.x, cur.y);
106.             pri(S.top().i);
107.         }
108.         else printf("%d %d end\n", cur.x, cur.y);
109.     }
110.     return 0;
111. }
112. void pri(int i){
113.     if(i == 1) printf("down\n");
114.     else if(i == 2) printf("right\n");
115.     else if(i == 3) printf("up\n");
116.     else printf("left\n");
117. }
118. struct node{
119.     int x, y , i;
120.     //x,y 表示坐标
121.     //i 表示要递归哪个方向，初始 i 为 1
122.     //当 i 为 5 的时候表示 4 个方向已经递归过
123.     node(int x, int y, int i){
124.         this->x = x;
125.         this->y = y;
126.         this->i = i;
127.     }
128. };
129. class linkstack{
130.     struct mynode{
131.         int x, y , i;
132.         mynode* next;
```

```
133.         mynode(int x, int y, int i){
134.             this->x = x;
135.             this->y = y;
136.             this->i = i;
137.             this->next = NULL;
138.         }
139.         mynode(const node& a){
140.             mynode(a.x, a.y, a.i);
141.         }
142.     };
143.     private:
144.         int mysize;
145.         mynode* head;
146.     public:
147.         linkstack(){
148.             mysize = 0;
149.             head = NULL;
150.         }
151.         void push(node& a){
152.             mynode* temp = new mynode(a.x, a.y, a.i);
153.             mysize++;
154.             if(head == NULL){
155.                 head = temp;
156.             }else{
157.                 temp->next = head;
158.                 head = temp;
159.             }
160.         }
161.         node top(){
162.             node ret(head->x, head->y, head->i);
163.             return ret;
164.         }
165.         void pop(){
166.             if(!head) return;
167.             mynode* temp = head->next;
168.             delete head;
169.             head = temp;
170.             mysize--;
171.         }
172.         int size(){
173.             return mysize;
174.         }
175.         bool empty(){
176.             return mysize == 0;
177.         }
```

```
178. };
179. //4、迷宫问题 结束
```