

数据结构实验报告

学号-姓名	桑龙龙-20030540015	实验时间	2020 年 10 月 17 日
诚信声明	本实验及实验报告所写内容为本人所作，没有抄袭。 		
实验题目	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		
实验过程中遇到的主要问题	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。		
实验小结	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。		
数据结构 （自定义数据类型）	<pre> 1. struct node{ 2. int x, y , i; 3. //x,y 表示坐标 4. //i 表示要递归哪个方向，初始 i 为 1 5. //当 i 为 5 的时候表示 4 个方向已经递归过 6. node(){} 7. node(int x, int y, int i){ 8. this->x = x; 9. this->y = y; 10. this->i = i; 11. } 12. }; 13. template <class T> 14. class linkstack{ 15. struct mynode{ 16. T a; 17. mynode* next; 18. mynode(T &a){ 19. this->a=a; 20. next=NULL; 21. } 22. }; </pre>		

```

23.     private:
24.         int mysize;
25.         mynode* head;
26.     public:
27.         linkstack(){
28.             mysize = 0;
29.             head = NULL;
30.         }
31.         void push(T &a){
32.             mynode* temp = new mynode(a);
33.             mysize++;
34.             if(head == NULL){
35.                 head = temp;
36.             }else{
37.                 temp->next = head;
38.                 head = temp;
39.             }
40.         }
41.         T top(){
42.             return head->a;
43.         }
44.         void pop(){
45.             if(!head) return;
46.             mynode* temp = head->next;
47.             delete head;
48.             head = temp;
49.             mysize--;
50.         }
51.         int size(){
52.             return mysize;
53.         }
54.         bool empty(){
55.             return mysize == 0;
56.         }
57. };
58.

```

主要算法
(或算法说明)

```
1. //1、数制转换
2. #include<stdlib.h>
3. #include <stdio.h>
4. int stack[30];
5. //栈
6. void d2b(long long ori);
7. //十进制转二进制
8.
9. int main(){
10.     long long ori;
11.     scanf("%lld\n",&ori);
12.     d2b(ori);
13.     return 0;
14. }
15. void d2b(long long ori){
16.     //十进制转二进制
17.     if(ori == 0)
18.     {//如果 ori 为 0
19.
20.         putchar('0');
21.         return;
22.     }else if(ori < 0)
23.     {//如果 ori 为负数
24.
25.         putchar('-');
26.         ori = -ori;
27.     }
28.     int top=0;
29.     while(ori){
30.         stack[top++] = ori % 2;
31.         ori /= 2;
32.     }
33.     while(top){
34.         putchar('0' + stack[(top--) - 1]);
35.     }
36. }
37. //1、数制转换 结束
```

```

38. //4、迷宫问题
39. #include<stdlib.h>
40. #include <stdio.h>
41. #define N 256
42. /*
43. 使用语言:C++
44. 输入格式
45. 第一行两个空格分隔的正整数 r, c
46. 随后 r 行, 每行 c 个由 01 组成的字符
47. 例:
48. 5 5
49. 01000
50. 00100
51. 10000
52. 11110
53. 00000
54. 注释: 0 表示可以走, 1 表示不可以走
55. */
56. int dir[5][2]={0,0,1,0,0,1,-1,0,0,-1};
57. //方向数组 dir[1 2 3 4]表示向下右上左四个方向
58. int r,c;
59. //表示图的行数和列数
60. bool grid[N][N];
61. //表示一个 r*c 的 10 地图
62. bool vis[N][N];
63. //访问标记
64. char path[N][N];
65. //路径记录
66.
67. void pri(int i);
68.
69. struct node{
70.     int x, y , i;
71.     //x,y 表示坐标
72.     //i 表示要递归哪个方向, 初始 i 为 1
73.     //当 i 为 5 的时候表示 4 个方向已经递归过
74.     node(){
75.
76.     }
77.     node(int x, int y, int i){
78.         this->x = x;
79.         this->y = y;
80.         this->i = i;
81.     }

```

```

82. };
83. //链栈
84. template <class T>
85. class linkstack{
86.     struct mynode{
87.         T a;
88.         mynode* next;
89.         mynode(T &a){
90.             this->a=a;
91.             next=NULL;
92.         }
93.     };
94.     private:
95.         int mysize;
96.         mynode* head;
97.     public:
98.         linkstack(){
99.             mysize = 0;
100.            head = NULL;
101.        }
102.        void push(T &a){
103.            mynode* temp = new mynode(a);
104.            mysize++;
105.            if(head == NULL){
106.                head = temp;
107.            }else{
108.                temp->next = head;
109.                head = temp;
110.            }
111.        }
112.        T top(){
113.            return head->a;
114.        }
115.        void pop(){
116.            if(!head) return;
117.            mynode* temp = head->next;
118.            delete head;
119.            head = temp;
120.            mysize--;
121.        }
122.        int size(){
123.            return mysize;
124.        }
125.        bool empty(){
126.            return mysize == 0;

```

```

127.     }
128. };
129.
130.
131.
132.
133. int main(){
134.     //输入部分
135.     scanf("%d %d\n", &r, &c);
136.     char a;
137.     for(int i = 0; i < r ; i++){
138.         for(int j = 0; j < c; j++){
139.             scanf("%c", &a);
140.             grid[i][j] = a == '1' ? false : true;
141.         }
142.         getchar();
143.     }
144.     //主程序部分
145.     //如果入口 grid[0][0]处就是 1，表示被堵死
146.     if(!grid[0][0]){
147.         printf("end\n");
148.         return 0;
149.     }
150.     linkstack<node> S;
151.     //声明链栈
152.     node cur(0, 0, 1);
153.     S.push(cur);
154.     bool find_path = false;
155.     while(!S.empty()){
156.         //模拟递归
157.         cur = S.top();
158.         S.pop();
159.         if(cur.i == 5){
160.             //(cur.x,cur.y)坐标递归完成
161.             //并将其访问标记重新标记为未访问
162.             vis[cur.x][cur.y] = false;
163.             continue;
164.         }else{
165.             //标记(cur.x,cur.y)被访问标记
166.             vis[cur.x][cur.y] = true;
167.         }
168.         if(cur.x == r - 1 && cur.y == c - 1 ){
169.             //到达终点（右下角），停止循环
170.             find_path = true;
171.             break;

```

```

172.     }
173.     cur.i++;
174.     S.push(cur); //重新入站
175.     cur.i--;
176.     //next_x 和 next_y 下一个要访问的点
177.     int next_x = dir[cur.i][0] + cur.x;
178.     int next_y = dir[cur.i][1] + cur.y;
179.     if(next_x >= r || next_x < 0 || next_y >= c || next_y
        < 0) continue;
180.     if(vis[next_x][next_y] || !grid[next_x][next_y]) cont
        inue;
181.     //如果 next_x,next_y 出了 grid 的范围
182.     //或者本身是一堵墙（即 grid[x][y]==false）或者被访问过
183.     path[next_x][next_y] = cur.i;
184.     //记录路径
185.     node next(next_x, next_y, 1);
186.     S.push(next);
187. }
188. while(!S.empty()) S.pop();
189. cur = {r-1, c-1, path[r-1][c-1]};
190. while(cur.x != 0 || cur.y != 0){
191.     S.push(cur);
192.     int x = cur.x, y = cur.y;
193.     cur.x = x - dir[cur.i][0];
194.     cur.y = y - dir[cur.i][1];
195.     cur.i = path[cur.x][cur.y];
196. }
197. S.push(cur);
198. while(!S.empty()){
199.     cur = S.top();
200.     S.pop();
201.     if(!S.empty()){
202.         printf("%d %d ", cur.x, cur.y);
203.         pri(S.top().i);
204.     }
205.     else printf("%d %d end\n", cur.x, cur.y);
206. }
207. return 0;
208. }
209. void pri(int i){
210.     if(i == 1) printf("down\n");
211.     else if(i == 2) printf("right\n");
212.     else if(i == 3) printf("up\n");
213.     else printf("left\n");
214. }

```

	215. //4、迷宫问题 结束
--	------------------