**河南开封科技传媒学院实验报告**

••••••••••••••••••••••••••••••••• 密 ••••••••••••••••••••••••••••••••• 封 ••••••••••••••••••••••••••••••••• 线 •••••••••••••••••••••••••••••••••

**20 -20 学年第 学期**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名： | 刘东 | | 学号： | 2136101045 | 专业： | 软件工程 | |
| 所在学院： | | 理工学院 | | | 指导老师： | | 袁帅 |
| 实验时间： | | 2023/3/15 | | | 实验地点： | | 6503 |
| 课程名称： | | 数据结构 | | | | | |
| 实验题目： | | 第四次实验 | | | | | |
| 实验目的： | | | | | | | |
| 实验内容：   1. 用代码实现一个栈并完成以下要求： 2. 输入0123入栈后再出栈并打印输出； 3. 对栈顶元素出栈后输出剩余元素并打印输出； 4. 判断此时栈是否为空； 5. 判断此时栈的长度； 6. 判断此时栈顶元素大小。   思考题：   1. 迷宫问题求解，用代码实现一个M\*N的迷宫从入口到出口的路线问题（可观看参考ppt） 2. 如果能实现1请尝试修改算法，实现寻找到的路径为最短路径。 | | | | | | | |
| 实验代码：  1，  #include<iostream>  using namespace std;  const int MAXSIZE = 1e4;  #define OK 1  #define ERROR 0  #define TRUE 1  #define FALSE 0  typedef int Status;  typedef struct {  int data[MAXSIZE];  int top;  } SqStack;  //打印元素  Status visit(int c) {  cout << c << " ";  return OK;  }  //初始化栈  Status InitStack(SqStack \*S) {  S->top = -1;  return OK;  }  //判断栈的长度  int StackLength(SqStack S) {  return S.top + 1;  }  /\* 若栈S为空栈，则返回TRUE，否则返回FALSE \*/  Status StackEmpty(SqStack S) {  if (S.top == -1)  return TRUE;  else  return FALSE;  }  //得到栈顶元素  Status GetTop(SqStack S, int \*e) {  if (S.top == -1) return ERROR;  else \*e = S.data[S.top];  return OK;  }  //插入元素e作为栈顶元素  Status Push(SqStack \*S, int e) {  if (S->top == MAXSIZE - 1) {  return ERROR;  }  S->top++;  S->data[S->top] = e;  return OK;  }  //删除栈顶元素  Status Pop(SqStack \*S) {  if (S->top == -1) return ERROR;  S->top--;  return OK;  }  //从栈顶到栈底显示每个元素  Status StackTraverse(SqStack S) {  int i;  i=0;  while (i <= S.top) {  visit(S.data[i++]);  }  cout << endl;  return OK;  }  int main() {  SqStack s;  if(InitStack(&s)==OK)  for(int j = 0;j < 4;j++)  Push(&s,j);  cout << "（1） 输入0123入栈后再出栈并打印输出；" << endl;  cout << "输入：" <<endl;  StackTraverse(s);  int t = 0;  cout << "出栈：" << endl;  while (t <= s.top) {  int e;  GetTop(s, &e);  Pop(&s);  cout << e << " ";  }  cout << endl;    cout << "（2） 对栈顶元素出栈后输出剩余元素并打印输出； " << endl;  if(InitStack(&s)==OK)  for(int j = 0;j < 4;j++)  Push(&s,j);  Pop(&s);  StackTraverse(s);    cout << "（3） 判断此时栈是否为空；" <<endl;  if (StackEmpty(s) == 1) cout << "空栈" << endl;  else cout << "不为空栈" << endl;    cout << "（4） 判断此时栈的长度；" <<endl;  cout << StackLength(s) << endl;    cout << "（5） 判断此时栈顶元素大小。" <<endl;  int e;  GetTop(s, &e);  cout << e << endl;  return 0;  }  思考一：  #include<iostream>  using namespace std;  #include<stack>  const int M = 8, N = 8;  typedef struct Q {  int x, y;  }Qn;  stack<Qn> result;  void dfs(int mg[M + 2][ N + 2], Qn q) {  if (q.x > 0 && q.y > 0 && q.x < 9 && q.y < 9){  if (q.x == 8 && q.y == 8) {  cout << "找到出口" << endl;  return;  }    if (mg[q.x][q.y + 1] == 0 && q.y + 1 < N + 1) {  q.y++;  result.push(q);  dfs(mg, q);  return;  }  if (mg[q.x + 1][q.y] == 0 && q.x + 1 < M + 1) {  q.x++;  result.push(q);  dfs(mg, q);  return;  }  if (mg[q.x][q.y - 1] == 0 && q.y - 1 > 0) {  mg[q.x][q.y] = 1;  result.pop();  q.y--;  dfs(mg, q);  return;  }  if (mg[q.x - 1][q.y] == 0 && q.x - 1 > 0) {  mg[q.x][q.y] = 1;  result.pop();  q.x--;  dfs(mg, q);  return;  }    }  }  int main () {  int mg[M+2][N+2]=  {  {1,1,1,1,1,1,1,1,1,1},  {1,0,0,1,0,0,0,1,0,1},  {1,0,0,1,0,0,0,1,0,1},  {1,0,0,0,0,1,1,0,0,1},  {1,0,1,1,1,0,0,0,0,1},  {1,0,0,0,1,0,0,0,0,1},  {1,0,1,0,0,0,1,0,0,1},  {1,0,1,1,1,0,1,1,0,1},  {1,1,0,0,0,0,0,0,0,1},  {1,1,1,1,1,1,1,1,1,1}  };  Qn q;    q.x = 1, q.y = 1;  result.push(q);  dfs(mg, q);  cout << "倒叙路径如下：" << endl;  while(!result.empty()) {  cout << "{" << result.top().x << ", " << result.top().y << "}" << endl;  result.pop();  }    return 0;  }  思考二：  #include<iostream>  using namespace std;  #include<queue>  const int M = 8, N = 8;  typedef struct Q {  int x, y;  }Qn;  queue<Qn> result;  Qn pre[M + 2][N + 2];  int dx[] = {-1, 0, 1, 0};  int dy[] = {0, 1, 0, 1};  void bfs(int mg[M + 2][N + 2], Qn q) {  result.push(q);  mg[q.x][q.y] = 1;    while (result.size()) {  auto u = result.front();  result.pop();  for (int i = 0; i < 4; i++) {  Qn dq;  dq.x = u.x + dx[i], dq.y = u.y + dy[i];  if (dq.x < 1 || dq.x >= 9 || dq.y < 1 || dq.y >= 9) continue;  if (mg[dq.x][dq.y]) continue;  mg[dq.x][dq.y] = 1;  pre[dq.x][dq.y] = u;  result.push(dq);  }  }  }  void print(int x, int y) {  if (x== 0 && y == 0) return;  Qn p = pre[x][y];  print (p.x, p.y);  cout << "{" << x << ", " << y << "}" << endl;  }  int main () {  int mg[M+2][N+2]=  {  {1,1,1,1,1,1,1,1,1,1},  {1,0,0,1,0,0,0,0,0,1},  {1,0,0,1,0,0,0,1,0,1},  {1,0,0,0,0,1,1,0,0,1},  {1,0,1,1,1,0,0,0,0,1},  {1,0,0,0,1,0,0,0,0,1},  {1,0,1,0,0,0,1,0,0,1},  {1,0,1,1,1,0,1,1,0,1},  {1,1,0,0,0,0,0,0,0,1},  {1,1,1,1,1,1,1,1,1,1}  };  Qn q;    q.x = 1, q.y = 1;      bfs(mg, q);    cout << "倒叙路径如下：" << endl;    print(8, 8);  return 0;  } | | | | | | | |
| 实验结果截图及实验心得： | | | | | | | |
|  | | | | | | | |
|  | | | | | | | |
|  | | | | | | | |
|  | | | | | | | |
|  | | | | | | | |