山东大学 计算机科学与技术 学院

数据结构与算法 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201705130113 | 姓名： 黄瑞哲 | | 班级： 计科17.3 |
| 实验题目：栈 | | | |
| 实验学时：2 | | 实验日期：2018.11.1 | |
| 实验目的：   1. 掌握栈结构的定义与实现 2. 掌握栈结构的使用 | | | |
| 软件环境：  G++ 8.1.0  Visual Studio Community 2017 | | | |
| 1. 实验内容（题目内容，输入要求，输出要求） 2. 创建栈类，采用数组描述 3. 计算数学表达式的值。输入数学表达式，输出表达式的计算结果。数学表达式由单个数字和运算符“+”、“-”、“\*”、“/”、“（”、“）”组成。 4. 以一个m\*n的长方阵表示迷宫，0和1分别表示迷宫中的通路和障碍。设计一个程序，对任意设定的迷宫，求一条从入口到出口的通路，或得出没有通路的结论。 5. 数据结构与算法描述 （整体思路描述，所需要的数据结构与算法） 6. 数学表达式计算：将表达式转换为后缀表达式，对后缀表达式进行操作时，遇到操作符则提取栈顶两个数并将运算结果压入栈顶直至栈空。中缀转后缀的方法为：遍历表达式中的元素，用符号栈暂存结果，如果为数字则存入结果线性表中，如果为运算符’)’则不断弹出符号栈中的符号存入结果线性表中直至遇到’(‘，如果为其他运算符，则将符号栈中所有比当前运算符优先级高的运算符弹出存入结果线性表中，再讲当前运算符压入符号栈。 7. 采用深度优先搜索。对于当前点，尝试他的四周，如果有点是合法的，则进入这个点继续搜索。如果这个点是终点则遍历系统栈中的节点，倒序输出即为路径。 8. 测试结果（测试输入，测试输出，结果分析）      1. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）   （1）计算表达式时需要先将中缀表达式转为后缀表达式，然后可以方便的用栈来求解。但是该方法不能正确求解形如3\*(-3)带有负数的表达式。可以尝试转化为3\*(0-3)的形式来求解。   1. 附录：实现源代码（本实验的全部源程序代码，程序风格清晰易理解，有充分的注释）   /\*linkedSatck.h\*/  #ifndef \_\_linkedStack\_H\_  #define \_\_linkedStack\_H\_  #include "stack.h"  #include <linearList/chain.h>  #include <stdexcept>  using namespace std;  template<typename T>  class linkedStack : public stack<T>  {  public:  linkedStack() :stackTop(nullptr), stackSize(0) {};  linkedStack(const linkedStack<T>&);  ~linkedStack();  bool empty() const { return stackSize == 0; }  int size() const { return stackSize; }  T top() const;  void pop();  void push(const T&);  void clear();  linkedStack<T>& operator=(const linkedStack<T>&);  protected:  chainNode<T>\* stackTop;  int stackSize;  };  template<typename T>  linkedStack<T>::linkedStack(const linkedStack<T>& s)  {  if (s.stackSize == 0)  {  stackTop = nullptr;  stackSize = 0;  }  else  {  stackTop = new chainNode<T>(s.stackTop->element);  chainNode<T>\* sourceNode = s.stackTop->next;  chainNode<T>\* currentNode = stackTop;  while (sourceNode != nullptr)  {  currentNode->next = new chainNode<T>(sourceNode->element);  currentNode = currentNode->next;  sourceNode = sourceNode->next;  }  stackSize = s.stackSize;  }  }  template<typename T>  linkedStack<T>::~linkedStack()  {  while (stackTop != nullptr)  {  chainNode<T>\* nextNode = stackTop->next;  delete stackTop;  stackTop = nextNode;  }  }  template<typename T>  T linkedStack<T>::top() const  {  if (stackSize == 0) throw out\_of\_range("stack is empty");  return stackTop->element;  }  template<typename T>  void linkedStack<T>::pop()  {  if (stackSize == 0) throw out\_of\_range("stack is empty");  chainNode<T>\* nextNode = stackTop->next;  delete stackTop;  stackTop = nextNode;  stackSize--;  }  template<typename T>  void linkedStack<T>::push(const T& theElement)  {  stackTop = new chainNode<T>(theElement, stackTop);  stackSize++;  }  template<typename T>  void linkedStack<T>::clear()  {  while (stackTop != nullptr)  {  chainNode<T>\* nextNode = stackTop->next;  delete stackTop;  stackTop = nextNode;  }  stackSize = 0;  }  template<typename T>  linkedStack<T>& linkedStack<T>::operator=(const linkedStack<T>& s)  {  if (this == &s) return \*this;  if (s.stackSize == 0)  {  stackTop = nullptr;  stackSize = 0;  }  else  {  stackTop = new chainNode<T>(s.stackTop->element);  chainNode<T>\* sourceNode = s.stackTop->next;  chainNode<T>\* currentNode = stackTop;  while (sourceNode != nullptr)  {  currentNode->next = new chainNode<T>(sourceNode->element);  currentNode = currentNode->next;  sourceNode = sourceNode->next;  }  stackSize = s.stackSize;  }  return \*this;  }  #endif //\_\_linkedStack\_H\_  /\*test1.h\*/  #ifndef LIB\_H\_  #define LIB\_H\_  #include <string>  #include <linearList/arrayList.h>  struct item  {  bool isDigit;  double num;  char op;  bool operator!=(const item& it) const {return isDigit != it.isDigit || num != it.num || op != it.op;}  };  bool isOp(char);  void translate(const std::string&, arrayList<item>&);  void infix2suffix(const arrayList<item>&, arrayList<item>&);  double calculate(const arrayList<item>&);  #endif  /\*test1.cpp\*/  #include "test1.h"  #include "pch.h"  #include <stack/linkedStack.h>  #include <sstream>  using namespace std;  //字符串转数字  template<typename T>  T str2num(const string &str)  {  istringstream iss(str);  T num;  iss >> num;  return num;  }  //优先级换算  int priority(char symbol)  {  switch (symbol)  {  case '+':  case '-':  return 0;  case '\*':  case '/':  return 1;  case '(':  case ')':  return 2;  }  return -1;  }  //判断是否为运算符  bool isOp(char symbol)  {  switch (symbol)  {  case '+':  case '-':  case '\*':  case '/':  case '(':  case ')':  return true;  }  return false;  }  //将字符串分解为单个数字和运算符  void translate(const string& str, arrayList<item>& vecIn)  {  int nCount = str.length();  int nIndex = 0;  while (nIndex < nCount)  {  if (isOp(str[nIndex]))  {  vecIn.push\_back({ false, 0, str[nIndex++] });  continue;  }  string num;  int nTemp = nIndex;  while (nTemp < nCount && (isdigit(str[nTemp]) || str[nTemp] == '.')) num += str[nTemp++];  if (nTemp != nIndex)  {  vecIn.push\_back({ true, str2num<double>(num), 0 });  nIndex += num.length();  }  else nIndex++;  }  }  //中缀转后缀  void infix2suffix(const arrayList<item>& infixIn, arrayList<item>& suffixOut)  {  linkedStack<char> symbolStack;  for (auto it = infixIn.begin(); it != infixIn.end(); ++it)  {  if (it->isDigit) suffixOut.push\_back(\*it);  else  {  if (it->op == ')')  {  while (!symbolStack.empty())  {  char op = symbolStack.top();  symbolStack.pop();  if (op == '(') break;  suffixOut.push\_back({ false, 0, op });  }  }  else  {  while (!symbolStack.empty() && symbolStack.top()!='(' && priority(it->op) <= priority(symbolStack.top()) )  {  suffixOut.push\_back({ false, 0, symbolStack.top() });  symbolStack.pop();  }  symbolStack.push(it->op);  }  }  }  while (!symbolStack.empty())  {  suffixOut.push\_back({ false, 0, symbolStack.top() });  symbolStack.pop();  }  }  //计算后缀表达式的值  double calculate(const arrayList<item>& vecIn)  {  linkedStack<double> numStack;  for (auto it = vecIn.begin(); it != vecIn.end(); ++it)  {  if (it->isDigit) numStack.push(it->num);  else  {  double num1 = numStack.top(); numStack.pop();  double num2 = numStack.top(); numStack.pop();  switch (it->op)  {  case '+':  numStack.push(num2 + num1);  break;  case '-':  numStack.push(num2 - num1);  break;  case '\*':  numStack.push(num2\*num1);  break;  case '/':  numStack.push(num2 / num1);  break;  default:  break;  }  }  }  return numStack.top();  }  /\*pch.h\*/  #ifndef PCH\_H  #define PCH\_H  // TODO: 添加要在此处预编译的标头  #include <linearList/arrayList.h>  #include <stack/linkedStack.h>  #include "test1.h"  #endif //PCH\_H  /\*main.cpp\*/  #include "pch.h"  #include <iostream>  using namespace std;  int e[55][55];  bool vis[55][55];  void test1()  {  cout << "请输入合法的数学表达式: " << endl;  string str;  getline(cin, str);  arrayList<item> vecIn, vecOut;  try  {  translate(str, vecIn);  infix2suffix(vecIn, vecOut);  cout << "计算结果为: " << endl;  cout << calculate(vecOut) << endl;  }  catch (...)  {  cout << "非法表达式" << endl;  }  }  struct Runtime  {  int x, y;  int i;  };  void print(linkedStack<Runtime>& s)  {//输出路径  if (s.size() == 1)  {  Runtime cur = s.top(); s.pop();  cout << "(" << cur.x << ", " << cur.y << ')';  return;  }  Runtime cur = s.top(); s.pop();  print(s);  cout << " -> (" << cur.x << ", " << cur.y << ')';  }  void test2()  {  int n, m, sx, sy, ex, ey;  int dx[4] = { 0, -1, 0, 1 }, dy[4] = { 1, 0, -1, 0 };  cout << "请依次输入迷宫的行列、入口坐标、出口坐标、迷宫: " << endl;  cin >> n >> m >> sx >> sy >> ex >> ey;  for (int i = 1; i <= n; ++i)  for (int j = 1; j <= m; ++j) cin >> e[i][j];  //模拟系统栈进行dfs  linkedStack<Runtime> s;  int i;  vis[sx][sy] = true;  s.push({ sx, sy, 0 });  while (!s.empty())  {  Runtime cur = s.top(); s.pop();  for (i = cur.i; i < 4; ++i)  {  if (i == 0 && cur.x == ex && cur.y == ey)  {  cout << "路径为: " << endl;  print(s);  cout << " -> (" << cur.x << " ," << cur.y << ')' << endl;  return;  }  int tx = cur.x + dx[i];  int ty = cur.y + dy[i];  if (tx > 0 && tx <= n && ty > 0 && ty <= m && !vis[tx][ty] && e[tx][ty] == 1)  {  vis[tx][ty] = true;  cur.i = i + 1;  s.push(cur);  s.push({ tx, ty, 0 });  break;  }  }  }  cout << "无解" << endl;  }  int main()  {  test1();  test2();  return 0;  } | | | |