**实 验 报 告（三）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | LL（1）语法分析 | | | 专 业 | 计算机科学与技术 |
| 课程名称 | 编译原理 | 指导老师 | 赵智超 | 班 级 | 三 |
| 姓 名 | 孔伟恒 | 学 号 | 18100140316 | 评 分 |  |
| 实验地点 | 1C06217 | 实验日期 | 2019/10/23 |
| 一、实验目的  1.理解预测分析表方法的实现原理。  2. 使了解语法分析的功能，掌握语法分析程序设计的原理和构造方法，训练掌握开发应用程序的基本方法。 | | | | | |
| 二、实验内容（含实验原理介绍）  编写一通用的预测法分析程序，要求有一定的错误处理能力，出错后能够使程序继续运行下去，直到分析过程结束。可通过不同的文法（通过数据表现）进行测试。  给定算术表达式文法，编写程序。  测试数据：  算术表达式文法  E→TE’  E’ → +TE’|- TE’|ε  T→FT’  T’ →\*FT’ |/ FT’ |%FT’|ε  F→(E) |id|num | | | | | |
| 三、实验过程及步骤（包含使用软件或实验设备等情况）  #include <iostream>  #include <cstring>  #include <cstdlib>  #include <map>  #include <set>  #include <stack>  #include <vector>  #include <algorithm>  using namespace std;  map <char, int> getnum;  vector <string> proce;  char text[100];  int table[105][105];  int num = 0, numvt = 0;  string first[100];  string follow[200];  void readin () {  memset (table, -1, sizeof (table));  getnum['#'] = 0;  text[0] = '#';  cout << "请输入所有的终结符: " << '\n';  char x;  do {  cin >> x;  getnum[x] = ++num;  text[num] = x;  } while (cin.peek () != '\n');  numvt = ++num;  getnum['@'] = numvt;  text[num] = ('@');  printf ("请输入所有的非终结符: \n");  do {  cin >> x;  getnum[x] = ++num;  text[num] = x;  } while (cin.peek () != '\n');  printf ("输入产生式集合(空字用'@'表示)，以'end'结束：\n");  string pro;  while (cin >> pro && pro != "end") {  string ss;  ss += pro[0];  for (int i = 3; i < pro.size(); i++) {  if (pro[i] == '|') {  proce.push\_back (ss);  ss.clear();  ss += pro[0];  } else {  ss += pro[i];  }  }  proce.push\_back (ss);  }  }  void jiaoji (string &a, string b) {  set <char> se;  for (int i = 0; i < a.size (); i++) {  se.insert (a[i]);  }  for (int i = 0; i < b.size (); i++) {  se.insert(b[i]);  }  string ans;  set <char> :: iterator it;  for (it = se.begin (); it != se.end (); it++) {  ans += \*it;  }  a = ans;  }  string get\_f (int vn, int & has\_0) {  if (vn == numvt) has\_0 = 1;  if (vn < numvt) return first[vn];  string ans;  for (int i = 0; i < proce.size (); i++) {  if (getnum[proce[i][0]] == vn) {  ans += get\_f (getnum[proce[i][1]], has\_0);  }  }  return ans;  }  void getfirst () {  for (int i = 1; i <= numvt; i++) {  first[i] += ('0' + i);  }  for (int j = 0; j < proce.size (); j++) {  int k = 0, has\_0 = 0;  do {  has\_0 = 0;  k++;  if (k == proce[j].size()) {  first[getnum[proce[j][0]]] += ('0' + numvt); break;  }  jiaoji(first[getnum[proce[j][0]]], get\_f (getnum[proce[j][k]], has\_0));  } while (has\_0);  }  }  void print\_first () {  printf ("first集: \n");  for (int i = 1; i <= num; i++) {  cout << "first [" << text[i] << "]: ";  for (int j = 0; j < first[i].size(); j++) {  cout << text[first[i][j] - '0'] << ' ';  }  cout << '\n';  }  cout << '\n';  }  void getfollow() {  jiaoji(follow[getnum[proce[0][0]]], "0");  for (int j = 0; j < proce.size(); j++) {  for (int i = 1; i < proce[j].size(); i++) {  if (getnum[proce[j][i]] <= numvt)continue;  int k = i; int has\_0;  do {  has\_0 = 0;  k++;  if (k == proce[j].size()) {  jiaoji(follow[getnum[proce[j][i]]], follow[getnum[proce[j][0]]]);  break;  }  jiaoji(follow[getnum[proce[j][i]]], get\_f(getnum[proce[j][k]], has\_0));  } while (has\_0);  }  }  }  void gettable() {  for (int i = 0; i < proce.size(); i++) {  if (proce[i][1] == '@') {  string flw = follow[getnum[proce[i][0]]];  for (int k = 0; k < flw.size(); k++) {  table[getnum[proce[i][0]]][flw[k] - '0'] = i;  }  }  string temps = first[getnum[proce[i][1]]];  for (int j = 0; j < temps.size(); j++) {  if (temps[j] != ('0' + numvt)) {  table[getnum[proce[i][0]]][temps[j] - '0'] = i;  } else {  string flw = follow[getnum[proce[i][1]]];  for (int k = 0; k < flw.size (); k++) {  table[getnum[proce[i][0]]][flw[k] - '0'] = i;  }  }  }  }  }  string get\_proce(int i) {  if (i < 0)return " ";  string ans;  ans += proce[i][0];  ans += "->";  for (int j = 1; j < proce[i].size(); j++)  ans += proce[i][j];  return ans;  }  void print\_table() {  cout << "预测分析表：" << endl;  for (int i = 0; i < numvt; i++)  cout << '\t' << text[i];  cout << '\n';  for (int i = numvt + 1; i <= num; i++) {  cout << text[i];  for (int j = 0; j<numvt; j++) {  cout << '\t' << get\_proce(table[i][j]);  }  cout << '\n';  }  cout << '\n';  }  void print\_follow() {  cout << "follow集：" << endl;  for (int i = numvt + 1; i <= num; i++) {  cout << "follow [" << text[i] << "]: ";  for (int j = 0; j < follow[i].size(); j++)  cout << text[follow[i][j] - '0'] << " ";  cout << '\n';  }  cout << '\n';  }  string word;  bool analyze() {  stack <char> sta;  sta.push('#'); sta.push(proce[0][0]);  int i = 0;  while (!sta.empty()) {  int cur = sta.top();  sta.pop();  if (cur == word[i]) {  i++;  } else if (cur == '#') {  return 1;  } else if (table[getnum[cur]][getnum[word[i]]] != -1) {  int k = table[getnum[cur]][getnum[word[i]]];  cout << proce[k][0] << "->";  for (int j = 1; j<proce[k].size(); j++) {  cout << proce[k][j];  }  cout << '\n';  for (int j = proce[k].size() - 1; j>0; j--) {  if (proce[k][j] != '@') {  sta.push(proce[k][j]);  }  }  } else {  return 0;  }  }  return 1;  }  void fun () {  readin();  getfirst();  getfollow();  getfollow();  gettable();  print\_first();  print\_follow();  print\_table();  cout << "请输入字：" << endl;  cin >> word;  cout << (analyze() ? "succeed! 该字符有效，所用产生式如上。" : "ERROR !") << endl;  }  int main() {  fun ();  return 0;  } | | | | | |
| 1. 实验结果（含算法说明、程序、数据记录及分析等，可附页）   对符号串i+i\*i的分析过程如下所示 :  深度截图_选择区域_20191028140951  深度截图_选择区域_20191028141017 | | | | | |
| 五、实验思考题  １、预测分析法和递归子程序法的区别和联系？  ２、在实验过程中遇到了什么问题，是怎么解决的？ | | | | | |
| 六、实验总结（含实验心得体会，收获与不足等） | | | | | |

注：双面打印