**编译原理课程实验报告**

**实验3：语义分析**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 夏敏 | | 院系 | | 软件学院 | | | 学号 | | | 1143710222 | |
| 任课教师 | | 陈鄞 | | | | 指导教师 |  | | | | | |
| 实验地点 | | 软件学院三楼机房 | | | | 实验时间 | 周日下午一二节 | | | | | |
| 实验课表现 | | 出勤、表现得分 | |  | | 实验报告  得分 |  | | | 实验总分 | |  |
| 操作结果得分 | |  | |
| **一、需求分析** | | | | | | | | | | 得分 | |  |
| 要求：阐述语义分析系统所要完成的功能。  使用语义分析研究上下文无关文法所产生的语言进行翻译。通过现有的SDT文法翻译文法方案，把token和文法联系起来，利用语法分析生成的分析树，结合SDT中的综合属性与继承属性，把语法分析树进行转换，生成中间代码，以便于后续的操作生成三地址指令以及机器码等。当出现语法错误时，能够产生错误信息并定位错误，给出错误信息。 | | | | | | | | | | | | |
| **二、文法设计** | | | | | | | | | | 得分 | |  |
| 要求：给出如下语言成分所对应的语义动作  ➢ 声明语句（包括变量声明、数组声明、记录声明和过程声明）  **<declaration\_list> ::=**  **<declaration\_list> ::= <declaration> P22 <declaration\_list>**  **<declaration\_specifiers> ::= <type\_specifier> ID P31**  **<declarator>    ::= ( <parameter\_type\_list> )**  **<declarator>    ::=**  **def P22():**  **global current\_symbol\_table\_pos**  **global current\_symbol\_index**  **s = symbol\_for\_str(LAST\_STACK\_TOP\_SYMBOL).father.children[0]**  **SYMBOL\_TABLE.append(Entry(s.attr['type'], s.attr['length'], s.attr['name']))**  **current\_symbol\_index += 1**  **current\_symbol\_table\_pos += s.attr['length']**  **def P31():**  **f = symbol\_for\_str(LAST\_STACK\_TOP\_SYMBOL).father**  **f.attr['name'] = f.children[1].lexical\_value**    ➢ 表达式及赋值语句（包括数组元素的引用和赋值）  <expression>    ::= ID = <unary\_expression> P62 <factor\_expression> P61  def P61():  f = symbol\_for\_str(LAST\_STACK\_TOP\_SYMBOL).father  if len(f.children) < 3:  f = f.father.father.father.father  l = f.children[0]  r = f.children[2]  fac = f.children[4]  lv = search\_for\_symbol(l.lexical\_value)  if lv is None:  syntax\_error('undefined ' + l.lexical\_value)  return  if lv.type != r.attr['type']:  syntax\_error('type mismatch')  return  result = None  if 'op' in fac.attr:  if fac.attr['op'] == '+':  result = f.attr['value'] + fac.attr['factor']  if fac.attr['op'] == '\*':  result = f.attr['value'] \* fac.attr['factor']  else:  result = r.attr['value']  fac.attr = {}  code\_output(lv.name + ' := ' + str(result))    def P62():  f = symbol\_for\_str(LAST\_STACK\_TOP\_SYMBOL).father.father.father.father  f.attr['type'] = f.children[2].attr['type']  f.attr['value'] = f.children[2].attr['value']  ➢ 分支语句：if\_then\_else  selection\_stmt>::= if ( <primary\_expression> ) P81 <stmt> P82    def P81():  global CURRENT\_CONDITION\_NODE  global last\_token  f = symbol\_for\_str(LAST\_STACK\_TOP\_SYMBOL).father  CURRENT\_CONDITION\_NODE = f  e = f.children[2]  code\_output('IF ' + last\_token + ' GOTO ' + str(CODE\_SIZE + 2))  code\_output(None)  f.attr['back'] = CODE\_SIZE - 1  def P82():  prev = CURRENT\_CONDITION\_NODE.attr['back']  CODE\_RESULT[prev] = 'GOTO ' + str(CODE\_SIZE)  ➢ 循环语句：do\_while  <iter\_stmt>     ::= while ( <primary\_expression> ) P91 <stmt> P92  def P91():  global CURRENT\_CONDITION\_NODE  global last\_token  f = symbol\_for\_str(LAST\_STACK\_TOP\_SYMBOL).father  CURRENT\_CONDITION\_NODE = f  e = f.children[2]  code\_output('IF ' + last\_token + ' GOTO ' + str(CODE\_SIZE + 2))  code\_output(None)  f.attr['back'] = CODE\_SIZE - 1  def P92():  prev = CURRENT\_CONDITION\_NODE.attr['back']  CODE\_RESULT[prev] = 'GOTO ' + str(CODE\_SIZE + 1)  code\_output('GOTO ' + str(prev - 1))  ➢ 过程调用语句  <external\_declaration> ::= <func\_declaration>  <func\_declaration>::= <declaration\_specifiers> <declarator> <declaration\_list> <compound\_stmt> | | | | | | | | | | | | |
| **三、系统设计** | | | | | | | | | | 得分 | |  |
| 要求：分为系统概要设计和系统详细设计。  （1）系统概要设计：给出必要的系统宏观层面设计图，如系统框架图、数据流图、功能模块结构图等以及相应的文字说明。    （2）系统详细设计：对如下工作进行展开描述  ✓ 核心数据结构的设计  class Production(object):     def \_\_init\_\_(self, left, right, select=None):         self.left = left         self.right = right         self.select = set()       def \_\_str\_\_(self):         return self.left + ' -> ' + str(self.right) + ' Select: ' + str(self.select)  class Symbol(object):     def \_\_init\_\_(self, symbol, first\_set=None, follow\_set=None, sym\_type='N'):         self.symbol = symbol         self.first\_set = first\_set         self.follow\_set = follow\_set         self.sym\_type = sym\_type         self.is\_nullable = False         self.attr = {}         self.father = None         self.children = []         self.lexical\_value = None       def \_\_str\_\_(self):         return self.symbol + ' Derive\_empty:' + str(self.is\_nullable) + ' First:' + str(self.first\_set) + ' Follow:' + str(self.follow\_set)       def is\_terminal(self):         return self.sym\_type == 'T'  class Entry(object):     def \_\_init\_\_(self, type, length, name):         self.type = type         self.length = length         self.name = name     def \_\_str\_\_(self):         return self.name + ' ' + self.type + ' ' + str(self.length)  ✓ 主要功能函数说明  grammar\_scanner     从外存载入文法，并进行处理，方便之后使用  getFirst     利用现有算法提取FIRST集  getFollow     利用现有算法提取FOLLOW集  syntax\_parse     模拟堆栈进行语法分析  get\_parsing\_table     构建预测分析表  do\_parsing     进行分析    ✓ 程序核心部分的程序流程图 | | | | | | | | | | | | |
| **四、系统实现及结果分析** | | | | | | | | | 得分 | | |  |
| 要求：对如下内容展开描述。  （1） 系统实现过程中遇到的问题；  原使用的文法过于繁杂，产生SDT的过程中，需要做的工作量太大，重写了一份文法，功能不是很全面  （2） 针对一测试程序输出其语义分析结果；  int main() {     int a;     float b;     int c;     float e;     c=10;     q=5;     if(c) {         a = 1 + 10;         b = 10.9 + 8.9;     }     b = 1.11 \* 8.9;     while(a) {         b = 10.44;         e = 990.45;         c = 90;     }     c = 80;  }    int func1 () {  }    SYMBOL TABLE  ------------  a int 4  b float 4  c int 4  e float 4    CODE  ------------  0: c := 10  1: IF c GOTO 3  2: GOTO 5  3: a := 11  4: b := 19.8  5: b := 9.879  6: IF a GOTO 8  7: GOTO 12  8: b := 10.44  9: e := 990.45  10: c := 90  11: GOTO 6  12: c := 80    （3） 输出针对此测试程序经过语义分析后的符号表；  SYMBOL TABLE  ------------  a int 4  b float 4  c int 4  e float 4    （4） 输出针对此测试程序对应的语义错误报告；  7:8 Syntax error: undefined q  （5） 对实验结果进行分析。  本次实验的功能，能够进行简单的语义分析，能够将C语言代码生成中间代码。由于写的文法功能有限，以及时间比较紧，功能上有不足，只能实现简单的程序进行语义分析和报错  注：其中的测试样例需先用已编写的词法分析程序进行处理。 | | | | | | | | | | | | |
| 指导教师评语：  日期： | | | | | | | | | | | | |