**哈尔滨工业大学**

**编译原理 2024春**

**实验二**

|  |  |
| --- | --- |
| 学院： | 计算学部 |
| 姓名： | 徐柯炎 |
| 学号： | 2021110683 |
| 指导教师： | 单丽莉 |

# 实验目的

1. 巩固对语义分析的基本功能和原理的认识。
2. 能够基于语法指导翻译的知识进行语义分析。
3. 理解并处理语义分析中的异常和错误。

# 实验环境

* GNU Linux Release: Ubuntu 12.04, kernel version 3.2.0–29。
* GCC version 4.6.3。
* GNU Flex version 2.5.35。
* GNU Bison version 2.5。

# 实验内容

## 实现功能

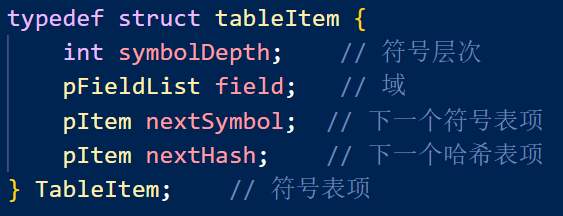
本程序实现了实验指导书上所有错误类型的检查，包括：

1. 变量（包括数组、指针、结构体）或过程未经声明就使用；
2. 变量（包括数组、指针、结构体）或过程名重复声明；
3. 运算分量类型不匹配；
4. 操作符与操作数之间的类型不匹配。

## 数据结构

1. 语法树节点的数据结构：

实验二的语法树节点的数据结构沿用了实验一的结构体，包括以下几个变量：

1. 行号int lineNo；
2. 节点类型NodeType type；
3. 节点名称char\* name；
4. 节点值char\* val；
5. 下一个孩子节点struct node\* child；
6. 下一个兄弟节点struct node\* next；
7. 语义分析时符号表的数据结构：

实验二的语义分析符号表采用一张哈希表来实现，表中的每一项都是如右图所示的结构体，下面解释一下每一项的作用：

1. int symbolDepth：符号层次，用来指明符号的作用域；
2. pFieldList field：域，用来存放符号的名字以及类型；
3. pItem nextSymbol：下一个符号表项，层次相同时下一个符号项；
4. pItem nextHash：下一个哈希表项，当哈希值一样的时候下一个符号项。

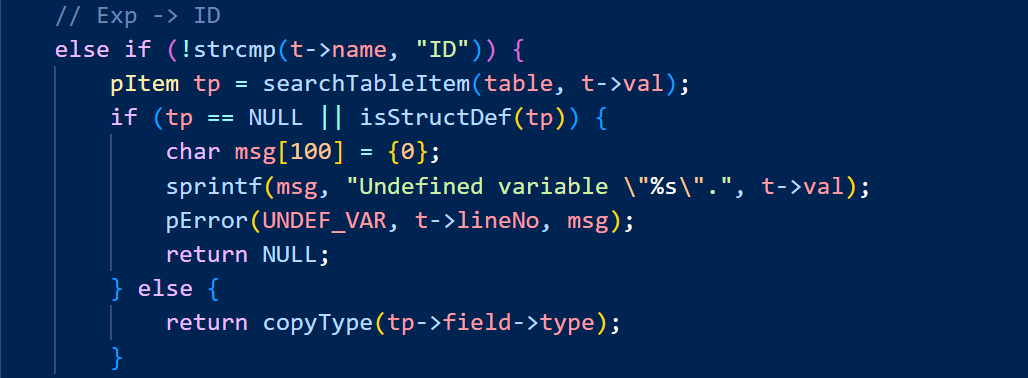
而在本实验中，每个域存放了符号的名字以及类型，而类型分为4种，也就是基础类型（int，float），结构体，函数，数组，分别用一个结构体表示并存放其类型和数值。

## 语义分析

本实验语义分析的大体过程如下：

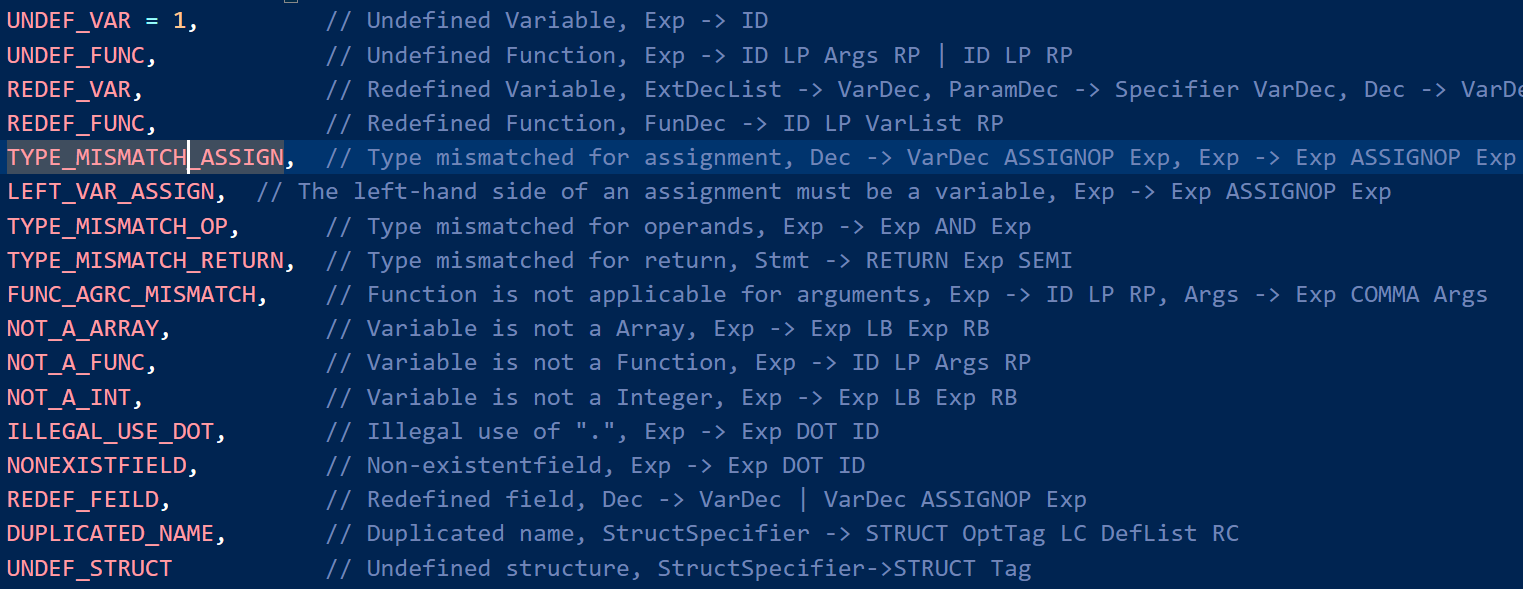
1. 首先进行语法分析和词法分析，构建语法分析树；
2. 有了语法分析树后，从根节点开始向下逐级分析，在必要的地方对错误类型进行检查分析。

以错误类型1：变量在使用时未经定义为例：



当语义分析到exp节点时，如果子节点为ID，也就是应当定义过的变量，这时去符号表中查找是否存在该变量，如果在符号表中没有查找到该符号，说明变量没有经过定义就使用了，于是报错，在command上打印”undefined variable...”，分析结束。

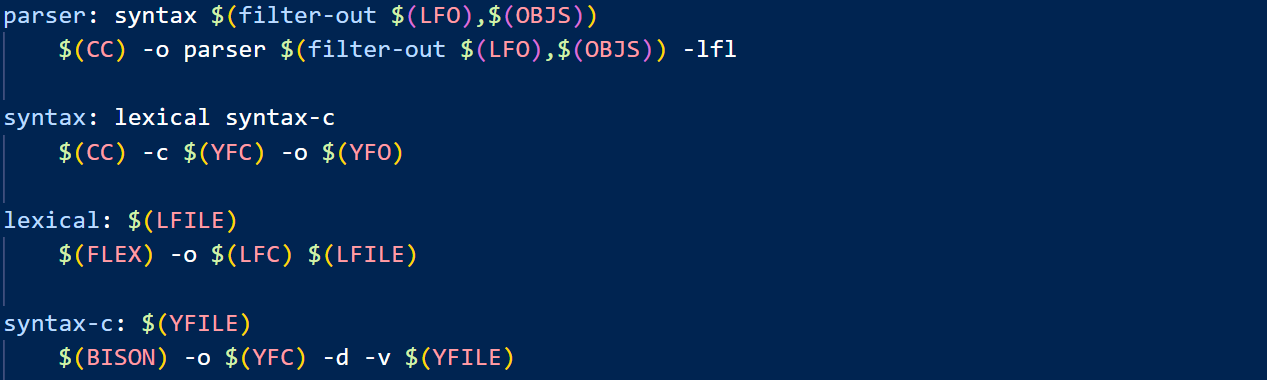
最后附上所有错误类型需要检查的文法：



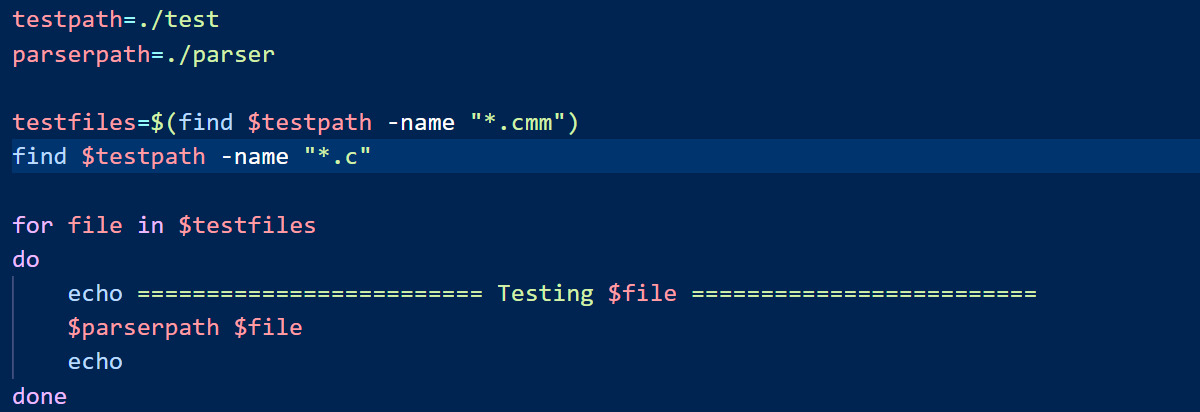
## 编译过程和测试过程

本程序采用makefile编译，并采用bash脚本进行测试样例的测试，部分截图如下：

Makefile：



test.sh



# 实验结果

部分测试结果如下：



# 实验总结

1. 学习了如何在词法分析和语法分析的基础上更进一步进行语义分析；
2. 增强了编程能力、分析问题的能力和动手能力；
3. 对语义分析有了更深层次的认识。