**编译系统实验报告**

**实验2 – 语义分析**

|  |
| --- |
| 程序实现的功能及如何实现： |
| 程序主要实现的功能如下：   1. 可以对C--源代码进行语义分析和类型检查，并打印分析结果 2. 在程序遇到源代码中的语义错误时，输出报错提示：   Error type【错误类型】at Line【行号】：【说明文字】   1. 如果程序中不存在语义错误，则程序不输出任何内容。 2. 注意：当输入文件中存在多个错误时，程序不一定能报告全部的错误，但是对于“最为本质”的错误，程序可以将其发现并输出对应的说明文字。 3. 注意：本次实验只完成了老师所要求的基本功能，在实验指导书中说明不做的功能（用荧光笔涂色标记）没有完成。   关于以上功能，具体的实现方法及程序文件简要介绍如下：   1. 关于每个程序文件的功能，简要介绍如下：   lexical.l：该文件中保存着符号对应的正则表达式，以及根据符号表中的每一个符号构建语法树的步骤。  node.h：该文件中以结构体表示语法树的每一个节点。  main.c：整个语义分析程序的执行文件。  semantic.h：保存着函数声明以及重要的结构体定义，具体将在后文讲到。  semantic.c：语义分析的核心功能代码，执行遍历语法树，检索错误的功能。  syntax.y：保存着语法树的构建过程对应的代码。   1. 关键的结构体定义 2. 语法树节点的结构体Node   节点的信息包括：当前节点的子节点，右兄弟节点，行号，名字，节点为语法单元还是词法单元。   1. 保存当前符号信息的Type\_   其中包含当前标识符的类型以及具体的数据。   1. 保存函数参数的结构体FUNCTION   保存着函数名，参数列表，返回值，声明和定义的数量，错误发生行数等关键信息。   1. 用正则表达式完成不同类型数据的表示。具体如图所示： 2. 整体程序执行流程   程序主要分为三个部分：首先通过syntax.y文件，按步骤构建语法树，再通过semantic.h与semantic.c遍历语法树构建符号表、遍历语法树检查语句类型，遇到错误的类型则输出对应的错误。 |
| 如何编译程序： |
| 最终的代码文件如图所示：  其中：  lexical.l，node.h，main.c，semantic.h，semantic.c，syntax.y是编写的程序文件。  testfile文件夹中保存着测试文件，对应着指导书中的示例2.1~2.17。  parser 是gcc编译后得到的可执行文件。  其余文件是中间过程生成的文件。  执行以下几条命令以编译程序：  flex -o ./lex.yy.c ./lexical.l  bison -o ./syntax.tab.c -d -v ./syntax.y  gcc -c ./syntax.tab.c -o ./syntax.tab.o  gcc -o parser ./main.o ./semantic.o ./syntax.tab.o -lfl -ly  在本实验中，我将以上命令写入脚本，在Linux下使用时，切入Lab2文件夹，直接执行 run.sh 便可完成程序编译。  完成之后，输入 ./parser ./test/testfile.cmm 便可以用程序对进行词法分析与语法分析。 |
| 程序执行结果： |
| 所要求检测的语义错误1,3,5,6,7,10,12如图所示 |