山东大学 软件 学院

**编译原理** 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202200300108 | 姓名：王博凡 | | 班级：软工2班 |
| 实验编号：实验一 | | | |
| 实验题目：词法分析程序设计与实现 | | | |
| 实验学时：6 | | 实验日期：2024.10.23 | |
| 实验目的：  **为PL/0语言建立一个词法分程序GETSYM（函数）** | | | |
| 硬件环境：  主机  机型：ASUS TUF Gaming A16 FA607PI  CPU: AMD Ryzen 9 7940HX with Radeon Graphics +六核  内存：三星 32GB DDR5 5200MHz（16GB + 16GB）  GPU: NVIDIA GeForce RTX 4070 Laptop GPU ( 8GB / 华硕)  虚拟机：  WSL2+Ubuntu22.04LTS | | | |
| 软件环境：  宿主机：Windows11  虚拟机：WSL2+Ubuntu22.04LTS | | | |
| 实验步骤与内容：  **1.实验题目分析**  实验要求写一个一个用于PL/0语言的词法分析器，该程序负责从输入源代码中识别单词并进行分类，最终生成一系列符号（使用Token表示）。每个单词的类别（使用SYM表示）以及标识符（使用ID表示）和数字常量（使用NUM表示）都需要被正确地识别和存储。  **2.实验模型**  词法分析器的核心任务是将输入的源代码进行逐字符处理，识别其中的关键字、标识符、数字和符号，最终将每个识别到的词法单元（Token）输出到文件中，以此构建程序的框架。程序的主框架包含三个主要模块：文件输入输出、词法分析以及结果输出。首先，通过 ifstream 和 ofstream 进行文件读写，读取输入文件中的每一行并输出到控制台，以便查看源程序内容。接着，核心的词法分析逻辑是通过 getSym 函数逐字符解析输入，识别每一个符号并将其映射为对应的 Token。最后，分析的结果会通过 processFile 函数输出到结果文件中，按照不同的类型对Token进行处理并输出。  流程图如下：  输出图片  **3.主要的数据结构**  在代码中，使用了几种数据结构：Token 结构体，用于存储表示Token的信息。哈希表 unordered\_map 和字符数组 tokenTypeStrings，用于各种关键字、符号的对应关系。  Token是词法分析器的核心数据结构，用于存储每个词法单元的类型（SYM）、标识符（ID）和数值（NUM）：   1. **struct** Token 2. { 3. TokenType sym; 4. string id; 5. string num; 7. };   其中，TokenType是一个枚举类型，用来标识不同的Token类型，例如 SYM\_IDENT、SYM\_NUMBER 代表标识符和数字，PLUS\_SYM、MINUS\_SYM 等符号对应运算符，CONST\_SYM、VAR\_SYM 代表关键字。  tokenTypeStrings[] 数组是一个辅助数组，其顺序和TokenType的枚举顺序一一对应，可以便捷地将这些枚举值映射为对应的字符串名称，以便在输出时可以直接将 TokenType 转换成可读的形式。  此外，哈希表用于完成keyword或symbol到SYM的映射： keywordMap 用来存储关键字到 TokenType 的映射，而 symbolMap 则用于将符号字符映射到相应的 TokenType。  **具体处理思路：**  getSym 函数从输入文件中逐字符读取，并根据字符类型将其识别为标识符、数字或符号。    这里方法接受ifs输入流，首先读取一个字符，进行空格和注释的处理：首先跳过空白字符，并识别注释，支持 // 和 /\*\*/ 两种注释格式。通过循环读取字符，遇到 // 时跳过整行内容（**其中\n会被识别为一个字符**），遇到 /\*...\*/ 时跳过多行注释内容。最后对于第一个字符为/但是并非注释的字符，将其放回输入流中供后续使用（**此后多个流程设计此操作，不再赘述**）。  **对标识符和关键字的处理：**    当读取到字母开头的字符时，代码通过token.id接受并将其拼接起来，后续再判断是否为关键字（通过在map中查找实现），并对token.sym进行相应的赋值。同时，若字符串超过了10个字符，则将其视为错误符号（ERROR\_SYM）。  **对数字的处理：**    当读取到数字开头的字符时，则进入数字的识别状态，将连续的数字字符拼接成一个完整的数值。若数字后面紧跟字母，判断为非法符号，标记为 ERROR\_SYM。  **符号的识别：**    首先对符号通过映射表直接查找，随后，如果是该语言中两个字符组成的符号（<=，>=，:=）的开头，进行再读一位并识别的处理，判断是否能组合成多字符符号。若不能，则将第二个字符放回输入流，重新解析。  最后，如果字符都不属于以上的情况，将其标志为ERROR\_SYM  **4.调试排错**  实验过程中主要遇到了两个问题，分别是符号的识别以及注释的处理。在符号识别方面，针对由两个字符组成的运算符（如 <=, >=, :=），最初的实现是每一个符号都会拼接后续所有的符号并识别，导致像);这样的符号组合会被识别为一个符号并判误。在注释处理方面，一开始没有将识别为非注释的字符放回流中，导致识别的字符出现了跳跃现象。  **5.运行结果**  使用了两个input，其输入及结果如下：  输入1：    可以看到：输入1中包含了两种注释以及换行，得到的结果成功跳过了这些字符并正确处理程序。控制台：    程序将源程序成功打印出来。  输入2：    输入2中包含了连续的空格，根据结果可以看出程序成功跳过了连续的空格并正确识别了源程序。  控制台： | | | |
| 结论分析与体会：  在本次实验中，主要实现了一个基础的词法分析器，能够正确识别PL/0语言中的标识符、关键字、数字和符号，并通过一定的逻辑跳过注释和空白字符。实验中遇到的主要问题是多字符符号的拼接识别，以及注释的处理。一开始对于两个字符组成的运算符处理不当，导致符号识别出错，后来通过逐字符判断、拼接后再查找映射表的方式解决了这个问题。在注释处理上，最初忽略了将非注释的字符放回输入流，导致某些字符未能正确解析，修复后程序可以准确跳过注释部分。  实验的结果成功达到了设计目标，程序能够正确识别源代码中的词法单元，并能够处理较复杂的源代码结构，例如多行注释、连续空格等情况。控制台输出也验证了程序能将输入源代码完整打印。  在实验中得到的启发主要体现在对语言词法分析的深入理解，特别是如何处理特殊的符号组合、注释跳过等常见的词法分析难点。同时，整个实验过程还帮助我熟悉了文件的读写操作、输入流的回退处理等关键技术。  在未来的改进中，可以进一步优化词法分析器的错误处理机制，例如更详细地报告词法错误的位置，并扩展支持更多的PL/0语言特性。此外，当前的符号识别较为简单，还可以通过引入更加复杂的状态机来提高分析器的健壮性和扩展性。 | | | |