实验 1 词法分析程序分析报告

西南民族大学 计算机科学与技术 1202 欧长坤 201231102123

一、流程分析

1.1 main()函数流程分析

本流程对实验原本提供的流程进行了一些改动，使得程序可以从文本文件中读入需要进行词法分析的程序，并将词法分析的部分独立出来，置于 scaner.h 和 scanner.cpp 中，并统一封装在了 start_lexical_analysis () 中。

流程图如图 1 所示。

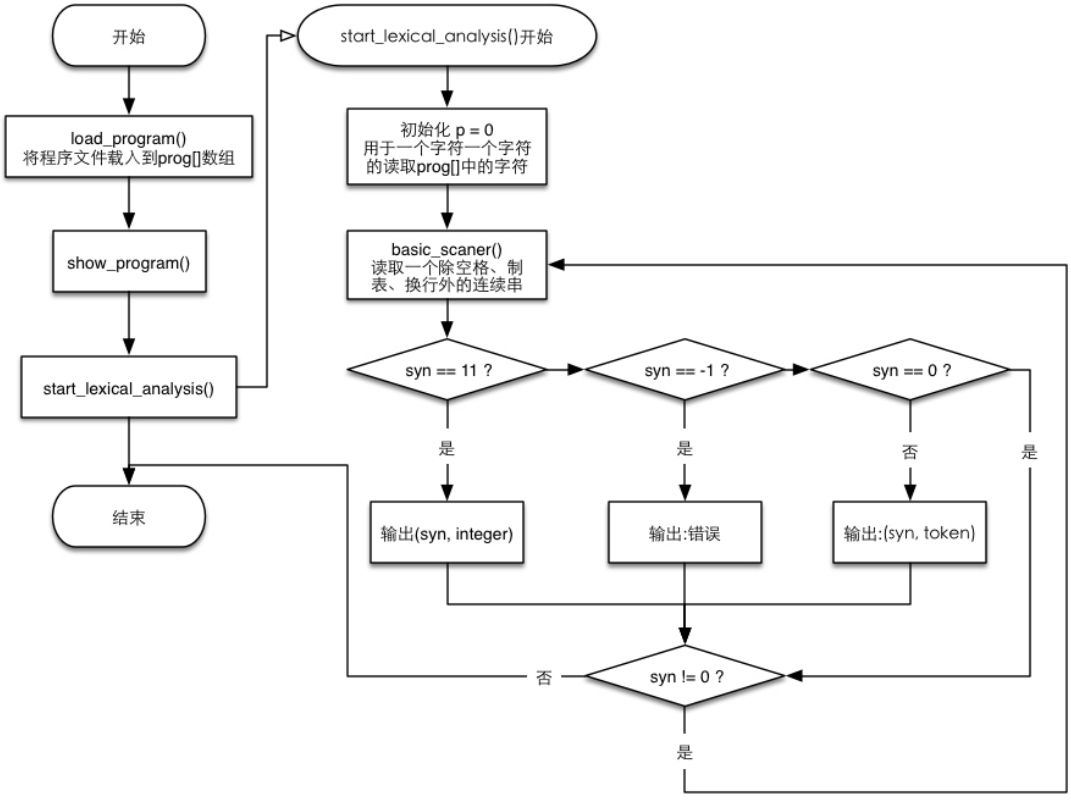

图 1 词法分析程序的 main()函数流程图

main()函数的执行流程如下：

1. 程序开始后，会使用 utils.h 中声明的 load_program() 函数来接受需要进行词法分析的程序的文本文件 data.txt，将其读入 scaner.cpp 中定义的全

局变量 prog[] 字符数组中。

2. 接下来，会使用 utils.h 中声明的 show_program() 函数来输出读入的程序，方便判断是否完成将程序读入。

3. 调用 scaner.h 中声明的 start_lexical_analysis() 函数，start_lexical_analysis()函数会输出对程序的词法分析结果至此，结束程序运行。

而对于 start_lexical_analysis()函数而言，不涉及具体的词法分析过程，负责输出 basic_scaner()词法分析函数的分析结果。执行流程如下：

1. 进入 start_lexical_analysis()后，函数会初始化索引变量 p，p 用于记录词法分析所分析 prog[]数组所进行的位置索引。

2. 接下来开始执行 basic_scaner()，进行词法分析

3. 词法分析完成后，位于 scaner.cpp 中的全局变量 syn 和 token[]均被更新，这时会判断 syn 的值，如果是 11 则表示 syn 为整型数字，则会输出(syn, integer);如果是-1 则表示 syn 所分析到得词语是语言中未定义的情况;如果 syn 的值为 0，则表示遇到了文件的结束符号 EOF，程序分析完毕；对于其他情况，则会直接输出(syn, token)。

下面是 syn 所表示含义的表格：

syn	token	syn	token	syn	token
0	EOF	10	标识符	20	<=
1	function	11	整形数字	21	=
2	If	12		22	==
3	Then	13	+	23	!=
4	While	14	-	24	
5	do	15	*	25	,
6	End	16	/	26	;
7		17	>	27	(
8		18	>=	28	)
9		19	<	29	{
-1	未定义			30	}

1.2 basic_scaner()函数流程分析

本流程与实验原本提供的逻辑基本一致，它是 main()函数流程图中得 basic_scaner()的内部执行情况的详细描述。流程图如图 2 所示。

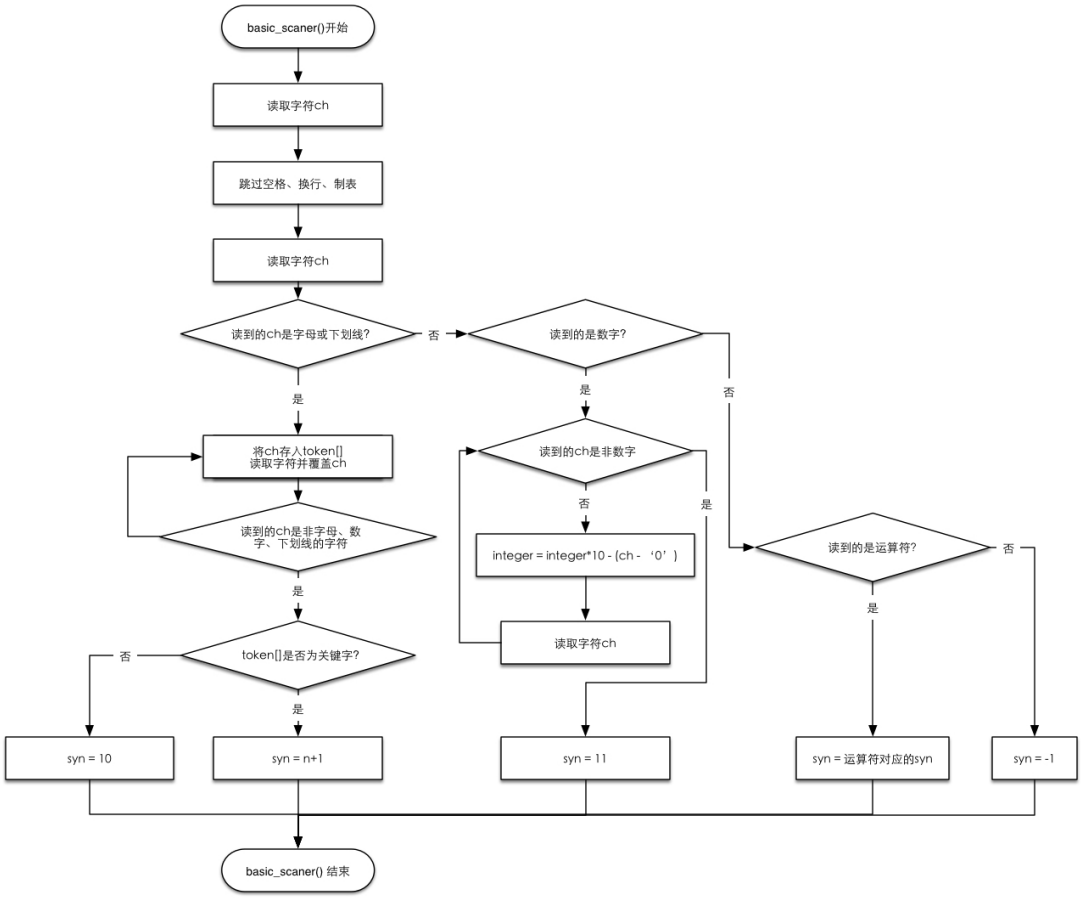

图 2 词法分析程序的 basic_scaner()函数流程图

basic_scaner()函数的执行流程如下：

1. 函数开始执行后，会读取一个字符，如果它是空格、换行、制表，则会跳过；
2. 如果读到的 ch 是字母或者是下划线，则说明接下来的单词可能是标识符，也可能是关键字；如果发现是关键字，则 syn=n+1，否则为标识符，则 syn=10；
3. 如果读到的 ch 是数字，则继续读后面的数字，直到读到非数字为止，并将读到的数字字符转成整数，使用 int 变量来保存，且 syn=11；
4. 如果读到的 ch 是运算符，则 syn 的值为对应运算符的值；
5. 如果读到的是其他未定义的字符，则 syn 的值为-1；
6. 处理回退情况，结束 basic_scaner()的运行。

二、词法分析器的状态转换图

根据实验所提供的源程序没有涉及实现浮点数的识别，所以词法分析器的状态转换图未考虑浮点数情况，如图 3 所示。

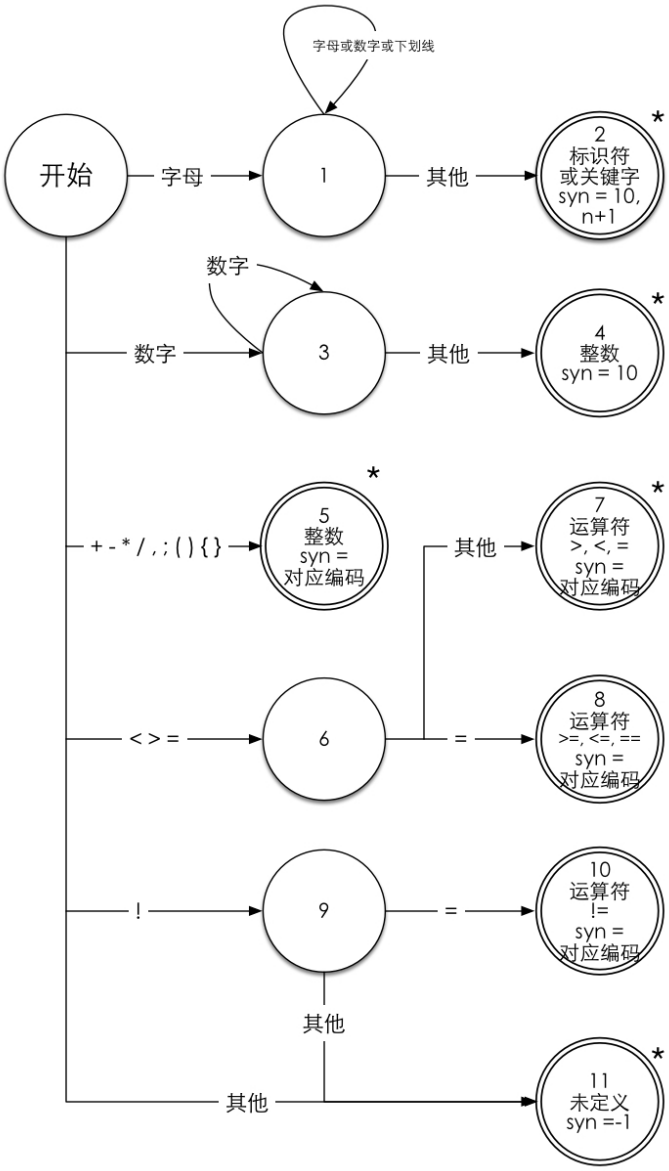

图 3 词法分析程序的状态转换图

三、调试过程简述

源程序一共出现了下述几个问题：

1. 全局变量过度滥用，没有详细区分函数模块，代码混乱，实验中一共将源代码分离成了 main.cpp, utils.h, utils.cpp, scaner.h, scaner.cpp 五个文件；
2. 源程序 main()函数中，对字符的读取使用的是 prog[++p]，这将造成 prog[]数组会浪费掉 prog[0]这一个单元，在 scaner 函数中这种情况依然存在；
3. scanner()最初有描述 while(ch != '#') 出现明显的错误，这会导致直接跳过整个程序串导致让后面的代码只能分析 prog[]数组中的最后一个'#'字符；
4. 其他位置的代码逻辑基本正确，但由于上述问题的存在导致了解决上面的问题后，还需要对其余的每个部分做细节修改。

四、功能扩展描述

本次实验一共扩展了五个功能：

1. 可以识别'{', '}'
2. 修改了关键字 'endfunc' 替换为 'end'
3. 当出现 '\t' 制表符的时候同样可以跳过
4. 标识符可以使用 '_' 甚至是开头
5. 增加了运算符 '->'

五、程序源代码

程序源程序使用 Makefile 进行编译运行。
代码已经开源至：http://github.com/euryugasaki/compiler-of-training
本实验报告涉及的源码位于：实验 1 – 词法分析。