词法分析与语法分析

梁宇钦 151180076

April 20, 2018

1. 功能实现

1.1 词法分析

实现了实验要求的词法识别和分析的所有要求. 能够正确识别词法正确的输入串, 能够判断并报告非法的输入串.

1.2 语法分析

基本实现了实验的所有要求. 能够正确判断上下文无关语法是否合法, 并能够准确报错. 能够按实验要求生成语法树, 并打印出来.

2. 编译和测试

这部分内容在文件 “README.MD” 中有详细说明, 这里仅做简要说明.

2.1 目录树

```
aSimpleCC
├── README.md
└── syntax.d
    ├── lexical.l
    ├── main.c
    ├── Makefile
    ├── README.MD
    ├── runtest.sh
    ├── syntaxtree.c
    ├── syntaxtree.h
    ├── syntax.y
    ├── test.d
    └── test_result
```

本次实验的所有内容都在 syntax.d 目录下面. 主要文件包括 lexical.l, syntax.y, syntaxtree.h, syntaxtree.c 和 main.c 等文件. 其中 lexical.l 是词法文件, 用 flex 编译. Syntax.y 是文法文件, 用 bison 编译. Syntaxtree.h 和 syntaxtree.c 是语法树的数据结构文件.

2.2 编译

直接在终端输入命令:

```
make
```

就可以了编译了.

2.3 测试

2.3.1 测试已经提供的样例

要测试实验指导提供的样例, 也是非常简单的. 直接运行脚本 “runtest.sh” 即可. 建议采用以下的命令运行脚本:

```
sh runtest.sh
```

其结果会存到 test_result 目录下面. 同时也会打印屏幕上.

2.3.1 测试其他未提供的样例

命令格式如下:

```
./parser filename.c
```

其结果会打印屏幕上面.

3. 语法树结构

语法树是一颗多叉树, 其数据结构如下图所示:

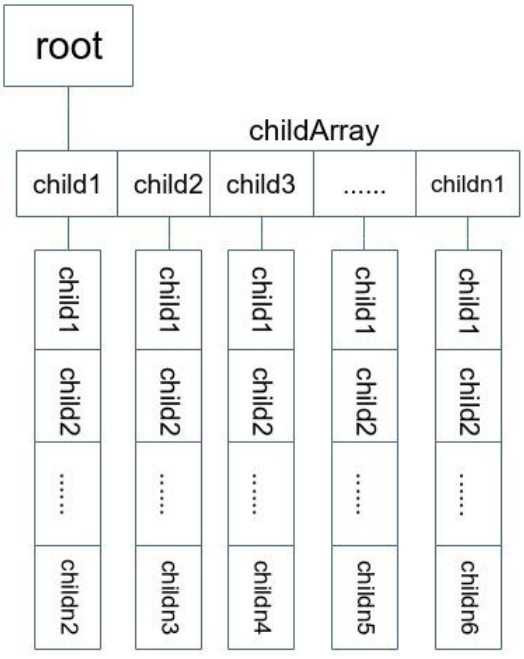

图 1. 语法树数据结构示意图

具体实现上, 语法树作为一个结构体实现. 且在多叉树这一个层面上, 只管理跟语法树相关的操作: 暂时只实现了创建树, 销毁树, 加入孩子节点等操作. 其他的树的操作因为还用不到所以还没有实现.

孩子节点的管理全部交到 childArray 这一个数据结构里面去管理.

childArray 也是一个结构体. 目前它管理孩子节点用的是动态数组的方式(如果后面发现对树的操作已插入等居多,可轻松改为链表方式). 已经实现的操作有: 从最后插入, 弹出最后一个, 插入, 删除等等常用的动态数组的操作.

这样管理的数据结构的层次十分清晰, 方便修改和维护, 也方便拓展功能.