babycc

胡俊良 2017302580220

January 10, 2020

1 环境说明

1.1 工具链

- Flex
- Bison 3.0 或更新，必须支持 Bison C++ API
- GCC 9/Clang 6 或更新，必须支持 C++17

1.2 macOS

- 使用 Homebrew 安装最新版工具链

```
brew install flex bison llvm
```

- make 即可

```
make clean && make run
```

1.3 Linux

以下 Ubuntu 为例

- 下载安装最新版 LLVM 工具链

```
sudo bash -c "$(wget -O - https://apt.llvm.org/llvm.sh)"
```

- 安装 Flex 和 Bison

```
sudo apt install -y flex bison
```

- 编译并运行程序

```
flex scanner.l
bison --report=state --graph --report=all parser.y
clang++ -std=c++17 -g -O0 -fsanitize=address \
    scanner.cxx parser.cxx -o babycc
for file in tests/*.c ; do
    cat ${file} | ./babycc 2>&1 | tee ${file}.output
done
```

1.4 测试说明

程序测试样例放在 tests 文件夹下，测试脚本会读入 tests 文件夹下每一个 C 语言源程序送入 babycc 主程序进行分析。

每个文件的输出结果保存在对应源码同一目录下的.output 文件中。

2 功能说明

词法规则中单个字母的定义详见源码。

词法规则只匹配正数，负数被认为是减号后接一个正数。

2.1 非十进制整数

```
// 十六进制前缀 + 十六进制数位 + 可选整型后缀
{HP}{H}+{IS}?
// 0 + 八进制数位 + 可选整型后缀
"0"{O}*{IS}?
```

2.2 指数形式小数

```
// 十进制数位 + 指数 + 可选后缀
{D}+{E}{FS}?
// 可选整数部分 + 小数部分 + 可选指数 + 可选后缀
{D}*"."{D}+{E}?{FS}?
// 整数部分 + 可选指数 + 可选后缀
{D}+"."{E}?{FS}?
// 十六进制前缀 + 十六进制数位 + 指数 + 可选后缀
{HP}{H}+{P}{FS}?
```

```
// 十六进制前缀 + 可选整数部分 + 小数 + 指数 + 可选
后缀
{HP}{H}*"."{H}+{P}{FS}?
// 十六进制前缀 + 整数部分 + 指数 + 可选后缀
{HP}{H}+"."{P}{FS}?
```

2.3 单多行注释

由于 Flex 的任意字符通配符不包括换行符，故单行注释可直接用//开头的任意字符串匹配。

```
"//".*  { /* consume //-comment */ }
```

多行注释则先匹配开头的/*，然后不断吃掉输入字符，直到看到 */为止。

```
"/*"    { comment(); }
```

相应吃注释程序源码：

```
void comment() {
    for (int c; (c = yyFlexLexer::yyinput()) ≠ 0;)
        if (c == '*') {
            while ((c = yyFlexLexer::yyinput()) == '*');
            if (c == '/') return;
            if (c == 0) break;
        }
    yyFlexLexer::LexerError("unterminated comment");
}
```

3 程序特点

3.1 智能内存管理

所有的终结符、非终结符均用一个 Node 对象表示，其实是 variant 的 naïve 实现。

节点内用：

- 一个 enum class 表示终结符或非终结符的类型
- 一个向量表示该节点在语法树中的孩子们

- 一个用于标记当前节点在输入流中起始位置的成员

```
class Node {
public:
    std::string label;
    SymbolType type;
    std::pair<int, int> position;
    std::vector<std::shared_ptr<Node>> children;
    std::string to_string();
};
```

Bison 里分析栈中的节点全部被包装在一个智能指针 std::shared_ptr<Node> 中，故当节点被弹出或分析过程非正常推出时，所有节点消耗的内存能被自动释放。

3.2 Pretty Printing

语法树会以类似命令行工具 tree 输出格式的形式输出，方便检查从属关系，如下：

```
└──Def (2)
    ├──Specifier (2)
    │   └──TYPE (2) int
    ├──DecList (3)
    │   └──Dec (3)
    │       └──VarDec (3)
    │           └──ID (3) i
    └──SEMI (3) ;
```