编译原理实验 1

姓名：张灵毓　　**学号：**171240524
邮箱：171240524@smail.nju.edu.cn

2020 年 3 月 14 日

实现的实验功能

词法错误的识别

- 对于符合 C–词法定义的字符，可以将其识别，对于未定义的字符以及不符合词法单元定义的字符，程序将会报 A 类错误。
- 除此之外，对于不符合规定的整形数和浮点数，都认为是词法错误，直接在词法分析器部分识别报错。

语法错误的识别

- 除了实验要求中的要求 1.3 相关的语法错误外，还可以识别一些常见的语法错误比如: 丢失封号，丢失括号，定义时缺少 Specifier 等等。
- 识别过程是首先 yyparse() 函数发现不符合产生式的情况，接着调用 yyerror() 函数向 stderror 打印错误类型行号和说明信息。随后进行错误恢复继续识别。错误恢复具体内容见 syntax.y 文件.
- 关于说明信息的打印，Bison2.5 手册中提到可以在定义部分加上 $\%error-verbose$, 则 yyerror() 中的 msg 参数会更加详细地打印相关信息, 如果使用的 bison 版本较高比如 3.4.1,3.5，$\%error-verbose$ 可能会在编译时提出 warning, 这是因为在较新版本的 bison 中更建议使用 $\%defineparse.errorverbose$, 但该警告不影响使用。

打印语法树

一些准备

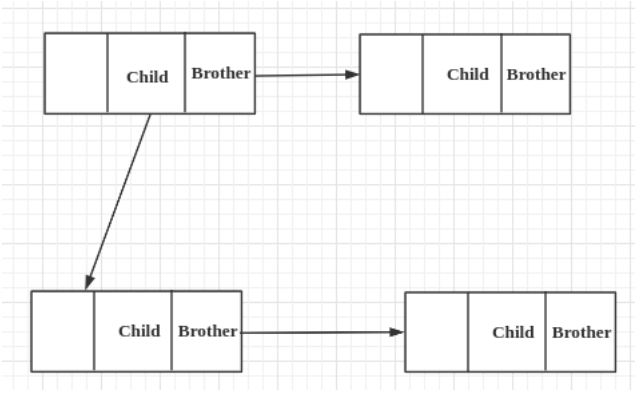
- 宏定义 YY_USER_ACTION, 定义每个语法单元的行号。
- 使用 $\%type$ 申明非终结符号。

- 语法树结点

单元名称 (name)	子结点数量 (num_child)	行号(row)
		列数(col)
是否是终结符号 (S_or_w)	数据 (value)	访问标识 (visit)
子结点指针 (child)		兄弟结点指针 (brother)

, 并将语法单元的属性值都改成 *struct Node**

- 树结构



树的建立

- 终结符:

```
{float} {
//建立结点(名字, 行数, 终结符还是非终结符)
yylval.ptr = Create("FLOAT",yylineno,0); // 见SynTree.c
// int,float,type,id的属性值
yylval.ptr->value.type_float = atof(yytext);
return FLOAT;
}
```

- 非终结符:

```
ExtDefList : ExtDef ExtDefList {
                    //建立结点(名字, 行数, 终结符还是非终结符)
                    $$ = Create("ExtDefList",@1.first_line,1); // 见SynTree.c
                    //建树添加结点(父结点, 子结点)
                    AddNode($$,$1); // 见SynTree.c
                    AddNode($$,$2);
                    // 将当前的根结点保存在全局变量root中
                    root = $$;
                }
|    /* empty*/ { $$ = NULL; }
;
```

语法树的输出

- 对构建好的语法树从根结点开始进行深度优先搜索 (前序遍历)，见 SynTree.c 中的 OutputTree()。
- 语法树结点信息输出缩进：在对子结点进行深度优先搜索之前，先修改子结点的列数。

```
1           child->col = parent->col + 2;
```