编译原理实验 1

孙伟杰 191250127

2022 年 3 月 13 日

完成了所有文档中的要求，如构建语法树，输出不同的错误信息，错误恢复等。
使用方式即在 Code/路径下在 terminal 中输入 make parser，然后./parser 即可

输出不同的错误信息即在 lexical.l 中对没有定义的规则自动输出信息，在 syntax.y 中定义 yyerror
在进行语法树的构建时，由于规则众多，手动进行每一个的添加会过于繁杂，在手册中找到了 YYLLOC_DEFAULT 宏，他在每次有规则被匹配时都会被调用

```
# define YYLLOC_DEFAULT(Cur, Rhs, N)                      \
do                                                        \
  if (N)                                                  \
    {                                                     \
      (Cur).first_line   = YYRHSLOC(Rhs, 1).first_line;   \
      (Cur).first_column = YYRHSLOC(Rhs, 1).first_column; \
      (Cur).last_line    = YYRHSLOC(Rhs, N).last_line;    \
      (Cur).last_column  = YYRHSLOC(Rhs, N).last_column;  \
    }                                                     \
  else                                                    \
    {                                                     \
      (Cur).first_line   = (Cur).last_line   =            \
        YYRHSLOC(Rhs, 0).last_line;                       \
      (Cur).first_column = (Cur).last_column =            \
        YYRHSLOC(Rhs, 0).last_column;                     \
    }                                                     \
while (0)
```

其中 YYRHSLOC 可以得到右边第 i 个语法单元的 yylloc。

而 bison 可以让我们自定义 yyltype 从而修改 yylloc 的定义，在其中加入了语法树的节点，也就是说每个语法单元的位置信息中现在还包括它在语法树中的位置。

```
%code requires{
  #define YYLTYPE YYLTYPE
  typedef struct YYLTYPE
  {
    int first_line;
    int first_column;
    int last_line;
    int last_column;
    char *filename;
    struct StNode * node;
  } YYLTYPE;
}
```

进而可以修改 DEFAULT 宏，在有规则匹配时自动进行语法树的构建，即

将右边的语法单元插入左边语法单元的子树中。
在进行语法树的打印时，我们将匹配 program 的 action 中 program 的 yylloc 中的语法树节点作为 root 节点然后进行树的遍历打印
进行错误恢复时, 对 stmt,exp,compst 等等进行错误恢复的定义