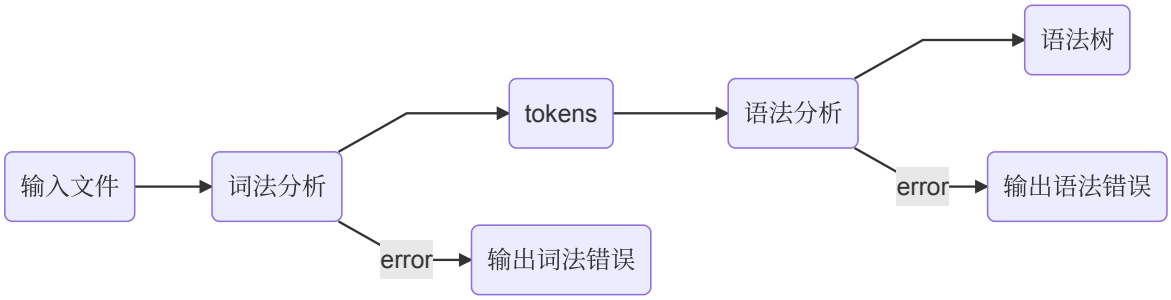
编译原理Lab1实验报告

计算机科学与技术系 201870214 宋骥原

计算机科学与技术系 201220217 申宸

实验内容

词法分析与语法分析的模拟



实验成果

我们小组实现了所有基础功能，并完成了选做1.1

实验思路

文件结构

文件名	文件意义
lexical.l	词法分析的正则表达式匹配与处理
syntax.y	语法分析的处理与语法树的构建
main.c	主函数，同时输出语法树
type.h	自定义结构体与自定义枚举类型

词法分析

我们的词法分析是应用了Flex，具体做法是定义一些正则表达式，再将输入流中的字符进行正则匹配，匹配完成后，将获得一系列的token，并将其传给语法分析器，在这个过程中，我们需要对标识符（ID）维护一个符号表，方便与语法分析器进行交互。

符号表的维护

符号表的表项目前我们定义如下，后续可以随着需要再行添加

```c
typedef struct symbol_node
{
    char *name;
    int type;
} symbol_node;
```

词法错误处理

同时我们也要在词法分析阶段对词法错误报出，只需要识别不到相应的词法单元，我们就需要将该字符进行报错输出，即我们所需要的Type A错误。

```
. {is_error = 1;
   printf("Error type A at Line %d: Mysterious characters \'%s\'\n",
          yylineno, yytext);}
```

与语法分析的交互

由于我们的词法分析的yylval默认是int类型，这对于我们后续进行语法分析是及其不利的，因此我们在bison源代码中定义了如下union类型，用以处理不同类型的词法单元。

```
%union{
    int type_op;//操作符
    int type_int;//int值
    int type_id;//ID在符号表中的偏移量
    int type_sym;//C--内置符号
    int type_type;//C--类型关键字
    int type_brac;//C--括号
    int type_key;//C--普通关键字
    float type_float;//float值

    struct Node *root;//对于非终结符，这里我们用来存储以该语法单元所对应的语法树的根节点
}
```

如此以来，我们就可以根据词法单元不同的类型来赋予yylval不同的值即可，方便后续处理。

拓展的处理

在这里，我们小组实现了对于八进制和十六进制数的识别和处理，具体正则表达式如下：

`int8 "0"{1}[0-7]+`

`int16 "0x"{1}([0-9]|[a-fA-F])+`

之后我们只需要将yytext翻译成相关的十进制数据，按照int类型存入即可。

语法分析

语法分析阶段就需要我们根据词法分析处获取的词法单元，根据一系列的规则构造相应的LALR自动机，并根据词法单元输入流进行移入/规约操作，最终实现语法分析。

产生式的处理与语法树的构建

我们根据C--的语法，构造相关的上下文无关文法，并在每一个文法后面添加一定的处理，在本次实验中，我们的处理就是构造语法分析树，具体分为以下两种类型。

对于语法分析树的结点，我们使用如下结构体类型来描述：

```
typedef struct Node
{
    int is_terminal;//是否是终结符
    int type;//类型
    union
    {
        int int_val;
        float float_val;
    } type_val;//存储结点的值，如int代表的真值，ID在符号表的偏移量等，分为两种类型
    int branches_num;//直接子结点数目
    int line;//该语法单元第一次出现的行数
    struct Node *nodelist[MAX_BRANCHES_LEN];//直接子结点的指针列表
} Node;
```

按照产生式的不同，我们有如下两种处理方式

1、该产生式体部完全为终结符，则我们需要构造一个新的结点，代表该终结符，同时再构造一个新的结点用来表示该产生式的头部，并将其连接即可。

2、该产生式体部存在非终结符，则由于我们LALR自底向上分析的特点，产生式体部的结点已经构建完毕，因此我们直接将其与产生式头部结点连接即可，当然，该产生式头部结点也需要我们新创建。

在以上过程中，我们由于上述定义YYTYPE的union类型中，我们用存储非终结符所对应的语法树的根节点，因此我们假设有如下产生式：

$$S \rightarrow S\ T$$

我们可以直接通过`$$ = new_node();add_node($$, $1);add_node($$, $2);`即可实现连接操作，大大降低了实验的难度。

语法错误的处理

对于语法错误，我们需要处理error，根据自定样例的测试，我将其插入了部分产生式的体部，例如

`IF LP error RP Stmt ELSE Stmt`，以此来实现错误的规约规则，同时处理错误后，需要yyerrok，以确保后续连续字符错误的正确处理。

ELSE二义性与SUB MINUS的处理

这里我们使用了%prec，使得某一规则与某一不存在的终结符具有相同优先级，以此来实现二义性的避免与SUB MINUS的处理。

写在最后

学海无涯