Lab1 实验报告

151220128 吴一楠

一、实现的功能

检查错误类型：

通过全局变量 noerror 来判断是否出错，如果出错将不输出语法树

错误类型 A：所有未被匹配的内容会出错

```
{noerror=0;printf("Error type A at Line %d: Mysterious character
'%s'.\n",yylineno,yytext);}
```

错误类型 B：通过 yyerror()实现

识别八进制和十六进制：

正则表达式匹配

```
0{octn}*
0x{hexn}+
```

识别指数形式的浮点数：

正则表达式匹配

```
(({digit}*\.{digit}+)|({digit}+\.))|((({digit}*\.{digit}+)|({digit}+\.?))[eE][+-]?{digit}+)
```

二、编译程序

通过 makefile 进行编译

```
parser:
        @bison -d syntax.y
        @flex lexical.l
        @$(CC) main.c $(SC) -lfl -ly -o parser
```

make parser 能够直接编译得到程序 parser，随后执行./parser xxx 即可

三、实验过程中的问题

一开始能够识别浮点数，加入语法树之后不能识别，后发现多了换行

暂未实现更加精确的报错：错误类型 B 无法分类并纠错

```
wu@ubuntu:~/Desktop/lab$ ./parser 1.c
Error type A at Line 4: Mysterious character '~'.
```

```
wu@ubuntu:~/Desktop/lab$ ./parser 1.c
Error type B at Line 4:syntax error
```

四、其他

节点的创建采用了可变参数，将出了 float 以外的其他所有类型整合在一起（也可将 float 整合，但无法完成浮点数的识别，误以为此处出错，浮点数单独采用一个函数）

```c
syntax_node* init_syntax_node(int lineno,char symbol[],char* text,...) {
        syntax_node* p=malloc(sizeof(syntax_node));
        strcpy(p->symbol,symbol);
        p->lineno=lineno;
        p->is_terminal=1;
        p->int_val=0;
        p->nr_child=0;
        if(text=="INT"){
                va_list valist;
                va_start(valist,text);
                p->int_val=va_arg(valist,int);
                va_end(valist);
        }
        else if(text!=NULL){
                strcpy(p->inf,text);
        }
        return p;
}

syntax_node* init_syntax_floatnode(int lineno,char symbol[],float value) {
         syntax_node* p=malloc(sizeof(syntax_node));
         strcpy(p->symbol,symbol);
         p->lineno=lineno;
         p->is_terminal=1;
         p->nr_child=0;
         p->float_val=value;
         return p;
}
```