《编译原理》课程实验报告

实验名称	实验五　中间语言翻译器的构造
实验内容	1.自选一段含有顺序结构、分支结构和循环结构的代码，将其翻译成中间代码形式。 2.中间代码形式可以是四元式或汇编格式。

一、实验目的：

设计中间代码生成器，进一步锻炼设计、实现、分析和维护编译程序等方面的能力，

二、主要数据结构：

线性表和散列表

三、主要设计思想与算法：

首先构造出文法，然后设计词法分析器，将分析出的单词送入语法分析器，分析过程中使用语法制导分析，对特定产生式的归约调用语义子程序，产生相应四元式形式的中间代码。

四、实验结果及测试用例：

测试用例：

文法：

<programme>-><ST>|<ST><programme>

<ST>-><assginST>|<bool_expr>;|<opST>|<branchST>|<whileST>

<assginST>-><var_type><opST>

<ari_expr>-><ari_expr>+<ari_expr>|<ari_expr>*<ari_expr>|(<ari_expr>)|<const>|<var>

<bool_expr>-><bool_expr_and><bool_expr>|<bool_expr_or><bool_expr>|<ari_expr><relop><ari_expr>|!<bool_expr>|<ari_expr>

<bool_expr_and>-><bool_expr><and>

<bool_expr_or>-><bool_expr><or>

<branchST>-><T><ST>

<C>-><if><bool_expr><then>

<T>-><C><ST><else>

<whileST>-><Wd><ST>

<W>-><while>

<Wd>-><W><bool_expr><do>

<opST>-><var>=<ari_expr>;

测试代码：

```
while a >= 10 && a != 11 || 0 do
    a=0;
int a = 2 * (10 + 20);
float b = 10.1;
if a then
    if b then
        int c=10;
    else int c=100;
else if b>0 then
    a=0;
else
    a=99;
while a<b do
    if 9>8 then
        a=b+10;
    else a=0;
```

实验结果：

```
--------------------
1 ('j>=', 'a', 10, 3)
2 ('j', '_', '_', 5)
3 ('j!=', 'a', 11, 7)
4 ('j', '_', '_', 5)
5 ('jnz', 0, '_', 7)
6 ('j', '_', '_', 1)
```

```
7 ('=', 0, '_', 8)
8 ('+', 10, 20, 'TEMP8')
9 ('*', 2, 'TEMP8', 'TEMP9')
10 ('=', 'TEMP9', '_', 'a')
11 ('=', 10.1, '_', 'b')
12 ('jnz', 'a', '_', 14)
13 ('j', '_', '_', 20)
14 ('jnz', 'b', '_', 16)
15 ('j', '_', '_', 18)
16 ('=', 10, '_', 'c')
17 ('j', '_', '_', 25)
18 ('=', 100, '_', 'c')
19 ('j', '_', '_', 25)
20 ('j>', 'b', 0, 22)
21 ('j', '_', '_', 24)
22 ('=', 0, '_', 'a')
23 ('j', '_', '_', 25)
24 ('=', 99, '_', 'a')
25 ('j<', 'a', 'b', 27)
26 ('j', '_', '_', 0)
27 ('j>', 9, 8, 33)
28 ('j', '_', '_', 32)
29 ('+', 'b', 10, 'TEMP29')
30 ('=', 'TEMP29', '_', 'a')
31 ('j', '_', '_', 25)
32 ('=', 0, '_', 'a')
--------------------
```

五、实验总结：

通过本实验，我对编译程序从词法分析、语法分析到中间代码生成的整体过程与

练习有了进一步理解。

在编写代码过程中，词法分析对于文法文件的识别整理出终结符、非终结符是第一个遇到的难点，如何进行分类处理是解决关键。

设计文法的过程较为顺利，但针对所设计的文法进行语法分析构成了问题，该文法并不是适用于我们所学过的优先分析法，于是我上网查询了关于 LR 分析的资料，并基于开源代码进行了相应更改，尤其是对于文法二义性的处理，设定优先级来消除冲突。

然后在针对产生式编写语义子程序，并在语法分析的归约过程中进行调用。虽然遇到了很多困难，但处理的过程让我感到了编程和解决困难的快乐。

但由于个人能力有限，语法制导翻译并不尽善尽美，可能存在未知的 bug 并未得到测试修复。由于文法设计较为粗糙随意，可能为后续编程带来较多困难。