华东师范大学 计算机科学与技术系

Small 语言编译器

设计说明书

毛杰文 10102130253

2013-10-22

1、介绍

本系统为一个小型编程语言——small 语言的编译器和中间代码解释器。

Small 语言特点：

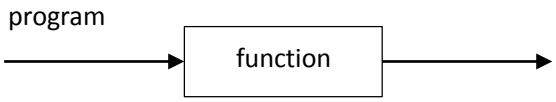
1、 语法类似于 Pascal，包含赋值语句、条件转移语句、循环语句、输入输出语句和表达式操作。
2、 没有函数调用。
3、 所有的变量都是全局的。
4、 只有整数类型和一维整数数组类型。
5、 不需要进行变量声明，通过第一次给一个变量赋值或读入以隐式声明一个变量。
6、 注释支持单行注释（以“//”开头）和块注释（以“/*”开头，以“*/”结尾，可以多行）

2、编译器系统结构

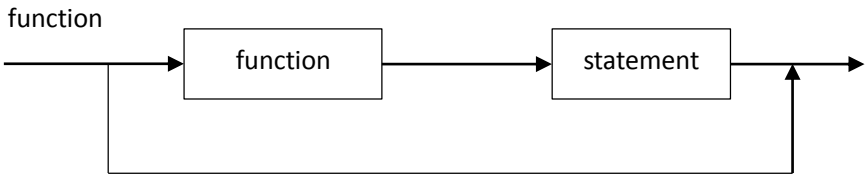
2.1 编译器

2.1.1 Small 语言语法图

I. Program := function



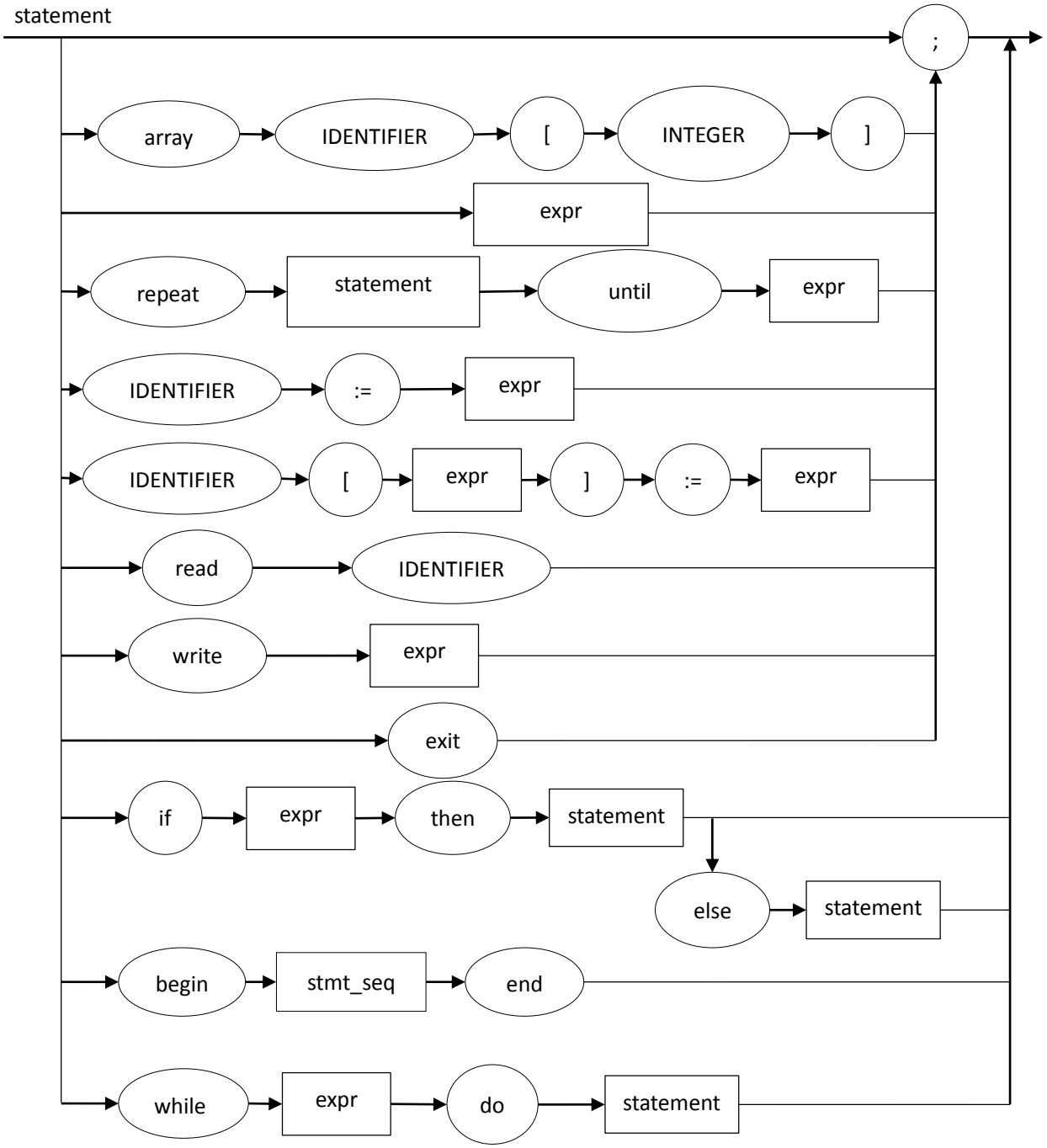
II. function := function statement|NULL



III. statement:

';'
|ARRAY IDENTIFIER'['INTEGER']';'
|expr';'
|REPEAT statement UNTIL expr';'
|IDENTIFIER ASSIGN expr';'
|IDENTIFIER'['expr']' ASSIGN expr';'

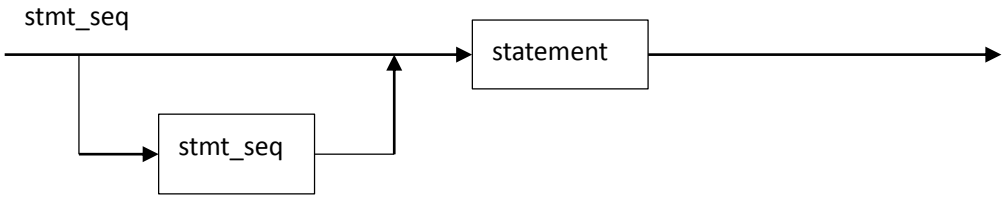
|READ IDENTIFIER';'
|WRITE expr';'
|IF expr THEN statement [ELSE statement]
|BEGINSYM stmt_seq END
|WHILE expr DO statement
|EXIT';



IV. stmt_seq:

statement

|stmt_seq statement



V. expr:

INTEGER

|IDENTIFIER

|IDENTIFIER['expr']'

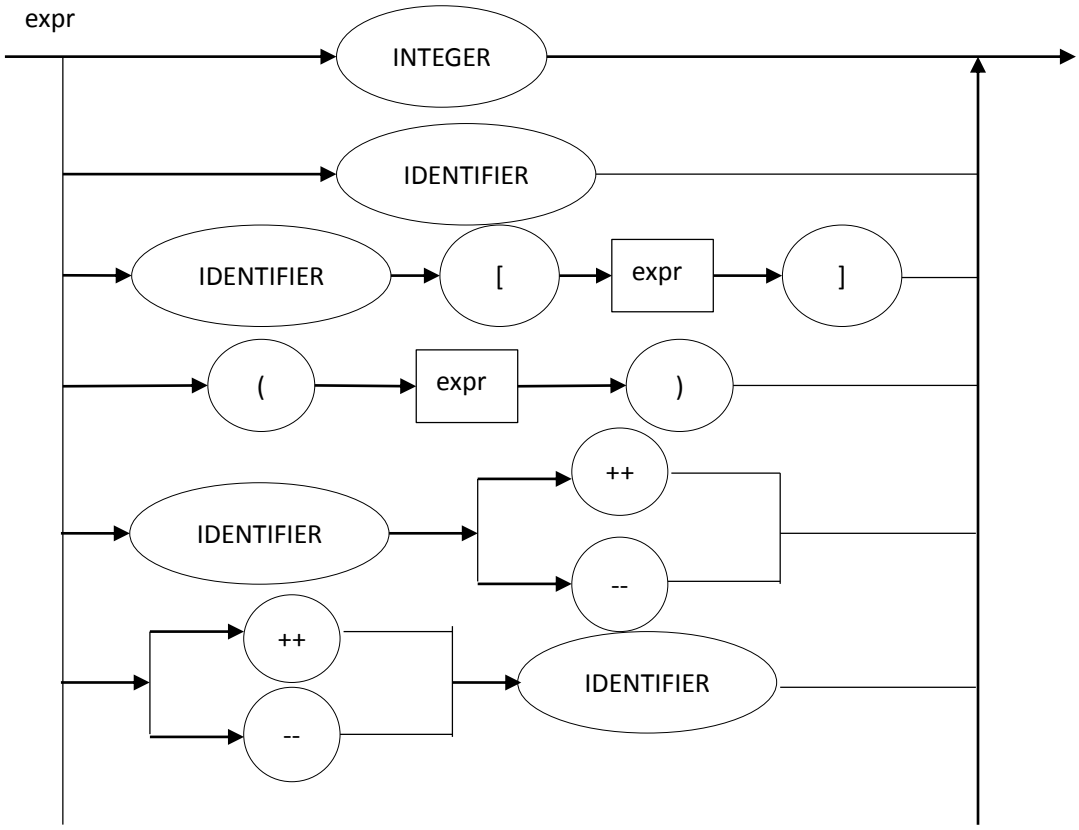
|'('expr')'

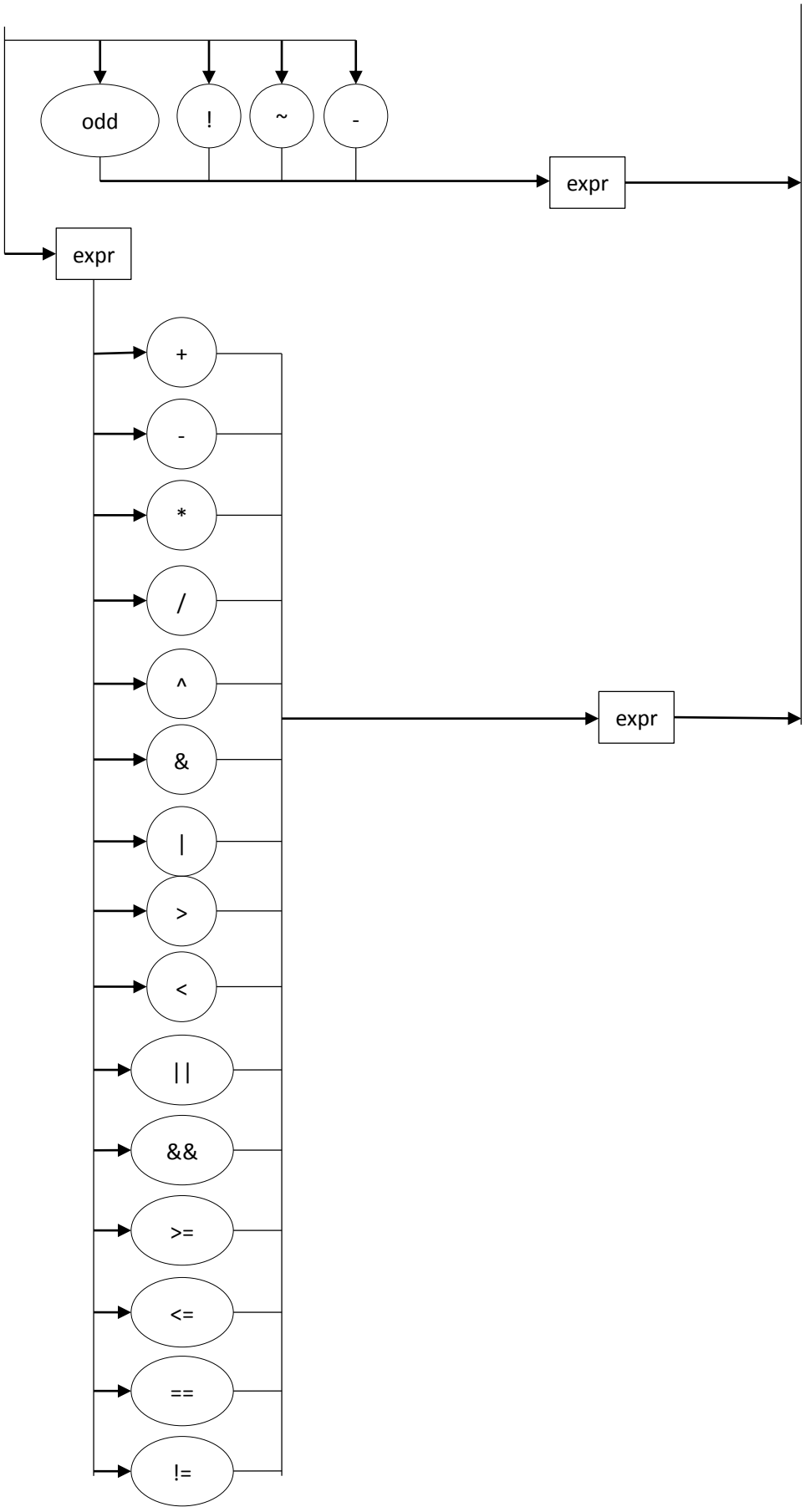
|IDENTIFIER("++"|"--")

|("++"|"--")IDENTIFIER

|(odd|'-'|'!'|'~') expr

|expr ('+'|'-'|'*'|'/'|'%'|'>'|'<'|">="|"<="|"=="|"!="|"&&"|"||"|'^'|'&'|'|')expr




odd
!
~
-
expr
expr
+
-
*
/
^
&
|
>
<
| |
&&
>=
<=
==
!=
expr

2.1.2 判断语法类型

经分析可知，2.1.1 所示语法图符合 LALR(1)文法，可使用自底向上的方式进行分析。

2.1.3 程序总体结构

Small 语言的编译程序将采用 flex 和 bison 进行。语法定义同 2.1.1。
主要算法：将使用 flex 和 bison 分析出的语法结构构造成一棵语法树，对此语法树进行深度遍历后生成中间代码，再由解释器解释中间代码执行。
各文件说明如下：

文件名	说明
smallcc.l	small 语言的词法分析文件
smallcc.y	small 语言的语法、语义分析文件
common.h	一些全局变量的声明
smallh.h	对语法树数据结构的定义
ex.c	对语法树解析的函数、代码生成函数
Intepreter.cs	实现解释器接口类
Table.cs	符号表类
Program.cs	解释器主程序

2.1.4 语法出错定义

目前 small 语言可能存在的语法错误如下：
1、 语句后漏掉分号
2、 If 语句缺少 then
3、 While 语句缺少 do
4、 语句块没有结束符
5、 其他错误

2.2 虚拟机

2.2.1 虚拟机组织结构

该虚拟机包含如下组织结构：
1、 代码区：编译器生成的所有中间代码都会进入代码区，准备被取出调用。
2、 程序计数器（PC）：控制程序流程
3、 数据栈：存放所有由程序生成的数据对象
4、 符号表：存放所有出现的变量或数组。其中，变量保存它们的名字和值，数组保存其名

字和首地址。

5、 栈顶指针 top

2.2.2 虚拟机指令格式

虚拟机指令格式为双字节指令格式：

Opcode	Oprand

Opcode 为指令码

Oprand 为操作数（有的指令没有操作数，通常是对栈顶的两个元素进行操作）

2.2.3 虚拟机指令集

指令码	说明
push	将一个整数或一个标识符压入运行栈
pop	将一个整数或一个标识符从运行栈内弹出，若弹出的是标识符，则标识符在符号表中的值被更新
jz	根据 flag 置位与否决定是否跳转
jmp	无条件跳转到指定标号处
out	将栈顶元素输出到显示终端
in	由用户输入标识符的值，将标识符在符号表中的值更新
neg	栈顶元素取相反数
add	将栈顶的两个整数相加后放入栈顶，原来的两个数被弹出
sub	栈顶两整数相减
mul	栈顶两整数相乘
div	栈顶两整数相除
compLT	栈顶两整数相比较，若底部元素小于顶部元素则 flag=1
compGT	栈顶两整数相比较，若底部元素大于顶部元素则 flag=1
compLE	栈顶两整数相比较，若底部元素小于等于顶部元素则 flag=1
compGE	栈顶两整数相比较，若底部元素大于等于顶部元素则 flag=1
compEQ	栈顶两整数相比较，若底部元素等于顶部元素则 flag=1
compNE	栈顶两整数相比较，若底部元素不等于顶部元素则 flag=1
halt	停止程序执行，不再执行后续语句
stm	将数存储入数组名所指向的地址加上偏移量的位置
ldm	将数组名所指向的地址加上偏移量的位置的值取出
new	说明该标识符是一个数组的首地址
alloc	必须紧跟在 new 之后，为数组开辟空间
and	仅跟在 comp 系列指令之后，将栈顶两数做逻辑与运算，改变控制寄存器的值
or	同上，将栈顶两数做逻辑或运算
not	同上，将栈顶数做逻辑非运算（非零值变成 0，0 变成 1）
bitand	将栈顶两数做位与运算

bitor	将栈顶两数做位或运算
bitnot	将栈顶数做取反运算
Bitxor	将栈顶两数做异或运算
inc	栈顶元素作自增运算
dec	栈顶元素作自减运算

3、对原语言的主要扩展点

3.1 语法修正

鉴于处理上的方便，将语法改成了类 C 语言的结构。即语句块必须由 begin-end 包括起来。另外，同样鉴于处理上的方便，所有的语句后都必须加分号。end 后不需要加分号。
取消了原 small 语法对表达式结构的细分。通过在 yacc 中说明运算符的优先级来区分不同的运算。运算的优先级参考 C 语言。

3.2 功能扩展

3.2.1 增加 while-do 语句

语法：”while” expr “do” statement。由于 repeat-until 和 do-while 在功能上重复，因此只做了 while-do 语句。

3.2.2 增加自增自减运算符“++”“--”

可以支持单个变量的自增++，自减--运算。有如下四种情况：
1、 前置运算符，即++a,--a
2、 后置运算符，即 a++,a--
3、 前置赋值，即 b = ++a,b=--a;
4、 后置赋值，即 b = a++, b= a--;
同时在 ex.c 文件中加入一个变量 wait，即是否需要等待语句分析结束才处理自增自减运算符

Wait 的值	说明
0	没有自增（自减）操作，不做处理
1	后置自增操作，若有赋值语句，则语句等价为 b=a;a=a+1;
2	后置自减操作，和上一条类似
3	前置自增（自减）操作，直接输出 inc 或 dec 的操作符

另外在语句处理末尾也加入判断，避免了因没有赋值号导致的漏操作。
增加两个中间代码操作符：inc 和 dec，为栈顶元素增加 1 或减少 1

3.2.3 增加 ODD 判断奇偶

语法：”odd” expr。若 expr 的值是奇数，返回 1，否则返回 0

3.2.4 求余运算%

语法：expr ‘%’ expr。求 expr1 除以 expr2 所得余数

3.2.5 布尔运算 and or not

由于 small 语言没有定义 bool 型变量，因此采用整数表示 bool 值。

1、 逻辑与”&&”：
    若参与逻辑与运算的数中有 0，则整个表达式为 0，否则为 1
2、 逻辑或”||”：
    若参与逻辑或运算的数中有 1，则整个表达式为 1，否则为 0
3、 逻辑非”!”：
    对原来的逻辑值取反。非零值变为 0，0 变为 1。

PS：当进行了比较运算之后，栈顶会依据比较结果赋值为 0 或 1。

3.2.6 位运算

本 small 扩展支持位运算，分别采用’&’（位与），’|’（位或），’~’（按位取反）,’^’（异或）表示。

3.2.7 跳过行注释

Small 语言不允许注释嵌套

3.2.8 数组

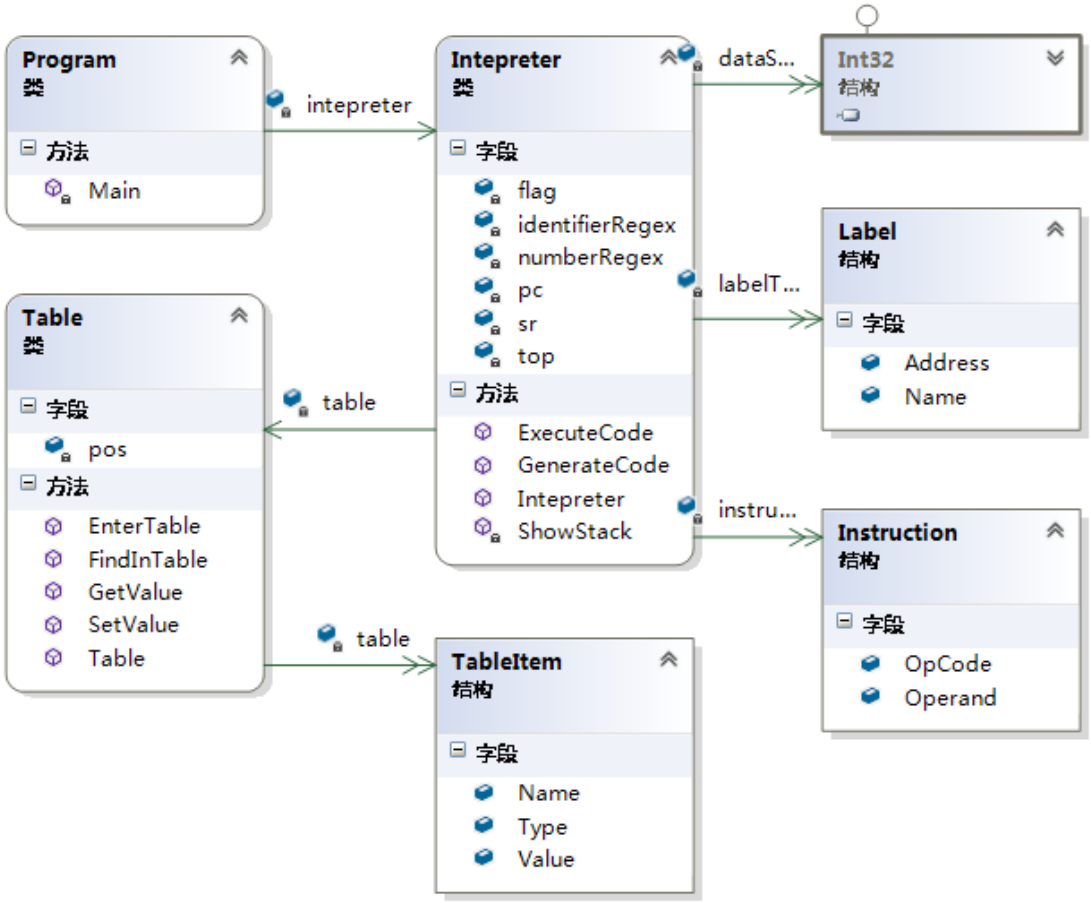
使用数组前必须声明，例如：

```
array a[10]
```

不支持数组声明大小时使用变量。但在使用数组元素时（不论读写）均可在方括号内使用表达式。

4、 解释器

4.1 解释器类图



4.2 公共接口说明

接口	说明
void ExecuteCode(bool[] switchs)	执行代码区中的代码 switchs：传入的开关参数：1-是否显示堆栈，2-是否显示指令
void GenerateCode()	将编译器生成的中间代码读入，在内存中产生“指令”
Intepreter(string path)	构造方法，对各项参数初始化 path：打开文件的路径

4.3 其他说明

由于解释器是用 C#语言写成的，因此必须安装.NET Framework，最低版本要求为.NET Framework 3.5。