Eeyore 文档说明

1 概述

Eeyore /'ɪ:juə(r)/ 是一种三地址码，用作 MiniC 语法分析后的输出格式。Eeyore 的设计同样遵循简洁的原则，使代码易读易调试。

2 语法描述

Eeyore 要求**每条语句单独占一行**。

2.1 变量

- Eeyore 中的变量有三种：原生变量，临时变量，函数参数。这三种变量分别以`'T','t','p'`开头，后接一个整数编号，编号从 0 开始，每个函数单独编号。如 `p0,p1,t0,T0`。
- Eeyore 的变量声明形如 `var t0` 和 `var 8 T0`，前者声明了一个 int 型临时变量，后者声明了有 2 个元素的原生 int 数组。
- **注意！函数参数不需要声明**。
- 除函数参数变量外，其余变量不允许重名。
- 函数内声明的变量作用域为变量声明语句到函数结束语句，函数外声明的变量作用域为变量声明语句到程序最后。
- 所谓原生变量，是指 MiniC 中使用的变量转到 Eeyore 中对应的变量。相应地，临时变量是指 MiniC 中没有显式对应变量的变量。

  其实这两种变量在语义上**没必要做如此区分**，Eeyore 区分二者是为了方便用户调试。用个例子来说明，把左边的 MiniC 语句翻译到 Eeyore：

MiniC	Eeyore
	`var T0`
	`var T1`
`int a;`	`var T2`
`int b;`	`var t0`
`int c;`	`t0 = 2 * T2`
`a = b + 2 * c;`	`var t1`
	`t1 = T1 + t0`
	`T0 = t1`

上面 T0,T1,T2 是原生变量，分别对应 MiniC 中的 a,b,c。t0,t1 是临时变量，分别对应中间运算结果 2*c, b+2*c。

2.2 表达式

Eeyore 表达式有以下特点:

- 允许直接把整数用作运算分（如 t0 = 2 * T2)。
- Eeyore 表达式支持的单目运算符有'!','-'
- 支持的双目运算符有'!=','==','>'.'<','&&','||','+','-','*','/','%'，前 6 个是逻辑运算符。
- 数组操作语句形如 T0 [t0] = t1 和 t0 = T0 [t1]。
- 注意！因为 MiniC 的 base 语法集只有 int 和 int 数组类型，数组操作语句中括号内的数应当是 4 的倍数。

2.3 函数

- Eeyore 中的函数以'f_' 开头，后接函数名，如 f_main,f_getint。
- 函数定义语句形如 f_putint [1]，中括号内的整数表示该函数的参数个数，函数结束处应有函数结束语句，形如 end f_xxx。
- 函数外的变量声明语句被视为全局变量声明，函数内的视为局部变量声明。
- 函数调用语句形如：t0 = call f_xxx。
- 传参数指令形如：param t1，所有传参都是**传值**，多个参数需依次传入。
- 作为参数的变量，在 param Variable 语句之后，到函数调用前，不可修改。(这样限制的目的是使寄存器分配时避免繁琐的分类讨论)
- 函数返回语句形如 return t0。

2.4 标号与跳转

- Eeyore 中的标号以小写字母'l' 开头，后接整数编号，编号从 0 开始，如 l0,l1。标号用来指明跳转语句的跳转地点，标号声明语句形如 l0:。
- 跳转语句分两种：无条件跳转、条件跳转。如 goto l1 和 if t0 < 1 goto l0。

2.5 缩进

Eeyore 没有缩进要求，但是允许缩进，为了之后代码调试的便利，我们建议正确使用缩进。

2.6 注释

Eeyore 允许单行注释，与 C 语言注释类似使用//，处理时自动忽略改行从//之后所有内容。

2.7 系统库支持

Eeyore 模拟器提供对输入输出的系统调用支持，对应的函数原型如下：

- `int getint()` //从标准输入读取一个整数
- `int putint(int x)` //输出 x 到标准输出
- `int getchar()` //从标准输入中读取一个字符
- `int putchar(int x)` //输出 ASCII 为 x 的字符

具体对应的 MiniC 代码和 Eeyore 代码用法参见章节“**示例**”

3 BNF

⟨*Declaration*⟩ ::= 'var' ⟨*INTEGER*⟩? Variable

⟨*FunctionDecl*⟩ ::= Function '[' ⟨*INTEGER*⟩ ']' '\n' ((Expression | Declaration)'\n')*
'end' Function

⟨*RightValue*⟩ ::= Variable | ⟨*INTEGER*⟩

⟨*Expression*⟩ ::= Variable '=' RightValue OP2 RightValue
| Variable '=' OP1 RightValue
| Variable '=' RightValue
| Variable '[' RightValue ']' = RightValue
| Variable = Variable '[' RightValue ']'
| 'if' RightValue LogicalOP RightValue 'goto' Label
| 'goto' Label
| Label ':'
| 'param' RightValue
| Variable '=' 'call' Function
| 'return' RightValue

⟨*Identifier*⟩ ::= ⟨*IDENTIFIER*⟩

⟨*Variable*⟩ ::= ⟨*VARIABLE*⟩

⟨*Label*⟩ ::= ⟨*LABEL*⟩

⟨*Function*⟩ ::= ⟨*FUNCTION*⟩

4 示例

MiniC	Eeyore

	var T0
	var 40 T1
	f_main [0]
	  T0 = call f_getint
	  var t0
	  t0 = T0 - 10
	  var t1
	  t1 = t0 > 0
	  if t1 == 0 goto l0
	  return 1
	l0:
	  var T2
	  var T3
	  T3 = 0
	  T2 = T3
	l1:
	  var t2
	  t2 = T3 < T0
	  if t2 == 0 goto l2
	  var t3
	  t3 = 4 * T3
	  var t4
	  t4 = call f_getint
	  T1 [t3] = t4
	  var t5
	  t5 = T1 [t3]
	  T2 = T2 + t5
	  T3 = T3 + 1
	l2:
	  param T2
	  call f_putint
	  return 0
	end f_main

int getint();
int putint(int x);
int n;
int a[10];
int main()
{
  n = getint();
  if (n > 10)
  return 1;
  int s;
  int i;
  i = 0;
  s = i;
  while (i < n) {
  a[i] = getint();
  s = s + a[i];
  ++i;
  }
  putint(s);
  return 0;
}	

5 Eeyore 模拟器使用方式

```
Usage ./Eeyore [-d] <filename>

-d : enable debug mode

- e.g. ./Eeyore -d test.in
```

出现"> "提示符表示进入debug模式，支持如下指令：

```
+ p <pc/symbol/label/funciton name>
    - e.g. p pc, p c1, p t1
    - Print the value of the symbol
+ s <number>
    - e.g. s 10
    - Run n Step
+ n
    - e.g. n
    - Run 1 Step
+ u <number/function/label>
    - e.g. u 10, u f_g, u l1
    - Run until pc equal to number or until the function or label.
+ r
    - e.g. r
    - Disable the debug mode and run until the program exit.
```