##### 0序言

Tina是一个小巧的动态类型语言；它文法风格非常贴近于C，在此基础上增加了动态类型、自动内存管理以及新的复合结构特性；它能够被编译器翻译成一种机器无关了中间语言的字节码文件，只需要一个简单的虚拟机(目前虚拟机使用标准C编写，生成的文件只有几十KB)，就能够在各个平台上运行。

##### 1基本使用方法

目前的编译器及虚拟机合为一体，其可执行程序默认为tina(在windows下默认为tina.exe)。如下是其参数及其使用方法。

--compile | -c 文件名 ：将单个.tina文件翻译成字节码文件，输入的文件名不包含“.tina”后缀。

--run | -r 文件名 ：载入并执行单个.tzw字节码文件，输入的文件名不包含“.tzw”后缀。

--excute | -x 文件名 ：载入并执行一个存放在.tbl文件中的字节码文件序列，输入的文件名不包含“.tbl”后缀。

--build | -b 文件名 ：将指定的.tbl文件中指定的.tina源代码逐一翻译成.tzw的字节码文件，输入的文件名不包含“.tbl”后缀。

##### 2 函数

Tina脚本语言最基本的执行过程为函数，此函数的概念与彼C语言中函数的概念基本相同，与此类似的还有在一些语言中称为子例程或之类的概念。

2.1:函数的定义

函数定义形式为： function func\_name (arg1 , arg2 , arg3….. ){ …….}：

其中function为函数定义的保留字，func\_name为函数的名称。一对小括号所包围的arg1 , arg2 , arg3…..为该函数输入的形式参数列表，该列表可以为空，表示该函数不接受参数。

2.2:函数的返回值

与C语言类似，函数可以通过return 保留字返回一个值，作为函数的返回值。

2.3:函数的动态性

与C语言不同的是，Tina中的函数的类型是完全动态的，从上面所说的参数列表可以看出，Tina完全不对传入进来的参数的类型进行限制；同时，Tina也完全不对返回值的类型进行限制，提供这样的机制的原因是我深深相信，这种动态化的函数可以使开发人员在开发当中从重重的类型转换之中获得解脱，并使得函数获得高度的复用。

2.3:第一类值

Tina中的函数属于第一类值(first class)，这意味着函数地位与其他类型的值相平等，譬如整数，能够被无差别的存储在变量中，被当做参数传给另一个函数，甚至能作为一个函数的返回值。

2.4:可见域与声明

Tina中的函数不同于C，不需要也不允许提前声明。在同一源代码文件中，在任何地方定义，都可以正确的执行代码。同时，Tina中的函数，除了main函数之外，其余的都是本地可见的，在一个代码文件中定义的函数除了只有main函数之外，其余的函数都只能在该文件中使用。

##### 3 变量

Tina中的变量是动态类型的，这意味着，变量本身无所谓是何种类型，决定变量性质的是变量某一刻所存贮的值——只有值有类型。目前，在Tina里共有这么几种类型：int 整数，real 实数，bool 布尔值，char 字符类型，nil 空类型，struct 结构体类型，vector 向量类型，tuple 元组类型。关于后三种值类型，我们在之后的章节进行讨论。

3.1 变量的定义

Tina中的变量必须先定义在使用，定义的方式为 Var variable\_name;当然，通常使用带初始化式的定义式会更加方便，如 var varIable\_name=3;其值的类型尤其初始的字面量确定。一下的例子可以表示：

var a=5; **//整数**

var a=5.1; **//实数**

var a=false; **/\*布尔值\*/**

var a=’a’; **/\*字符\*/**

var a=nil; **/\*空类型\*/**

刚才的例子同时也展示了Tina中支持的两种风格的注释，其使用方法与ANSI C 95或以上的C版本注释方法完全相同。

3.2 变量的作用域

除了结构体原型成员之外，**没有全局变量这一概念**，所有的变量要么属于一个结构体原型，要么属于某个函数的局部变量。函数定义之后，会在被声明的代码块中可见，但是内层的代码块中若有完全同名的变量会在内部将其屏蔽。比如一下的这个例子。

{

var a=5;

{

var a=4;

print a; **/\*打印的值为4\*/**

}

print a; **//打印的值为5**

}

print是一个保留字，它能够将右侧的表达式最后的结果打印到标准输出(stdout)中。上例中，内层的a就会屏蔽掉外层的a，直到代码执行跳出内层的代码块，此时a又代表外部的变量。

3.3 变量的初始值

Tina虚拟机严格保证所有的变量的初值都为nil，这样可以避免程序员在忘记初始化变量时发生严重的后果。

##### 4 控制流

Tina提供三种控制流保留字，分别是while for if，其使用方法与C语言类似，区别有二：其一，三种控制流中的表达式的结果值**必须为布尔值**，如果其结果不为bool值，将会产生一个已知的运行时异常，其具体的捕获、处理方式由虚拟机的实现定义；其二，for语句中的三个控制表达式**均不能省略**，同样的，若省略，将会产生一个已知的运行时异常，其具体的捕获、处理方式由虚拟机的实现定义。

##### 5 struct复合结构

向C语言致敬！我将C语言的struct关键字沿用到了Tina中，其目的就是为了表示一个复合类型。

5.1:基本概念原型

定义一个struct实际上等于定义了一个struct的原型，在前几节说到的不能有全局变量的例外就在此处。每一个struct被定义后就被认为是一个始终存在于内存中的全局变量，这个全局变量是一个复合类型——它包含了若干成员。

因此，我们说构造某个类型的struct对象，实际上等于将指定的原型拷贝到你新创建的对象的空间内，并触发初始化函数。

5.1:定义的形式

形如：

struct a\_struct{

var a;

fuction b()

{

…………

}

function a\_struct()

{

….

}

}

上例中的b我们有时候也称为“方法”(method)，我们提供一个特殊的关键字——self来简化方法的调用，self关键字的使用等同于一个变量，但是它始终指向当前结构体对象本身。这个概念类似于Objective C的self或者C PP中的this。

与结构体同名的成员方法，我们称之为构造函数、构造方法或构造器(constructor),在结构体对象被创建时，构造方法将被调用。

当然，我们说变量的类型是动态的，结构体的成员也不例外，故上例中的b在初始时，可能为一个方法，但绝对不代表在运行时b为一个方法。

5.2:创建

创建形式如下:

var a =obj(1,2,3,4,5);

##### 6 容器

Tina目前提供了两种容器类型，tuple——元组，vector——向量。两者的区别在于前者可以存储不同类型的值，后者所存储的值的类型必须严格一致。

6.1:构造方法

要构造一个元组/向量，方法如下：

var a= vector(1,2,3,4,5,6,7);

var b=tuple(1,2,3,4,5,6,7);

对了使用其他编程语言常有的字符串形式，我们提供了一种更加简便的构造方法，但是，请读者始终记住，在Tina中，**字符串不过是存储着字符类型的向量而已**。

var a = “hello world!!”; **/\*简便方式\*/**

var a = vector(‘h’,’e’,’l’,’l’,’o’,’ ’,’w’,’o’,’r’,’l’,’d’,’!’,’!’); **/\*等价的普通方式\*/**

6.1使用方式

不论是元组还是向量，都可以用”[]”下标运算符来访问其元素的值，与C或很多语言不同的是，索引由1开始，下例将打印一个向量中的所有元素：

var a =vector(1,2,3,4,5,6,7,8);

for(var i=1; i<=8;i++)

{

print a[i];

}