# L语言程序设计·概念版

# L语言前瞻

L语言是以汉字为基础的编程语言。旨在接近人类更加理解的编程语言。L语言属于上层语言，它的底层实现语言是C++，所以你也可以认为L语言本身就是一种面向对象的编程语言。函数的参数调用规则参照了Objective-C。

过去的编程语言，我们人类是主角，是我们让计算机根据我们编写的带有我们自己的主观意见的句子或者段去解释执行。噢，这太委屈计算机了。L语言编写起来更加是计算机自己想做什么，只不过是我们在教它怎么做而已——它就像一个婴儿一样。

L语言是采用中文编程，则一条语句段（作用域）结束就为句号“。”；逗号“，”作为断句即使语句更加清晰，它的作用好比英文编程中的空格，但功能不局限于空格。

大家可以回想一下，我们如何用汉字写一篇文章吧，是的，L语言编程就好比在写一篇另类的文章。

关于注释，英文编程采用的是双斜杠和/\*\*/，回想一下，我们的作文是如何给文章中的段落或者句子作注解，而且它只用于看，不用于念出来呢，没错它就是“（）”。当然在L语言里，英文编程所采用的注释方法依然有效。

我保留了数字和+-\*/=%^&|这些操作符，因为整体来讲它们是方便的，并不影响中文编程的纯洁性。

加减乘除等操作符，我赋予了他们新的含义。当初学习C语言的时候，老师让你把变量a加1用代码表示出来，我猜你可能这样写过：a+1; 你是不是在懊恼：“我这是a+1啊！”，从编译器角度来说，a+1这条语句产生了副作用，因为它没有指定谁来接收a+1产生的临时值。既然是自然语言编程，在L语言中，a+1就是给a加了1，是的，你没有看错！

我不喜欢用很多的空格去分离类型和实体，或者其它。我不认为这样是一种人类友好的表达想法的方式，虽然我也认为空格在当代英文编程中给人们调错和编码带来了方便。所以在L语言中，你很少看见空格，你看到的更多的是逗号和句号，连每条语句后面的分号都替换成了逗号。句号一般代表的结束，用来表示一段作用域的结束再好不过了。而分号在L语言中有了其它含义，我参考更多的是分号在中文表达的含义：并列（主要）、转折、承接、因果。

我也支持用英文作为数据实体的表示（即变量），因为用汉字有时候确实不能准确表达一种抽象的含义，如果你能也可以，反正都可以。

我把程序的函数抽象成（或许用“实例化”更准确吧）一篇文章的段落，段落是以两个汉字的空格开始，所以L语言的函数就是这样，两个汉字空格起手，参数紧跟空格之后，用【】包括其中，之后换行开始“段落”的书写，直至遇到下一个两个汉字的空格或者文件尾，函数结束。但最后一个字符一定是句号“。”，如果不算注释的话。但是，这个时候，我又在懊恼，我怎么忘了函数名，毕竟我们需要调用函数来执行功能，所以需要一个标记来标识一个被调用的函数。

# 数据类型

学习的空余时间，我在抽象数据类型。英文编程，亦或者说西方编程，他们很好的抽象了数据类型，例如：int、double、long等，经历了时间的洗礼，这些东西被保留和开发了出来。易语言，用中文描述式地把他们转换成了汉字的数据类型。

我在想，L语言要不要这样，int->整数，double->小数，这样生硬地翻译过来，太别扭，而且这样与易语言有什么区别，它已经在网络上被吐槽翻了。如何正确识别和抽象，这就是我来制定数据类型的一大难点。

# 关键字lists

关键字是不能被重新定义为数据类型的，但是可以被定义为函数的名字。关键字之间亦或者关键字与变量之间，不需要和其它东西用空格隔开，因为他们的含义或外加条件式的含义（例如函数名）在全局范围内是唯一的。我又要来写文章举例了——我们写文章的时候用到过空格吗？当然，行首除外。

是：一般结合判断子句来构成真假判断。

对：真值

错：假值

如果、就、否则、不然：构成选择结构语句。

不：否定字，参与一切与真值相关的操作，形成反值效应。

# 顺序程序设计

# 选择结构程序设计

L语言有两种选择结构：如果…就…语句；据…选择语句。

如果语句：

如果语句的用法很灵活，有如下关键字可以和它配合，就，不然，否则。接下来我用一段C语言代码和L语言代码进行对比，读者就会很容易懂了。

C语言：

if (a == 0){

printf(“%d\n”, a);

}

else{

++a;

}

L语言：

**如果**a**是**0**就**，说：a；（“说”是一个内置的功能函数调用，冒号“：”后面是参数传递）

**否则**（不然）a+1。

可以看出，L语言表达使人更容易理解，而且也很简洁。我们再来一段长的C语言if-else的表达式，看看，我们L语言是怎么表达的。

C语言：

if (a == 0){

printf(“%d\n”, a);

}

else if (a == 5){

--a;

}

else{

++a;

}

L语言：

**如果**a**是**0**就**，说：a；

**如果**a**是**5**就**，**--**a；

**否则**（不然也可以）a+1。

这里的如果子句，只要有一句结果是真值就不会执行后续语句，只有都不满足所有如果子句，才会执行否则子句。

“就”关键字后面跟的是子句，可以用多个逗号隔开，直至遇到“；”分号。这里为什么后面是分号呢，因为在这里，如果、如果、否则，构成了一个选择结构，对于系统来说，他们是一个段落一个整体。假如，我们这样写：

**如果**a**是**0**就**，说：a。

**如果**a**是**5**就**，**--**a。

**否则**（不然也可以）a+1。

用C语言表达就是：

if (a == 0){

printf(“%d\n”, a);

}

if (a == 5){

--a;

};

else{

++a;

}

注意第二个if后面的分号。显然这种写法是有语法错误的。

是的，L语言的写法也是有语法错误的，**否则**一定要和**如果**构成子句，否则是语法错误。

如果语句的嵌套：

if (a == 0){

a++;

if (a % 2 == 0){

printf(“%d\n”, a);

}

}

**如果**a**是**0**就**，

a++，

**如果**a%2**是**0**就**，

**说**：a。

if (a == 0){

if (a % 2 == 0){

printf(“%d\n”, a);

a++;

}

}

**如果**a**是**0**就**，

**如果**a%2**是**0**就**，

**说**：a，

a++。

选择语句：

如果语句只有两个分支选择（虽然你可以嵌套），而选择语句就提供了多条分支，理论上是无限制的分支。让我们来看看语法规则吧。

据a选择，

条件1：子句1；

条件2：子句2；

其它：子句N。

这里分号“；”代表和其他条件保持并列关系，即执行到这儿就会跳出选择语句。可以参考中文分号“；”的用法，假如你把分号“；”写成了句号“。”那么会打破这种并列关系，执行这个条件后会继续往后面执行。其它表示默认的情况下该执行的入口。我们依然会用C语言和L语言来作对比。

C语言：

switch(grade){

case ‘A’: printf(“85~100\n”);break;

case ‘B’: printf(“70~84\n”);break;

case ‘C’: printf(“60~69\n”);break;

case ‘D’: printf(“0~59\n”);break;

default: printf(“error grade!\n”);break;

}

L语言：

据grade选择，

‘A’：说：“85~100\n”；

‘B’：说：“70~84\n”；

‘C’：说：“60~69\n”；

‘D’：说：“0~59\n”；

其它：说：“error grade!\n”。

一定要记得是分号。其它子句后面是句号，以此来完成这个选择结构。当然在条件子句中你也可以使用“句号”，如果这样可以带来你想要的效果的话。

# 循环结构设计

C语言或者后续的语言，有while、do…while、for（变种）、for each循环，我也在思考着，“是否照着这些用汉语模仿一下，仔细一想，太low了”。所以，我制定了以下几种循环机制。

当且仅当...就，语句块。

这里我采用了数学在定义某一个东西的时候，常用的唯一条件用语，没办法，谁让我的高中班主任是数学老师呢。

C语言：

while(true){

//TODO:…

}

L语言：

当且仅当对就，

（TODO：…）。

C语言的for语句经常用于遍历，通过“遍历”这种“体感”，我发现可以这么定义遍历。

有时候，我们需要索引来告诉我们，现在的遍历到达的位置，所以：

# 利用数组处理批量数据（待定）

# 用函数（待定）实现模块化程序设计

# 结构化数据