VBS基础教学

2022-04-30 08:14　167阅读 · 1喜欢 · 0评论

代码猴Z科技
粉丝：95　文章：5

已关注

VBS(VBScript的进一步简写)是基于Visual Basic的脚本语言. Microsoft Visual Basic是微软公司出品的一套可视化编程工具, 语法基于Basic. 脚本语言, 就是不编译成二进制文件, 直接由宿主(host)解释源代码并执行, 简单点说就是你写的程序不需要编译成.exe, 而是直接给用户发送.vbs的源程序, 用户就能执行了

VBScript(Microsoft Visual Basic Script Editon).，微软公司可视化BASIC脚本版). 正如其字面所透露的信息,
VBS(VBScript的进一步简写)是基于Visual Basic的脚本语言. Microsoft Visual Basic是微软公司出品的一套可视化编程工具, 语法基于Basic. 脚本语言, 就是不编译成二进制文件, 直接由宿主(host)解释源代码并执行, 简单点说就是你写的程序不需要编译成.exe, 而是直接给用户发送.vbs的源程序, 用户就能执行了.

第一篇（共六篇）

我知道菜鸟现在最关心的就是用什么工具来开发VBS程序了, 答案是:记事本(Notepad)，我不是开玩笑, 其实任何一种文本编辑器都可以用来开发VBS开发, 只不过记事本是由系统自带的, 比较好找而已. 尽管如此, 我还是建议你去下载一个专业的文本编辑器, 因为这些工具可以提供 "语法高亮"等功能, 更加方便开发, 用哪一个随你喜好, 我比较喜欢Edit Plus ScriptCryptor也不错　OK, 我们先来写一个VBScript程序热热身。

复制代码代码如下:

```
REM 输入并回显你的名字
'使用InputBox和Msgbox函数
'（上面及本行可以不必写入源代码中，下面才是运行的代码）
Dim name,msg
msg="请输入你的名字:"
name=Inputbox(msg)
Msgbox name
```

把上面的程序清单输入到记事本里面, 然后保存为以.vbs为扩展名的文件（如果在文件名上没有“.***”，就打开控制面板-文件夹选项-查看-取消隐藏已知文件类型的扩展名).然后双击, 观察运行结果。

注意:为了使你学得更好，推荐自己输入程序清单, 尽量不要复制->粘贴

我来解释一下这个程序, 第一行和第二行的开头分别是"REM"语句和" ' ", 这两个东西的作用是相同的, 表示之后的文字是注释, 就是说符号后的什么也不干,只是用来说明这段程序的功能, 版权信息等等. 注释行是程序最重要的部分之一, 尽管它不是必需的, 但对于其他人阅读源代码, 以及自己分析源代码是很有好处的. 好的习惯是在必要的地方加上清晰, 简洁的注释。

Dim用来声明一个变量, 在VBS中, 变量类型并不是那么重要, 就是说VBS会帮你自动识别变量类型, 而且变量在使用前不一定要先声明, 程序会动态分配变量空间. 在VBS中你不用考虑name储存的是一个整数还是一个小数(学名叫"浮点数"), 也不用考虑是不是字符串(一串字符, 比如:"Hello World"), VBS会自动帮你搞定. 所以第三行语句可以删除, 效果不会变, 但我强烈反对这么做, 一个变量的基本原则就是:先声明,后使用.变量名用字母开头,可以使用下划线,数字, 但不能使用vbs已经定义的字, 比如dim, 也不能是纯数字。

下一行被称之为"赋值", "="是赋值符号, 并不是数学中的等于号, 尽管看起来一样.这是正统的理解, 你要理解成等于也没有什么不可. 赋值号的左边是一个变量, 右边是要赋给变量的值, 经过赋值以后, msg这个变量在程序中等同于"请输入你的名字:"这个字符串,但当msg被再次赋值的时候, 原值就会消失. 不光字符串, 其他任何变量都这样被赋值, 例如: a=2, b=12.222等等。

再往下,Inputbox和Msgbox是VBS内建的函数, 一个函数就相当于一个"黑箱", 有输入(参数)和输出(返回)值, 你可以不用了解函数是怎么运作的, 只要了解这个函数能干什么就行了, 我们也可以定义自己的函数, 不过那要等到以后再讲. 现在我们只要了解, 一个函数可以有返回值也可以没有, 可以有参数也可以没有. 例如Inputbox就是有返

回值的函数, 我们用赋值号左边的变量来"接"住InputBox的返回值--就是你输入的内容. 在inputbox右边的括号里是参数列表, 每个参数用","分隔开, 每个参数有不同的功效, 比如第一个参数会显示在提示里, 我们把msg这个变量作为第一个参数传给了Inputbox 函数, 而msg="请输入你的名字:", 所以我们在对话框的提示栏就会看到"请输入你的名字:" 第二个参数是对话框的标题, 我们用直接量(学名叫"常量", 这里是"字符串常量")传递给函数, 当然你也可以传递变量. Inputbox还有很多参数, 比如你在"名称"后面再加一个","然后输入随便一串字符(字符串,用双引号""包裹起来的一串字符叫做字符串)然后运行, 看看结果. 你会发现用于输入的文本框有了默认的值, 这就是第三个参数的作用。

Msgbox函数是用来输出的函数, 在VBS中没有专门的输出函数(BASIC中的print,C中的printf), 所以我们只能用对话框来观察输出结果, Msgbox的必要参数只有一个, 就是要输出的内容, 在这种情况下, 我们不需要理会msgbox的返回值. 关于Msgbox和Inputbox我们以后还会在讨论, 今天只是热热身, 到此为止。

要点:

1) 注释(以REM或'开头)行在程序中不起作用, 但能让别人更容易读懂你的程序.
2) 变量好像一个盒子, 或一个代号, 可以代表你想代表的东西. 变量赋值使用"="
3) 以""包裹起来的字符称之为"字符串"
4) 函数像一个"黑箱", 有参数和返回值, 用"="左边的变量可以接住返回值
5) Inputbox函数弹出一个输入对话框,Msgbox则用于输出

作业:

1) 试验Inputbox的第三个参数
2) 写一段程序输出你的年龄
3) 写一段程序进行3次输入, 分别输入你和你父母的姓名(要求显示提示), 并分3次输出

第二篇（共六篇）

今天讲各种"量"和基本运算

先说常量, 这个比较简单.

什么是常量呢, 常量就是其值不可变化的量.

常量分为两种:第一种, 自然常量. 这叫是因为它们本身就是常量, 你怎么更改21的值呢? 他永远都是21, 不可能变成46 。

如果你在程序中使用"21=46", 这样的语句将会引发一个错误. 同样的, 字符串也是常量 (还记得字符串吗? 就是包裹在""之间的一串字符), "Hello World"就是一个例子, 如果你用"Hello World"="Bye"这样的语句同样会引发一个错误. 你能举出自然常量的更多例子吗?

第二种, 是我们自己定义的常量,这种量也使用代号, 它们也被赋值, 但和变量的不同点在于, 他们在定义的时候被赋值,以后就不能改变了,如果企图改变将会引发一个错误. 定义一个常量,我们使用"const"这个关键字(关键字的意思是系统定义了有特殊功能的字, 不能作为变量名或常量名使用)

格式是:const 常量名=常量值.
例如:
const PI=3.1415962
const NAME="记忆碎片"

这样我们就定义了两个常量, PI和NAME, 一般说来, 常量名全部使用大写, 但也可以不用, 随你喜好. 将一些在程序中不需要改变的值定义为常量是个好习惯, 这样能防止不必要的意外. 另外, 使用自定义常量也可以减少你的工作量. 比如:

```
msgbox "Hello World"
msgbox "Hello World"
msgbox "Hello World"
msgbox "Hello World"
msgbox "Hello World"
```

这个程序输出五次Hello World, 如果你想要改变输出为Bye-Bye, 就必须修改全部程序, 当然你可以手动修改5次, 但如果你要输出1000次呢? 常量就可以替我们解决这个问题:

```
const hw="Hello World"
msgbox hw
msgbox hw
msgbox hw
msgbox hw
msgbox hw
```

这样当你要修改输出的时候只要修改hw的值就行了.

好了, 现在我们来看看编程的第一块重要"基石":变量. 解释变量最好的办法我觉得是"盒子", 一个变量好像一个盒子, 里面只能装一个东西, 当你要装进去别的东西的时候必须把原有的东西拿出来. 这个"盒子"是有名称的, 当你在程序中使用变量的时候, 系统会打开盒子取出里面的东西, 让这些东西参与处理, 而不是盒子. 有些语言是很依赖"盒子"里面装些什么东西, 这样才能找到合适的"盒子"(比如C语言), 但VBS给我提供的是能够自动伸缩的"魔术盒", 我们不用关心装进去的是什么东西, VBS会自动调整盒子的大小. 例如:

```
Dim a1,a2,a3
a1=14
a2=12.23
a3="Hello"
```

而不用像C语言那样麻烦: 或者是VB的正规声明(VB可以声明也可以不用)那样:

复制代码代码如下:

```
int a1; Dim a1 as integer
float a2; Dim a2 as Double
char* a3; Dim a3 as string
a1=14; a1=14
a2=12.23; a2=12.23
a3="Hello"; a3="Hello"
```

嗯……扯远了……

变量有什么用呢? 哇, 那用处可大了. 最简单, 你并不能确定程序运行时变量的值, 比如前一节课我们编的输入姓名的程序, 你不能确定InputBox返回的是什么(还记得Inputbox的返回值吗? 就是你输入的内容), 所以你就没有办法应对各种情况, 但我们用name这个"盒子"把用户的名字装起来, 到用的时候我们只要知道name这个盒子的名字就行了, 系统会自己打开它并把里面的内容拿出来用. 再举个例子, 我们编写一个程序计算一个矩形的面积, 比如这个程序要发给小学生使用:

复制代码代码如下:

```
dim a,b,s
a=15
b=12
s=a*b
```

```
msgbox s
```

这样就可以求出长为15,宽为12的矩形的面积了, 是不是很简单? 当然, 这个程序也可以这样写:

```
dim s
s=15*12
msgbox s
```

这样看起来程序短了许多, 也节约内存, 但两种都不是鼓励的做法, 为什么? 请看下面.
现在, 我们的程序要变得像点样子才行, 谁的程序写出来还要别人修改源代码才能用啊?
所以, 我们要接受用户的输入, 还记得吗? InputBox函数.
修改后程序如下:

复制代码代码如下:

```
dim a,b,s
a=inputbox("请输入矩形的长:")
b=inputbox("请输入矩形的宽:")
s=a*b
msgbox s
```

ok, 这么一修改, 无论用户输入怎样的数据, 我们都能计算出矩形的面积了. 如果你用s=15*12能改吗? 当然不行。

我想你已经发现了, vbs中的数学计算和真正的算术没有什么不同, +,-,*,/,(),[],{}都是一样的用法, 比如:

```
dim ans
ans=12+32/4+[(23-10)*2]
msgbox ans
```

四则运算的法则在编程中同样生效, 你可以在编程中重新获得小学时候的乐趣(你恨数学?那就别学电脑了)。

在编程中有一个有趣的运算符是"mod", 这个运算符叫做"取余运算符", 就是取得一次除法的余数, 例如

```
dim a
a=16 mod 5
```

你知道a等于几吗? Bingo! 没错, 就是1. 因为16 / 5 =3....1, 所以mod计算的结果就是1.
还有一个运算符是"^"(就是键盘"6"上面的小箭头), 他表示"乘幂"(或者是"方") 比如:

```
dim a,b,c
a=2
b=a^2
c=a^3
msgbox b
msgbox c
```

则b=a*a=4, c=a*a*a=8

好了，我们一次不要讲太多, 这次就讲到这里, 现在总结一下.

要点:

1) 常量分为自然常量和自定义常量, 常量的值不可修改
2) 变量就像盒子, 我们不在乎盒子里装的是什么, 但一定要知道盒子的名字

3) 四则运算在编程中没有任何不同
4) MOD是取余运算

作业:

1) 编一个程序, 计算圆形的面积, 半径由用户给出 (使用Inputbox) PI取值3.14159
2) 编一个程序取得20 / 3 的余数

第三篇（共六篇）

首先, 我来解决一下上次课程的几个疑问

第一, 那个余数问题, 16 / 5 = 3……1, 是因为我改过前面的部分, 后面的忘了改了, 不好意思。

第二, 请看一下程序清单:

（1）

复制代码代码如下:

```
Dim a,b,c
a=inputbox("a是:","输入半径")
b=Inputbox("b是:","输入半径")
c=a*2+b*2
msgbox c
```

这个 输入1、2时是6

（2）

复制代码代码如下:

```
Dim a,b,c
a=inputbox("a是:","输入半径")
b=Inputbox("b是:","输入半径")
c=(a+b)*2
msgbox c
```

这个输入1、2时是24

为什么会不一样呢? 在数学上c=(a+b)*2 和 c=a*2+b*2是等价的, 在VBS中也是如此. 问题出在"+"上, 在VBS中, +不仅仅是加号的意思还表示把两个字符串连接起来,例如"Hello"+"World"="HelloWorld" ，have you understood? 你还记得InoutBox函数的返回值吗? 是字符串! 这就看出问题了吧, 在编程中"1"不等于(<>)1, "1"是一个字符, 而1是一个数, 所以a,b都是字符串变量, "1"+"2"="12", 这就好像我们小时跟伙伴开玩笑问他们1+1=?一样, 我们总是笑着说"错啦,应该是11".但为什么, a可以*2却不发生错误呢? 这时VBS比较智能的一个表现, 如果这个字符串的内容是一个数且对他进行数学运算, 则把字符串强制转换成数参与运算, 如果字符串代表一个数, 但不参加数学运算, 而是参加字符串运算(合并)则当作字符串处理, 所以你看到a+b=12, 这时候a+b的结果(12)是一个字符串, 当它要乘以2的时候就被强制转换成了数字12, 这样我就得到了结果24。

怎么修改这个程序呢? 我们需要用到另一个内建的函数:int, int函数的功能是将输入值转化成整数值, 我们这样修改:

```
c=(int(a)+int(b))*2
```

这个意思就是把a作为参数传递给int函数, int函数就会返回那个整数(你的输入值), 然后让返回值参与运算, 这样就得到了正确答案.所以,以后如果你用的是inputbox函数的话,最好用int语句加工一下:比如c=int(c) 'c是你自己的变量

另外，还有一个函数：CDbl。如果你想把字符转换为数字，但又不取整时就用这个：

复制代码代码如下:

```
Dim a,b,c,d,e
a=inputbox("a是:","输入半径")
b=Inputbox("b是:","输入半径")
c=CDbl(a)
d=CDbl(b)
e=(c+d)*2
msgbox e
```

你输入1.2，1.3时就会输出5。

上面的实例也可以综合写成:

```
Dim a,b
a=CDbl(inputbox("a是:","输入半径"))
b=CDbl(Inputbox("b是:","输入半径"))
Msgbox (a+b)*2
```

这样输入1.2，1.3时依然就得出5。不过这样写比较适合学习过一段时间VBS的朋友.

大家是不是觉得这个课程有点枯燥？呵呵, 变量和运算符部分的确是这样的, 不过多多练习也就好了, 这次, 我们写写真正好玩的东西: 流程控制语句. 这个部分开始才是真正的编程.

首先介绍判断结构.

在此之前, 我们先介绍一种简单的变量类型:布尔值(Boolean), 这种变量只有两个可能值:True,Flase,即真或假. 这种变量在某些情况下很有用(比如"开关"). 我们定义一个Boolean变量的方法和其他变量一样, 赋值也一样, 例如:

复制代码代码如下:

```
dim a,b
a=true
b=false
```

注意,true和"true"（加双引号）是不一样的, "true"是字符串,true是布尔值, 千万不能混淆.

回到if语句上来, 我们先来看看简化版的if语句:if 判断式 then 语句体　我们来看一个例子:

复制代码代码如下:

```
dim a,b
a=12
b=13
if b>a then msgbox "B大于A"
```

我们只看最后一行, a>b这个式子(表达式)有一个返回值, 是Boolean型的. 因为这个式子只有两种可能:b大于a, b不大于a, 所以这个式子也只有两种可能性, 即真或者假. if语句判断这个表达式的返回值是真还是假, 如果是真(true)则执行then后面的语句, 如果是假, 则不执行, 你把a的值改成14看看还会不会弹出对话框?

当我们要在判断之后执行多行语句怎么办呢, 我们需要用语句块来解决, 在这里可以叫块if

复制代码代码如下:

```
dim a,b
a=12
b=13
if a<b then
msgbox "A小于B"
msgbox "B大于A"
end if
```

两个msgbox函数夹在if和end if之间, 这个部分就是语句块, 块里的每一条语句之前请空出4~8(一个<Tab>键)个格, 这不是必需的, 但是是一个好习惯, 以便看清楚程序的结构. 这样我们就能运行多于一个的语句, 请注意if...then...end if 这三个关键部分不要掉了. OK, 我出一个题, 输入一个数, 如果小于100就输出"错误", 如果大于100就输出"正确", 我这里有两个程序版本:

复制代码代码如下:

```
dim a
a=inputbox("请输入一个大于100的数")
a=int(a) 'inputbox返回的是字符串, 我们把他变成整数 : )
if a>100 then msgbox "正确"
if a<100 then msgbox "错误"
```

还有一个更简单的

复制代码代码如下:

```
dim a
a=inputbox("请输入一个大于100的数")
a=int(a) 'inputbox返回的是字符串, 我们把他变成整数
if a>100 then
msgbox "正确"
else
msgbox "错误"
end if
```

看到多了一个else了吧, else的作用就是当要判断的表达式为false时执行的. 这样程序就可以处理两种不同的情况了. 不要忘了用end if结尾

嘿嘿, 我是“变态者”, 现在我要你处理三种情况, <100,=100,>100, 还要写在一个if结构里, 你怎么办, 我给你答案:

复制代码代码如下:

```
dim a
a=inputbox("请输入一个大于100的数")
a=int(a) 'inputbox返回的是字符串, 我们把他变成整数
if a>100 then
msgbox "正确"
else if a=100 then
msgbox "老大, 你耍我?"
else
```

```
msgbox "错误"
end if
end if
```

这次输入100看看, 是什么? else if语句可以在if结构中多次出现, 以灵活判断不同的情况 (如果你要判断得太多, 就请使用"选择结构", 过会儿就讲), 当所有elseif都处理完了,而没有符合情况的时候再执行else中的语句.另一个例子:

复制代码代码如下:

```
Dim a,b,c,d
a=inputbox("a是:","输入半径")
b=Inputbox("b是:","输入半径")
d=Inputbox("答案:","输入答案")
c=a*2+b*2 '这里没有问题, 会自动转换
if d=c then
msgbox "你好聪明"
else
msgbox "你好猪头 自己的题还不会!"
end if
```

哈哈, 无论你回答得多么正确你都是猪头, 不是我耍你, 还是文章开始时候的inputbox的返回类型在耍你, d是inputbox的返回值, 他是一个字符串, 而c是一次整数计算的结果, 他是一个整数. 一个字符串无论如何也不等于一个整数, 尽管他们字面上是一样的:"8"<>(不等于号)8 所以if的判断式的值永远是false, 总是执行else部分的语句. 我们可以这么修改

复制代码代码如下:

```
Dim a,b,c,d
a=inputbox("a是:","输入半径")
b=Inputbox("b是:","输入半径")
d=Inputbox("答案:","输入答案")
d=int(d)

'在这里我们取出了d的值, 变成整数, 再放回"d"这个盒子里

c=a*2+b*2
if d=c then
msgbox "你好聪明"
else
msgbox "你好猪头 自己的题还不会!"
end if
```

这样就成功了. 这也是Inputbox函数的一个讨厌的地方, 没办法, vbs没有其他好的输入方式了.

说到if, 我们不得不说一说逻辑运算符, 今天介绍两种, "and" 和 "or" 学会了if语句之后, 我举一个例子, 你一看就明白了.

复制代码代码如下:

```
dim a,b
a=inputbox("输入一个数 >10")
b=inputbox("输入另一个数 >10")
```

```
a=int(a)
b=int(b)
if a>10 and b>10 then
msgbox "正确"
else
msgbox "错误"
end if
```

这段程序让你输入两个值, 必须都大于10, 只要有一个不大于, 就输出错误

复制代码代码如下:

```
dim a,b
a=inputbox("输入一个数 >10")
b=inputbox("输入另一个数 >10")
a=int(a)
b=int(b)
if a>10 or b>10 then
msgbox "正确"
else
msgbox "错误"
end if
```

这段程序让你输入两个值, 只要有一个大于10, 就返回成功. 其实and和or很好理解, 我读"if a>10 or b>10 then" 这一句, 用华语是这样:"如果a大于10或者b大于10, 那么...". 这样是不是就很好理解了呢.

OK, 我们再来看一种新结构, , 学完这个 , 今天的课就结束, 已经午夜了, 我都累死了.

当你的程序要处理很多种不同的判断情况的时候elseif..then会让程序看起来很杂乱, 所以就有了一种select case结构专门对付这种情况, select case的语法结构很简单:

```
select case 变量名
case 值
语句
case 值
语句
case else
语句
end select
```

我们举个例子就能很简单的说明：

复制代码代码如下:

```
dim a
a=inputbox("输入一个1--3的值")
a=int(a) '处理inputbox返回字符串的问题
select case a
case 1
msgbox "壹"
case 2
msgbox "贰"
case 3
msgbox "叁"
case else
```

```
msgbox "输入错误"
end select
```

这个例子把1,2,3这三个阿拉伯数字转化成中国大写数字, 这个程序写成if...elseif 的形式如下

```
dim a
a=inputbox("请输入1--3的值")
a=int(a)
if a=1 then
msgbox "壹"
elseif a=2 then
msgbox "贰"
elseif a=3 then
msgbox "叁"
else
msgbox "输入错误"
end if
```

怎么样, 麻烦吧, 还是select好吧.
OK, 今天到此结束, 总结一下:

要点:

1) inputbox返回的是一个字符串, 而不是一个数, 必须用a=int(a)这种形式转化成数
2) bool变量的值只有两种:true,false
2.5) and两边的表达式都是true, 则返回true. or两边的表达式有一个是true, 就返回true
3) if 语句的格式
4) select...case的格式

作业:

1) 使用3个bool值, 储存你的3兄弟姐妹是否是男性 (提示:sister1male=false)
2) 给定一个个数, 大于10而且小于20输出"正确", 否则输出"错误"
3) 输入12,或者15, 输出"正确", 否则输出"错误"
4) 把5以内的正整数都转换成中国大些数字
5) 自己随便设计一个程序, 应用今天的知识

第四篇（共六篇）

大家好,今天写第4章：循环结构

我们先来看一道题:商场进行每日结算, 要求累加出今天的营业额, 每次输入一个数, 这道题其实很简单, 但就我们现在学过的知识要完成这道题相当麻烦, 我们来分析一下. 首先, 我们需要知道买卖的次数, 这样才能控制输入的次数,但是, 这种设计是非常低效的, 每天都要重新设计程序. 假定今天进行了5次交易, 以下是源程序:

复制代码代码如下:

```
dim sum
sum=0 '初始化变量
sum=sum + int(inputbox("请输入交易额"))
'sum=sum+x    这种形式是把本身的值取出来, 进行一次运算, 再放回本身, 这种方法很有用处
'这里使用了函数嵌套, 把inputbox的返回值直接传给int函数, 转化成整数, 下同
sum=sum + int(inputbox("请输入交易额"))
sum=sum + int(inputbox("请输入交易额"))
sum=sum + int(inputbox("请输入交易额"))
```

```
sum=sum + int(inputbox("请输入交易额"))
msgbox sum
```

看到了吗, 我通过把计算过程复制了5遍才设计好了程序, 这种程序在汽车交易所等交易次数少的地方还能凑合着用, 如果放到超市岂不是要复制, 粘贴几千遍? 我们今天讲的内容就可以克服这种缺陷, 首先, 我们来讲以下Do...Loop语句.

do...loop的结构看上去非常简单, 就是:do...loop, 仅此而已, 这个结构不断执行do和loop之间的语句(学名叫:循环体), 永不停止. 举个例子来说:

```
do
msgbox "这个信息会不断重复出现, 要停止程序请使用任务管理器(Ctrl+Alt+Del)中止wscript进程"
loop
```

运行这个程序, 当你点销掉一个对话框马上会出来另一个, 你永远点不完, 总有下一个. 谁会运行这样的程序? 除非是给别人捣乱(我就干过这种事), 所以在do..loop结构中还有一个语句:exit do, 这个语句将终止循环, 跳到loop后面的语句继续执行.举个例子来说:

```
dim a '注意:常量不需要在dim里面声明,否则会引发错误
const pass="123456" '这是一个字符串 请用""包裹起来. 设定密码为常量, 不可变更
do
a=inputbox("请输入密码")
if a=pass then
msgbox "密码校验成功"
exit do
end if
loop
```

这个程序会一直不停的问你密码, 知道你输入了正确的密码为止.(if可以嵌套在另一个if当中, 也可以嵌套在循环体当中, 所以一定要用缩进, 来分清楚程序的各个部分). 这个程序是很经典的, 早期的程序都是这么做的. 但是我们是Hacker, 所以我们了解系统的安全性, 这种无限次认证程序很容易被穷举破解, 我们要来限定认证的次数. 修改程序如下

```
dim a,ctr
ctr=0 '设置计数器
const pass="pas123_" '上面的那个是弱密码, 这次改的强一点
do
a=inputbox("请输入密码")
if a=pass then
msgbox "认证成功"
exit do
else
if ctr=3 then
msgbox "已经达到认证上限, 认证程序关闭"
exit do
else
ctr=ctr+1 注意：这一句是赋值句，要从右往左读，即每出错一次就把ctr加上1，然后再放回ctr里面，使得这个常量加1

msgbox "认证出错, 请检查密码"
end if
end if
loop
```

运行这个程序试试看, 当你出了3次错误以后, 就会停止再次询问密码, 关闭程序. telnet认证用来限制次数的程序与此大同小异. 要注意的是嵌套的if语句, 请仔细读一下这个程序, 可能比较难懂, 也请你试着自己设计一下类似

的程序.

其实, 要在do...loop加上验证的功能, 并不一定要用if, 我们可以直接利用do. 我来介绍一下while关键字, while可以放在do或者是loop后面, 然后再接一个表达式, 当表达式的值为true的时候(表达式成立),才运行循环体.我们来看一下修改后的程序"

复制代码代码如下:

```
dim a,ctr
ctr=0
const pass="pas123_"
do while ctr<3
a=inputbox("请输入密码")
if a=pass then
msgbox "认证成功"
msgbox "(你可以在这里加一段成功后得到的信息)"
exit do
else
ctr=ctr+1 '如果密码出错就增加一次错误认证计数
msgbox "认证出错, 请检查密码"
end if
loop
```

这样实现的功能和上一个例子完全一样, 我们再来看看把while放在loop后面:

复制代码代码如下:

```
dim a,ctr
ctr=0
const pass="pas123_"
do
a=inputbox("请输入密码")
if a=pass then
msgbox "认证成功"
msgbox "(你可以在这里加一段成功后得到的信息)"
exit do
else
ctr=ctr+1 '如果密码出错就增加一次错误认证计数
msgbox "认证出错, 请检查密码"
end if
loop while ctr<3
```

功能是一样的, 为什么要放在loop后面呢? 你把ctr的值改成3就知道了, while在do后面的程序会直接退出, 而在loop后面还会允许一次认证, 到了loop才结束.和while相反的是until, 用法和while一样, 不过他只有当后面的表达式的值为false(表达式不成立)的时候才执行循环体, 请自己试验一下

ok, 我们来看另外一种循环结构,for....next, 这种循环结构是基于计数的, 也是在编程中最常见到的循环结构.

复制代码代码如下:

```
dim i
for i=0 to 5
msgbox i
next
```

看到了吗?每次输出的i都是递增的, 但我们没有明确指出i要递增, 当i达到5的时候, 循环就结束了, 因为由0开始, 所以循环体执行了6次, 这一点很重要, 大部分东西都是从0开始而不是1. 这个程序也可以写成

do的形式:

复制代码代码如下:

```
dim i
i=0
do while i<5
msgbox i
i=i+1 '因为do不能自动计数, 必须手动加
loop
```

怎么样, 还是for比较好用吧. for在编程中很有用途, 我们再举一个例子, 顺便讲一下嵌套循环.

复制代码代码如下:

```
dim i,j
for i=1 to 9
for j=1 to 9
str=str & i * j & " " '&是和并字符串的符号
next '每个next对应一个for
next
msgbox str
```

(这样出现的结果是一次性的，如果你需要依次出现把msgbox str)提前到next之前
看看运行结果, 是否令你会想起小学时代的数学老师(丑陋的嘴脸).

要注意, 这里有一个"大"的for, 和一个小的for, 当小的for执行完一个周期以后, 大的for才执行一次(换句话说, 大的for执行一次, 小的要执行9次), 所以一共执行了九九八十一次. 在大的for里可以不仅仅是一个小的for, 也可以加上另外的语句. 我们来修改一下源程序:

复制代码代码如下:

```
dim i,j
for i=1 to 9
for j=1 to 9
str=str & i * j & " "
next '每个next对应一个for
str=str & vbCrlf 'vbCrlf相当于键盘上的回车键,因为你不能在键盘上输入,所以系统定义了一个默认的常量
next
msgbox str
```

这次运行完成以后, 输出结果按照乘数进行了分割, 每小for运行完一次, 就换一行(通过vbcrlf).

这次的内容对菜鸟可能比较难懂, 掌握的办法只有一个:多实践. 另外, 我在论坛看到很多人还问:"VBScript要用什么工具编?"我就很气愤, 我在第一篇里面已经说明:用记事本编辑源代码, 然后保存为以.vbs为扩展名的程序就可以了, 请大家不要用其他工具编写,否则很容易引起错误.

另外, 国产的一种垃圾软件"超级X霸"抢占了vbs这个扩展名, 请把那个垃圾卸载掉.
我们来总结一下:

要点:

1) do..loop和exit do的用法
2) while当表达式true的时候执行循环体,until反之
3) for...next是计数循环, 每次执行计数器递加
4) 嵌套循环的作用和写法
4.5) &用于连接字符串
5) vbCrLf相当于键盘上的回车键

作业:

1) 在我国的数学经典著作"九章算术"中有这样一道题:百钱买百鸡, 公鸡5钱一只, 母鸡3钱一只, 小鸡1钱3只)求得是能有多少种办法买这些鸡. 如果看不懂的话我用大白话说说:有人要去买鸡, 用100块钱正好买了100只鸡, 价格如下:公:5$, 母:3$, 小:1$ for 3, 让你求一共多少种卖法(公母小怎么搭配). 请用循环解决这个问题.

第五篇（共六篇）

今天我们来了解语言本身的最后一个论题:数组.

要理解"数组", 这个概念我觉得另一种翻译对学习来说更加容易:"阵列", 没错, 数组就是一个阵列, 一个数据的阵列. 最简单的例子是数据库系统, 假设你要储存20名学生的英语成绩, 如果不是用数组, 你则要创建20个不同的变量, 累死. 数组就是类型相同(重要!)的一组数据(或者n组), 用来储存相关的量, 最简单的数组是一维数组, 我们就先来学习它吧.

什么是一维数组呢? 在3维以下,你可以利用几何知识来理解"维"的概念,一维相当于一条线, 二维则是一个矩形, 三维是一个长方体. 我知道这么讲是很抽象的, 我们先举个一维数组的例子就比较容易了解了.

```
dim a(9) '从零开始
for i=0 to 9
a(i)=i '填充每一个数组元素
msgbox a(i) '输出数组元素
next
```

我们可以看到, 定义一个数组的方法和定义一个变量没有什么不同, 同样是使用dim语句. 定义一维数组的方法如下:

dim 数组名(元素数量), 这里大家要注意一点, 这里定义的元素数量总是比你要的要少一个, 因为一个数组的起点是0号数据而不是1, 所以大家一定要小心: 你需要10个数据, 就定义"9", 需要100个就定义99, 依此类推. 数组的元素可以看成一个个独立的变量, 你可以像独立的变量那样使用他们. 数组元素的量可能是毫无关系的, 比如第一个数组元素储存你的年龄, 第二个储存今年西瓜的销售量, 但这种做法是不鼓励的, 甚至是不被接受的, 不要这么干, 这样的情况请定义独立的变量. for语句在数组中可算是大显身手, 还记得for吗? 它累加一个变量, 我们可以把这个变量应用在数组中正好用来读取或者填充按照顺序排列的数组元素, 上面就是这样一个例子. 数组其实是很简单的东西(再BASIC语言里面), 数组难的是怎么捣弄这些循环, 让他们按照你的要求运转. 这个等到二维数组再说, 我们先看看如何手工填充数组.

如果你这个都想不到的话, 那你真是白学了:

复制代码代码如下:

```
dim name(7),str '一共八个学生, str变量是用来把他们储存成一个字符串以便输出
for i=0 to 7
name(i)=inputbox("请输入第" & i+1 & "个学生的名字")
str=str & " " & name(i)
next
```

```
msgbox str
```

这样我们就有了一个小小的数据库, 它们的数据排列可以看成这样:
name(0),name(1),name(2).....name(7)

看到了吧, 所以我说我们可以把它看成是"一条线", 等到我们学到了文件操作, 就可以把他们输出到文件中去了. 一维数组有很多用处, 我们来看一下一个复杂的例子. 我们要储存3各学生的名字, 身高, 成绩这三种数据, 由于名字是字符串, 而身高可能是浮点数(带小数点的数), 成绩则可能是整数, 所以我们不能把他们储存在一个数组里面(不要忘记, 数组织只能存储同类的数据), 所以我们要建3个数组, 以下是例程:

复制代码代码如下:

```
dim name(2), high(2), mark(2) '定义三个数组分别储存3个人的名字, 身高和得分
dim ctr '计数器
for ctr=0 to 2
name(ctr)=inputbox("请输入第" & ctr+1 & "个学生的姓名")
high(ctr)=inputbox("请输入第" & ctr+1 & "个学生的身高")
mark(ctr)=inputbox("请输入第" & ctr+1 & "个学生的得分")
next
```

OK, 我们已经填充好了数据, 现在我们的小小数据库只能按顺序输入, 我们要让它看起来像点样子, 我们来给他设计查询功能:

复制代码代码如下:

```
'接着上面的程序
dim cname, temp '要查询的名字, 和一个临时变量, 用来储存数据的位置
cname=inputbox("请输入你要查询的名字:")
for ctr=0 to 2 '遍历所有name数组的成员, 寻找要查询的名字
if name(ctr)=cname then
temp=ctr '记录数据位置
exit for '退出循环, 和exit do的用法一样
end if '不要忘了end if
next
msgbox "姓名:" & name(temp) & " " & "身高:" & high(temp) & " " & "得分:" & mark(temp)
```

嘿嘿, 有意思吧, 其实在这个程序里面, 那个temp变量完全没有必要, 只是为了更清楚地说明问题. 因为当exit for以后ctr变量的值就不会改变, 储存的正好是对应数据在数组中的位置, 写这个temp变量是为了照顾到以后要学C++的朋友(C++可以在for语句里声明新变量, 只在这个for结构中有效, 所以到了外部就不能访问了). 也就是说可以简化成如下:

复制代码代码如下:

```
dim cname
cname=inputbox("请输入你要查询的名字:")
for ctr=0 to 2
if name(ctr)=cname then exit for '因为只有exit for就不需要块if了
next
msgbox "姓名:" & name(ctr) & " " & "身高:" & high(ctr) & " " & "得分:" & mark(ctr)
```

这是最直接的路子. 好好重读一下上面所有的源代码, 然后自己写几个程序, 完全搞清楚一维数组及其应用以后再看后面的二位数组. 二维数组好像是一个一维数组的集合, 就好像"线积成面"一样, 由n各一维数组组成二维数组, 这是初学者比较好理解的办法(比较精确的是用"编号"的概念去理解, 因为4维以上的数组就比较难以用欧几里

德几何概念去解释了). 二维数组是很好解释的, 我们来看一下:

复制代码代码如下:

```
dim a(2,2) '从零开始, 一共有3 X 3 = 9 个数据
dim i,j '需要两个计数器
for i=0 to 2
for j=0 to 2 '使用嵌套循环
a(i,j)="X"
next
next
```

我们创建了一个这样的二维数组(那些","是我用来分割元素的,并不存在), 了解二维数组的了吗? 不清楚我们再讲一下

```
编号 0 1 2
0 X,X,X
1 X,X,X
2 X,X,X
```

二维数组的看起来是不是就是一个矩形呢?(你在内存中看不到这样的矩形, 只是便于你理解), 每个数据都有编号, 由两个数来定位, 这个很像(非常像)你在国际象棋棋盘上寻找一个格, 我们用类似"C6","A2"这样的"垂直坐标"进行定位, 对, "垂直坐标", 很确切. 我们要使用一个二维数组元素的时候可以和普通变量一样使用, 只要指定数组元素的"定位点"就可以了, 例如a(0)(1)=1, b(2)(1)="你好", 诸如此类. 要注意的是二维数组也只能储存类型相同的元素, 而且上标(起点)也从0开始. 计算一个二维数组的元素个数只要把两个下标+1(以得到实际的值)再乘起来就可以了, 非常类似于求一个矩形的面积.

假如需要储存的都是同一类型的数据, 我们就可以用二维数组, 比如要储存5个人的姓名,国籍,民族, 就可以使用二维数据

复制代码代码如下:

```
dim info(4,2) '一共五个人, 要储存的数据类型有3项
dim i,j
for i=0 to 4
for j=0 to 2
dim opt '定义一个变量用于存储数据项提示
select case j '判断应该输入的是什么数据
case 0
opt="姓名"
case 1
opt="国籍"
case 2
opt="民族"
end select
info(i,j)=inputbox("请输入第" & i+1 & "个人的" & opt)
next
next
'输出太麻烦了, 我懒得动, 你知道那么回事就行了
```

这样就不需要定义3个一维数组了.

多位数组(三维以上)的定义和使用方法与二维数组一样, 但不太好在欧几里德几何空间里加以解释, 幸好我们并不太常用那么多维的数组.定义一个三位数组:dim a(1,2,3) '一共24各数组元素.

要点:

1) 一维数组是"线", 二维数组是"面", 三维数组是"体" (多维数组就乱套)
2) 数组的下标从0开始
3) for循环在数组的应用中起了很大作用, 二维数组需要嵌套循环

作业:

上次出的"百鸡问题", 大家喜欢吗? 以后我们就做这种需要动脑的题目, 那些简单的实践, 大家一定要多做!
1) 定义一个数组, 包含5个元素, 都是随机整数(随便输入), 要求把他们按照从大到小的顺序排列起来
2) 有两个二维数组a(4,4)和b(4,4) (元素值随便), 交换两个数组(原来的a的所有元素值变成b的, b的所有元素值变成a的)

第六篇（最后一篇）

今天我们学习基础篇的最后一个部分：自定义函数和过程. 我们每天都在和函数打交道, inputbox()是函数, msgbox)是函数, int(也是函数...这些函数都是系统内建的, 我们只能用不能改. 今天, 我就教大家怎样自己制作一个函数.

首先我们要了解, 为什么要用函数, 我们用"实例"说话, 先看一个例子: 给出两个数, 输出较大的那一个.

复制代码代码如下:

```
dim a1,a2,b1,b2,c1,c2
a1=2:a2=4 '":"可以让你把多个语句写在一行上
b1=32:b2=67
c1=12:c2=898
if a1>a2 then
msgbox a1
elseif a1<a2 then
msgbox a2
end if
if b1>b2 then
msgbox b1
elseif b1<b2 then
msgbox b2
end if
if c1>c2 then
msgbox c1
elseif c1<c2 then
msgbox c2
end if
```

多么麻烦呀, 我们把相同的比较过程复制了好几遍, 早期语言没有结构化(没有过程和函数)的时候, 程序员们的确是这么干的, 那个年代也没有剪贴板这一说, 大家都是重新输入代码. 后来工作简化了:

复制代码代码如下:

```
dim a1,a2,b1,b2,c1,c2
a1=2:a2=4
b1=32:b2=67
c1=12:c2=898
msgbox co(a1,a2)
```

```
msgbox co(b1,b2)
msgbox co(c1,c2)
function co(t1,t2) '我们使用function定义了一个新的函数
if t1>t2 then
co=t1 '通过"函数名=表达式"这种方法返回结果
elseif t2>t1 then
co=t2
end if
end function
```

我们在这里是用了一个新的关键字:funciton, 这个关键字表示一个新函数开始, 格式:

```
function 函数名(参数1, 参数2...参数n) '列表可以是空的, 但括号不能省略, 参数之间用","分割
...
exit function '结束函数, 不是必需的
...
end function
```

函数是一个模块, 只有你调用的时候才会运行, 也就说, 当你编写了一个函数, 然后在程序中并不调用它, 那么这个函数永远不会运行. 一般来说, 我们编写程序是按照:

```
主程序
......
......
......
函数1
......
......
函数2
......
......
```

详细解释一下: 函数中最重要的是参数和返回值. 参数是在函数名后面的()里定义的, 用","分割, 使用参数的时候我们也用","分割. 说到这里我想起一件事, 昨天有个朋友给我发消息问我:

```
msgbox name1,name2,name3
```

这个错在哪里? 为什么不能同时显示出三个变量? 这就是因为你用了",", 这个符号表示你输入的三个量作为三个不同参数传递给msgbox)函数, msgbox()函数只会显示出第一个参数, 第二个参数的作用是出现在标题栏. 所以你应该用"&"或者"+"把三个字符串变量连接起来, 作为第一个参数传递给msgbox()函数. 程序员说参数的时候经常说到"形参", "实参"这样的"黑话", 我来解释一下. "形参"是"形式参数"的简称, "实参"是"实际参数"的简称, 实参是指你调用函数的时候传递给函数的量, 可以使变量或者常量(直接量), 例如:co(12,24)中的12,24就是实参. 形参是你在函数定义时定义的变量, 这些变量用来"接住"传递过来的量, 例如function co(t1,t2t1,t2就是形参.

在VBScript中, 参数传递是一种传值, 而不是传址(听不明白不要紧, 学了C语言的指针你就明白了), 所以我们进行的参数传递实际上是进行了一次变量赋值, 例如我们调用co(a1,a2), 实际上程序会执行一步:t1=a1,t2=a2这样的操作. 同样因为传值传址的原因, VBScript只能返回一个值, 我们先来看看什么叫"返回". 当一个过程调用了另一个过程的时候(比如主程序调用了函数), 控制权就到了被调用过程那里, 当这个过程执行完毕以后, 会回到调用它的地方继续执行, 这个就叫做"返回", 返回的时候可以带一个值叫做"返回值"(这是"通俗"的理解). 在vbs继承了basic的传统, 返回的时候采用"函数名=返回值"的办法, 这个"返回值"是指一个表达式(在编程中, 任何东西都是表达式, 比如变量a, 常数0, "Hello",c=1+2等等这都是表达式). 比如

有一个函数是ht, 则返回的方法是:ht=你要返回的值. 注意:返回以后, 后面的语句将不再执行.

调用一个函数我就不用讲了吧:变量=函数名(参数)

有时候我们并不需要返回什么值, 这个时候我们可以使用一种称之为"子程序"的结构. 子程序或称之为过程与函数的差别

就在于:1) 没有返回值, 2) 使用sub关键字定义, 3) 通过Call调用.举个例子:

复制代码代码如下:

```
dim yname
yname=inputbox("请输入你的名字:")
call who(yname)
sub who(cname)
msgbox "你好" & cname
msgbox "感谢你阅读我的课程"
msgbox "这是基础部分的最后一课"
end sub
```

你一定看明白了, 很简单的. 退出一个过程和退出一个函数一样:exit sub(函数:exit function).

要注意, 子程序(过程)是比较特殊的一个结构, C等语言是没有这个概念的, C语言中的一切都是函数, 没有返回值的函数

在C语言中只要使用void修饰符定义就行了.

今天没有什么可讲的了, 基础篇就这么结束了, 目前你已经有了基本的编程概念(面向过程的结构化编程), 可以选择学习另外一种语言(比如C或Pascal), 现在的基础会有一定的帮助. 如果你想要继续学习vbs或通过它更详细的了解编程在转型可以跟我继续学习, 但因为我的假期结束所以更新的时间可能会比较慢, 请大家原谅．初步计划如下:

进阶篇:

变量的深入讨论
├ 变量类型
├ 变量的有效范围
├ 数组的深入讨论
├ 动态数组
├ 函数的深入讨论
├ 数组作为函数参数
├ 多个返回值
├ 字符串操作
├ 其他
├ 面向对象编程(OOP)的基本知识
├ 文件操作
├ FSO对象
├ 其他相关部分
├ VBS与网页
├ HTML中嵌入VBS
┗ VBS与表单(设计你的程序界面 wow!)

实战篇

├ 病毒编程
┗ Socket编程(TCP/UDP)

这只是大概的内容, 我想应该会有变化, 到时候你就会读到的. 今天的内容请多多实践, 作业就是把前面的各个

课程在温习一下. 对要离开这个课程去更进一步学习的朋友:祝你再学习编程的道路上一帆风顺.
编辑本段vbScript常用运算符与函数

基本运算
+ 数字加法及字符串连接
- 数字减法
* 数字乘法
/ 数字除法
Mod 求余数
\ 求商数
& 字符串连接
^ 次方
= 相等
<> 不相等
>= 大于或等于
> 大于
<= 小于或等于
< 小于
Not 非
And 且
Or 或
Xor 异或
循环及决策
if ....then 若...则...
if ...then...else 若...则...非
else if... 非若
select case... 群组选择条件
end select
for ... next 计数循环
while...wend 条件循环(一)
do while...loop 条件循环(二)
do...loop while 条件循环(三)
do until...loop 条件循环(四)
do...loop until 条件循环(五)

数学函数

abs绝对值
Sgn 正负号
Hex 转换成十六进制
Oct 转换成八进制
Sqr 平方根
Int 取整数
Fix 取整数
Round 取整数
Log 以e为底的对数
Sin 正弦函数
Cos 余弦函数
Tan 正切函数
字符串处理函数
IsNull 判断对象是否为空
Len 字符串长度
Mid 取部分字符串
Left 从字符串开头取部分字符串
Right 从字符串结尾取部分字符串

Lcase 转换成小写
Ucase 转换成大写
Trim 清除字符串开头及结尾的空格符
Ltrim 清除字符串开头空格符
Rtrim 清除字符串结尾空格符
Replace 替换字符串部分字符
Instr 判断是否包含于另一个字符串(从起始搜寻)
InstrRev 判断是否包含于另一个字符串(从结尾搜寻)
Space 任意字符数的空格符
String 任意字符数的任一字符
StrReverse 反转字符串
Split 以某字符分割字符串

数据类型转换函数

Cint 转换成整形
Cstr 转换成字符串
Clng 转换成长整数
Cbool 转换成布尔函数
Cdate 转换成日期函数
CSng 转换成单精度
CDbl 转换成双精度

日期时间函数

Date 现在日期
Time 现在时间
NOw 现在日期时间
DateAdd 增加日期
DateDiff 两日期差
DateSerial 日期设定
Datevalue 日期设定
Year 现在年份
Month 现在月份
Day 现在天
Hour 现在时刻
Minute 现在分钟
Second 现在秒钟
Timer 午夜距现在秒数
TimeSerial 时间设定
Timevalue 时间所属部分
WeekDay 星期名称
MonthName 月份名称

其它函数

Array 产生数组
Asc 字符ASCII码
Chr ASCII码字符
Filter 过滤数组
InputBox 输入窗口
Join 合并数组中的元素
MsgBox 信息窗口
Lbound 数组下界

Ubound 数组上届

指令

Const 设定常数
Dim 定义变量或者数组
Erase 清除数组
ReDim 重新声明数组
Randomize 起始随机数
Rnd 取得随机数
ASP对象
Session对象
IsEmpty 测试Session变量是否存在
TimeOut 设定Session变量生存周期
Abandon 强制清除Session变量
Application对象
IsEmpty 测试Application变量是否存在
Lock 锁定Application变量
Unlock 解除Lock指令的锁定
Cookies对象
Expires 设定Cookies变量的生存周期
Connection对象
Open 打开与数据库的连接
Execute 打开Recordset对象
Close 关闭Connection对象
Recordset对象
movefirst 将记录指针移至第一条
movelast 将记录指针移至最后一条
movenext 将记录指针移至下一条
moveprevious 将记录指针移至上一条
bof 测试是否为recordset的起始
eof 测试是否为recordset的结束
open 打开Recoreset对象
close 关闭recordset对象
fields 读取数据的子对象
fileds.count 字段个数
pagesize 每页记录条数
absolutepage 设定为某页
pagecount 总页数
Absoluteposition 直接跳至某条记录