计 算 机 语 言

计 算 机 语 言
资料格式：未知
资料大小：0 Bytes
上 传 者：哥怕谁
下载次数：1877
上传时间：2010-7-19 9:47:44

说明：计 算 机 语 言
关键词： 计算机语言 计算机知识

计 算 机 语 言

计算机语言（Computer Lnguage）指用于人与计算机之间通讯的语言。计算机语言是人与计算机之间传递信息的媒介。
计算机程序设计语言的发展，经历了从机器语言、汇编语言到高级语言的历程。
计算机语言主要分为三类：

哥怕谁的资料（23）
UML用户指南(第2版)9
上传者：哥怕谁
上传时间：07/19/2010
UML用户指南(第2版)8
上传者：哥怕谁
上传时间：07/19/2010
UML用户指南(第2版)7
上传者：哥怕谁
上传时间：07/19/2010
UML用户指南(第2版)6
上传者：哥怕谁
上传时间：07/19/2010

- 低级语言
- 高级语言
- 专用语言

1、低级语言

- 机器语言、汇编语言和符号语言。
- 汇编语言源程序必须经过汇编，生成目标文件，然后执行。

2、高级语言

- BASIC（True basic、Qbasic、Virtual Basic）、C、PASCAL、FORTRAN、智能化语言（LISP、Prolog）等等。
- 高级语言源程序可以用解释、编译两种方式执行。通常用后一种。

我们使用的C语言就是使用的后者。

3、专用语言

CAD系统中的绘图语言和DBMS的数据库查询语言。

1.1.机器语言

机器语言是指一台计算机全部的指令集合

电子计算机所使用的是由"0"和"1"组成的二进制数，二进制是计算机的语言的基础。计算机发明之初，人们只能降贵纡尊，用计算机的语言去命令计算机干这干那，一句话，就是写出一串串由"0"和"1"组成的指令序列交由计算机执行，这种计算机能够认识的语言，就是机器语言。使用机器语言是十分痛苦的，特别是在程序有错需要修改时，更是如此。

因此程序就是一个个的二进制文件。一条机器语言成为一条指令。指令是不可分割的最小功能单元。而且，由于每台计算机的指令系统往往各不相同，所以，在一台计算机上执行的程序，要想在另一台计算机上执行，必须另编程序，造成了重复工作。但由于使用的是针对特定型号计算机的语言，故而运算效率是所有语言中最高的。机器语言，是第一代计算机语言。

1.2.汇编语言

为了减轻使用机器语言编程的痛苦，人们进行了一种有益的改进：用一些简洁的英文字母、符号串来替代一个特定的指令的二进制串，比如，用"ADD"代表加法，"MOV"代表数据传递等等，这样一来，人们很容易读懂并理解程序在干什么，纠错及维护都变得方便了，这种程序设计语言就称为汇编语言，即第二代计算机语言。然而计算机是不认识这些符号的，这就需要一个专门的程序，专门负责将这些符号翻译成二进制数的机器语言，这种翻译程序被称为汇编程序。

汇编语言同样十分依赖于机器硬件，移植性不好，但效率仍十分高，针对计算机特定硬件而编制的汇编语言程序，能准确发挥计算机硬件的功能和特长，程序精炼而质量高，所以至今仍是一种常用而强有力的软件开发工具。

1.3.高级语言

1.3.1.高级语言的发展

从最初与计算机交流的痛苦经历中，人们意识到，应该设计一种这样的语言，这种语言接近于数学语言或人的自然语言，同时又不依赖于计算机硬件，编出的程序能在所有机器上通用。经过努力，1954年，第一个完全脱离机器硬件的高级语言--FORTRAN问世了，40 多年来，共有几百种高级语言出现，有重要意义的有几十种，影响较大、使用较普遍的有FORTRAN、ALGOL、COBOL、BASIC、LISP、SNOBOL、PL/1、Pascal、C、PROLOG、Ada、C++、VC、VB、Delphi、JAVA等。特别要提到的：在C语言诞生以前，系统软件主要是用汇编语言编写的。由于汇编语言程序依赖于计算机硬件，其可读性和可移植性都很差；但一般的高级语言又难以实现对计算机硬件的直接操作（这正是汇编语言的优势），于是人们盼望有一种兼有汇编语言和高级语言特性的新语言——C语言。

高级语言的发展也经历了从早期语言到结构化程序设计语言，从面向过程到非过程化程序语言的过程。相应地，软件的开发也由最初的个体手工作坊式的封闭式生产，发展为产业化、流水线式的工业化生产。

60年代中后期，软件越来越多，规模越来越大，而软件的生产基本上是个自为战，缺乏科学规范的系统规划与测试、评估标准，其恶果是大批耗费巨资建立起来的软件系统，由于含有错误而无法使用，甚至带来巨大损失，软件给人的感觉是越来越不可靠，以致几乎没有不出错的软件。这一切，极大地震动了计算机界，史称"软件危机"。人们认识到：大型程序的编制不同于写小程序，它应该是一项新的技术，应该像处理工程一样处理软件研制的全过程。程序的设计应易于保证正确性，也便于验证正确性。1969年，提出了结构化程序设计方法，1970年，第一个结构化程序设计语言--Pascal语言出现，标志着结构化程序设计时期的开始。

80年代初开始，在软件设计思想上，又产生了一次革命，其成果就是面向对象的程序设计。在此之前的高级语言，几乎都是面向过程的，程序的执行是流水线似的，在一个模块被执行完成前，人们不能干别的事，也无法动态地改变程序的执行方向。这和人们日常处理事物的方式是不一致的，对人而言是希望发生一件事就处理一件事，也就是说，不能面向过程，而应是面向具体的应用功能，也就是对象（Object）。其方法就是软件的集成化，如同硬件的集成电路一样，生产一些通用的、封装紧密的功能模块，称之为软件集成块，它与具体应用无关，但能相互组合，完成具体的应用功能，同时又能重复使用。对使用者来说，只关心它的接口（输入量、输出量）及能实现的功能，至于如何实现的，那是它内部的事，使用者完全不用关心，C++、Virtual Basic、Delphi就是典型代表。

高级语言的下一个发展目标是面向应用，也就是说：只需要告诉程序你要干什么，程序就能自动生成算法，自动进行处理，这就是非过程化的程序语言。

机器语言

英 文： Machine Language

别 名： 低级语言，二进制代码语言

定 义：

机器语言是直接用二进制代码指令表达的计算机语言，指令是用0和1组成的一串代码，它们有一定的位数，并分成若干段，各段的编码表示不同的含义，例如某台计算机字长为16位，即有16个二进制数

组成一条指令或其它信息。16个0和1可组成各种排列组合，通过线路变成电信号，让计算机执行各种不同的操作。
如某种计算机的指令为1011011000000000，它表示让计算机进行一次加法操作；而指令1011010100000000则表示进行一次减法操作。它们的前八位表示操作码，而后八位表示地址码。从上面两条指令可以看出，它们只是在操作码中从左边第0位算起的第6和第7位不同。这种机型可包含256（=2的8次方）个不同的指令。
特 点：
机器语言或称为二进制代码语言，计算机可以直接识别，不需要进行任何翻译。每台机器的指令，其格式和代码所代表的含义都是硬性规定的，故称之为面向机器的语言，也称为机器语言。它是第一代的计算机语言。机器语言对不同型号的计算机来说一般是不同的。
缺 点：
1.大量繁杂琐碎的细节牵制着程序员，使他们不可能有更多的时间和精力去从事创造性的劳动，执行对他们来说更为重要的任务。如确保程序的正确性、高效性。
2.程序员既要驾驭程序设计的全局又要深入每一个局部直到实现的细节，即使智力超群的程序员也常常会顾此失彼，屡出差错，因而所编出的程序可靠性差，且开发周期长。
3.由于用机器语言进行程序设计的思维和表达方式与人们的习惯大相径庭，只有经过较长时间职业训练的程序员才能胜任，使得程序设计曲高和寡。
4.因为它的书面形式全是"密"码，所以可读性差，不便于交流与合作。
5.因为它严重地依赖于具体的计算机，所以可移植性差，重用性差。

这些弊端造成当时的计算机应用未能迅速得到推广。

脚本语言

1.脚本语言(JavaScript,VBscript等)介于HTML和C,C++,Java,C#等编程语言之间。
HTML通常用于格式化和链结文本。而编程语言通常用于向机器发出一系列复杂的指令。
2.脚本语言与编程语言也有很多相似地方，其函数与编程语言比较相象一些,其也涉及到变量。与编程语言之间最大的区别是编程语言的语法和规则更为严格和复杂一些.
3.与程序代码的关系:脚本也是一种语言，其同样由程序代码组成。
注：脚本语言一般都有相应的脚本引擎来解释执行。他们一般需要解释器才能运行。JAVASCRIPT,ASP,PHP,PERL,Nuva都是脚本语言。C/C++编译、链接后，可形成独立执行的exe文件。
4.脚本语言是一种解释性的语言,例如vbscript,javascript,installshield script,ActionScript等等,它不象c\c++等可以编译成二进制代码,以可执行文件的形式存在.
脚本语言不需要编译，可以直接用，由解释器来负责解释。

5.脚本语言一般都是以文本形式存在,类似于一种命令.
举个例子说,如果你建立了一个程序,叫aaa.exe,可以打开.aa为扩展名的文件.

你为.aa文件的编写指定了一套规则(语法),当别人编写了.aa文件后,你的程序用这种规则来理解编写人的意图,并作出回应.那么,这一套规则就是脚本语言.

汇编语言

汇编语言(Assembly Language)是面向机器的程序设计语言.汇编语言是一种功能很强的程序设计语言,也是利用计算机所有硬件特性并能直接控制硬件的语言。汇编语言"作为一门语言，对应于高级语言的编译器，需要一个"汇编器"来把汇编语言原文件汇编成机器可执行的代码。高级的汇编器如MASM,TASM等等为我们写汇编程序提供了很多类似于高级语言的特征，比如结构化、抽象等。在这样的环境中编写的汇编程序，有很大一部分是面向汇编器的伪指令，已经类同于高级语言。现在的汇编环境已经如此高级，即使全部用汇编语言来编写windows的应用程序也是可行的，但这不是汇编语言的长处。汇编语言的长处在于编写高效且需要对机器硬件精确控制的程序。

在汇编语言中，用助记符(Memoni)代替操作码，用地址符号(Symbol)或标号(Label)代替地址码。这样用符号代替机器语言的二进制码，就把机器语言变成了汇编语言。因此汇编语言亦称为符号语言。

使用汇编语言编写的程序，机器不能直接识别，要由一种程序将汇编语言翻译成机器语言，这种起翻译作用的程序叫汇编程序，汇编程序是系统软件中语言处理系统软件。汇编语言把汇编程序翻译成机器语言的过程称为汇编。

汇编语言比机器语言易于读写、调试和修改，同时具有机器语言全部优点。但在编写复杂程序时，相对高级语言代码量较大，而且汇编语言依赖于具体的处理器体系结构，不能通用，因此不能直接在不同处理器体系结构之间移植。

汇编语言的特点:
1.面向机器的低级语言，通常是为特定的计算机或系列计算机专门设计的。
2.保持了机器语言的优点，具有直接和简捷的特点。
3.可有效地访问、控制计算机的各种硬件设备，如磁盘、存储器、CPU、I/O端口等。
4.目标代码简短，占用内存少，执行速度快，是高效的程序设计语言。
5.经常与高级语言配合使用，应用十分广泛。
汇编语言的应用:
1.70%以上的系统软件是用汇编语言编写的。
2.某些快速处理、位处理、访问硬件设备等高效程序是用汇编语言编写的。
3.某些高级绘图程序、视频游戏程序是用汇编语言编写的。
汇编语言是我们理解整个计算机系统的最佳起点和最有效途径

人们经常认为汇编语言的应用范围很小，而忽视它的重要性。其实汇编语言对每一个希望学习计算机

科学与技术的人来说都是非常重要的，是不能不学习的语言。
所有可编程计算机都向人们提供机器指令，通过机器指令人们能够使用机器的逻辑功能。
所有程序，不论用何种语言编制，都必须转成机器指令，运用机器的逻辑功能，其功能才能得以实现。
机器的逻辑功能，软件系统功能构筑其上，硬件系统功能运行于下。
汇编语言直接描述机器指令，比机器指令容易记忆和理解。通过学习和使用汇编语言，能够感知、体会、理解机器的逻辑功能，向上为理解各种软件系统的原理，打下技术理论基础；向下为掌握硬件系统的原理，打下实践应用基础。
学习汇编语言，向上可以理解软件，向下能够感知硬件，是我们理解整个计算机系统的最佳起点和最有效途径。

C语言

目录·定义
·C语言的发展历史
·C语言的优点
·C语言的缺点
·C源程序的结构特点
·学习C语言

定义

C语言是一种计算机程序设计语言。它既有高级语言的特点，又具有汇编语言的特点。它可以作为系统设计语言，编写工作系统应用程序，也可以作为应用程序设计语言，编写不依赖计算机硬件的应用程序。因此，它的应用范围广泛。
C语言对操作系统和系统使用程序以及需要对硬件进行操作的场合，用C语言明显优于其它解释型高级语言，有一些大型应用软件也是用C语言编写的。
C语言具有绘图能力强，可移植性，并具备很强的数据处理能力，因此适于编写系统软件，三维，二维图形和动画。它是数值计算的高级语言。
常用的C语言IDE（集成开发环境）有Microsoft Visual C++，Borland C++，Watcom C++ ,Borland C++，Borland C++ Builder,Borland C++ 3.1 for DOS,Watcom C++ 11.0 for DOS,GNU DJGPP C++，Lccwin32 C Compiler 3.1,Microsoft C,High C,Turbo C等等.....

C语言的发展历史

C语言的原型ALGOL 60语言。（也称为A语言）
1963年，剑桥大学将ALGOL 60语言发展成为CPL(Combined Programming Language)语言。
1967年，剑桥大学的Matin Richards 对CPL语言进行了简化，于是产生了BCPL语言。

1970年，美国贝尔实验室的Ken Thompson将BCPL进行了修改，并为它起了一个有趣的名字“B语言”。意思是将CPL语言煮干，提炼出它的精华。并且他用B语言写了第一个UNIX操作系统。
而在1973年，B语言也给人“煮”了一下，美国贝尔实验室的D.M.RITCHIE在B语言的基础上最终设计出了一种新的语言，他取了BCPL的第二个字母作为这种语言的名字，这就是C语言。
为了使UNIX操作系统推广，1977年Dennis M.Ritchie 发表了不依赖于具体机器系统的C语言编译文本《可移植的C语言编译程序》。即是著名的ANSI C。
1978年Brian W.Kernighian和Dennis M.Ritchie出版了名著《C语言程序》（The C Programming Language），从而使C语言成为当时世界上流行最广泛的高级程序设计语言。
1988年，随着微型计算机的日益普及, C语言出现了许多版本。由于没有统一的标准,使得这些C语言之间出现了一些不一致的地方。为了改变这种情况,美国国家标准研究所(ANSI)为C语言制定了一套ANSI标准, 成为现行的C语言标准3.C语言的主要特点 。C语言发展迅速, 而且成为最受欢迎的语言之一, 主要因为它具有强大的功能。许多著名的系统软件, 如DBASE Ⅲ PLUS、DBASE Ⅳ 都是由C 语言编写的。用C语言加上一些汇编语言子程序, 就更能显示C语言的优势了,象PC- DOS 、WORDSTAR等就是用这种方法编写的。

C语言的优点
1. 简洁紧凑、灵活方便
C语言一共只有32个关键字,9种控制语句，程序书写自由，主要用小写字母表示。它把高级语言的基本结构和语句与低级语言的实用性结合起来。 C 语言可以象汇编语言一样对位、字节和地址进行操作, 而这三者是计算机最基本的工作单元。
2. 运算符丰富
C的运算符包含的范围很广泛，共有种34个运算符。C语言把括号、赋值、强制类型转换等都作为运算符处理。从而使C的运算类型极其丰富表达式类型多样化，灵活使用各种运算符可以实现在其它高级语言中难以实现的运算。
3. 数据结构丰富
C的数据类型有：整型、实型、字符型、数组类型、指针类型、结构体类型、共用体类型等。能用来实现各种复杂的数据类型的运算。并引入了指针概念,使程序效率更高。另外C语言具有强大的图形功能,支持多种显示器和驱动器。且计算功能、逻辑判断功能强大。
4. C是结构式语言
结构式语言的显著特点是代码及数据的分隔化,即程序的各个部分除了必要的信息交流外彼此独立。这种结构化方式可使程序层次清晰, 便于使用、维护以及调试。C语言是以函数形式提供给用户的,这些函数可方便的调用,并具有多种循环、条件语句控制程序流向,从而使程序完全结构化。
5. C语法限制不太严格，程序设计自由度大
虽然C语言也是强类型语言，但它的语法比较灵活，允许程序编写者有较大的自由度。

6. C语言允许直接访问物理地址，可以直接对硬件进行操作
因此既具有高级语言的功能，又具有低级语言的许多功能，能够象汇编语言一样对位、字节和地址进行操作,而这三者是计算机最基本的工作单元，可以用来写系统软件。
7. C语言程序生成代码质量高，程序执行效率高
一般只比汇编程序生成的目标代码效率低10へ20%。
8. C语言适用范围大，可移植性好
C语言有一个突出的优点就是适合于多种操作系统, 如DOS、UNIX,也适用于多种机型。

C语言的缺点
1. C语言的缺点主要是表现在数据的封装性上，这一点使得C在数据的安全性上做的有很大缺陷，这也是C和C++的一大区别。
2. C语言的语法限制不太严格，对变量的类型约束不严格，影响程序的安全性，对数组下标越界不作检查等。从应用的角度，C语言比其他高级语言较难掌握。
[C语言指针]
指针就是C语言的一大特色,可以说C语言优于其它高级语言的一个重要原因就是因为它有指针操作可以直接进行靠近硬件的操作,但是C的指针操作也给它带来了很多不安全的因素。C++在这方面做了很好的改进,在保留了指针操作的同时又增强了安全性。Java取消了指针操作，提高了安全性。

C源程序的结构特点
1.一个C语言源程序可以由一个或多个源文件组成。
2.每个源文件可由一个或多个函数组成。
3.一个源程序不论由多少个文件组成，都有一个且只能有一个main函数，即主函数。
4.源程序中可以有预处理命令(include 命令仅为其中的一种)，预处理命令通常应放在源文件或源程序的最前面。
5.每一个说明，每一个语句都必须以分号结尾。但预处理命令，函数头和花括号“}”之后不能加分号。
6.标识符，关键字之间必须至少加一个空格以示间隔。若已有明显的间隔符，也可不再加空格来间隔。

学习C语言
在初学C语言时，可能会遇到有些问题理解不透，或者表达方式与以往数学学习中不同（如运算符等），这就要求不气馁，不明白的地方多问多想，鼓足勇气进行学习，待学完后面的章节知识，前面的问题也就迎刃而解了，这一方面我感觉是我们同学最欠缺，大多学不好的就是因为一开始遇到困难就放弃，曾经和好多同学谈他的问题，回答是听不懂、不想听、放弃这样三个过程，我反问，这节课你听过课吗？回答又是没有，根本就没听过课，怎么说自己听不懂呢？相应的根本就没学习，又谈何学的好？

学习C语言始终要记住“曙光在前头”和“千金难买回头看”，“千金难买回头看”是学习知识的重要方法，就是说，学习后面的知识，不要忘了回头弄清遗留下的问题和加深理解前面的知识，这是我们学生最不易做到的，然而却又是最重要的。学习C语言就是要经过几个反复，才能前后贯穿，积累应该掌握的C知识。

那么，我们如何学好《C程序设计》呢？

一．学好C语言的运算符和运算顺序

这是学好《C程序设计》的基础，C语言的运算非常灵活，功能十分丰富，运算种类远多于其它程序设计语言。在表达式方面较其它程序语言更为简洁，如自加、自减、逗号运算和三目运算使表达式更为简单，但初学者往往会觉的这种表达式难读，关键原因就是对运算符和运算顺序理解不透不全。当多种不同运算组成一个运算表达式，即一个运算式中出现多种运算符时，运算的优先顺序和结合规则显得十分重要。在学习中，只要我们对此合理进行分类，找出它们与我们在数学中所学到运算之间的不同点之后，记住这些运算也就不困难了，有些运算符在理解后更会牢记心中，将来用起来得心应手，而有些可暂时放弃不记，等用到时再记不迟。

先要明确运算符按优先级不同分类，《C程序设计》运算符可分为15种优先级，从高到低，优先级为1~15，除第2、3级和第14级为从右至左结合外，其它都是从左至右结合，它决定同级运算符的运算顺序.

二．学好C语言的四种程序结构

（1）顺序结构

顺序结构的程序设计是最简单的，只要按照解决问题的顺序写出相应的语句就行，它的执行顺序是自上而下，依次执行。

例如；a＝3，b＝5，现交换a，b的值，这个问题就好像交换两个杯子水，这当然要用到第三个杯子，假如第三个杯子是c，那么正确的程序为：c＝a；a＝b；b＝c；执行结果是a＝5，b＝c＝3如果改变其顺序，写成：a＝b；c＝a；b＝c；则执行结果就变成a＝b＝c＝5，不能达到预期的目的，初学者最容易犯这种错误。顺序结构可以独立使用构成一个简单的完整程序，常见的输入、计算，输出三步曲的程序就是顺序结构，例如计算圆的面积，其程序的语句顺序就是输入圆的半径r，计算s＝3.14159*r*r,输出圆的面积s。不过大多数情况下顺序结构都是作为程序的一部分，与其它结构一起构成一个复杂的程序，例如分支结构中的复合语句、循环结构中的循环体等。

（2）分支结构

顺序结构的程序虽然能解决计算、输出等问题，但不能做判断再选择。对于要先做判断再选择的问题就要使用分支结构。分支结构的执行是依据一定的条件选择执行路径，而不是严格按照语句出现的物理顺序。分支结构的程序设计方法的关键在于构造合适的分支条件和分析程序流程，根据不同的程序流程选择适当的分支语句。分支结构适合于带有逻辑或关系比较等条件判断的计算，设计这类程序时往往都要先绘制其程序流程图，然后根据程序流程写出源程序，这样做把程序设计分析与语言分开，使得问题简单化，易于理解。程序流程图是根据解题分析所绘制的程序执行流程图。

学习分支结构不要被分支嵌套所迷惑，只要正确绘制出流程图，弄清各分支所要执行的功能，嵌套结构也就不难了。嵌套只不过是分支中又包括分支语句而已，不是新知识，只要对双分支的理解清楚，分支嵌套是不难的。下面我介绍几种基本的分支结构。

①if(条件)
{
分支体
}

这种分支结构中的分支体可以是一条语句，此时“{ }”可以省略，也可以是多条语句即复合语句。它有两条分支路径可选，一是当条件为真，执行分支体，否则跳过分支体，这时分支体就不会执行。如：要计算x的绝对值，根据绝对值定义，我们知道，当x>=0时，其绝对值不变，而x<0时其绝对值是为x的反号，因此程序段为：if(x<0) x=-x;

②if(条件)
{分支1}
else
{分支2}

这是典型的分支结构，如果条件成立，执行分支1，否则执行分支2，分支1和分支2都可以是1条或若干条语句构成。如：求ax^2+bx+c=0的根

分析：因为当b^2-4ac>=0时,方程有两个实根，否则（b^2-4ac<0）有两个共轭复根。其程序段如下：

```
d=b*b-4*a*c;
if(d>=0)
{x1=(-b+sqrt(d))/2a;
x2=(-b-sqrt(d))/2a;
printf(“x1=%8.4f,x2=%8.4f\n”,x1,x2);
}
else
{r=-b/(2*a);
i =sqrt(-d)/(2*a);
printf(“x1=%8.4f+%8.4fi\n”r, i);
printf(“x2=%8.4f-%8.4fi\n”r,i)
}
```

③嵌套分支语句：其语句格式为：

```
if(条件1) {分支1}；
else if（条件2） {分支2}
else if（条件3） {分支3}
```

……
else if（条件n）{分支n}
else {分支n+1}
嵌套分支语句虽可解决多个入口和出口的问题，但超过3重嵌套后，语句结构变得非常复杂，对于程序的阅读和理解都极为不便，建议嵌套在3重以内，超过3重可以用下面的语句。
④switch开关语句：该语句也是多分支选择语句，到底执行哪一块，取决于开关设置，也就是表达式的值与常量表达式相匹配的那一路，它不同if…else 语句，它的所有分支都是并列的，程序执行时，由第一分支开始查找，如果相匹配，执行其后的块，接着执行第2分支，第3分支……的块，直到遇到break语句；如果不匹配，查找下一个分支是否匹配。这个语句在应用时要特别注意开关条件的合理设置以及break语句的合理应用。
（3）循环结构：
循环结构可以减少源程序重复书写的工作量，用来描述重复执行某段算法的问题，这是程序设计中最能发挥计算机特长的程序结构，C语言中提供四种循环，即goto循环、while循环、do –while循环和for循环。四种循环可以用来处理同一问题，一般情况下它们可以互相代替换，但一般不提倡用goto循环，因为强制改变程序的顺序经常会给程序的运行带来不可预料的错误，在学习中我们主要学习while、do…while、for三种循环。常用的三种循环结构学习的重点在于弄清它们相同与不同之处，以便在不同场合下使用，这就要清楚三种循环的格式和执行顺序，将每种循环的流程图理解透彻后就会明白如何替换使用，如把while循环的例题，用for语句重新编写一个程序，这样能更好地理解它们的作用。特别要注意在循环体内应包含趋于结束的语句（即循环变量值的改变），否则就可能成了一个死循环，这是初学者的一个常见错误。
在学完这三个循环后，应明确它们的异同点：用while和do…while循环时，循环变量的初始化的操作应在循环体之前，而for循环一般在语句1中进行的；while 循环和for循环都是先判断表达式，后执行循环体，而do…while循环是先执行循环体后判断表达式，也就是说do…while的循环体最少被执行一次，而while 循环和for就可能一次都不执行。另外还要注意的是这三种循环都可以用break语句跳出循环，用continue语句结束本次循环，而goto语句与if构成的循环，是不能用break和 continue语句进行控制的。
顺序结构、分支结构和循环结构并不彼此孤立的，在循环中可以有分支、顺序结构，分支中也可以有循环、顺序结构，其实不管哪种结构，我们均可广义的把它们看成一个语句。在实际编程过程中常将这三种结构相互结合以实现各种算法，设计出相应程序，但是要编程的问题较大，编写出的程序就往往很长、结构重复多，造成可读性差，难以理解，解决这个问题的方法是将C程序设计成模块化结构。
(4)模块化程序结构
C语言的模块化程序结构用函数来实现，即将复杂的C程序分为若干模块，每个模块都编写成一个C函数，然后通过主函数调用函数及函数调用函数来实现一大型问题的C程序编写，因此常说：C程序=主函数+子函数。 因此，对函数的定义、调用、值的返回等中要尤其注重理解和应用，并通过上机调试加

以巩固。
三．掌握一些简单的算法
编程其实一大部分工作就是分析问题，找到解决问题的方法，再以相应的编程语言写出代码。这就要求掌握算法，根据我们的《C程序设计》教学大纲中，只要求我们掌握一些简单的算法，在掌握这些基本算法后，要完成对问题的分析就容易了。如两个数的交换、三个数的比较、选择法排序和冒泡法排序，这就要求我们要清楚这些算法的内在含义
结语：当我们把握好上述几方面后，只要同学们能克服畏难、厌学、上课能专心听讲，做好练习与上机调试，其实C语言并不难学
C源程序的关键字----------
所谓关键字就是已被C语言本身使用, 不能作其它用途使用的字。例如关键字不能用作变量名、函数名等
由ANSI标准定义的C语言关键字共32个 :
auto double int struct break else long switch case enum register typedef char extern return union const float short unsigned continue for signed void default goto sizeof volatile do if while static
根据关键字的作用，可分其为数据类型关键字、控制语句关键字、存储类型关键字和其它关键字四类。
1 数据类型关键字（12个）：
(1) char ：声明字符型变量或函数
(2) double ：声明双精度变量或函数
(3) enum ：声明枚举类型
(4) float：声明浮点型变量或函数
(5) int： 声明整型变量或函数
(6) long ：声明长整型变量或函数
(7) short ：声明短整型变量或函数
(8) signed：声明有符号类型变量或函数
(9) struct：声明结构体变量或函数
(10) union：声明联合数据类型
(11) unsigned：声明无符号类型变量或函数
(12) void ：声明函数无返回值或无参数，声明无类型指针（基本上就这三个作用）
（2）控制语句关键字（12个）：
A循环语句
(1) for：一种循环语句(可意会不可言传）
(2) do ：循环语句的循环体
(3) while ：循环语句的循环条件

(4) break：跳出当前循环
(5) continue：结束当前循环，开始下一轮循环
B条件语句
(1)if: 条件语句
(2)else ：条件语句否定分支（与 if 连用）
(3)goto：无条件跳转语句
C开关语句
(1)switch :用于开关语句
(2)case：开关语句分支
(3)default：开关语句中的“其他”分支
D
return ：子程序返回语句（可以带参数，也看不带参数）
3 存储类型关键字（4个）
(1)auto ：声明自动变量 一般不使用
(2)extern：声明变量是在其他文件正声明（也可以看做是引用变量）
(3)register：声明积存器变量
(4)static ：声明静态变量
4 其它关键字（4个）：
(1)const ：声明只读变量
(2)sizeof：计算数据类型长度
(3)typedef：用以给数据类型取别名（当然还有其他作用
(4)volatile：说明变量在程序执行中可被隐含地改变

C++语 言

C++语言是一种优秀的面向对象程序设计语言，它在C语言的基础上发展而来，但它比C语言更容易为人们学习和掌握。C++以其独特的语言机制在计算机科学的各个领域中得到了广泛的应用。面向对象的设计思想是在原来结构化程序设计方法基础上的一个质的飞跃，C++完美地体现了面向对象的各种特性。

Bjarne Stroustrup（C++的设计者）对C++的设计和演化的描述
C++的设计和演化（The Design and Evolution of C++）
C++程序设计语言是由来自AT&T Bell Laboratories的Bjarne Stroustrup（即本文作者）设计和实现的，它兼具Simula语言在组织与设计方面的特性以及适用于系统程序设计的C语言设施。C++最初的版本被称作“带类的C（C with classes）”[Stroustrup,1980]，在1980年被第一次投入使用；当时它只支持系统程序设计（§3）和数据抽象技术（§4.1）。支持面向对象程序设计的语言设施在1983年被加入C++；之后，面向对象设计方法和面向对象程序设计技术就逐渐进入了C++领域。在1985年，C++第一次投

入商业市场[Stroustrup,1986][Stroustrup,1986b]。在1987至1989年间，支持范型程序设计的语言设施也被加进了C++[Ellis,1990][Stroustrup,1991]。
随着若干独立开发的C++实现产品的出现和广泛应用，正式的C++标准化工作在1990年启动。标准化工作由ANSI（American National Standard Institute）以及后来加入的ISO（International Standards Organization）负责。1998年正式发布了C++语言的国际标准[C++,1998]。在标准化工作进展期间，标准委员会充当了一个重要的角色，其发布的C++标准之草案在正式标准发布之前，一直被作为过渡标准而存在。而作为标准委员会中的积极分子，我是C++进一步发展工作中的主要参与者。与以前的C++语言版本相比，标准C++更接近我理想中的那个C++语言了。关于C++的设计和演化，
在[Stroustrup,1994]、[Stroustrup,1996]和[Stroustrup,1997b]中有详细的叙述。至于标准化工作末期产生的C++语言定义，在[Stroustrup,1997]有详细叙述。
1 C++的设计目标（C++ Design Aims）
C++的设计目标，就是要让C++既具有适合于系统程序设计的C语言所具有的可适应性和高效性，又能在其程序组织结构方面具有像Simula那样的语言设施（Simula所支持的这种程序组织结构通常被称为面向对象程序设计风格）。在设计的时候，还做了很大的努力，使得引借自Simula的高层次的程序设计技术能够应用于系统程序设计之中。这即是说，C++所提供的抽象机制能够被应用于那些对效率和可适应性具有极高要求的程序设计任务之中。
上述的C++之设计目标可以小结如下：
[设计目标]
|| 对于要解决实际问题的程序员而言，C++使程序设计变得更有乐趣；
|| C++是一门通用目的的程序设计语言，它：
——是一个更好的C；
——支持数据抽象；
——支持面向对象程序设计；
——支持范型程序设计。
对范型程序设计的支持在C++设计的后期才被作为一个明确、独立的目标来实现。而在C++演化过程的大部分时间里，我一直把范型程序设计以及支持它的语言特性划归在“数据抽象”的大标题之下。
2 C++的设计原则（Design Principles）
在[Stroustrup,1994]中，C++的设计规则被分为基本规则、基于设计的规则、语言的技术性规则以及基于低层次程序设计的规则四个方面，分列在下文中。
[基本规则（General rules）]
|| C++的每一步演化和发展必须是由于实际问题所引起的；
|| C++是一门语言，而不是一个完整的系统；
|| 不能无休止的一味追求完美；
|| C++在其存在的“当时”那个时期必须是有用处的；

|| 每一种语言特性必须有一个有根据的、明确的实现方案；
|| 总能提供一种变通的方法；
|| 能为意欲支持的每一种程序设计风格提供易于理解的支持方法；
|| 不强制于人。
可以注意到，基本规则的最后三条暗示了两点：对适用于真实世界中各种应用的便捷工具的强调；对程序员的技术和取向（偏好）的充分考虑。从一开始，C++面向的就是那些要做实际项目的程序员。所谓的"完美"被认为是不可能达到的，这是由于C++用户在需求、背景和待解决问题上存在着太大的不同。况且，在一门通用目的的程序设计语言的整个生存期之内，连对"完美"一词的诠释都可能会有极大的改变。由此可知，在语言的演化过程中，来自用户的反馈和语言实现者们积累的经验才是最为重要的。

[基于设计的规则（Design-support rules）]
|| 支持良好的设计方案；
|| 提供用于程序组织的语言设施；
|| 心口如一（Say what you mean）；
|| 所有的语言特性必须具有切实有效的承受能力；
|| 开启一个有用的特性比避免所有的误用更为重要；
|| 能将独立开发的部件组合成完整的软件。
C++的一个目标就是提供更易用并具有一定承受能力的设计思想和程序设计技术，进一步提高程序的质量。这些技术中的绝大部分都源自Simula [Dahl,1970][Dahl,1972][Birtwistle,1979]，并通常被作为面向对象程序设计和面向对象设计思想来讨论。然而，C++的设计目标总还是在于要支持一定范围内的各种程序设计风格和设计思想。这与一般在语言设计方面的观点形成一定对比。一般在语言设计上总是试图将所有系统内建于单独一个被重点支持的、带有强制性的程序设计风格之中（或称典范paradigm）。
[语言的技术性规则（Language-technical rules）]
|| 与静态型别系统（Static type system）没有内在的冲突；
|| 像对内建（built-in）型别一样对用户自定义型别提供很好的支持；
|| 个异化（locality）行为是可取的；
|| 避免产生顺序上的依赖关系；
|| 在对语言产生疑惑时，可以选取其特性中最易掌握的部分；
|| 可以因为不正当的语法使用而产生问题（Syntax matters (often in perverse ways)）
|| 削弱对预处理器的使用。
当然，这些规则要具体结合更多关于基本目标的上下文环境来考虑。应该注意到的是，在"与C有较高的兼容性"、"不损失效率"以及"具有便捷的可用性来解决实际问题"这三个方面的要求，与在"完整的型别安全性"、"完全的通用性"以及"完善的抽象之美"这三个方面的要求形成对立。

C++从Simula中借鉴了用户自定义型别（class,§4.1）和类层次机制。然而，在Simula及许多类似的语言中，其对用户自定义型别的支持与其对内建型别的支持存在着根本上的不同。例如，Simula中不允许在栈中为用户自定义型别的对象分配空间，并且只允许通过指针（这在Simula中称为引用——reference）来对这些对象进行访问。而相反的，内建型别的对象只在栈中被分配空间，不能在动态存储区中分配，而且不能使用指针指向它。这种在对待内建型别与对待用户自定义型别上的差异，暗示着对效率问题的严格考虑。比如，当作为一个在动态存储区中被分配的对象之引用时，如果该对象属于自定义型别（比如complex,§4.1），那么就会为运行期及空间带来负荷；而这些负荷在有些应用中被认为是不可接受的。这些正是C++意欲涉足解决的问题。同时，在用法上的不同也决定了：不可能在范型程序设计中统一对待那些语义上近似的型别。

在维护一个较庞大的程序时，一个程序员不可避免的会基于某些不完整的知识来对程序作一些修改，只关注全部程序代码中的一小部分。

基于此，C++提供了class（§4）、namespace（§5.2）和访问控制（§4.1），使设计决策的各异化（locality）成为可能。

在基于一趟编译（one-pass compilation）的语言中，某些顺序上的依赖性是不可避免的。例如在C++中，一个变量或者函数在其被声明之前是无法使用的。然而，C++中类成员的名字规则和重载解析（overload resolution）的规则还是在独立于声明顺序的原则下被制定出来，以便将发生混乱和错误的可能性降至最低。

[基于低层次程序设计的规则（Low-level programming support rules）]

II 使用传统的（笨拙的）连接器（linker）；

II 与C语言不存在无故的不兼容性；

II 不给C++之下层级的更低层语言留出余地（汇编语言除外）；

II 你不会为你所不使用的部分付出代价（零负荷规则）；

II 在产生疑惑时，能提供完全自主控制的途径。

在C++的设计中只要在不严重影响其对强型别检查（strong type checking）的支持的地方，都尽量做到与C的“source-link”方式相兼容。除了某些微小的细节差别之外，C++将C[Kernighan,1978][Kernighan,1988]作为一个子集包含了进来。C++与C的兼容性使得C++程序员立刻就能有一个完整的语言和工具集可用。还有两点也很重要，一是有大量关于C的高质量的教学素材已经存在，二是C++程序员可以利用C++与C的兼容性而直接并有效的使用大量现成的程序库。在决定将C作为C++的基础的时候，C还没有像后来那样出类拔萃、炙手可热，所以在考虑这个问题的时候，与C语言所提供的可适应性和高效性相比，C语言的流行程度只是个次要的考虑因素。

然而，与C的兼容性也使得C++在某些语法和语义上保留了C的一些瑕疵之处。比如，C语言的声明语法就实在远不及优美；而其内建型别的隐式转换规则也是混乱无章法的。还有另一个大问题，就是许多从C转向C++的程序员并没有认识到，代码质量上的显著提高只能通过在程序设计风格上的显著改变来达到。

java语言

Java简介

Java是由Sun Microsystems公司于1995年5月推出的Java程序设计语言（以下简称Java语言）和Java平台的总称。用Java实现的HotJava浏览器（支持Java applet）显示了Java的魅力：跨平台、动感的Web、Internet计算。从此，Java被广泛接受并推动了Web的迅速发展，常用的浏览器现在均支持Java applet。另一方面，Java技术也不断更新。

Java平台由Java虚拟机（Java Virtual Machine）和Java 应用编程接口（Application Programming Interface、简称API）构成。Java 应用编程接口为Java应用提供了一个独立于操作系统的标准接口，可分为基本部分和扩展部分。在硬件或操作系统平台上安装一个Java平台之后，Java应用程序就可运行。现在Java平台已经嵌入了几乎所有的操作系统。这样Java程序可以只编译一次，就可以在各种系统中运行。Java应用编程接口已经从1.1x版发展到1.2版。目前常用的Java平台基于Java1.4，最近版本为Java1.6。

Java分为三个体系JavaSE，JavaEE，JavaME。

Java语言

Java语言是一个支持网络计算的面向对象程序设计语言。Java语言吸收了Smalltalk语言和C++语言的优点，并增加了其它特性，如支持并发程序设计、网络通信、和多媒体数据控制等。主要特性如下：

1、Java语言是简单的。Java语言的语法与C语言和C++语言很接近，使得大多数程序员很容易学习和使用Java。另一方面，Java丢弃了C++ 中很少使用的、很难理解的、令人迷惑的那些特性，如操作符重载、多继承、自动的强制类型转换。特别地，Java语言不使用指针，并提供了自动的废料收集，使得程序员不必为内存管理而担忧。

2、Java语言是一个面向对象的。Java语言提供类、接口和继承等原语，为了简单起见，只支持类之间的单继承，但支持接口之间的多继承，并支持类与接口之间的实现机制（关键字为implements）。Java语言全面支持动态绑定，而C++ 语言只对虚函数使用动态绑定。总之，Java语言是一个纯的面向对象程序设计语言。

3、Java语言是分布式的。Java语言支持Internet应用的开发，在基本的Java应用编程接口中有一个网络应用编程接口（java.net），它提供了用于网络应用编程的类库，包括URL、URLConnection、Socket、 ServerSocket等。Java的RMI(远程方法激活)机制也是开发分布式应用的重要手段。

4、Java语言是健壮的。Java的强类型机制、异常处理、废料的自动收集等是Java程序健壮性的重要保证。对指针的丢弃是Java的明智选择。Java的安全检查机制使得Java更具健壮性。

5、Java语言是安全的。Java通常被用在网络环境中，为此，Java提供了一个安全机制以防恶意代码的攻击。除了Java语言具有的许多安全特性以外，Java对通过网络下载的类具有一个安全防范机

制（类ClassLoader），如分配不同的名字空间以防替代本地的同名类、字节代码检查，并提供安全管理机制（类SecurityManager）让Java应用设置安全哨兵。

6、Java语言是体系结构中立的。Java程序（后缀为java的文件）在Java平台上被编译为体系结构中立的字节码格式（后缀为class的文件），然后可以在实现这个Java平台的任何系统中运行。这种途径适合于异构的网络环境和软件的分发。

7、Java语言是可移植的。这种可移植性来源于体系结构中立性，另外，Java还严格规定了各个基本数据类型的长度。Java系统本身也具有很强的可移植性，Java编译器是用Java实现的，Java的运行环境是用ANSI C实现的。

8、Java语言是解释型的。如前所述，Java程序在Java平台上被编译为字节码格式，然后可以在实现这个Java平台的任何系统中运行。在运行时，Java平台中的Java解释器对这些字节码进行解释执行，执行过程中需要的类在联接阶段被载入到运行环境中。

9、Java是高性能的。与那些解释型的高级脚本语言相比，Java的确是高性能的。事实上，Java的运行速度随着JIT(Just-In-Time)编译器技术的发展越来越接近于C++。

10、Java语言是多线程的。在Java语言中，线程是一种特殊的对象，它必须由Thread类或其子（孙）类来创建。通常有两种方法来创建线程：其一，使用型构为Thread(Runnable) 的构造子将一个实现了Runnable接口的对象包装成一个线程，其二，从Thread类派生出子类并重写run方法，使用该子类创建的对象即为线程。值得注意的是Thread类已经实现了Runnable接口，因此，任何一个线程均有它的run方法，而run方法中包含了线程所要运行的代码。线程的活动由一组方法来控制。Java语言支持多个线程的同时执行，并提供多线程之间的同步机制（关键字为synchronized）。

11、Java语言是动态的。Java语言的设计目标之一是适应于动态变化的环境。Java程序需要的类能动态地被载入到运行环境，也可以通过网络来载入所需要的类。这也有利于软件的升级。另外，Java中的类有一个运行时刻的表示，能进行运行时刻的类型检查。

Java语言的优良特性使得Java应用具有无比的健壮性和可靠性，这也减少了应用系统的维护费用。Java对对象技术的全面支持和Java平台内嵌的API能缩短应用系统的开发时间并降低成本。Java的编译一次，到处可运行的特性使得它能够提供一个随处可用的开放结构和在多平台之间传递信息的低成本方式。特别是Java企业应用编程接口（Java Enterprise APIs）为企业计算及电子商务应用系统提供了有关技术和丰富的类库。

1、JDBC（Java Database Connectivity）提供连接各种关系数据库的统一接口。

2、EJB(Enterprise JavaBeans)使得开发者方便地创建、部署和管理跨平台的基于组件的企业应用。

3、Java RMI(Java Remote Method Invocation)用来开发分布式Java应用程序。一个Java对象的方法能被远程Java虚拟机调用。这样，远程方法激活可以发生在对等的两端，也可以发生在客户端和服务器之间，只要双方的应用程序都是用Java写的。

4、Java IDL(Java Interface Definition Language) 提供与CORBA(Common Object Request Broker

Architecture)的无逢的互操作性。这使得Java能集成异构的商务信息资源。

5、JNDI(Java Naming and Directory Interface)提供从Java平台到的统一的无逢的连接。这个接口屏蔽了企业网络所使用的各种命名和目录服务。

6、JMAPI（Java Management API）为异构网络上系统、网络和服务管理的开发提供一整套丰富的对象和方法。

7、JMS(Java Message Service)提供企业消息服务，如可靠的消息队列、发布和订阅通信、以及有关推拉（Push/Pull）技术的各个方面。

8、JTS(Java transaction Service)提供存取事务处理资源的开放标准，这些事务处理资源包括事务处理应用程序、事务处理管理及监控。

在Java技术中，值得关注的还有JavaBeans，它是一个开放的标准的组件体系结构，它独立于平台，但使用Java语言。一个JavaBean是一个满足JavaBeans规范的Java类，通常定义了一个现实世界的事物或概念。一个JavaBean的主要特征包括属性、方法和事件。通常，在一个支持JavaBeans规范的开发环境（如Sun Java Studio 和IBM VisualAge for Java）中，可以可视地操作JavaBean，也可以使用JavaBean构造出新的JavaBean。JavaBean的优势还在于Java带来的可移植性。现在，EJB (Enterprise JavaBeans) 将JavaBean概念扩展到Java服务端组件体系结构，这个模型支持多层的分布式对象应用。除了JavaBeans，典型的组件体系结构还有DCOM和CORBA，关于这些组件体系结构的深入讨论超出了本书的范围。

Java开源项目

Spring Framework 【Java开源 J2EE框架】

Spring 是一个解决了许多在J2EE开发中常见的问题的强大框架。 Spring提供了管理业务对象的一致方法并且鼓励了注入对接口编程而不是对类编程的良好习惯。Spring的架构基础是基于使用JavaBean属性的 Inversion of Control容器。然而，这仅仅是完整图景中的一部分：Spring在使用IoC容器作为构建完关注所有架构层的完整解决方案方面是独一无二的。 Spring提供了唯一的数据访问抽象，包括简单和有效率的JDBC框架，极大的改进了效率并且减少了可能的错误。Spring的数据访问架构还集成了 Hibernate和其他O/R mapping解决方案。Spring还提供了唯一的事务管理抽象，它能够在各种底层事务管理技术，例如JTA或者JDBC事务提供一个一致的编程模型。 Spring提供了一个用标准Java语言编写的AOP框架，它给POJOs提供了声明式的事务管理和其他企业事务--如果你需要--还能实现你自己的 aspects。这个框架足够强大，使得应用程序能够抛开EJB的复杂性，同时享受着和传统EJB相关的关键服务。Spring还提供了可以和IoC容器集成的强大而灵活的MVC Web框架。

【SpringIDE：Eclipse平台下一个辅助开发插件】.

WebWork 【Java开源 Web框架】

WebWork 是由OpenSymphony组织开发的，致力于组件化和代码重用的拉出式MVC模式J2EE Web框架。WebWork目前最新版本是2.1，现在的WebWork2.x前身是Rickard Oberg开发的WebWork，但现在WebWork已经被拆分成了Xwork1和WebWork2两个项目。 Xwork简洁、灵活功

能强大，它是一个标准的Command模式实现，并且完全从web层脱离出来。 Xwork提供了很多核心功能：前端拦截机（interceptor），运行时表单属性验证，类型转换，强大的表达式语言（OGNL – the Object Graph Notation Language），IoC（Inversion of Control倒置控制）容器等。 WebWork2建立在Xwork之上，处理HTTP的响应和请求。WebWork2使用ServletDispatcher将HTTP请求的变成Action(业务层Action类), session（会话）application（应用程序）范围的映射，request请求参数映射。WebWork2支持多视图表示，视图部分可以使用 JSP, Velocity, FreeMarker, JasperReports，XML等。在WebWork2.2中添加了对AJAX的支持，这支持是构建在DWR与Dojo这两个框架的基础之上.【EclipseWork：用于WebWork辅助开发的一个Eclipse插件】

Struts 【Java开源 Web框架】

Struts 是一个基于Sun J2EE平台的MVC框架，主要是采用Servlet和JSP技术来实现的。由于Struts能充分满足应用开发的需求，简单易用，敏捷迅速，在过去的一年中颇受关注。Struts把Servlet、JSP、自定义标签和信息资源(message resources)整合到一个统一的框架中，开发人员利用其进行开发时不用再自己编码实现全套MVC模式，极大的节省了时间，所以说Struts是一个非常不错的应用框架。【StrutsIDE：用于Struts辅助开发的一个Eclipse插件】

Hibernate 【Java开源 持久层框架】

Hibernate 是一个开放源代码的对象关系映射框架，它对JDBC进行了非常轻量级的对象封装，使得Java程序员可以随心所欲的使用对象编程思维来操纵数据库。 Hibernate可以应用在任何使用JDBC的场合，既可以在Java的客户端程序实用，也可以在Servlet/JSP的Web应用中使用，最具革命意义的是，Hibernate可以在应用EJB的J2EE架构中取代CMP，完成数据持久化的重任。Eclipse平台下的Hibernate辅助开发工具:【Hibernate Synchronizer】【MiddlegenIDE】

Quartz 【Java开源 Job调度】

Quartz 是OpenSymphony开源组织在Job scheduling领域又一个开源项目,它可以与J2EE与J2SE应用程序相结合也可以单独使用。Quartz可以用来创建简单或为运行十个，百个，甚至是好几万个Jobs这样复杂的日程序表。Jobs可以做成标准的Java组件或 EJBs。Quartz的最新版本为Quartz 1.5.0。

Velocity 【Java开源 模板引擎】

Velocity 是一个基于java的模板引擎（template engine）。它允许任何人仅仅简单的使用模板语言（template language）来引用由java代码定义的对象。当Velocity应用于web开发时，界面设计人员可以和java程序开发人员同步开发一个遵循MVC架构的web站点，也就是说，页面设计人员可以只关注页面的显示效果，而由java程序开发人员关注业务逻辑编码。Velocity将java代码从web页面中分离出来，这样为web站点的长期维护提供了便利，同时也为我们在JSP和PHP之外又提供了一种可选的方案。Velocity的能力远不止web站点开发这个领域，例如，它可以从模板（template）产生SQL和PostScript、XML，它也可以被当作一个独立工具来产生源代码和报告，或者作为其他系统的集成组件使用。Velocity也可以为Turbine web开发架构提供模板服务（template

service）。Velocity+Turbine提供一个模板服务的方式允许一个web应用以一个眞正的MVC模型进行开发。【VeloEclipse ：Velocity在Eclipse平台下的一个辅助开发插件】

IBATIS 【Java开源 持久层框架】

使用ibatis 提供的ORM机制，对业务逻辑实现人员而言，面对的是纯粹的Java对象， 这一层与通过Hibernate 实现ORM 而言基本一致，而对于具体的数据操作，Hibernate 会自动生成SQL 语句，而ibatis 则要求开发者编写具体的SQL 语句。相对Hibernate等 “全自动”ORM机制而言，ibatis 以SQL开发的工作量和数据库移植性上的让步，为系统设计提供了更大的自由空间。作为“全自动”ORM 实现的一种有益补充，ibatis 的出现显 得别具意义。

Compiere ERP&CRM 【Java开源ERP与CRM系统】

Compiere ERP&CRM为全球范围内的中小型企业提供综合型解决方案，覆盖从客户管理、供应链到财务管理的全部领域，支持多组织、多币种、多会计模式、多成本计算、多语种、多税制等国际化特性。易于安装、易于实施、易于使用。只需要短短几个小时，您就可以使用申购-采购-发票-付款、报价-订单-发票-收款、产品与定价、资产管理、客户关系、供应商关系、员工关系、经营业绩分析等强大功能了。

Roller Weblogger 【Java开源 Blog博客】

这个weblogging 设计得比较精巧，源代码是很好的学习资料。它支持weblogging应有的特性如：评论功能，所见即所得HTML编辑，TrackBack，提供页面模板，RSS syndication，blogroll管理和提供一个XML-RPC 接口。

Eclipse 【Java开源 开发工具】

Eclipse平台是IBM向开发源码社区捐赠的开发框架，它之所以出名并不是因为IBM宣称投入开发的资金总数—4千万美元，而是因为如此巨大的投入所带来的成果：一个成熟的、精心设计的以及可扩展的体系结构。

XPlanner 【Java开源 项目管理】

XPlanner 一个基于Web的XP团队计划和跟踪工具。XP独特的开发概念如iteration、user stories等，XPlanner都提供了相对应的的管理工具，XPlanner支持XP开发流程，并解决利用XP思想来开发项目所碰到的问题。 XPlanner特点包括：简单的模型规划，虚拟笔记卡(Virtual note cards),iterations、user stories与工作记录的追踪，未完成stories将自动迭代，工作时间追踪，生成团队效率，个人工时报表，SOAP界面支持。

HSQLDB 【Java开源 DBMS数据库】

HSQLDB(Hypersonic SQL)是纯Java开发的关系型数据库,并提供JDBC驱动存取数据。支持ANSI-92 标准 SQL语法。而且他占的空间很小。大约只有160K,拥有快速的数据库引擎。

Liferay 【Java开源 Portal门户】

代表了完整的J2EE应用，使用了Web、EJB以及JMS等技术，特别是其前台界面部分使用Struts框架技术，基于XML的portlet配置文件可以自由地动态扩展，使用了Web Services来支持一些远程信息

的获取，使用 Apahce Lucene实现全文检索功能。
主要特点：
1、提供单一登陆接口，多认证模式（LDAP或SQL）;
2、管理员能通过用户界面轻松管理用户，组，角色；
3、用户能可以根据需要定制个性化的portal layout;
4、能够在主流的J2EE应用服务器上运行，如JBoss+Jetty/Tomcat,JOnAS；
5、支持主流的数据库，如PostgreSQL,MySQL；
6、使用了第三放的开源项目，如Hibernate, Lucene, Struts；
7、支持包括中文在内的多种语言；
8、采用最先进的技术 Java, EJB, JMS, SOAP, XML；

JetSpeed 【Java开源 Portal门户】

Jetspeed 是一个开放源代码的企业信息门户(EIP)的实现，使用的技术是Java和XML。用户可以使用浏览器，支持WAP协议的手机或者其它的设备访问Jetspeed架设的信息门户获取信息。Jetspeed扮演着信息集中器的角色，它能够把信息集中起来并且很容易地提供给用户。
Jetspeed具有如下的特征：
* 即将成为标准化的Java Portlet API
* 基于模板的布局, 包括JSP和Velocity
* 通过开放的内容同步技术支持远程XML内容交换
* 定制默认的主页
* 使用数据库进行用户认证
* 内存缓存技术, 加快页面的响应
* 通过Rich Site Summary技术, 支持同步内容
* 和Cocoon, WebMacro, Velocity集成.
* Wireless Markup Language (WML) 支持
* 使用XML格式的配置文件注册portlet.
* 完整的Web Application Archive (WAR) 支持
* Web应用程序开发的基础设施
* 可以在本地缓存远程内容
* 与Avantgo同步
* 可移植到所有支持JDK1.2和Servlet 2.2的平台
* 与Turbine模块和服务集成
* 可以根据用户, 安装媒体类型和语言的不同设定, 产生不同的个性化服务
* 持续化服务使得所由的portlet能够容易的存储每个用户的状态, 页面和portlet
* 使用皮肤技术使得用户可以选择portlet的颜色和显示属性

* 自定义功能是的管理员可以选择portlet以及定义个人页面的布局
* 在数据库中存储PSML
* 通过Jetspeed的安全portlets管理用户, 组,角色和权限
* 基于角色对访问portlet进行控制

JOnAS 【Java开源 J2EE服务器】

JOnAS 是一个开放源代码的J2EE实现，在ObjectWeb协会中开发。整合了Tomcat或Jetty成为它的Web容器，以确保符合Servlet 2.3和JSP 1.2规范。JOnAS服务器依赖或实现以下的Java API：JCA、JDBC、JTA 、JMS、JMX、JNDI、JAAS、JavaMail 。

JFox3.0 【Java开源 J2EE服务器】

JFox 是 Open Source Java EE Application Server，致力于提供轻量级的Java EE应用服务器，从3.0开始，JFox提供了一个支持模块化的MVC框架，以简化EJB以及Web应用的开发! 如果您正在寻找一个简单、轻量、高效、完善的Java EE开发平台，那么JFox正是您需要的。

JFox 3.0 拥有以下特性：

1. 重新设计的 IoC 微内核，融入 OSGi 模块化思想
2. 设计成嵌入式架构，能够和任何 Java Web Server集成部署
3. 支持 EJB3，JPA规范，支持容器内和容器外两种方式运行EJB和JPA组件
4. 支持 EJB 发布成Web Service
5. 采用 JOTM(http://jotm.objectweb.org/)提供事务处理，支持两阶段提交(2PC)
6. 采用 XAPool(http://forge.objectweb.org/projects/xapool/) 提供 XA DataSource，支持智能连接池管理
7. 内置 MVC 框架，实现自动Form Mapping,Validator,Uploading等功能，支持JSP/Velocity/Freemarker页面引擎，并支持直接在Action中注入EJB
8. 支持多应用模块部署，让中大型应用充分享受模块化开发带来的优势
9. 提供 Manager 管理模块，可以查看和管理各种运行时参数
10. 提供根据 JFox 特色重写的 Petstore 应用模块

Html

HTML简介

HTML（HyperTextMark-upLanguage）即超文本标记语言或超文本链接标示语言，是WWW的描述语言。设计HTML语言的目的是为了能把存放在一台电脑中的文本或图形与另一台电脑中的文本或图形方便地联系在一起，形成有机的整体，人们不用考虑具体信息是在当前电脑上还是在网络的其它电脑上。我们只需使用鼠标在某一文档中点取一个图标，Internet就会马上转到与此图标相关的内容上去，而这些信息可能存放在网络的另一台电脑中。 HTML文本是由HTML命令组成的描述性文本，HTML命令可以说明文字、图形、动画、声音、表格、链接等。HTML的结构包括头部（Head）、主体（Body）两大部分，其中头部描述浏览器所需的信息，而主体则包含所要说明的具体内容。

另外，HTML是网络的通用语言,一种简单、通用的全置标记语言。它允许网页制作人建立文本与图片相结合的复杂页面，这些页面可以被网上任何其他人浏览到，无论使用的是什么类型的电脑或浏览器。
神奇吗?一点都不神奇,因为现在你看到的就是这种语言写的页面！
也许你听说过许多可以编辑网页的软件,事实上,你不需要用任何专门的软件来建立HTML页面；你所需要的只是一个文字处理器（如Mcrosoft Word\记事本\写字板等等）以及HTML的工作常识。其实你很快就会发现,基础的HTML语言简直容易死了。
HTML只不过是组合成一个文本文件的一系列标签。它们像乐队的指挥,告诉乐手们哪里需要停顿,哪里需要激昂。
HTML标签通常是英文词汇的全称（如块引用:blockquote）或缩略语（如“p”代表Paragragh），但它们的与一般文本有区别，因为它们放在单书名号里。故Paragragh标签是<p>,块引用标签是<blockquote>。有些标签说明页面如何被格式化（例如，<p>开始一个新段落），其他则说明这些词如何显示(<b>使文字变粗)还有一些其他标签提供在页面上不显示的信息——例如标题。
关于标签，需要记住的是，它们是成双出现的。每当使用一个标签——如<blockquote>,则必须以另一个标签</blockquote>将它关闭。注意“blockquote”前的斜杠，那就是关闭标签与打开标签的区别。 但是也有一些标签例外。比如，<input>标签就不需要。
基本HTML页面以<html>标签开始，以</html>结束。在它们之间，整个页面有两部分——标题和正文。标题词——夹在<head>和</head>标签之间——这个词语在打开页面时出现在屏幕底部最小化的窗口。正文则夹在<body>和</body>之间——即所有页面的内容所在。页面上显示的任何东西都包含在这两个标签之中。
那么让我们建立一个简单的范例吧，非常容易的。第一步，当然是要建立一个新的文本文件（记住，如果你在使用比较复杂的文字处理器，就应该用“纯文本”或“普通文本”来保存），将它命名为“xxxx.html”。(随便你起一个什么名字,但记住,要用英文)
扩展名也可是HTM

然后你可以用浏览器将它打开，你会看见最简单的自己做的页面。

Asp

目录·概述
·ASP的工作原理
·ASP的运行环境
·ASP的意涵与特性
·ASP常用内置函数
·ASP中Application和Session对象

概述

ASP是Active Server Page的缩写，意为“活动服务器网页”。ASP是微软公司开发的代替CGI脚本程序的一种应用,它可以与数据库和其它程序进行交互，是一种简单、方便的编程工具。ASP的网页文件的格式是.asp，现在常用于各种动态网站中。 ASP是一种服务器端脚本编写环境，可以用来创建和运行动态网页或Web应用程序。ASP网页可以包含HTML标记、普通文本、脚本命令以及COM组件等。利用ASP可以向网页中添加交互式内容（如在线表单），也可以创建使用HTML网页作为用户界面的web应用程序。 与HTML相比，ASP网页具有以下特点：

（1）利用ASP可以实现突破静态网页的一些功能限制，实现动态网页技术；

（2）ASP文件是包含在HTML代码所组成的文件中的，易于修改和测试；

（3）服务器上的ASP解释程序会在服务器端制定ASP程序，并将结果以HTML格式传送到客户端浏览器上，因此使用各种浏览器都可以正常浏览ASP所产生的网页；

（4）ASP提供了一些内置对象，使用这些对象可以使服务器端脚本功能更强。例如可以从web浏览器中获取用户通过HTML表单提交的信息，并在脚本中对这些信息进行处理，然后向web浏览器发送信息；

（5）ASP可以使用服务器端ActiveX组件来执行各种各样的任务，例如存取数据库、发现哦那个Email或访问文件系统等。

（6）由于服务器是将ASP程序执行的结果以HTML格式传回客户端浏览器，因此使用者不会看到ASP所编写的原始程序代码，可防止ASP程序代码被窃取。

ASP的工作原理

当在Web站点中融入ASP功能后，将发生以下事情：

1、用户向浏览器地址栏输入网址，默认页面的扩展名是.asp。

2、浏览器向服务器发出请求。

3、服务器引擎开始运行ASP程序。

4、ASP文件按照从上到下的顺序开始处理，执行脚本命令，执行HTML页面内容。

5、页面信息发送到浏览器。

ASP的运行环境

asp需要运行在PWS或IIS下。PWS或IIS服务在windows98或windows2000的光盘上附带着，可以通过“添加/删除程序”中的“添加/删除windows组件”来安装。

一般asp需与access数据库或SQL Server数据库结合使用，编出功能强大的程序。

能够运行ASP的web服务器软件

Windows2000默认安装的是IIS5.0(internet information server)，而windows xp默认安装的是IIS5.1，windows 2003默认安装的IIS6.0。
PWS(personal web server)运行在windows98环境下的简单个人网页服务器。

ASP的意涵与特性
ASP(Application Service Provider,应用软体租赁服务提供者)即是指「透过网路以租赁方式提供应用软体服务的业者」,即是指业者以应用软体为主体,透过网路一对多地传递服务,这种以服务为主的交易模式促使企业可藉由租赁的方式,以更符合成本效益的方式拥有软体的使用权,并且亦能因为业者集中式的管理而大幅降低企业维护的成本.
基本上,ASP即具有「软体服务化,服务网路化」,「资讯委外服务与网路结合」与「产品通路化,通路产品化」等三大特性,其甚至可以被视为是ISP（Internet Service Provider）与ITS（Information Technology Service）的结合.
ASP的英文是Application Service Provider，中文的标准翻译就是“应用服务提 供商”，是指为商业或者个人客户提供管理应用解决方案的公司或者企业。最近 ASP被媒体炒做十分火热，不是IT行业的人面对一堆技术名词专业术语很难弄清楚ASP的内容，本文试图用浅显的语言来为广大的读者揭开ASP神秘的面纱，对于IT行业的大热门ASP领域有一些基本的认识。
1. ASP是什么东西？
简单地讲，ASP就是为客户提供服务的服务商，它和会计事物所、婚姻介绍所没有什么本质方面的区别。不同的是ASP主要是通过INTERNET（国际互联网络）作为主要工作和业务工具，采用一对多的方式，向企业、公司提供标准化的应用软件以及相关的技术咨询、管理租赁的服务，ASP的概念最早是1998年由美国人提出来的。目前被全球各大IT厂商看好并被认为是可以推动网络经济发展的，有稳固基础的第三种网络商业模式。
和传统的外包服务（Outsourcing）相比，ASP的主要区别在于：ASP是一对多的经营模式，提供的服务有兼容性和可协调性，并且ASP的收费方式一般是按月收费。
业界认为：ASP一般有这样一些“成员”：电信运营商、传统IT服务厂商、互联网络接入服务商（ISP）、独立软件供应商（ISV）、系统集成商和单纯的ASP公司。
根据流行的观点，ASP有如下五个核心内涵：
a. ASP着重应用为中心，提供对于应用方面的访问和管理。
b. ASP服务可以为用户提供没有在服务器、人员、系统和系统授权等前期资源投入情况下就可以在“定制”的全新应用系统环境进行访问的服务，如ISP，而这样的服务一般按月份ASP收取服务费。
c. ASP采用集中管理的方式---ASP一般都有一个管理中心，所有的客户通过INTERNET来进行远程访问，获得技术支持和咨询服务。
d. 一对多的服务，也就是讲，ASP提供的是标准化的产品包，产品都是最低程度的自定义或者没有实

现客户定制化，对于行业用户来讲已经达到实用方便的标准。
e. 按照合同交付，在ASP客户的眼中，ASP是一家根据客户协议内容提供相关服务，保证应用服务系统服务可以得到确实履行的机构。
2. ASP为什么会火？
从大的方面来讲，困扰国内企业生存、发展的核心问题是管理问题。随着互联网络的普及和应用的深入，企业用户可以随时随地直接租用ASP的服务器和软件系统来进行自己的业务管理，这样做的好处在于；第一，企业可以节省大笔用于IT建设方面的资金，大幅度降低企业管理信息化的成本。第二，ASP的用户可以采用各种方式获得应用和服务，软件类服务产品完全可以通过网络在非常短的时间内组成一个完善的、高效的、先进的企业管理系统，迅速获得企业一体化的运营管理方案。
网络经济发展突飞猛进，电子商务一日千里，网络和网站从门户到内容、从注意力到垂直性，目前逐渐转向热衷ASP也是一个主要的原因。
笔者资料中，国内最早对ASP触电的是网友“飞鸟”，在1998年6月自发组织了研究、交流和探讨ASP技术的“飞鸟之家”，现在已经发展成为chinaasp.com，成为国内最早的ASP应用技术服务提供商网站。上海的互易网络有限公司结合国内实际情况，推出了为国内企业服务的ASP平台互易网，向企业提供以电子商务为核心的，企业内、外部网络设施和应用的远程构架和托管服务，创造虚拟企业门户（EP）直接将ASP应用到商业增殖环节中去。
此外，ISP也全面转向ASP的怀抱，成为ISP进一步发展的产物。软件商对于ASP更是情有独钟。业界最新的消息是，中国第三电信“网通”已经制定ASP发展战略，国内最大的管理咨询公司“汉普”将把旗下八个子公司定位在企业内部资源计划管理（ERP）领域的ASP中，北京“联成互动”瞄准客户关系管理（CRM）领域的ASP，北京“数码方舟”定位在网络办公的ASP，HP正在和中国建设银行讨论共建金融领域的ASP。
ASP正在IT经济大潮中显山露水，其发展前景不可估量。
3. ASP的发展阶段和面临的问题
以网络服务商、软件厂商和ISP为主力的各种IT角色，正在根据自己的优势条件出发对ASP领域进行多种方面的尝试。就目前阶段来讲，ASP提供的服务不计其数五花八门，没有标准化和量化的概念，硬件厂商向ASP的“土壤”和势力方向靠拢，软件厂商和ISP则直接参与到ASP业务的第一线。
笔者估计，经过一段时间的试探和发展，ASP将向服务集成方面发展，产品和服务初步的标准化将很快建立起来，接着进入到市场细化和标准制定、ASP产品成熟时期，ASP的稳步增长，最终将成为IT行业商务模式的核心！
目前在ASP发展的道路上，主要面临的问题是观念的转变方面：用租赁代替购买，服务集成代替产品经销商、服务经济代替产品经济等等。具体到实际方面来讲，安全和服务的质量是ASP和客户共同关心的头等大事，要实现ASP提出的“租赁高科技”的口号，ASP任重而道远！

ASP常用内置函数

1，日期/时间函数

这些函数包括对“年”、“月”、“日”、“时”、“分”、“秒”、“星期”等的显示。

（1）Now函数：根据计算机系统设定的日期和时间，返回当前的日期和时间值。使用方法now()；

（2）Date函数：只返回当前计算机系统设定的日期值。使用方法：date()；

（3）Time函数：只返回当前计算机系统设定的时间值。使用方法：time()；

（4）Year函数：返回一个代表某年的整数。使用方法：year(date)，其中date参数是任意的可以代表日期的参数，比如“year(date())”就表示是从“date()”得出的日期中提取其中“年”的整数。

另外，还可以这样应用：“year(#5 20,2006#)”表示提取“2006年5月20日”中“年”的整数值。关于“5 20,2006”，也可使用“5-20-2006”、“5/20/2006”等形式表现，即“某月某日”和“某年”的组合。同时注意使用“#”进行包括以表示日期值。

（5）Month函数：返回1到12之间的整数值，表示一年中某月。使用方法：month(date)。关于参数date的说明和year函数相同。但要注意日期的正确性，比如“#13-31-2006#”，根本就没有“13”月，肯定是错误的了。

（6）Day函数：返回1到31之间的整数值，表示一个月中的某天。使用方法：day(date)。关于参数date的说明和year函数相同。同样要注意日期的正确性，比如“#2-30-2006#”其中对“2”月定义的“30”日这天就是错误的。

（7）Hour函数：返回0到23之间的整数值，表示一天中的某个小时。使用方法：hour(time)。其中参数time是任意的可代表时间的表达式。比如“hour(time())”就表示是从“time()”得出的时间中提取其中“小时”的整数。同样，参数time还可以这样应用“hour(#11:45:50#)”表示从“11”时“45”分“50”秒中提取当前小时数。当然，定义的时间要符合时间的规范。

（8）Minute函数：返回0到59之间的整数值，表示一小时中的某分钟。使用方法：minute(time)。time参数的说明和hour函数相同。

（9）Second函数：返回0到59之间的整数值，表示一分钟中的某秒。使用方法：second(time)。time参数的说明和hour函数相同。

（10）Weekday函数：返回一个星期中某天的整数。使用方法：weekday(date)。关于参数date的说明和year函数相同。该函数返回值为“1”到“7”，分别代表“星期日”、“星期一”……“星期六”。比如当返回值是“4”时就表示“星期三”。

（11）WeekDayName函数：返回一个星期中具体某天的字符串。相对weekday函数而言即翻译出“星期几”，使用方法：weekdayname(weekday)。参数weekday即星期中具体某天的数值。比如“weekdayname(weekday(date()))”就表示当前是“星期几”。因为“date()”表示的是当前的时间，而“weekday(date())”就表示的是一星期中具体某天的整数。

当然weekdayname函数最终显示的字符串内容还与当前操作系统语系有关，比如中文操作系统将显示“星期一”这类的中文字符，而英文操作系统则显示为“Mon”（Monday简写）。

此外，在VBScript中还有一些关于时间间隔的计算函数：
（1）DateAdd函数：返回指定时间间隔的日期、时间。可以计算出相隔多少年、或相隔几个月、又或相隔几个小时等的新日期、时间。使用方法：dateadd(interval, number, date)。
其中参数interval表示需要添加的时间间隔单位。其是以字符串的形式表达的，比如“yyyy”表示年，“q”表示季度，“m”表示月份，“d”表示天数，“ww”表示周数，“h”表示小时数，“n”表示分钟数，“s”表示秒数。而参数number则表示添加的时间间隔数。其是以数值的形式表达的，可以为负值。参数date则要求是日期、时间的正确格式。
比如dateadd("d",100,"2006-5-20")就表示2006年5月20号以后的100天的日期值：2006-8-28。再比如dateadd("h",-12,"2005-5-20 10:00:00")就表示2005年5月20号上午10点前的12小时的日期时间：2005-5-19 22:00:00。
（2）DateDiff函数：返回两个日期时间之间的间隔。可计算出两个日期相隔的年代、小时数等。使用方法：datediff(interval,date1,date2)。
参数interval和dateadd函数中的interval参数内容描述相同，date1和date2参数分别就是相互比较的两个日期时间。另外，当date1的日期时间值大于date2时，将显示为负值。
比如DateDiff("yyyy","1982-7-18",date)表示某人的出生到现在已经多少年了。又比如DateDiff("d","1982-7-18","2062-7-18")则计算了80年过了多少天：29220。
2，字符串处理函数
在脚本的功能处理中，通常需要对一些字符串进行一些修饰性处理。比如过滤掉字符串中的敏感字眼以符合最终显示的要求；又比如一段较长的字符串，需要提取开头的几个字符时。
（1）Asc函数：返回字符串中第一个字母对应的ANSI字符代码。使用方法：asc(string)。其中string参数表示字符串。
（2）Chr函数：返回指定了ANSI 字符代码对应的字符。使用方法：chr(chrcode)。参数chrcode是相关的标识数字。该函数的功能和asc函数形成对应。
比如：asc(“a”)表示小写字母“a”的ANSI 字符“97”；同样chr(97)表示的就是“小写字母a”。另外chr(chrcode)中参数chrcode值为0到31的数字时，表示不可打印的ASCII码。比如“chr(10)”表示换行符，“chr(13)”表示回车符等，这常用于输入和显示格式的转换中。
（3）Len函数：返回字符串内字符的数目（字节数）。使用方法：len(string)。比如len(“love”)的值就是4。
（4）LCase函数：返回所有字符串的小写形式。使用方法：lcase(string)。比如lcase(“CNBruce”)返回为“cnbruce”。
（5）UCase函数：返回所有字符串的大写形式。与lcase函数形成对应。同样，ucase(“CNBruce”)返回为“CNBRUCE”。
（6）Trim函数、LTrim函数和RTrim函数：分别返回前导和后续不带空格、前导不带空格或后续不带空格的字符串内容。比如：

trim(“ cnbruce ”)返回为“cnbruce”，前导和后续都不带空格；
ltrim(“ cnbruce ”)返回为“cnbruce ”，前导不带空格;
rtrim(“ cnbruce ”)返回为“ cnbruce”，后续不带空格；
该函数常用于注册信息中，比如确保注册用户名前或后的空格。
（7）Left函数：返回从字符串的左边算起的指定数目的字符。使用方法：left(string,length)。比如left(“brousce”,5)返回为“brous”，即前五位字符。
（8）Right函数：返回从字符串的左边算起的指定数目的字符。使用方法：right(string,length)。比如right(“brousce”,4)返回为“usce”，即后四位字符。
（9）instr函数：返回某字符串在另一字符串中第一次出现的位置。比如现在查找字母“A”在字符串“A110B121C119D1861”中第一次出现的位置，则可以 instr(my_string,"A110B121C119D1861")
（10）Mid函数：从字符串中返回指定数目的字符。比如现在的“110”则应该是从字符串“A110B121C119D1861”的第2位取得3个单位的值：mid("A110B121C119D1861",2,3)
（11）Replace函数：在字符串中查找、替代指定的字符串。replace(strtobesearched,strsearchfor,strreplacewith)其中strtobesearched是字符串，strsearchfor是被查找的子字符串，strreplacewith是用来替代的子字符串。比如 replace(rscon,"<","<") 则表示将rscon中所有“<”的字符替换为“<”
3，类型转换函数
Cbool(string) 转换为布尔值
Cbyte(string) 转换为字节类型的值
Ccur(string) 转换为货币类值
Cdate(string) 转换为日前类型的值
Cdbl(string) 转换为双精度值
Cint(string) 转换为整数值
Clng(string) 转换为长整型的值
Csng(string) 转换为单精度的值
Cstr(var) 转换为字符串值
Str(var) 数值转换为字符串
Val(string) 字符串转换为数值
4，运算函数
Abs(nmb) 返回数子的绝对值
Atn(nmb) 返回一个数的反正切
Cos(nmb) 返回一个角度的余炫值
Exp(nmb) 返回自然指数的次方值
Int(nmb) 返回数字的整形（进位）部份

Fix(nmb) 返回数字的整形（舍去）部份
Formatpercent(表达式) 返回百分比
Hex(nmb) 返回数据的16进制数
Log(nmb) 返回自然对数
Oct(nmb) 返回数字的8进制数
Rnd 返回大于“0”而小于“1”的随机数，但此前需 randomize 声明产生随机种子
Sgn(nmb) 判断一个数字的正负号
Sin(nmb) 返回角度的正铉值
Sqr(nmb) 返回数字的二次方根
Tan(nmb) 返回一个数的正切值
5，其他函数
IsArray(var) 判断一个变量是否是数组
IsDate(var) 判断一个变量是否是日期
IsNull(var) 判断一个变量是否为空
IsNumeric(var) 判断表达式是否包含数值
IsObject(var) 判断一个变量是否是对象
TypeName(var) 返回变量的数据类型
Array(list) 返回数组
Split(liststr) 从一个列表字符串中返回一个一维数组
LBound(arrayP 返回数组的最小索引
Ubound(array) 返回数组的最大索引
CreateObject(class) 创建一个对象
GetObject(pathfilename) 得到文件对象

ASP中Application和Session对象
一、Application对象的成员概述
Application对象成员包括Application对象的集合、方法和事件。
1. Application对象的集合
Contents集合：没有使用元素定义的存储于Applicaiton对象中的所有变量的集合
StaticObjects:使用元素定义的存储于Application对象中的所有变量的集合
例：在default.asp中有如下赋值
application("a")="a"
application("b")=128

application("c")=false
则有contents集合
application.contents(1)="a" '也可写为application.contents("a")="a"
application.contents(2)=128 '也可写为application.contents("b")=128
application.contents(3)=false '也可写为application.contents("c")=false
在此笔者推荐你在调用时使用类如application.contents("a")的方法，因为这样更为直观，如果用序号来表示的话则要考虑赋值的先后顺序。
2.Application对象的方法
Contents.Remove("变量名")：从Application.Contents集合中删除指定的变量
Contents.RemoveAll() ：把Application.Contents集合中的所有变量删除
Lock() ：锁定Application对象，使得只有当前的ASP页对内容能进行访问
Unlock() ：解除对Application对象的锁定
例：在default.asp中：

```
application("a")="a"
application("b")=128
application("c")=false
response.write application.contents(1)&"
"
response.write application.contents(2)&"
"
response.write application.contents(3)&"
"
response.write "After Remove b:"
application.contents.remove("b")
response.write application.contents(1)&"
"
response.write application.contents(2)&"
"
```

执行结果：

```
a
128
False
After Remove b:
a
```

False

如果要删除集合中所有变量用application.contents.removeall即可，至于Lock和Unlock方法在实际中经常用到，读者也比较熟悉，在此就不在累赘。

3.Application对象事件

OnStart：第一个访问服务器的用户第一次访问某一页面时发生

OnEnd ：当最后一个用户的会话已经结束并且该会话的OnEnd事件所有代码已经执行完毕后发生，或最后一个用户访问服务器一段时间（一般为20分钟）后仍然没有人访问该服务器产生。

想要定义application对象的OnStart和OnEnd事件里做什么需要将代码写在Global.asa这个文件里（下文有举例），并且将该文件放在站点的根目录下（一般是Inetpub\wwwroot\）

二、Session对象的成员概述

Session对象的成员比Application对象多一项属性，即：集合、属性、方法、事件

1.Session对象的集合

Contents ：没有使用元素定义的存储于特定Session对象的所有变量的集合。

StaticObject：使用元素定义的、存储于Session对象中的所有变量的集合。

例：在default.asp中有如下赋值

```
session("a")="a"
session("b")=128
session("c")=false
```

则有contents集合

```
session.contents(1)="a" '也可写为session.contents("a")="a"
session.contents(2)=128 '也可写为session.contents("b")=128
session.contents(3)=false '也可写为session.contents("c")=false
```

2.Session对象的属性

CodePage: 可读/可写。整型。定义用于在浏览器中显示页内容的代码页。代码页是字符集的数字值，不同的语言使用不同的代码页。例如，ANSI代码页为1252，日文代码页为932，简体中文代码页为936。

LCID : 可读/可写。整型。定义发送给浏览器的页面地区标识。LCID是唯一地标识地区的一个国际标准缩写，例如，2057定义当前地区的货币符号是"£"。

SessionID: 只读。长整型。返回本会话的会话标识符。每创建一个会话，由服务器自动分配一个标识符。可以根据它的值判断两个用户是谁先访问服务器。

Timeout : 可读/可写。整型。为会话定义以分钟为单位的超时限定。如果用户在这个时间内没有刷新或请求任何一个网页，则该用户产生的会话自动结束。缺省值是20。

以上属性在实际应用中作用不大，而且基本上不需要怎么修改，这几个属性也没什么特殊的地方。

3.Session对象的方法

Contents.Remove("变量名"): 从Session.contents集合中删除指定的变量

Contents.Removeall() : 删除Session.contents集合中的所有变量

Abandon() : 结束当前用户会话并且撤消当前Session对象。

Session对象的Contents.Remove("变量名")和Contents.Removeall()方法与Application对象的基本上没什么区别，为帮助理解，大家可以参照上面的例子将Application改为Session。这里要说明一下的是Contents.Removeall()和Abandon()的区别，执行这两个方法都会释放当前

用户会话的所有Session变量，不同的是Contents.Removeall()单纯地释放Session变量的值而不终止当前的会话，而Abandon()除了释放Session变量外还会终止会话引发Session_OnEnd事件，希望大家注意两者的区别。

4.Session对象的事件

OnStart: 当ASP用户会话产生时触发，一旦有任一用户对本服务器请求任一页面即产生该事件。

OnEnd : 当ASP用户会话结束时触发，当使用Abandon()方法或超时也会触发该事件。

这两个事件和Application的OnStart、OnEnd事件一样，也是必须放在Global.asa文件里，下面就重点和大家研究一下这四个事件的使用。

三、Global.asa

ASP的Application和Session对象体现了其他ASP内置对象所没有的特征--事件。每一个访客访问服务器时都会触发一个OnStart事件（第一个访客会同时触发Application和Session的OnStart事件，但Application先于Session），每个访客的会话结束时都会触发一个OnEnd事件（最后一个访客会话结束时会同时触发Application和Session的OnEnd事件，但Session先于Application）。

OnStart和OnEnd这两个事件一般应用在虚拟社区中统计在线人数、修改用户的在线离线状态等。要具体定义这两个事件，需要将代码写在Global.asa文件，并将该文件放在站点的根目录下（缺省是\Inetpub\wwwroot\）。另外，Application和Session对象规定了在OnEnd事件里除了Application对象外其他ASP内置对象（Response、Request、Server、Session...）一概不能使用。以下举一个虚拟社区统计在线人数的例子来说明如何使用这两个事件。

文件说明：

global.asa 位于d:\Inetpub\wwwroot\目录下

default.asp 位于d:\Inetpub\wwwroot\目录下，虚拟社区登录页面

login.asp 位于d:\Inetpub\wwwroot\目录下，用于检测用户输入的用户名及密码

index.asp 位于d:\Inetpub\wwwroot\目录下，虚拟社区首页

bbs.mdb 位于d:\Inetpub\wwwroot\目录下，存储用户信息的数据库

数据库（ACCESS）结构：

===bbs表===

id 用户ID，长整型

name 用户名，文本型

```
    code 密码，文本型
    online 在线状态，是/否
===global.asa===
    ＜script LANGUAGE="VBScript" RUNAT="Server"＞
    Sub Application_OnStart
        application("online")=0
    End Sub
    sub Application_OnEnd
    nd Sub
    Sub Session_OnStart
    End Sub
    Sub Session_OnEnd
        if session.contents("pass") then '判断是否为登录用户的Session_OnEnd
            application.lock
            application("online")=application("online")-1
            application.unlock
        end if
    End Sub
    ＜/script＞
    ===login.asp===
    .....'密码验证，连接数据库，检测用户输入的用户名及密码是否正确
    if 密码验证通过 then
        session("name")=rs("name")
        session("id")=rs("id")
        session("pass")=true
    else
        rs.close
        conn.close
        response.write "密码错误！"
        response.end
    end if
    application.lock
    application("online")=application("online")+1
    conn.Execute ("update bbs set online=1 where id="&session("id"))'将用户的状态设为在线
```

```
application.unlock
rs.close
conn.close
response.redirect "index.asp" '初始化数据后跳转到社区首页
==========
```

在本例中，用application("online")变量记录已经登录社区的在线人数，因为一旦有用户访问服务器而不管用户是否登录，都会产生OnStart事件，所以不能在OnStart事件里使Applicaiton("online")加一。因为不管是否是登录用户的会话结束都会产生OnEnd事件（假如有访客访问了服务器但并不登录社区，他的会话结束后也会产生OnEnd事件），所以在Session_OnEnd事件里用了句if语句来判断是否为已登录用户的OnEnd事件，如果是才将在线人数减一。

这只是一个统计在线人数的简单例子，对于一个完整的虚拟社区来说，仅仅统计有多少人在线是不够的，在本例中数据库里有个online字段是用来记录用户的在线状态，用户登录的时候，在login.asp里将online设为1，但用户离线时并没有将online设为0，要完善它，就要修改一下Session_OnEnd事件，在该事件里将online设为0。

```
===ｇｌｏｂａｌ・ｓａｓ===
＜script LANGUAGE="VBScript" RUNAT="Server"＞
Sub Application_OnStart
   application("online")=0
   set application("conn")=Server.CreateObject("ADODB.Connection")
   application("db")=Server.MapPath("\bbs.mdb") '此处最好使用绝对路径\bbs.mdb，下文有详细介绍
End Sub
sub Application_OnEnd
   set application("conn")=nothing
End Sub
   Sub Session_OnStart
End Sub
Sub Session_OnEnd
   if session.contents("pass") then '判断是否为登录用户的Session_OnEnd
        application("con").open ="driver={Microsoft Access Driver (*.mdb)};dbq="&application("db")
         application.lock
        application("online")=application("online")-1
        application("con").Execute ("update friends set online=0 where id="&session.contents("id"))
        application.unlock
```

```
            application("con").close
        end if
    End Sub
    </script>
    ==============
```

至此，完整的代码已经完成了。因为在Application和Session的OnEnd事件里不能使用Server对象，所以要将数据库的连接及数据库在服务器上的物理地址（d:\inetpub\wwwroot\bbs.mdb）存储在application变量中，并在Application_OnStart事件中预先处理。同理，在Session_OnEnd事件中不能用session("pass")来代替session.contents("pass")（以下有详尽说明）。

四、本文实例中值得引起注意的两点

1. OnEnd事件里的session.contents

刚开始接触global.asa的朋友经常会将上面Session_OnEnd事件里的

if session.contents("pass") then　写成

if session("pass") then，

这样的话系统不会提示错误，但是永远也不会执行then后面的内容，这是因为在OnEnd事件里禁止使用Session对象，但是可以用Session对象的集合来调用session变量。因为IIS并没提示任何错误信息，所以笔者曾经在这上面浪费了很多时间。在此希望大家引以为鉴！

2. Application_OnStart事件里用Server.MapPath获取数据库的物理地址时应使用绝对地址为了说明这个问题，大家可以做个实验：将上面Application_OnStart事件里的

application("db")=Server.MapPath("\bbs.mdb")改为:

application("db")=Server.MapPath("bbs.mdb")

然后在d:\inetpub\wwwroot\目录下建立一个test子目录，写一个temp.asp在test目录里。

```
    ====test.asp====
    <%response.write application("db")%>
    ================
```

再将temp.asp拷贝一份放在根目录下（d:\inetpub\wwwroot\）。用记事本打开global.asa，再打开两个浏览器，浏览器A输入地址http://localhost/temp.asp，按回车，将在浏览器上输出：

d:\inetpub\wwwroot\bbs.mdb

然后，在记事本的窗口上点"文件"菜单，选"保存"（使global.asa的修改时间改变，从而使IIS重启动所有服务），再在浏览器B输入地址http://localhost/test/temp.asp，按回车，在浏览器上输出的是：

d:\inetpub\wwwroot\test\bbs.mdb

global.asa文件虽然是放在站点根目录下，但是如果在server.mappath中使用的是相对地址，而触发Application_OnStart事件的用户第一次访问的页面又不是属于根目录的话，得到数据库的物理地址将不会是期望的结果，希望大家要特别小心。

类别：Web | 编辑 | 删除
| 评论(0) | 浏览(0)
全面解析ASP Server对象
2007-07-27 12:40
Server对象提供对服务器上访问的方法和属性.大多数方法和属性是作为实用程序的功能提供的。

语法:

Server.property|method

属性（property）

Server对象只有一个属性：ScriptTimeout 程序能够运行的最大时间

方法（Methods）

CreateObject 建立一个对象实例.

Execute 执行一个asp文件

GetLastError 返回一个错误代码

HTMLEncode 对指定的HTML代码进行转换.

MapPath 将一个相对路径转化为一个绝对路径.

Transfer 将当前的所有状态信息发送给另一个asp文件

URLEncode 以URL形式转化指定的代码,包括空格

Server对象的方法详细说明

CreateObject

语法

Server.CreateObject(progID)

参数

progID

指定要创建的组件名称,格式如下: [Vendor.]Component[.Version].

要点：

一般来说,用由Server.CreateObject方法创建的对象拥有页面的范围.这就说,当这页的asp程序执行完后,这种对象会自动地消失.

为了创建一个拥有Session或Application范围的对象,你可以在Global.asa文件中使用

Execute

Execute 方法呼叫一个ASP文件并且执行它就像这个呼叫的ASP文件存在这个ASP文件中一样。这很像许多语言中的类的调用。

语法

Server.Execute(Path)

参数

Path

指定执行的那个asp文件的路径。如是它是一个绝对路径，那么它必须是一个在这个ASP应用程序相同的地方（目录）。

讲解

Server.Execute 方法提供了一种将一个复杂ASP应用程序分化为小块单位来执行的方法。通过这种方法，你能够建一个ASP图书馆，你能够随便在你需要时调用你图书馆中的ASP文件。这个就有点像SSI了！嘿嘿！

当IIS根据指定的ASP文件路径执行完这个ASP文件之后，就会自动返回以前的ASP文件。这个刚刚执行完的ASP文件有可能改变了HTTP head.但是和其它的ASP文件一样，当程序试图改变http head时，就会报错！

这个path参数可以包括一个询问信息。

如果在被呼叫和呼叫的ASP文件中都含有相同的子函数，那么这些子函数只在本ASP文件中起作用。举个例子，如果在下面的ASP1和ASP2两个文件中都含有放弃程序的子函数。首先ASP1呼叫ASP2，那么ASP2中的的OnTransactionAbort开始执行，当ASP2执行完毕，ASP1中的OnTransactionAbort才开始执行。

```
ASP1:
＜%@ Transaction=
Required
%＞
＜%
  Server.Execute ("Page22.asp")
  Sub OnTransactionAbort
  Sub OnTransactionCommit
%＞

Asp2.asp:
＜%@
  Transaction=Required
  Sub OnTransactionAbort
  Sub OnTransactionCommit
%＞
```

Example

ASP1
＜% Response.Write("I am going to execute ASP2 ")
Server.Execute("/myasps/asp2.asp")
%＞
ASP2
＜% Response.Write("Here I am")%＞

GetLastError

GetLastError 方法返回一个ASPError Object 来描述一个错误信息.这个方法只适用于在asp文件发送任何内容给用户机之前.

语法

Server.GetLastError ()

要点

如果一个500;100 用户错误已经被定义在一个asp应用程序中，它是指的一个以.asp为后缀的文件。这种情况下，在这个程序运行时当一个错误发生时，服务器就会自动的以Server.Transfer这种方式传送到这个正在执行的ASP页面。ASP应用程序就会将有效的处理这个错误。另外，这个ASPError Object一定要有效，这样你就能够看到服务器提供给你的错误信息来改这个文件了！

一般的Web Site 都是根据文件\iishelp\common\500-100.asp来构造的。你能够用它来执行一个asp错误，当然你能够自己定义了！。如果你想改变为另外一个asp文件的来执行这些用户错误。那么你可以用IIS中的snap-in.

注意：当IIS发现了一个asp文件或者global.asa文件中的一个错误，那么一个500;100用户错误产生。以下的程序将不能执行！

Example

下面的三个例子证明不同的错误会产生的用户错误。三个错误是：

编译错误

运行错误

逻辑错误

第一个例子证明了一个编译错误，就是当IIS试图包含一个文件时产生的。这个错误会产生是因为在这个包含文件中没有定义所需的参数。第二个例子显示的是一个运行错误，这个程序中断的原因是程序中没有“next".第三个例子显示的是一个逻辑错误，因为这个程序试图除以一个0. 不行啦！

Example 1
＜%
response.write "hello"
%＞

Example 2

```
<%
    dim I
    for i=1 to 1
    nxt
%>
```

Example 3

```
<%
    dim i,j
    dim sum
    sum=0
    j=0
    for i=1 to 10
      sum=sum+1
    next
    sum=sum/j
%>
```

HTMLEncode

HTMLEncode方法对指定的字符串进行HTML编码.

语法

```
Server.HTMLEncode( string )
```

参数

string 要进行编码的字符

例子

下面的程序:

```
<%= Server.HTMLEncode("The paragraph tag: ") %>
```

输出为:

The paragraph tag:

注意 程序执行后在浏览器中看到的是:

The paragraph tag:

但是如果你用"查看源文件"看一下的话,源代码就不是了.

MapPath

MapPath 方法将相对路径转化为服务器上的物理路径

语法

Server.MapPath(Path)

参数

Path

相对路径。这个路径是以"/"或"\"开头的路径，如果这个路径中没有"\",那么MapPath方法就会返回以当前目录为基础的路径。

讲解

MapPath 方法不能检查路径在这个服务器下是否存在。因为 MapPath 转化路径时是不管这个路径是否在这个服务器下存在的。

你能够用它来将一个相对路径转化为一个物理路径，然后再在这个路径下进行各种操作。

Example

在下面的例子中，data.txt文件存在 C:\Inetpub\Wwwroot\Script 目录中，而且一个test.asp 文件包括下面的代码。C:\Inetpub\Wwwroot 是该服务器的主目录 。

下面的例子中，首先用环境变量"PATH_INFO"获得当前文件的物理路径。

下面是Script 代码：

＜ %= server.mappath(Request.ServerVariables("PATH_INFO"))%＞

显示为：

c:\inetpub\wwwroot\script\test.asp

因为下面的例子中路径参数没有以"/"开头，所以它是以当前目录转化的，asp文件是放在C:\Inetpub\Wwwroot\Script中的.以下是 scripts的内容：

＜ %= server.mappath("data.txt")%＞

＜ %= server.mappath("script/data.txt")%＞

显示为：

c:\inetpub\wwwroot\script\data.txt

c:\inetpub\wwwroot\script\script\data.txt

以下的两个例子是以"/"开头的.以下是scripts的内容：

＜ %= server.mappath("\script")%＞

显示为：

c:\inetpub\wwwroot\script\data.txt

c:\inetpub\wwwroot\script

直接用"/"或"\"就会得到服务器的主目录：

＜ %= server.mappath("\")%＞

显示为：

c:\inetpub\wwwroot

c:\inetpub\wwwroot
Transfer
transfer 方法会把一个正在执行的asp文件的所有信息传给另外一人asp文件。
语法
Server.Transfer (path)
参数
Path
将要接收信息的asp文件的位置。
要点
当你调用Server.Transfer时，所有内建对象的状态信息都会包含在这次传送之中。这就是说，所有在保存在Session或Application中的信息都会被传送，而且，所有当前请求的信息都会被接收信息的asp文件所接受。
Example
下面的例子示范了从一个asp文件传送到另一个asp文件例子！
ASP1

```
＜ % Dim sessvar1 Response.Write Session.SessionID
   Response.Write ("")
   Response.Write("I am going to ASP2 ")
   Server.
Transfer
("/Myasps/ASP2.asp")
% ＞
```

ASP2

```
＜ % Response.Write Session.SessionID %＞
```

URLEncode
URLEncode 方法可以将指定字符串进行URL编码。
语法
Server.URLEncode(string)
参数
string 指定要转化的字符串
Example
下面是代码：

```
＜ % Respones.Write(Server.URLEncode("http://www.microsoft.com")) % ＞
```

显示为：

http%3A%2F%2Fwww%2Emicrosoft%2Ecom

属性：ScriptTimeout

ScriptTimeout 属性规定了程序的最大运行时间。

语法

Server.ScriptTimeout = NumSeconds

参数

NumSeconds

规定了程序的最大的运行时间（以秒计算）。缺省值是90秒

Remarks

一个缺省的Scritpt Timeout的值会能过ASPScriptTimeOUT属性来设置在Web sertvic 或 Web server上。在程序中，ScriptTimeout属性的值不能小于这个缺省值。举个例子吧，如果NumSeconds我们设置为10秒，而缺省值为90秒，那么程序就会中止在90秒以后，而不是10秒以后的。同样，如果我们设置ScriptTimeout的值为100秒，那么，程序就会在100秒之后中止，而不是90秒。

Example

下面的例了中程序将被设置为100秒后自动中止。

＜ % Server.ScriptTimeout = 100 %＞

下面的例子中将重新得到ScriptTimeout的值，然后把它存在Timout变量中

＜ % TimeOut = Server.ScriptTimeout %＞

ASP内置对象Request的ServerVariables集合列表

Request.ServerVariables("Url")
返回服务器地址
Request.ServerVariables("Path_Info")
客户端提供的路径信息
Request.ServerVariables("Appl_Physical_Path")
与应用程序元数据库路径相应的物理路径
Request.ServerVariables("Path_Translated")
通过由虚拟至物理的映射后得到的路径
Request.ServerVariables("Script_Name")
执行脚本的名称
Request.ServerVariables("Query_String")
查询字符串内容

Request.ServerVariables("Http_Referer")
请求的字符串内容
Request.ServerVariables("Server_Port")
接受请求的服务器端口号
Request.ServerVariables("Remote_Addr")
发出请求的远程主机的IP地址
Request.ServerVariables("Remote_Host")
发出请求的远程主机名称
Request.ServerVariables("Local_Addr")
返回接受请求的服务器地址
Request.ServerVariables("Http_Host")
返回服务器地址
Request.ServerVariables("Server_Name")
服务器的主机名、DNS地址或IP地址
Request.ServerVariables("Request_Method")
提出请求的方法比如GET、HEAD、POST等等
Request.ServerVariables("Server_Port_Secure")
如果接受请求的服务器端口为安全端口时，则为1，否则为0
Request.ServerVariables("Server_Protocol")
服务器使用的协议的名称和版本
Request.ServerVariables("Server_Software")
应答请求并运行网关的服务器软件的名称和版本
Request.ServerVariables("All_Http")
客户端发送的所有HTTP标头，前缀HTTP_
Request.ServerVariables("All_Raw")
客户端发送的所有HTTP标头,其结果和客户端发送时一样，没有前缀HTTP_
Request.ServerVariables("Appl_MD_Path")
应用程序的元数据库路径
Request.ServerVariables("Content_Length")
客户端发出内容的长度
Request.ServerVariables("Https")
如果请求穿过安全通道（SSL），则返回ON如果请求来自非安全通道，则返回OFF
Request.ServerVariables("Instance_ID")
IIS实例的ID号

Request.ServerVariables("Instance_Meta_Path")
响应请求的IIS实例的元数据库路径
Request.ServerVariables("Http_Accept_Encoding")
返回内容如：gzip,deflate
Request.ServerVariables("Http_Accept_Language")
返回内容如：en-us
Request.ServerVariables("Http_Connection")
返回内容：Keep-Alive
Request.ServerVariables("Http_Cookie")
Request.ServerVariables("Http_User_Agent")
返回内容：Mozilla/4.0(compatible;MSIE6.0;WindowsNT5.1;SV1)
Request.ServerVariables("Https_Keysize")
安全套接字层连接关键字的位数，如128
Request.ServerVariables("Https_Secretkeysize")
服务器验证私人关键字的位数如1024
Request.ServerVariables("Https_Server_Issuer")
服务器证书的发行者字段
Request.ServerVariables("Https_Server_Subject")
服务器证书的主题字段
Request.ServerVariables("Auth_Password")
当使用基本验证模式时，客户在密码对话框中输入的密码
Request.ServerVariables("Auth_Type")
是用户访问受保护的脚本时，服务器用於检验用户的验证方法
Request.ServerVariables("Auth_User")
代证的用户名
Request.ServerVariables("Cert_Cookie")
唯一的客户证书ID号
Request.ServerVariables("Cert_Flag")
客户证书标志，如有客户端证书，则bit0为0如果客户端证书验证无效，bit1被设置为1
Request.ServerVariables("Cert_Issuer")
用户证书中的发行者字段
Request.ServerVariables("Cert_Keysize")
安全套接字层连接关键字的位数，如128
Request.ServerVariables("Cert_Secretkeysize")

服务器验证私人关键字的位数如1024
Request.ServerVariables("Cert_Serialnumber")
客户证书的序列号字段
Request.ServerVariables("Cert_Server_Issuer")
服务器证书的发行者字段
Request.ServerVariables("Cert_Server_Subject")
服务器证书的主题字段
Request.ServerVariables("Cert_Subject")
客户端证书的主题字段
Request.ServerVariables("Content_Type")
客户发送的form内容或HTTPPUT的数据类型

BASIC

BASIC（Beginners' All-purpose Symbolic Instruction Code，又译培基），意思就是“初学者的全方位符式指令代码”，是一种设计给初学者使用的程序设计语言。BASIC是一种直译式的编程语言，在完成编写后不须经由编译及连结等手续即可执行，但如果需要单独执行时仍然需要将其建立成执行档。
BASIC的历史
1964年，两位美国计算机科学家G. Kemeny和Thomas E. Kurtz在FORTRAN语言的基础上创造了一种新的语言——BASIC，BASIC是一种适用于初学者的人机交互式语言。
Basic 的名字－－Beginner's All－purpose Symbolic Instruction Code（初学者通用的符号指令代码），原来被作者写做 BASIC，只是后来被微软广泛的叫做 Basic 了。
BASIC语言本来是为校园的大学生们创造的高级语言，目的是使大学生容易使用计算机。尽管初期的BASIC仅有几十条语句，但由于BASIC在当时比较容易学习，它很快从校园走向社会，成为初学者学习计算机程序设计的首选语言。
随着计算机科学技术的迅速发展，特别是微型计算机的广泛使用，计算机厂商不断地在原由的BASIC基础上进行功能扩充，出现了多种BASIC版本，例如TRS-80 BASIC、Apple BASIC、GWBASIC、IBM BASIC(即BASICA)、True BASIC。此时BASIC已经由初期小型、简单的学习语言发展成为功能丰富的使用语言。它的许多功能已经能与其他优秀的计算机高级语言相媲美，而且有的功能（如绘图）甚至超过其他语言。
1975 年，比尔·盖茨创立的 Microsoft，并成功的把 Basic 语言的编译器移植到使用 Intel 处理器的 ALR 计算机中，IBM 在 1982 年选定 Microsoft 创作 PC 的操作系统时，也选定了 Microsoft 的 Basic 作为其计算机的 ROM－Basic。微软还在其发布的 DOS 操作系统中免费加入了 GW－Basic、QBasic 等当时最好的 Basic 解释程序。
Quick BASIC是微软（Microsoft）公司1987年推出的。

1991年，伴随着MS-DOS5.0的推出，微软（Microsoft）公司同时推出了Quick BASIC的简化版QBASIC，将其作为操作系统的组成部分免费提供给用户。自从Windows操作系统出现以来，图形用户界面（GUI）的BASIC语言(即Visual Basic)已经得到广泛应用。
2001年Visual Basic .NET推出
2003年推出Visual Basic .NET 2003推出
2005年11月7日在Visual Studio 2005内推出Visual Basic 2005。
中文维基百科 BASIC 使用者
BASIC是一个与电脑相关的小作品。你可以通过编辑或修订扩充其内容。
程序设计语言编辑
工业编程语言： A+ | Ada | 汇编语言 | B | Brainfuck | C | C++ | C++/CLI | Cg | COBOL | Eiffel | Erlang | FORTRAN | IronPython | Java | JRuby | Jython | LISP | Lua | Nuva | Oberon | Objective-C | Ocaml | Pascal | Perl | Powerbuilder | Python | QBASIC | R | REXX | Ruby | Self | Smalltalk | SQL | Tcl/Tk | Visual Basic | PHP | Lua | C# | F# | J# | Visual Basic .NET
脚本编程语言： ActionScript | JavaScript | JScript | Nuva | PostScript | VBScript
学术编程语言： APL/J | Haskell | Logo | ML | Prolog | Scheme | SAC
其他编程语言： ALGOL | BASIC | Clipper | Forth | Modula-2/Modula-3 | MUMPS | PL/I | Simula
取自"https://secure.wikimedia.org/wikipedia/zh/w/index.php?title=BASIC&;variant=zh-cn"
页面分类: 电脑小作品 | 程序设计语言
BASIC是一种高级语言，它的英文含义是“初学者通用符号指令代码”，是在1965年5月，由美国科学家托马斯·库尔兹研制出来的。10多年后，（现微软公司的总裁）比尔·盖茨把它移植到PC上。三十多年来，BASIC语言一直是初学计算机语言者使用最广泛的一种高级语言。它能进行数值计算、画图、演奏音乐，功能十分强大，而学起来又是非常容易。
BASIC语言的主要特点是：（1）构成简单。 BASIC语言的最基本语句只有17种，而且它们都是常见的英文单词或其变形，如READ、END等，很容易学习和掌握。
（2）是一种“人机会话”式的语言。通过键盘操作，用BASIC语言编写完的程序，可以在计算机上边编写、边修改、边运行。而且还可以在运行中向人们提示信息的指出错误，要求人去改正，即实现了人和机器的对话。
（3）BASIC语言应用广泛。许多中、小学以至于大学都开设BASIC语言。
BASIC是Beginner's All-purpose Symbolic Instruction Code 的缩写，意为初学者通用符号指令代码语言，它是在1964年由美国的两位教授Thomas 和John G.Kemeny在Fortran语言的基础上设计的语言系统，这个简单、易学的程序设计语言当时只有17条语句，12个函数和3个命令，现在一般称其为基本BASIC。
C#
C#(读做 "C sharp"，中文译音“夏普”)是微软公司发布的一种面向对象的、运行于.NET Framework之上

的高级程序设计语言,并定于在微软职业开发者论坛(PDC)上登台亮相.C#是微软公司研究员Anders Hejlsberg的最新成果.C#看起来与Java有着惊人的相似;它包括了诸如单一继承,界面,与Java几乎同样的语法,和编译成中间代码再运行的过程.但是C#与Java有着明显的不同,它借鉴了Delphi的一个特点,与COM(组件对象模型)是直接集成的,而且它是微软公司.NET windows网络框架的主角.

在本文中,我将考察创建一种新计算机语言的一般动机,并将特别指明是什么原因导致了C#的出现.然后我将介绍C#和它与Java,c,c++的相似之处.其次我将讨论一些存在于Java和C#之间的高层次的,和基础的差别.我将以衡量在用多种语言开发大型应用程序的时候所需的知识(或者对这种知识的缺乏程度)来结束本文,而这正是.NET和C#的一个主要战略.目前,C#和.NET还只能以C#语言规则,以及Windows 2000的一个"d预览版本",还有MSDN上迅速增多的文档集子的形式获得(还没有最终定型).

微软c#语言定义主要是从C和C++继承而来的,而且语言中的许多元素也反映了这一点.C#在设计者从C++继承的可选选项方面比Java要广泛一些(比如说structs),它还增加了自己新的特点(比方说源代码版本定义).但它还太不成熟,不可能挤垮Java.C#还需要进化成一种开发者能够接受和采用的语言.而微软当前为它的这种新语言大造声势也是值得注意的.目前大家的反应是:"这是对Java的反击."

C#更象Java一些,虽然微软在这个问题上保持沉默.这也是意料中的事情,我觉得,因为Java近来很成功而使用Java的公司都报告说它们在生产效率上比C++获得了提高.

Java所带来的巨大影响和大家对它的广泛接受已经由工作于这种语言和平台之上的程序员数量明显的说明了(估计世界范围内共有两百五十万程序员使用Java).由这种语言写成的应用程序的数量是令人惊讶的并已经渗透了每一个级别的计算,包括无线计算和移动电话(比如日本发明的Java电话).C#能够在用户领域获得这样的礼遇吗?我们必须等待并观望,就象已经由SSI公司的CEO和主席Kalpathi S. Suresh指出来的那样,"我发现所有这些都是渐进的.如果C#不存在,我们总能回到Java或C和C++.这些都不完全是新技术;它们在更大的意义上来说只是大公司制造的市场噱头.我们必须给他们时间安顿下来看看这些是不是真的对IT工业有什么影响."

C#从Java继承而来的特点

类:在C#中类的申明与Java很相似.这是合理的因为经验告诉我们Java模型工作得很好.Java的关键字import已经被替换成using,它起到了同样的作用.一个类开始执行的起点是静态方法Main().下面的Hello World程序展示了基本的形式:

```
using System;
class Hello
{
static void Main()
{
Console.WriteLine("Hello, world");
}
}
```

在这个例子中,System这个名字指向一个包括了基本C#实用类集合的命名空间(namespace).这个命名空间包括了Console类,它在这个例子中被用来输出一个字符串.类可以是抽象的和不可继承的:一个被申明成abstract的类不能被实例化;它只能被用做一个基类.C#关键字sealed就象Java关键字final,它申明一个类不是抽象的,但是它也不能被用做另一个类的基类.界面:就象在Java中一样,一个界面是一组方法集合的抽象定义.当一个类或结构体实现一个界面的时候,它必须实现这个界面中定义的所有方法.一个单一的类可以实现几个界面.也许以后会出现一些微妙的差别,但是这个特点看起来与Java相比没有变化.布尔运算:条件表达式的结果是布尔数据类型,布尔数据类型是这种语言中独立的一种数据类型.从布尔类型到其他类型没有直接的转换过程.布尔常量true和false是C#中的关键字.错误处理:如Java中那样,通过抛出和捕捉异常对象来管理错误处理过程.内存管理:由底层.NET框架进行自动内存垃圾回收.

C#从C和C++继承的特点

编译:程序直接编译成标准的二进制可执行形式.但C#的源程序并不是被编译成二进制可执行形式，而是一中中间语言，类似于JAVA字节码。如果前面的Hello World程序被保存成一个文本文件并被命名为Hello.cs,它将被编译成命名Hello.exe的可执行程序.

结构体:一个C#的结构体与C++的结构体是相似的,因为它能够包含数据声明和方法.但是,不象C++,C#结构体与类是不同的而且不支持继承.但是,与Java相同的是,一个结构体可以实现界面.

预编译:C#中存在预编译指令支持条件编译,警告,错误报告和编译行控制.可用的预编译指令有:

#define
#undef
#if
#elif
#else
#endif
#warning
#error
#line []

没有了#include 伪指令.你无法再用#define 语句对符号赋值,所以就不存在源代码替换的概念--这些符号只能用在#if和#elif伪指令里.在#line伪指令里的数字(和可选的名字)能够修改行号还有#warning和#error输出结果的文件名.

操作符重载:一些操作符能够被重载,而另一些则不能.特别的是,没有一个赋值运算符能够被重载.能够被被重载的单目操作符是:

+ - ! ~ ++ -- true false

能够被重载的二元运算符是:

+ - * / % & | ^ << >> == != > < >= <=

C#独有的特点

C#最引人入胜的地方是它和Java的不同,而不是相似的地方.这一节(和这个系列第二部分的大部分地方)讲述了C#实现的和Java不同的地方或者Java根本没有的特点.
中间代码:微软在用户选择何时MSIL应该编译成机器码的时候是留了很大的余地.微软公司很小心的声称MSIL不是解释性的,而是被编译成了机器码.它也明白许多--如果不是大多数的话--程序员认为Java程序要不可避免的比C编写的任何东西都要慢.而这种实现方式决定了基于MSIL的程序(指的是用C#,Visual Basic,"Managed C++"--C++的一个符合CLS的版本--等语言编写的程序)将在性能上超过"解释性的"Java代码.当然,这一点还需要得到事实证明,因为C#和其他生成MSIL的编译器还没有发布.但是Java JIT编译器的普遍存在使得Java和C#在性能上相对相同.象"C#是编译语言而Java是解释性的,"之类的声明只是商业技巧.Java的中间代码和MSIL都是中间的汇编形式的语言,它们在运行时或其它的时候被编译成机器代码.
命名空间中的申明:当你创建一个程序的时候,你在一个命名空间里创建了一个或多个类.同在这个命名空间里(在类的外面)你还有可能声明界面,枚举类型和结构体.必须使用using关键字来引用其他命名空间的内容.
基本的数据类型:C#拥有比C,C++或者Java更广泛的数据类型.这些类型是bool, byte, ubyte, short, ushort, int, uint, long, ulong, float, double,和decimal.象Java一样,所有这些类型都有一个固定的大小.又象C和C++一样,每个数据类型都有有符号和无符号两种类型.与Java相同的是,一个字符变量包含的是一个16位的Unicode字符.C#新的数据类型是decimal数据类型,对于货币数据,它能存放28位10进制数字.
两个基本类:一个名叫object的类是所有其他类的基类.而一个名叫string的类也象object一样是这个语言的一部分.作为语言的一部分存在意味着编译器有可能使用它--无论何时你在程序中写入一句带引号的字符串,编译器会创建一个string对象来保存它.
参数传递:方法可以被声明接受可变数目的参数.缺省的参数传递方法是对基本数据类型进行值传递.ref关键字可以用来强迫一个变量通过引用传递,这使得一个变量可以接受一个返回值.out关键字也能声明引用传递过程,与ref不同的地方是,它指明这个参数并不需要初始值.
与COM的集成:C#对Windows程序最大的卖点可能就是它与COM的无缝集成了,COM就是微软的Win32组件技术.实际上,最终有可能在任何.NET语言里编写COM客户和服务器端.C#编写的类可以子类化一个以存在的COM组件;生成的类也能被作为一个COM组件使用,然后又能使用,比方说,JScript语言子类化它从而得到第三个COM组件.这种现象的结果是导致了一个运行环境的产生,在这个环境里的组件是网络服务,可用用任何.NET语言子类化.
索引下标:一个索引与属性除了不使用属性名来引用类成员而是用一个方括号中的数字来匿名引用(就象用数组下标一样)以外是相似的.

```
public class ListBox: Control
{
private string[] items;
public string this[int index]
```

```
{
get
{
return items[index];
}
set
{
items[index] = value;
Repaint();
}
}
}
```

可以用一个循环器来匿名引用字符串内部数组成员,就象下面这样:

```
ListBox listBox = ...;
listBox[0] = "hello";
Console.WriteLine(listBox[0]);
```

代理和反馈:一个代理对象包括了访问一个特定对象的特定方法所需的信息.只要把它当成一个聪明的方法指针就行了.代理对象可以被移动到另一个地方,然后可以通过访问它来对已存在的方法进行类型安全的调用.一个反馈方法是代理的特例.event关键字用在将在事件发生的时候被当成代理调用的方法声明.

补充：

C#简史——摘自《程序员》杂志2005-12月刊

C# 简史

编者按：时间过得眞快，居然现在就可以写C#的简史了。但是想想也不奇怪，C#可谓起点高、发展快的新一代语言，它的这五年走过了很多前辈十几年的路。公允地说，C#是目前兼顾系统开发和应用开发的最佳实用语言，并且很有可能成为编程语言历史上的第一个“全能”型语言。看过这篇简史，我们都应该明白，不要再把C#看成年轻后生了——只要是“马拉多纳”，就早晚当“球王”。

C# 1.0，纯粹的面向对象

当时间回溯到1998年底，微软正在忙于新一代COM的设计工作。此前，COM一直是组件化开发中非常成功的一种技术；但由于它仅提供了二进制层面上的统一，因此无法将类型信息和用于支持基础平台和开发工具的信息放到组件中。这时，Java正在逐步走向成熟。于是，微软学习Java的做法，将虚拟机的概念引入到了COM领域；同时，微软提出了“元数据”的概念，用于描述组件的类型信息和工具支持信息，并决定将其放入到组件当中。这种“COM虚拟机”的名字在经历了若干争论后，最终被定为CLR（Common Language Runtime，公共语言运行时）。与此同时，微

软提出了在该运行时上运作的语言应该遵循的一些规则，以及该虚拟机的类型系统和指令集——所有这些规范形成了最终的C L I（Common Language Infrastructure，公共语言基础设施），并提交给了ECMA委员会。同时，微软开发了CLI的一个实现，这就是大名鼎鼎的.NET了。

1998年12月，微软启动了一个全新的语言项目——COOL，这是一款专门为CLR设计的纯面向对象的语言，也正是本文的主角——C#的前身。历时

半年有余，1999年7月份，微软完成了COOL语言的一个内部版本。直到2000年2月份，微软才正式将COOL语言更名为C#。据说起这个名字是因为C#开发小组的人很讨厌搜索引擎，因此把大部分搜索引擎无法识别的“#” 字符作为该语言名字的一部分；还有一种说法是在音乐当中“#”是升调记号，表达了微软希望它在C的基础上更上一层楼的美好愿望——当然这些都只是传说，无从考证。又是历经了一系列的修改，微软终于在

2000年7月发布了C#语言的第一个预览版。因此人们一般认为C#是2000年发布的，并以此来计算它的“年龄”。在此后的一年多时间里，微软一直在修补各个测试版本中的BUG。直到2002年2月，微软终于推出了迟迟未上市的Visual Studio 7.0，并将其定名为“VisualStudio .NET 2002”。随着这套开发环境的出炉，开发者们终于看到了C#语言的第一个正式版本——C# 1.0。此后，微软马不停蹄，Visual Studio也恢复了往日

的开发进度。在2003年5月，微软如期推出了Visual Studio .NET 2003，同时也发布了C#的改进版本——C# 1.1。这一时期的C#（以下称为C# 1.x）提出了纯粹的面向对象概念，并在语言特性中展现得淋漓尽致。C++并非纯面向对象的，为了和C兼容以及提供更高的执行效率，它保留了很多模块化的东西。Java尽管号称是面向对象的，但实际上，对于对象所应该具备的三种构成结构——属性、方法和事件，Java仅提供了方法，其它两种结构都要通过方法来模拟。在C# 1.x中，所有面向对象的概念都在语言中得到了非常好的体现。同时，C#还通过类类型、值类型和

接口类型的概念形成了统一的类型系统。C#使用了大家所熟知的语法实现了方法，以至于很多人认为C#和Java、C++等面向对象语言“非常相像”，这使得从使用其他面向对象语言转到使用C#的过程非常简单。此外，C#还通过无参数列表的方法声名语法，结合get/set访问器实现了优雅的属性语法。其中的get访问器相当于获取属性值的方法，可以通过一些运算返回最终的结果，而不是简单地返回一个变量的值；而set访问器相当于设置属性值的方法，在其中可以进行一系列检测，最后将属性值赋给相应的变量。同时，通过同时提供get和set访问器、只提供get访问器和只提供set访问器，还可以很方便地实现可写、只读和只写的属性。C#的这种属性语法，使得一个属性在提供该属性的类的内部看来，非常像一组方法；而对于外部调用类看来，访问一个对象的属性和访问它的公共域没有任何区别。通过委托（稍后介绍），结合关键字event，C#提供了优雅的事件概念。使用+=运算符，开发者可以非常方便地将一个事件处理器关联到一个事件上，这个过程称之为“订阅”一个事件。由于委托内部封装了一个调用链表，因此可以方便地为一个事件添加多个事件处理器，这些处理器会自动地依次调用。多年的开发语言进化证明，函数指针是非常重要也是非常危险的语言特征之一。同时，基于函数指针的回调机制也Windows 核心概念之一。然而，由于函数指针很难验证参数的类型准确性，因此C#（确切地

说是CLI）提出了“委托”的概念，这是一种类型安全的函数指针链表。这意味着，C#不仅可以提供回调机制，同时调用回调的一方还无需在其内部维护函数指针列表，所要做的仅仅是声名一个具有恰当委托类型的公共成员即可；而提供回调的一方也只需通过构造一个带有指定方法的相应委托实例，并通过“+=”运算符添加到回调列表即可。

尽管C# 1.x提供了如此多的新鲜概念，但实际上，这些概念都是由CLI提出的。因此当将一个C#源程序编译为可执行文件时，编译器做的工作相对而言并不多。需要编译器代劳的是要将一个简单的委托定义语句翻译为一个继承System.MulticastDelegate类型定义。

C# 2.0，泛型编程新概念

微软本打算继续保证开发进度，并在2004年推出Visual Studio .NET 2004，但由于其间软件工程学尤其是软件管理学的大规模进步，微软所提供的这种仅具备开发和调试功能的IDE已经无法满足团队开发的需求。因此微软决定在项目设计和管理工具方面进行了进一步研发，并将其集成到Visual Studio中，以赢回原有的市场。因此，微软将Visual Studio.NET 2004“改名”为Visual Studio 2005，并决定推迟一年发布。不过，微软还是坚持在2004年的6月份发布了Visual Studio2005的第一个Beta 版，同时向开发者展示了C#语言的2.0版本。2005年4月，微软发布了
Visual Studio 2005 Beta2，这已经是具备了几乎全部功能的VisualStudio，包括的产品有SQL Server2005、Team Foundation Server和TeamSuite。这时的C#编译器已经能够处理C# 2.0中所有的新特性。C# 2.0为开发者带来的最主要的特性就是泛型编程能力。和面向对象思想一样，泛型思想也是一种已经成熟的编程思想，但依然是没有哪一种主流开发语言能够支持完备的泛型概念。这主要是因为泛型的概念在一定程度上对面向对象概念进行冲击，同时，由于在编译期间对类型参数的完全检测很难做到，很多问题会被遗留到运行时。C# 2.0别出心裁，对泛
型类型参数提出了“约束”的新概念，并以优雅的语法体现在语言之中。有了约束，结合编译器强大的类型推断能力，可以在编译时发现几乎所有“危险”的泛型应用。C# 2.0的另一个突出的特性就是匿名方法，用来取代一些短小的并且仅出现一次的委托，使得语言结构更加紧凑。匿名方法除了可以使得事件处理器的编写更加精简以外，还将开发者带入了程序设计的一个新的领域——函数式编程，曾经有高人就用匿名方法结合泛型编程实现了函数式编程中的重要结构—— Lambda 表达式。尽管这种实现显得很繁琐而且不易理解，但毕竟是实现了。最终,函数式编程还是被引入到了C#语言中,这将在下一节中为大家讲述。

此外，C# 2.0还进一步增强了语言的表达能力。在C# 2.0中，属性语法中的get和set访问器可以拥有不同的权限，这就使得定义一个在库的内部可读写，而在库的外部只读的属性成为可能。同时，C# 2.0还提供了迭代器的概念，这使得一个类无需实现IEnumerator 和IEnumerable接口即可实现一个可以进行遍历的类型，并且无需在类型中维护迭代状态。此时的.NET已经得到了很广泛的认可，并且因为元数据为组件带来了强大的自我描述能力，许多程序库厂商被吸引到.NET平台上来。随着.NET程序库数量的增长，逐渐暴露了命名的问题。在面向对象技术广泛发展
后，人们就意识到名字的管理问题，因此几乎所有的面向对象语言都提出了“命名空间”的概念；

而在C# 1.x时代，这个问题再一次出现。如果一个库厂商XX 希望以XX.System来命名他们自己的系统基础库，那么当开发者使用using System语句时就会产生歧义。为此。C# 2.0中提供了global关键字，这为.NET库中所有的命名空间提供了一个“根”,通过指定global::System和global::XX.System就可以区别两个库了。这一时期的C#编译器变得非常复杂，泛型的引入使得编译器不得不具备超强的类型推断能力。同时，迭代器的思想并非是在CLI层面上实现的，而是由编译器自动生成了实现IEnumerator和IEnumerable接口类型。C# 3.0，魔鬼在经历了一系列的改进和完善后，微软决定于2005年11月发布Visual Studio2005，该开发环境将正式支持C#2.0。由于此推出了数个预览版和测试版，大家的期待之情似乎已经不是那么强烈了。而2005年9 月份的PDC大会则为开发者们带来了另外的惊喜——C#3.0（研发代号“Orcas”——魔鬼）的技术预览版。

说到C# 3.0，就不得不提一下微软的LINQ 项目，LINQ（语言集成查询，Language Integrated Query）提出了一种通过面向对象语法来实现对非面向对象数据源的查询技术，可查询的数据源从关系型数据库延伸到一般意义上的集合（如数组和列表）以及XML。而C# 3.0则是率先实现了LINQ的语言。在C# 3.0中，我们可以用类似于SQL语句的语法从一个数据源中轻松地得到满足一定条件的对象集合。例如要查找一个字符串

数组names中所有长度大于5的字符串，就可以写：　　var longname = from n in names wheren.Length > 5 select n;这样我们就得到一个新的字符数组longname，其中包含了我们所需要的结果。这种语句称作查询语句，与SQL语句唯一的区别是C#中的查询语句往往把select子句放到最后（这反而倒有些类似于中文的阅读顺序了）。初次看到这样一个语句，我们可能会有很大疑问：这还是C#语言吗？这的确是合乎语法规则的C#代码，而且编译器可以识别这种语法。然而实际上，C#编译器并不会对这种语法进行实际的的编译，而是将其翻译为正常的方法调用：

```
var longname = names.Where(n => n.
Length > 5).Select(n);
```

然后再进行进一步的编译。在上面的例子中已经说明，names是一个存放有字符串的数组，而数组类型并没有Where的方法。的确，Where并非names的成员方法，微软也没有对数组类型进行任何改动。这是C# 3.0中另外一个重要的新特性：扩展方法。扩展方法是定义在其他静态类中的静态方法，其第一个参数的类型就是希望扩展的类型，并且这个参数被冠以this修饰符。扩展方法是静态的，但可以像调用被扩展类型的实例方法那样进行调用，看起来好像是被扩展类型自己的方法一样。这就为语言带来了很大的灵活性，我们可以将一组近似的功能如上面的Where 和Select等（这在LINQ中被称作“标准查询表达式”）定义在一个外部类中，这样既无须修改现有类型，又可以将功能组织在一起。当然，为了做到面向对象的封装性，扩展方法只能在被扩展类型的公共成员上进行操作，如果需要从内部对类型进行改进，就必须改变现有类型的代码。在Where方法的参数列表里，我们又发现了一种奇怪的语法：n => n.Length > 5。这就是我们上文提到过的Lambda 表达式。微软的官方规范中称，Lambda 表达式是匿名方法的一种自然进化。因此Lambda 表达式其实也是一种特殊的委托，由编译器负责生成一个匿名的委托类型，它接受一个字符串类型的参数n；返回值为布尔类型，表示n的长度是否大于5；其中的参数类型和返回值类型都是由编译器推断而来的。说到类型推断，还要解释的

一点就是上面的语句中出现的新关键字var。从出现的位置来看，var应该是一个类型。然而这又不是一个C#内建类型，也不是CLI提出的新类型；它只是一个“占位符”，它的确表示一个类型，但具体是什么类型需要编译器在编译期间进行推断。Lamda表达式的眞正意义不仅仅在于简化了委托的编写方式，更重要的是它把代码表达式体现为了数据。换句话说，Lambda表达式不仅可以被编译为一段可以执行的代码（类似于匿名方法），也可以将其翻译为一个数据结构——表达式树。而如何处理Lambda 表达式，是由编译器根据Lambda表达式的使用方式来自动确定的。当把一个Lambda表达式赋给一个具有委托类型的域、属性或变量时，编译器像编译匿名方法
一样将表达式体翻译成一段可执行代码；而当把一个L a m b d a 表达式赋给一个具有Expression<T>类型的域、属性或变量时，编译器就会将Lambda表达式解析为一个表达式树。对于翻译为代码的Lambda，可以向调用委托那样进行调用，而对于翻译为表达式树的Lambda表达式，就不可以了，会得到一个编译错误。但表达式树存在于一个由编译器生成的数据结构中，因此可以在运行时对其进行分析甚至修改。除了上面提到的一些重大改进之外，C# 3.0也对细微的语法进行了一些改进，使C#语言变得更加优雅和全面。值得说明的是，C# 3.0经过编译后生成的IL代码，完全是基于.NET 2.0的，C#语言已经远远跑在了他所栖生的平台前面。这一时期的C#语言离CLI已经越来越远了，编译器的工作也愈加繁重起来。首先很多语言结构（如查询表达式和Lambda 表达式）都不是CLI中提供的特性，因此需要编译器进行大量的转译工作；其次是这些语言结构带来的大量类型推断任务，也都是靠编译器来完成的。C#走到了3.0以后，已经完全不再是当年那个“简单”的语言了。它的开发者称其为“魔鬼”，而琳琅满目的新特性也的确让开发者们眼花缭乱，甚至感到恐惧。语言集成查询的引入，使得前一段时期内为开发者们广泛讨论的ORM概念得到了更加深入地体现，尤其是它所支持的数据源之广泛，让ORM理念变得已经不再必要了；而一些“.NET中的ORM实现”，似乎也成了完全不必要的扩展项目了。Lambda 表达式的引入，使得C#将可以轻松地完成特定领域（Domain-Specific）的开发。一个成功的开发人员在面对新鲜事物和新的困难时，兴奋是远大于恐惧的。让魔鬼来得更猛烈些吧！

COBOL

英文缩写: COBOL (Common business Oriented Language)
中文译名: COBOL语言
解 释: 一种适合于商业及数据处理的类似英语的程序设计语言。这种语言可使商业数据处理过程精确表达。
COBOL（面向商业的通用语言，又称为企业管理语言、数据处理语言等，Common Business Oriented Langauge）是最早的高级编程语言之一，是世界上第一个商用语言。
1 COBOL的历史
1959年5月，五角大楼委托格雷斯·霍波（G.Hopper）博士领导一个委员会并由Rear Admiral Grace Hopper公司主持开发，并于1961年由美国数据系统语言协会公布。正式发布于1960年4月，称为Cobol－60，现在最新的版本是Cobol－2002。
1963年，美国国家标准研究所(ANSI)进行了标准化，但是ANSI标准很少被遵循；因此，COBOL程序只

是部分可移植的。
2 COBOL的重要性
经过40多年的不断修改、丰富完善和标准化，COBOL已发展为多种版本的庞大语言，在财会工作、统计报表、计划编制、情报检索、人事管理等数据管理及商业数据处理领域，都有着广泛的应用。
COBOL的重要性可以用这句话来描述：世界上70%的数据都是用COBOL语言处理的，并且90%的ATM事务处理用的都是COBOL语言。每天在线处理的COBOL事务有300亿次。500强中有492家（包括全部的100强）使用了COBOL语言，目前在COBOL方面的投资已经超过3万亿美元，，据称用COBOL书写的程序超过了1000亿行，并且以每年大约50亿行代码的速度在增长。
由于COBOL在商业领域的雄厚基础，而且COBOL主要是应用于银行、金融和会计行业等非常重要的商业数据处理领域。所以，即使对于具有相当经验的IT公司来说，重新编写COBOL语言的可靠的应用软件也是不实际或是从商业角度上并不可行的，而且还要花上很长的时间，只要大型机存在，COBOL就不会消失，即使是对电脑界产生巨大影响的“千年虫”（Y2K）也没有改变COBOL的命运。
3 COBOL的特点
COBOL是一种面向数据处理的、面向文件的、面向过程(POL)的高级编程语言，是一种功能很强而又极为冗长的语言。
COBOL适合于具有循环处理周期的环境(例如打印工资支票)以及数据操纵量相当大的环境。COBOL主要应用于商业数据处理领域，对各种类型的数据进行收集、存储、传送、分类、排序、计算及打印报表、输出图象是它的强项。
COBOL语法与英文很接近，即使不懂电脑的人也能看懂程序。
强大的文件处理功能，大量的数据通常以文件的形式存储在磁盘上。
仅提供了加、减、乘、除及乘方这五种简单的算术运算，因而不适于进行科学计算。
未来的COBOL将支持XML等Web时代的新技术。
4 COBOL的程序结构
COBOL程序由4部(DIVISION)组成：IDENTIFICATION DIVISION.(标识部)、ENVIRONMENT DIVISION.(环境部)、DATA DIVISION.(数据部)、PROCEDURE DIVISION.(过程部)，而每个部又由若干节 (SECTION)组成。

Delphi

Delphi这个名字源于古希腊的城市名。它集中了第三代语言的优点。以Object Pascal为基础，扩充了面向对象的能力，并且完美地结合了可视化的开发手段。Delphi自1995年3 月一推出就受到了人们的关注，并在当年一举夺得了多项大奖。

Delphi的出现打破了V承可视化编程领域一统天下的局面。并且Delphi使用了本地编译器直接生成技术，使程序的执行性能远远高于其它产品生成的程序。它还是眞正的面向对象的编程语

言。PASCAL语言的严谨加上可视化的优势和强大的数据库功能使得它有充分的资本和微软的VB叫板。许多人当时都认为Pascal 是最有前途的程序设计语言，并预测Delphi将会成为可视化编程的主流环境。

Delphi在你编好程序后自动转换成.EXE文件它运行时速度比VB快，而且编译后不需要其他的支持库就能运行。它的数据库功能也挺强的，是开发中型数据库软件理想的编程工具。 Delphi适用于应用软件、数据库系统、系统软件等类型的开发。而且它拥有和VB差不多一样的功能，而且一样能应用API函数，这在控制Windows很有用。

Delphi是全新的可视化编程环境，为我们提供了一种方便、快捷的Windows应用程序开发工具。它使用了Microsoft Windows图形用户界面的许多先进特性和设计思想，采用了弹性可重复利用的完整的面向对象程序语言(Object-Oriented Language)、当今世界上最快的编辑器、最为领先的数据库技术。对于广大的程序开发人员来讲，使用Delphi开发应用软件，无疑会大大地提高编程效率，而且随着应用的深入，您将会发现编程不再是枯燥无味的工作——Delphi的每一个设计细节，都将带给您一份欣喜。

Delphi的基本形式

Delphi实际上是Pascal语言的一种版本，但它与传统的Pascal语言有天壤之别。一个Delphi程序首先是应用程序框架，而这一框架正是应用程序的“骨架”。在骨架上即使没有附着任何东西，仍可以严格地按照设计运行。您的工作只是在“骨架”中加入您的程序。缺省的应用程序是一个空白的窗体(Form)，您可以运行它，结果得到一个空白的窗口。这个窗口具有Windows窗口的全部性质：可以被放大缩小、移动、最大最小化等，但您却没有编写一行程序。因此，可以说应用程序框架通过提供所有应用程序共有的东西，为用户应用程序的开发打下了良好的基础。

Delphi已经为您做好了一切基础工作——程序框架就是一个已经完成的可运行应用程序，只是不处理任何事情。您所需要做的，只是在程序中加入完成您所需功能的代码而已。在空白窗口的背后，应用程序的框架正在等待用户的输入。由于您并未告诉它接收到用户输入后作何反应，窗口除了响应Windows的基本操作(移动、缩放等)外，它只是接受用户的输入，然后再忽略。Delphi把Windows编程的回调、句柄处理等繁复过程都放在一个不可见的Romulam覆盖物下面，这样您可以不为它们所困扰，轻松从容地对可视部件进行编程。

面向对象编程的概念

面向对象的程序设计(Object-Oriented Programming，简记为OOP)是Delphi诞生的基础。OOP立意于创建软件重用代码，具备更好地模拟现实世界环境的能力，这使它被公认为是自上而下编程的优胜者。它通过给程序中加入扩展语句，把函数“封装”进Windows编程所必需的“对象”中。面向对象的编程语言使得复杂的工作条理清晰、编写容易。

说它是一场革命，不是对对象本身而言，而是对它们处理工作的能力而言。对象并不与传统程序设计和编程方法兼容，只是部分面向对象反而会使情形更糟。除非整个开发环境都是面向对象的，否则对象产生的好处还没有带来的麻烦多。

而Delphi是完全面向对象的，这就使得Delphi成为一种触手可及的促进软件重用的开发工具，从而具有强大的吸引力。

一些早期的具有OOP性能的程序语言如C++,Pascal,Smalltalk等，虽然具有面向对象的特征，但不能轻松地画出可视化对象，与用户交互能力较差，程序员仍然要编写大量的代码。Delphi的推出，填补了这项空白。您不必自己建立对象，只要在提供的程序框架中加入完成功能的代码，其余的都交给Delphi去做。欲生成漂亮的界面和结构良好的程序丝毫不必绞尽脑汁，Delphi将帮助您轻松地完成。它允许在一个具有真正OOP扩展的可视化编程环境中，使用它的Object Pascal语言。这种革命性的组合，使得可视化编程与面向对象的开发框架紧密地结合起来。

FORTRAN

目录·FORTRAN简介
·FORTRAN开发历史
·Fortran的特性
·Fortran语言的Hello World程序
·Fortran编译器
·Fortran程序包

FORTRAN简介

FORTRAN是英文“FORmula TRANslator”的缩写，译为“公式翻译器”，它是世界上最早出现的计算机高级程序设计语言，广泛应用于科学和工程计算领域。FORTRAN语言以其特有的功能在数值、科学和工程计算领域发挥着重要作用。

FORTRAN开发历史

早在1951年，美国IBM公司约翰·贝克斯(John Backus)针对汇编语言的缺点着手研究开发FORTRAN语言，并于1954年在纽约正式对外发布。称约翰·贝克斯提出的FORTRAN语言为FORTRANⅠ，FORTRANⅠ虽然功能简单，但它的开创性工作，在社会上引起了极大的反响。到1957年第一个FORTRAN编译器在IBM704计算机上实现，并首次成功运行了FORTRAN程序。

在1958年，对FORTRANⅠ进行了扩充和完善，引进了子函数等概念，推出了商业化的FORTRANⅡ版本。之后，FORTRAN语言发展迅速，多种版本相继在其它计算机上实现。

在1962年，推出了FORTRAN Ⅳ。FORTRAN Ⅳ没有充分考虑兼容性，导致FORTRANⅡ程序不能在FORTRAN Ⅳ系统中运行，使其应用受到了很大限制，这时语言不兼容性问题和影响被突出表现出来。此前也出现过FORTRAN Ⅲ，但由于存在严重缺陷，没有在计算机上实现。

随着FORTRAN语言版本的不断更新和变化，语言不兼容性问题日益突出，语言标准化工作被提上了日程。1962年5月，美国标准化协会(简称ANSI)成立相关机构着手进行FORTRAN语言标准化的研

究工作，并于1966年正式公布了两个标准文本：美国国家标准FORTRAN(ANSI X3.9-1966)和美国国家标准基本FORTRAN(ANSI X3.10-1966)，前者相当于FORTRAN Ⅳ，后者相当于FORTRANⅡ。基本FORTRAN是美国国家标准FORTRAN的一个子集，从而实现了语言的向下兼容，初步解决了语言的兼容性问题。通常称美国国家标准FORTRAN为FORTRAN 66。FORTRAN 66的推出在国际上产生了广泛影响，1972年国际标准化组织(简称ISO)在FORTRAN 66基础上制定了FORTRAN语言三级国际标准：基本级、中间级和完全级。

20世纪60代末，结构化程序设计方法提出后，具有结构化特征的程序设计语言开始出现，如：ALGOL、PASCAL、MODULA、C等。如何将结构化特征引入FORTRAN 66引起计算机厂商和研究机构的高度重视，许多计算机厂商开始对FORTRAN 66进行不同程度的扩充，引入了结构化特征。针对这种情况，ANSI于1976年对FORTRAN 66(ANSI X3.9-1966)进行了修订，吸收了计算机厂商所扩充的一些行之有效的功能，同时增加了许多新内容。ANSI于1978年4月正式公布了新的美国国家标准(程序设计语言FORTRAN ANSI X3.9-1978)，同时宣布撤消ANSI FORTRAN 3.9-1966，通常称新标准为FORTRAN 77(该版本原计划1977年公布)。FORTRAN 77向下兼容FORTRAN 66。
在1980年，FORTRAN 77被ISO正式确定为国际标准ISO 1539-1980，该标准分全集和子集。FORTRAN 77推出后，由于具有结构化特征，在社会上得到了广泛应用，同时由于扩充了字符处理功能，在非数值处理领域也能大显身手。

20世纪80年代末，FORTRAN 77结构化和现代化的研究开始兴起，到1991年5月，ANSI公布了新的美国国家标准FORTRAN(ANSI 3.198-1991)。之后，ISO采纳该标准，并确定为国际标准ISO/IEC 1539-1：1991，新国际标准还采纳了我国计算机和信息处理标准化技术委员会程序设计分会提出的多字节字符集数据类型及相应的内部函数，为非英语国家使用计算机提供了极大的方便。通常称新标准为FORTRAN 90，FORTRAN 90向下兼容FORTRAN 77。之后不久又出现了FORTRAN 95。

FORTRAN 90的推出，使传统FORTRAN语言具有了现代气息。Fortran 2003 的规则已经由ISO组织制定发布。

Windows平台下，微软公司将FORTRAN 90无缝集成在Developer Studio集成开发环境之中，推出了Microsoft FORTRAN PowerStation 4.0，使FORTRAN 90真正实现了可视化编程，彻底告别了传统DOS环境(字符界面)，转到了现代Windows环境(视窗界面)，共享微软公司Windows平台的丰富资源。

在1997年3月，微软公司和数据设备公司(Digital Equipment Corp，简称DEC)强强联合，合作研究、开发和推出了功能更强的FORTRAN语言新版本：

Digital Visual FORTRAN 5.0，它是Microsoft FORTRAN PowerStation 4.0的升级换代产品。DEC公司在高性能科学和工程计算方面拥有世界领先技术，其高质量的FORTRAN编译器遍及全球。1998年1月，DEC与Compag公司合并，DEC成为Compag公司的全资子公司，于是Digital Visual FORTRAN更名为Compag Visual FORTRAN，其最新版本为Compag Visual FORTRAN 6.6。Compaq和HP合并之后，Compaq的Fortran小组和Intel的Fortran开发小组合并，开发出来Intel

Fotran编译器9，有linux和window2个版本，其windows版本为Intel Visual Fortran， 可以和微软的Visual Studio.net集成。Windows平台下还有PGI，Absoft，Intel等多个商业公司的Fortran编译器，还有大量小公司的免费Fortran编译器。

openMPI使Fortran等语言可以容易且免费的实现并行计算。

Linux平台下，其gcc编译器默认支持fortran，另外有Intel，Sun Studio，openMPI，Photran等共享编译器和PGI，Absoft，lachy， IBM，SGI，HP等多个版本的商业编译器。

支持Fortran 2003标准的编译器行将推出，新版本的Sun Studio 编译器已经支持部分 Fortran 2003 语法。

Fortran的特性

Fortran语言的最大特性是接近数学公式的自然描述，在计算机里具有很高的执行效率。

易学，语法严谨。

可以直接对矩阵和复数进行运算，这一点类似matlab。

自诞生以来广泛地应用于数值计算领域，积累了大量高效而可靠的源程序。

很多专用的大型数值运算计算机针对Fortran做了优化。

广泛地应用于并行计算和高性能计算领域。

Fortran90，Fortran95，Fortran2003的相继推出使Fortran语言具备了现代高级编程语言的一些特性。

Fortran语言的Hello World程序

下面是一个在标准输出设备上输出Hello World的简单程序，这种程序通常作为开始学习编程语言时的第一个程序：

```
WRITE(*，10)
10 FORMAT('Hello， world！')
STOP
END
```

Fortran编译器

Windows操作系统下：

Fortran Power Station 4.0 (FPS 4.0)，微软公司开发的Fortran编译器。1997年3月转让给DEC公司。

Digital Visual Fortran (DVF)，Fortran Power Station的DEC公司版本。

Compaq Visual Fortran (CVF)，1998年1月，DEC公司被康柏公司收购，Digital Visual Fortran更名为Compaq Visual Fortran。一个著名的版本是Compaq Visual Fortran 6.5。目前康柏公司已并入惠普公司。Compaq Visual Fortran的最新版是6.6。

Intel Fortran，英特尔公司的开发的Fortran编译器。

Lahey Fortran

Absoft Fortran
OpenWatcom
Linux操作系统下：
PGI Fortran
G77，GNU的Fortran77编译器，集成在GCC中。
GFORTRAN，GNU的最新的Fortran编译器，集成在GCC 4.0中，目的是支持Fortran95和一部分Fortran2003的功能，以替代G77。
Intel Fortran
Absoft Fortran
g95，跟GFORTRAN同为开放源代码的Fortran95编译器。

Fortran程序包

几个著名的Fortran程序包：
IMSL--国际数学和统计链接库
BLAS--Basic Linear Algebra Subroutines
LAPACK--Linear Algebra PACKage

FORTRAN90是ISO(国际标准化组织)于1991年推出的最新标准,我国国家标准是GB/T 3057-1996.除了保持FORTRAN77的全部优点之外,又 增加了许多具有现代特性的功能,使他成为具有良好的结构特性,鲜明的时代特性的程序设计语言.程序设计是计算机基础教育的基础与重点，高级语言程序设计课是继微机应用基础之后的一门必修的基础课，目的是向学生介绍程序设计的基础知识，使学生掌握高级语言程序设计的基本方法，具有应用计算机的初步能力，并培养学生掌握用计算机处理问题的思维方法。通过该课程的学习，要求学生了解FORTRAN语言的特点，基本成份及使用方法，具有阅读程序和初步编程的能力。进行算法的初步训练，掌握最基本算法的设计和实现方法。掌握结构化程序设计方法，能设计出良好风格的程序。具有调试程序的基本能力。

Javascript

Javascript是一种由Netscape的LiveScript发展而来的脚本语言，主要目的是为了解决服务器终端语言，比如Perl，遗留的速度问题。当时服务端需要对数据进行验证，由于网络速度相当缓慢，只有28.8kbps，验证步骤浪费的时间太多。于是Netscape的浏览器Navigator加入了Javascript，提供了数据验证的基本功能。

历史

在1992年，Nombas开始开发一种嵌入式脚本语言，叫做C-minus-minus(Cmm)。[待续...

能够具有交互性，能够包含更多活跃的元素，就有必要在网页中嵌入其它的技术。

如：Javascript、VBScript、Document Object Model（文件目标模块）、Layers和 Cascading Style

Sheets（CSS），这里主要讲Javascript。那么Javascript是什么东东？Javascript就是适应动态网页制作的需要而诞生的一种新的编程语言，如今越来越广泛地使用于Internet网页制作上。 Javascript是由Netscape公司开发的一种脚本语言（scripting language），或者称为描述语言。在HTML基础上，使用Javascript可以开发交互式Web网页。Javascript的出现使得网页和用户之间实现了一种实时性的、动态的、交互性的关系，使网页包含更多活跃的元素和更加精彩的内容。运行用Javascript编写的程序需要能支持Javascript语言的浏览器。Netscape公司 Navigator 3．0以上版本的浏览器都能支持Javascript程序，微软公司 Internet Explorer 3．0以上版本的浏览器基本上支持Javascript。微软公司还有自己开发的Javascript，称为JScript。 Javascript和Jscript基本上是相同的，只是在一些细节上有出入。 Javascript短小精悍， 又是在客户机上执行的，大大提高了网页的浏览速度和交互能力。 同时它又是专门为制作Web网页而量身定做的一种简单的编程语言。

虽然，在Dreamweaver的Behaviors可以为我们方便地使用Javascript程序而不用编写代码，但我们自己了解了Javascript的编程方法后，将能更加方便灵活地应用，也使Javascript的代码更简练。本专题通过对一系列典型程序的剖析，使你快速地掌握Javascript的编程技巧，设计出质量上乘的动态网页打下坚实的基础。在此之前，我们先了解一些Javascript 的基本概念。

JavaScript 有什么特点

JavaScript 使网页增加互动性。JavaScript 使有规律地重复的HTML文段简化，减少下载时间。JavaScript 能及时响应用户的操作，对提交表单做即时的检查，无需浪费时间交由 CGI 验证。JavaScript 的特点是无穷无尽的，只要你有创意。

Java 与 JavaScript 有什么不同

很多人看到 Java 和 JavaScript 都有“Java”四个字，就以为它们是同一样东西，连我自己当初也是这样。其实它们是完完全全不同的两种东西。Java，全称应该是 Java Applet，是嵌在网页中，而又有自己独立的运行窗口的小程序。Java Applet 是预先编译好的，一个 Applet 文件（.class）用 Notepad 打开阅读，根本不能理解。Java Applet 的功能很强大，可以访问 http、ftp等协议，甚至可以在电脑上种病毒（已有先例了）。相比之下，JavaScript 的能力就比较小了。JavaScript 是一种“脚本”（“Script”），它直接把代码写到 HTML 文档中，浏览器读取它们的时候才进行编译、执行，所以能查看 HTML 源文件就能查看JavaScript 源代码。JavaScript 没有独立的运行窗口，浏览器当前窗口就是它的运行窗口。它们的相同点，我想只有同是以 Java 作编程语言一点了。

开发 JavaScript 该用什么软件

一个 JavaScript 程序其实是一个文档，一个文本文件。它是嵌入到 HTML 文档中的。所以，任何可以编写 HTML 文档的软件都可以用来开发 JavaScript。在此我推荐大家用 FrontPage 2000 附带的 Microsoft 脚本编辑器（在 FrontPage 菜单 | 工具 | 宏 | Microsoft 脚本编辑器）。它是个像 Visual Basic / C++ 一样的程序开发器，能对正在输入的语句作出简要提示。配合 FrontPage 2000，使工作量大大减少。

一、Javascript在网页的用法

Javascript加入网页有两种方法：
1、直接加入HTML文档
这是最常用的方法，大部分含有Javascript的网页都采用这种方法，如：

```
<script language="Javascript">
<!--
document.writeln("这是Javascript！采用直接插入的方法！");
//-Javascript结束-->
</script>
```

在这个例子中，我们可看到一个新的标签：<script>……</script>，而<script language="Javascript”>用来告诉浏览器这是用Javascript编写的程序，需要调动相应的解释程序进行解释。
HTML的注释标签<!--和-->：用来去掉浏览器所不能识别的Javascript源代码的，这对不支持 Javascript 语言的浏览器来说是很有用的。
//-Javascript结束：双斜杠表示 Javascript的注释部分，即从//开始到行尾的字符都被忽略。 至于程序中所用到的document · write（）函数则表示将括号中的文字输出到窗口中去， 这在后面将会详细介绍。
另外一点需要注意的是，<script>……</script>的位置并不是固定的，可以包含在<head>.....</head>或<body>.....</body>中的任何地方。
2、引用方式 如果已经存在一个Javascript源文件（以js为扩展名），则可以采用这种引用的方式，以提高程序代码的利用率。其基本格式如下：

```
<script src＝url language="Javascript"></script>
```

其中的Url就是程序文件的地址。同样的，这样的语句可以放在HTML文档头部或主体的任何部分。如果要实现“直接插入方式”中所举例子的效果，可以首先创建一个Javascript源代码文件“Script.js”，其内容如下：

```
document.writeln("这是Javascript！采用直接插入的方法！");
```

在网页中可以这样调用程序：<script src＝"Script.js" language="Javascript"></script> 。
二、Javascript基本概念
在这里只作简单介绍，在以后的例子中结程序再作具体解释其作用。
1、运算符
运算符就是完成操和的一系列符号，它有七类：
赋值运算符、算术运算符、比较运算符、逻辑运算符、条件运算、位操作运算符和字符串运算符。
2、表达式
运算符和操作数的组合称为表达式，通常分为四类：赋值表达式、算术表达式、布尔表达式和字符串表达式。
3、语句
Javascript程序是由若干语句组成的，语句是编写程序的指令。Javascript提供了完整的基本编程语句，

它们是：
赋值语句、switch选择语句、while循环语句、for循环语句、do while循环语句、break循环中止语句和continue循环中断语句。
4、函数
函数是命名的语句段，这个语句段可以被当作一个整体来引用不着和执行。使用函数要注意以下几点：
1）函数由关键字function定义；
2）函数必须先定义后使用，否则将出错；
3）函数名是调用函数时引用的名称，它对大小写是敏感的，调用函数时不可写错函数名；
4）参数表示传递给函数使用或操作的值，它可以是常量，也可以是变量；
5）return语句用于返回表达式的值，也可以没有。
5、对象
Javascript的一个重要功能就是基于对象的功能，通过基于对象的程序设计，可以用更直观、模块化和可重复使用的方式进行程序开发。
一组包含数据的属性和对属性中包含数据进行操作的方法，称为对象。比如要设定网页的背景颜色，所针对的对象就是document，所用的属性名是bgcolor，如document.bgcolor="blue"，就是表示使背景的颜色为蓝色。
6、事件
用户与网页交互时产生的操作，称为事件。绝大部分事都由用户的动作所引发，如：用户按鼠标的按钮，就产生onclick事件，若鼠标的指针的链接上移动，就产生onmouseover事件等等。
在Javascript中，事件往往与事件处理程序配套使用。
学习Javascript比较快速有效的方法是先熟悉一些基本概念，然后找几个别人设计好的程序认真仔细地分析一遍，再稍作改动，再看看能否达到预期目的，不断地举一反三，既可以加深对一些参数、设计方法的理解，又可以快速地提高自己的水平。另外，再提醒一下：Javascript对大小写是敏感的，特别是一些对象、方法、属性的大小写一定要一致，要养成一种良好的习惯，否则在调试程序时可要累死你了。
7、变量
如 var myVariable = "some value";

很明显先前的这个仁兄只是对JAVASCRIPT很厉害,他所说的JAVA其实是错误的
Java是由Sun Microsystems公司于1995年5月推出的Java程序设计语言和Java平台的总称。用Java实现的HotJava浏览器（支持Java applet）显示了Java的魅力：跨平台、动感的Web、Internet计算。从此，Java被广泛接受并推动了Web的迅速发展，常用的浏览器现在均支持Java applet。另一方面，Java技术也不断更新。

Java平台由Java虚拟机（Java Virtual Machine）和Java 应用编程接口（Application Programming Interface、简称API）构成。Java 应用编程接口为Java应用提供了一个独立于操作系统的标准接口，可分为基本部分和扩展部分。在硬件或操作系统平台上安装一个Java平台之后，Java应用程序就可运行。现在Java平台已经嵌入了几乎所有的操作系统。这样Java程序可以只编译一次，就可以在各种系统中运行。

Java分为三个体系JavaSE，JavaEE,JavaME。

JSP

JSP(Java Server Pages)是由Sun Microsystems公司倡导、许多公司参与一起建立的一种动态网页技术标准，本文简单介绍JSP及其优点。

JSP(Java Server Pages)是由Sun Microsystems公司倡导、许多公司参与一起建立的一种动态网页技术标准。 JSP技术是用JAVA语言作为脚本语言的，JSP网页为整
个服务器端的JAVA库单元提供了一个接口来服务于HTTP的应用程序。
在传统的网页HTML文件(*.htm,*.html)中加入Java程序片段(Scriptlet)和JSP标记(tag)，就构成了JSP网页(*.jsp)。Web服务器在遇到访问JSP网页的请求时，首先
执行其中的程序片段，然后将执行结果以HTML格式返回给客户。程序片段可以操作数据库、重新定向网页以及发送 email 等等，这就是建立动态网站所需要的功能
。所有程序操作都在服务器端执行，网络上传送给客户端的仅是得到的结果，对客户浏览器的要求最低，可以实现无Plugin，无ActiveX，无Java Applet，甚至无
Frame。

JSP的优点：

.对于用户界面的更新，其实就是由 Web Server进行的，所以给人的感觉更新很快。

.所有的应用都是基于服务器的，所以它们可以时刻保持最新版本。

.客户端的接口不是很繁琐，对于各种应用易于部署、维护和修改。

Nuva

Nuva语言是一种面向对象的动态脚本语言。Nuva对应汉语的女娲一词。女娲是中国上古时代的神话传说人物。

1 Nuva语言的设计目的

Nuva语言的设计目的是用于基于模板的代码生成。除了用于代码生成领域外，Nuva语言也能用于开发应用程序，如文本和数据处理、GUI 应用程序等。

2 Nuva语言特点

2.1 语法简单灵活

Nuva语言采用类似伪码的语法风格，结构之间可以任意嵌套，关键字和运算符兼容大部分现有的编程语言，非常容易学习。

```
<.
if (a = b && c == d or e <> f)
?? foo()

function foo()
Result = 'foo'
end function
end if
.>
```

2.2 动态的，无类型约束

Nuva语言采用动态类型，使用时不需声明类型，赋值计算时自动进行类型转换，如下：

```
<.
var a = '1'
a ++
?? 'a' ~ a
// 结果为: a2
.>
```

2.3 支持面向对象

Nuva语言支持面向对象的编程方法，支持继承性和多态性。

2.4 自动垃圾回收

Nuva语言支持自动垃圾回收，程序员不需显式释放其所创建的对象。

2.5 模板专用的语言元素

Nuva语言为模板增加了专用的语言元素，方便模板的编写。

<. | .> | [. | .] 模板标记可以混合配对使用，对于格式要求很严格的场合非常有用。

```
[.='Hello, Nuva!'.]
<.='Hello, Nuva!'.>
[.='Hello, Nuva!'.>
<.='Hello, Nuva!'.]
```

凡[.之前的所有空白字符原样输出，.]之后的所有空白字符（包括换行）也原样输出；

如果行首到<.之间均为空白字符，则该部分空白字符不输出，否则原样输出；

如果.>到行尾之间均为空白字符，则该部分空白字符和换行不输出，否则也原样输出。

Nuva语言特有的file和assign结构能够非常方便的对输出进行组合、分解，从而方便了模板的编写。

3 Nuva虚拟机特点
3.1 内置了正则表达式引擎
Nuva虚拟机内置了正则表达式引擎，能够方便的进行文本处理。
```
<.
var text = System.File.Load('Regex_Test.nuva')
foreach(str = text.RegexMatchs('\w+', ))
?? str
end foreach
.>
```
输出如下的结果：
```
var
text
System
File
Load
Regex_Test
nuva
foreach
str
text
RegexMatches
w
str
end
foreach
```
3.2 内置了 O/R Mapping 引擎
Nuva虚拟机内置了 O/R Mapping 引擎，可以通过面向对象的方式直接存取数据库架构和数据。
3.3 内置了基于 HTML/XML 的界面引擎
Nuva虚拟机内置了基于 HTML/XML 的界面引擎，能够方便的编写 GUI 应用程序，典型的例子就是 Macrobject CodeAuto 代码生成器。
4 Nuva语言代码示例
4.1 Hello, Nuva!
```
<.. "Hello, Nuva!" Demo ..>
<.
```

```
//=====================================
// Hello, Nuva! (1)
//=====================================
?? 'Hello, Nuva!'
/*=====================================
Hello, Nuva! (2)
=====================================*/
function HelloNuva()
?? "Hello, Nuva!";
end function
HelloNuva();
/*=====================================
Hello, Nuva! (3)
=====================================*/
class Nuva()
function Hello()
?? 'Hello, Nuva!';
end function
end class
var n = Nuva();
n.Hello();
.>
```

4.2 foreach | O/R Mapping

```
<.
function Foreach_Demo()
// Load Schema from a Xml file
var schema = System.Data.LoadSchema(
System.Path.ProjectPath ~ '..\Northwind\Northwind.xobject'
);
?? '--------------------'
?? 'Tables Order by Name'
?? '--------------------'
foreach(table = schema.Tables.OrderbyName)
?? table.Name
```

```
end foreach
?? '--------------------------------'
?? 'Tables Filter by Name.Length < 10'
?? '--------------------------------'
foreach(table = schema.Tables | table.Name.Length < 10)
?? table.Name
end foreach
end function
.>
```

4.3 file | 生成文件

```
<.
function File_Demo()
?? \r\n ~ '--Read file and Output here--'
file('codeexamples.nuvaproj') end file
// Read file and write to 'Target', overwrite it if exist
file('codeexamples.nuvaproj', true)
Target = 'temp.target'
end file

?? \r\n ~ '--Read file dynamically and Output here--'
file()
FileName = System.Path.ProjectPath ~ 'output\temp.target'
end file

// Delete file
System.File.Delete(System.Path.ProjectPath ~ 'output\temp.target')
end function
.>
```

4.4 assign | 捕获输出

```
<.
function Assign_Demo()
// 'Result' assigned from the output in the assign statement
assign(Result).]
Generate Text ... @[.=System.Now.] ...
```

```
<.end assign
end function
.>
```

4.5 函数 | 递归调用

```
<.
/*-----------------------------------------------------
Factorial
-----------------------------------------------------*/
function Factorial ( N )
if (N <= 1)
Result = 1;
else
Result = N * Factorial(N - 1); // Recursion Call
end if;
end function;
.>
```

4.6 类 | 多态性

```
<.
function Class_Demo()
class ClassA()
function Do()
this.DynDo()
end function

function DynDo()
?? 'ClassA'
end function
end class

class ClassB = ClassA()
function DynDo()
?? 'ClassB'
end function
end class
```

```
var c1 = ClassA()
var c2 = ClassB()
c1.Do()
c2.Do()
end function
```

PASCAL语言

目录·一、PASCAL语言的来历
·二、PASCAL语言的发展
·三、PASCAL语言的影响
·四、PASCAL在学习和竞赛中的应用
·五、Pascal的基本运用

pascal

一、PASCAL语言的来历

Pascal是一种计算机通用的高级程序设计语言。它由瑞士Niklaus Wirth教授于六十年代末设计并创立。Pascal也可以是指人名，它的取名原本就是为了纪念十七世纪法国著名哲学家和数学家Blaise Pascal，而不是编程工具。以法国数学家命名的pascal语言现已成为使用最广泛的基于DOS的语言之一，其主要特点有：严格的结构化形式；丰富完备的数据类型；运行效率高；查错能力强。
Pascal语言还是一种自编译的语言，这就使它的可靠性大大提高了。
Pascal具有简洁的语法，结构化的程序结构。它是结构化编程语言，于70年代在ALGOL基础上研制出来的。它具有丰富的数据类型并提供了数据类型定义设施，其控制结构体现了结构程序设计原则。
它最初是为系统地教授程序设计而设计的，特点是简明化和结构化，适合教学科学计算与系统软件的研制。如今，在许多学校的计算机语言课上，学的都是Pascal语言。
Pascal是最早出现的结构化编程语言，具有丰富的数据类型和简洁灵活的操作语句，适于描述数值和非数值的问题。
正因为上述特点，Pascal语言可以被方便地用于描述各种算法与数据结构。尤其是对于程序设计的初学者，Pascal语言有益于培养良好的程序设计风格和习惯。IOI(国际奥林匹克信息学竞赛)把Pascal语言作为三种程序设计语言之一， NOI(全国奥林匹克信息学竞赛)把Pascal语言定为唯一提倡的程序设计语言，在大学中Pascal语言也常常被用作学习数据结构与算法的教学语言。

二、PASCAL语言的发展

在Pascal问世以来的三十余年间，先后产生了适合于不同机型的各种各样版本。其中影响最大的莫过于Turbo Pascal系列软件。它是由美国Borland公司设计、研制的一种适用于微机的Pascal编译系统。该编译系统由1983年推出1.0版本发展到1992年推出的7.0版本，其版本不断更新，而功能更趋完善。
Turbo Pascal语言是编译型程序语言，它提供了一个集成环境的工作系统，集编辑、编译、运行、调试等多功能于一体

Pascal有5个主要的版本，分别是Unextended Pascal、Extended Pascal、Object-Oriented Extensions to Pascal、Borland Pascal和Delphi Object Pascal。其中，Unextended Pascal、Extended Pascal和Object-Oriented Extensions to Pascal是由Pascal标准委员会所创立和维护的，Unextended Pascal类似于瑞士Niklaus Wirth教授和K.Jensen于1974年联名发表的Pascal用户手册和报告，而Extended Pascal则是在其基础上进行了扩展，加入了许多新的特性，它们都属于正式的Pascal标准；Object-Oriented Extensions to Pascal是由Pascal标准委员会发表的一份技术报告，在Extended Pascal的基础上增加了一些用以支持面向对象程序设计的特性，但它属于非正式的标准。Borland Pascal和Delphi Object Pascal是由Borland公司专门为其开发的编译工具设计的Pascal语言，前者是用于DOS的Turbo Pascal系列和Windows 3.x的Turbo Pascal for Windows的传统高级语言，后者是用于Windows的Delphi和Linux的Kylix的面向对象程序设计语言，它们都不是正式的Pascal标准，具有专利性。但由于Turbo Pascal系列和Delphi功能强大并且广为流行，Borland Pascal和Delphi Object Pascal已自成为一种标准，为许多人所熟悉。

三、PASCAL语言的影响

高级语言发展过程中，PASCAL是一个重要的里程碑。PASCAL语言是第一个系统地体现了E.W.Dijkstra和C.A.R.Hoare定义的结构化程序设计概念的语言。1971年，瑞士联邦技术学院尼克劳斯·沃尔斯（N.Wirth）教授发明了另一种简单明晰的电脑语言，这就是以电脑先驱帕斯卡的名字命名的PASCAL语言。PASCAL语言语法严谨，层次分明，程序易写，具有很强的可读性，是第一个结构化的编程语言。它一出世就受到广泛欢迎，迅速地从欧洲传到美国。沃尔斯一生还写作了大量有关程序设计、算法和数据结构的著作，因此，他获得了1984年度“图灵奖”。

四、PASCAL在学习和竞赛中的应用

在中国的信息学奥林匹克竞赛中，过去比较常用的Pascal编程工具是Turbo Pascal。Turbo Pascal是DOS下的一种16位编程工具，在Delphi出现之前，它是世界上最多人使用的Pascal编程工具，拥有编译速度极快的先进编译器和功能强大而又简便易用的集成开发环境（IDE），在微机程序员中广为流行，正是它的出现奠定了Pascal在DOS/Windows平台上不可动摇的根基，现在常见的版本有Turbo Pascal 5.5、Turbo Pascal 6.0和Borland Turbo Pascal with Objects 7.0。Turbo Pascal 6.0与Turbo Pascal 5.5相比，主要是IDE更为强大，而其程序设计功能改变不大，只是增加了一些新的功能，例如可以内嵌asm汇编语句等。而Borland Turbo Pascal with Objects 7.0（简称Borland Pascal 7.0）则有了新的飞跃，首先是IDE进一步加强，提供了程序浏览器，然后是程序设计功能有了很大的提升，新增了一些十分有用的标准子程序，支持比较完善的面向对象程序设计功能，并提供了DOS实模式、DOS保护模式和Windows模式三种程序编译模式，能够编写出可以使用扩充内存（XMS）的保护模式应用程序或者在Windows 3.x下运行的Windows程序，另外还提供了一个对象窗口库（OWL），使用它可以快速的开发出具有一致的视窗界面（DOS或Windows 3.x）的应用程序。Borland Pascal 7.0在1992年推出，是Turbo Pascal系列在DOS下的最后版本。

下面列出Turbo Pascal的编年史：

1983 Turbo Pascal 1.0
Turbo Pascal 2.0
Turbo-87 Pascal 提高实数运算速度并扩大值域
1985 Turbo Pascal 3.0 增加图形功能
Turbo BCD Pascal 特别适合应用于商业
1987 Turbo Pascal 4.0 提供集成开发环境(IDE)，引入单元概念
1988 Turbo Pascal 5.0 增加调试功能
1989 Turbo Pascal 5.5 支持面向对象的程序设计(OPP)
1990 Turbo Pascal 6.0 提供面向对象的应用框架和库(Turbo Vision)
1992 Turbo Pascal 7.0 面向对象的应用系统、更完善的IDE
Turbo Vision 2.0
1993 Borland Pascal 7.0 开发 Object Windows库
(For Windows) 提供对OLE多媒体应用开发的支持
1995 Delphi Visual Pascal

Turbo Pascal语言是编译型程序语言，它提供了一个集成环境的工作系统，集编辑、编译、运行、调试等多功能于一体。

现在，随着Turbo Pascal逐渐被淘汰，全国信息学奥林匹克竞赛决赛（NOI）和国际信息学奥林匹克竞赛（IOI）已经指定Free Pascal为比赛使用的Pascal编程工具。Free Pascal是由一个国际组织开发的32位Pascal编程工具，属于共享软件，可用于各种操作系统。根据编译选项的不同，它可以使用Borland Pascal兼容语法、Delphi 2 Object Pascal语法或者其它语法进行编写程序。由于它拥有32位的编译器，而且一直在更新发展中，因此它的功能比Borland Pascal更加强大，拥有许多现代程序设计的特征，但同时也很不成熟，存在很多漏洞。Free Pascal正处于发展初期，相应的函数库十分少，对程序员的吸引力远比不上拥有VCL和CLX的Delphi和Kylix。

五、Pascal的基本运用

Turbo Pascal系列软件作为开发系统软件与就任软件及实施科学计算和教学的有力工具，下发挥着越来越大的作用。也是国际和全国青少年信息学奥林匹克竞赛指定的语言之一。从历届信息学竞赛的情况看，它是最能出成绩和选手最欢迎的语言。以后的例子就以Turbo Pascal 7.0进行程序设计。

下面我们就以一个实例来看一看Pascal程序的结构，从中认识到Pascal语言程序的书写方式，以及其规范的标准设计方式。

例1：输入一个圆的半径，求出其圆周长。

设圆的半径为R，周长为L，我们知道公式如下:

$L=2\pi R$

它的Pascal程序如下:

```
program yzhch( input, output) ; {程序首部}
```

```
const {常量说明}
pi=3.14159
var {变量说明}
a,b,l,s:real;
begin {程序开始}
readln(r); {输入半径}
l:==2*pi*r; {计圆周长}
writeln('l=',l); {输出圆周长}
end. {结束程序}
```

从以上简单的例子可以看出，Turbo Pascal程序是由程序首部、程序说明部分和程序执行部分组成。具体如下所示：

```
program 程序名; {程序首部}
说明部分 {说明部分}
begin {程序开始}
语句1； {执行语句}
语句2； {执行语句}
…… {执行语句}
end. {结束程序}
```

上面程序由如下两部分组成:

1、程序首部

程序首部是程序的开头部分，由保留字program后，接程序名及程序参数表组成，结束时一定要有分号。程序名yzhch是用户自己定义的标识符，参数表一般是文件变量名，用于该程序与外界的数据交流。最常用的参数为input和output。Turbo Pascal程序首部中参数表可以省略。

2、程序说明部分

Pascal语言要求用户将在程序中所使用的标号、常量、类型、变量、记录、文件、以及过程和函数除了Pascal自己预先定义的标准量之外，都必须在说明部分说明后才能在程序执行部分使用。但各个内容部分是可选的，只有执行程序部分需要的时候才进行说明。

3、程序执行部分

紧接着说明部分的begin和end之间的部分为程序的执行部分。它由一系列语句组成，一条语句执行一定的功能，所有语句完成程序设计的任务。语句之间用“;”隔开，允许一行写多个语句，也允许一个语句写多行。最后一行的end后加“.”号表示结束。所跟其后的语句将无任何作用。Begin与end应配对出现，这是每一个Turbo Pascal程序都必须的。

注意：后面将学习到的语句中，也需要引用begin和end作为程序段的分隔标记，但其必须遵守语句规则。

数据类型、常量的变量

一、数据类型的概念

计算机处理数据对象是一个广义的概念。例如，125、12.76是数据，'xiang qj zhong'这一串字符也是数据。前者是数值数据，后者是字符串数据，是非数值数据。显然，为了表示这些数据，它们在内存中必须以不同方式存放。为处理这些数据，计算机对它们施加的运算也不同。为此，Turbo Pascal语言建立了数据类型的概念，对描述的数据进行分类。每一种数据类型定义了一个具有相同性质的数据集合。各种数据类型的数据具有不同的性质。程序中所用到的每一个数据，包括常量和变量都有一个和它相联系的类型。由此决定了数据所具有的值，也决定了对该数据所能进行的操作。

Turbo Pascal语言中数据具有丰富的类型，按它们的特点可以分为简单类型、构造类型、指针类型和过程类型四大类，如图下所示。

其中，标准类型用语言系统预先定义的标准标识符表示，整型用integer表示，实型用real表示，布尔型用boolean表示，字符型用char表示。

二、常量

常量是指在程序中使用的一些具体的整型数、实型数和字符串。

(1)整型数：如9、3、-5、0等。

(2)实型数：如3?1、-6.1E+20等。

(3)字符串：是用单引号括起来的一串字符，如，'book'、'96?5'、'ABC'等。

以上列举的都可以作为常量在程序中使用。为了提高程序的可读性并使程序便于修改，在程序中往往用一些标识符来代表具体的常量。

在Turbo Pascal语言中，可以给一些常量取个名字用一个标识符代表它，这就是常量定义。例如，Cost=60；Blank=' '。

经常量定义的标识符又称为常量标识符。

在Turbo Pascal语言中，常量定义要写在常量定义部分中。

常量定义部分的一般形式:

```
Const
(常量标识符1)= (常量1);
(常量标识符2)=(常量2);
(常量标识符n)=(常量n);
```

Const是保留字，表示开始一个常量定义部分，其后可以有若干个常量定义，这些常量定义之间要用“；”号分隔。例如：

```
Const
Cost=60;
A=Cost+30;
Pi =3.14159;
```

Turbo Pascal语言对常量定义有如下要求:
(1)常量定义要放在程序的常量定义部分，即程序首部之后，执行部分之前。
(2)必须遵循先定义后使用的原则，即只有已经定义的常量标识符，才能在程序中使用。

三、变量

在程序执行过程中其值可以改变的数据，称为变量。每个变量都要有一个名称，这就是变量名。变量名由用户自己定义，但必须符合标识符的规定。

在一个程序中，一个变量只能属于一种确定的数据类型。因此，程序中出现的每个变量都必须说明其数据类型，这样就规定了该变量的取值范围，也决定了对该变量所能执行的运算操作。

变量的类型，可以是标准数据类型integer、real、boolean和char，也可以是用户自定义的各种类型。

变量说明形式是：一个变量标识符或由逗号隔开的多个变量标识符在它的冒号"："后面说明其数据类型。

在Turbo Pascal程序中，变量说明要写在变量说明部分中。

变量说明部分的一般形式:

```
var
(变量说明1)；
(变量说明2)；
……(变量说明n)；
```

其中var是保留字，表示一个变量说明部分开始。一个var可以含有多个不同的变量说明，每个变量说明之间用分号隔开，有时称被分号隔开的变量说明为变量说明项。例如：

```
var
x，y：real；
ch1：char；
t，f：boolean；
```

注意:不同类型的变量一般不能互相串用。

这里还应指出，变量一经说明系统就在计算机内存中为其分配一个存贮空间。在程序中使用到变量时，就在相应的内存中存入数据或取出数据，这种操作称为变量的访问。

标准数据类型

Pascal向程序设计者提供了丰富的数据类型，它们用于专门的目的，但却都是由简单的、非构造型的数据类型所构成的。本节介绍Turbo Pascal中最为基本的几种数据类型：整型、实型、布尔型和字符型。它们都是系统定义的简单数据类型，称为标准数据类型，其对应的名字称为标准标识符。

1、整型

一个整型数据用来存放整数，整型数据可以是正整数、负整数和整数零。

Turbo Pascal中的整型常数必须按规定严格书写。

Turbo Pascal支持五种预定义整型，它们是短整型(Shortint)、整型(Integer)、长整型 (Longint)、字节型(Byte)和字类型(Word)，每一种类型规定了相应的整数取值范围以及所占内存字节数(一个字节为8个二进制位)。因此，用户在具体编程定义变量类型时，要根据它们的特点选用适当的类型，以达到理想的效果。当两个不同范围类型的操作数进行运算时，得到的结果属于较大范围的类型。如下表所示。
Turbo Pascal语言规定可以对整型数据进行算术运算符+、一、*、Div、Mod。
它们分别表示加、减、乘、整除和取余。这五种运算，要求参加运算的两个数都是整型数，运算结果也是整型数。前三种运算与一般的算术运算加、减、乘相同。Div整除运算，是两个整型数相除取整数部分(商的整数部分)，得到整型结果。Mod取余运算，是两个整型数相除取余数，余数的符号与被除数符号相同。例如：
3 Div 2 = 1 5 Div 7 = 0
6 Div (-4) = - 1 (-12) Div (-5) = 2
7 Mod 4 = 3 (14) Mod (-4) = 2
(-18) Mod (-6) = 0 6 Mod l7 = 6
由此可见，a Mod b，所得结果的符号与a相同，其值(绝对值)在0～｜b｜-1之间。运算符Mod与Div之间有如下关系：
a Mod b = a – (a Div b) * b (b<>0)
其中Mod运算的结果的符号与a的符号相同。
利用以上两种运算可以对正整数进行分离。例如：
n为四位数8531，可用下法分离出它的个、十、百、千位。
8531 Mod l0 = 1 (个位数)
(8531 Mod l00) Div l0 = 3 (十位数)
(8531 Mod l000) Div l00 = 5 (百位数)
8531 Mod l000 = 8 (千位数)
利用 a Mod b可以判断a能否被b整除。当a Mod b = 0时，a能被b整除。
2、实型
一个实型数据用来存放实数。实型数据可以是正实数、负实数和实数零。实型数据一般用小数或指数形式(亦称科学表示法)表示。例如：
+1993，33，3.5E+8(=3.5×105)， -0.5E-3(=-0.5×10-3，)，-20.0，，0.0等都是合法实型数。
Turbo Pascal支持一种预定义实型，它们是基本实型(Real)、单精度实型(Single)、双精度实型(Double)、扩展实型(Extended)和装配实型(Comp)。每一种类型规定了相应的实数取值范围和所占内存字节数，以及它们所能达到的精度，即有效数字位数。因此，用户在具体编程时应根据以上的参数适当选用，以达到最佳效果。如下表所示。
对于此类实型数据，若其绝对值大于上界，则产生上溢；绝对值小于下界，则产生下溢，下溢导致结果为0。Comp类型的取值范围是-263+1～238-1之间的整数，相当于十进制的-9.218～9.218。由

于Comp类型的数据表示成二进制形式的数，这种类型的变量有时处理起来比较方便，特别对于数值很大的整数间的计算，这种数据类型很有用。
Turbo Pascal语言允许实型数使用下列运算符进行运算。
运算符：+、-、*、/
分别表示加、减、乘和除。其中"/"叫实数除，即使两个整型数相除，其结果也总是实型，如： 7/2=3.5 6/3=2.0
3、字符型
用标准标识符Char标明字符型。字符型数据可以是字母、符号、数字(0-9)等ASCII码的所有字符。Turbo Pascal支持扩展ASCII码，共包括256个字符。但非印刷字符是不能在标准显示上显示或打印输出。在计算机内部，字符集的元素是以该元素在字符集内的顺序位置来标记的，位置取值范围为0~255，我们称这些整数为字符在字符集内的序数值或序号。每个字符型数据在内存中占一个字节。将字符用单引号括起来，即成字符常数，如，'X'，'7'，'?'。字符常数可按字符的序数值确定大小关系，也就是说它们的大小由它们所对应的ASCII码值决定，如：'Y'，'Z'，'A'<'a'。
由于采用ASCII码，字符依ASCII码序号排列。这样，字符与ASCII码序号有一一对应的映射关系。
4、布尔型
一个布尔型数据用来存放逻辑值，或称布尔值。Turbo Pascal支持预定义布尔型，以标准标识符Boolean表示。Boolean一词，系根据19世纪英国数学家George boole (1815-1864)的名字而得，George boole为现代布尔代数之父。布尔型数据的值只有两个：True(逻辑眞)和False(逻辑假)。布尔型是顺序类型，规定False<TRUE，FALSE的序号为0，TRUE的序号为1。
逻辑运算的结果只有两个：True（眞）和False（假）。Turbo Pascal提供了六种关系运算符和三种逻辑运算符：
=(等于)、<（小于）、<=（小于等于）、>（大于）、>=（大于等于）、<>（不等于）
NOT（非）、AND（与）、OR（或）
运算关系
函 数
标准函数。Turbo Pascal语言提供了自变量为整型量的标准函数有顺序函数算术函数和转换函数等。
标准函数是Turbo Pascal语言预先定义的，它们实际上是能完成特定功能的称步子程序的程序段。每个标准函数都用一个标识符来标识，每个标准函数都能完成一个特定的功能，在程序中可以直接调用它们。Turbo Pascal语言中某些标准函数与数学中的函数有相似之处。
一、整数类型函数
整型是顺序类型，即所有的整型数都是按一定的顺序排列的。如3的后序数是4，350的后序数是351。以后介绍的布尔型、字符型、枚举类型和子界类型等都是顺序类型。顺序函数可以对顺序类型数据进行操作，但要注意它们自变量的取值范围。
①前趋函数：Pred(x)函数值为x-l，例如：

Pred (6)=5 Pred (-21)=-22
②后继函数：Succ (x)函数值为x+l，例如：
Succ (l5)=16 Succ (-114)= -113
③绝对值函数：Abs (x)函数值为｜X｜，例如：
Abs (-119)=119 Abs (101)=101
④平方函数：Sqr (x)函数值为X2，例如：
Sqr (-5)=，25 Sqr (l0)= 100
以上四个函数的结果仍是整型数。
⑤奇函数：Odd (x)，函数的结果为布尔型。当X为奇数时，函数值为true；当X为偶数时，函数值为false。例如：
Odd (13)= True Odd (16)= False
⑥字符函数：Chr (X)，函数值是序号的ASCll字符，属字符型。例如：
Chr (65)='A' Chr (32)=' '
二、实数类型函数
在下列算术函数中，X可以是实型或整型数的表达式。对于函数Abs和Sqr，其结果类型和变量X的类型相同，其他算术函数的结果类型都是实型。
绝对值函数Abs(x)：函数值为x的绝对值
平方函数Sqr (x)：函数值为x的平方
小数函数Frac (x)：函数值为x的小数部分
整数函数Int (x)：函数值为x的整数部分
正弦函数Sin (x)：函数值为x的正弦，其申，的单位为弧度
余弦函数Cos (x)：函数值为x的余弦，其中，的单位为弧度
指数函数Exp (x)：函数值为了ex
对数函数Ln (X)：函数值为x的自然对数
平方根函数的Sqrt (x)：函数值为x的平方根
反正切函数Arctan(x)：函数值为x的反正切，单位为弧度
随机函数Random：无自变量时，函数值取(0,1)间的随机小数；有自变量且为Word类型时，函数值取(0，自变量)间的随机整数。
三、字符类型函数
Turbo Pascal语言提供如下自变量为字符型的标准函数，其中Chr为字符型。
后继函数Succ (ch)：例如，Succ ('8')='9' Succ ('E')='F'
对字符集的最后一个字符，Succ函数无意义。
前趋函数Pred (ch)：例如，Pred ('7')='6' Pred ('B')=' A'
序数函数Ord (ch)：:给出字符ch在ASCll字符集中的序号，结果为整型。

注意：Ord (’7’)<>7，正确的是：Ord (’7’)=Ord(’0’)+7=48+7=55
若ch是数字字符，则Ord (ch)-Ord ('0')是该数字字符的数值。例如：Ord (’7’)-Ord(’0’)=7
前面介绍的字符函数Chr (i)是Ord (ch)的逆函数。例如:
Chr (55)= ’7’ Chr (Ord(’A’))=’A’
三、布尔类型函数
Turbo Pascal语言提供布尔型函数主要是几个字符型函数。
Ord (B) 例如：Ord (false)=0 Ord (true)=1

表达式
运算是对数据进行加工处理的过程，得到运算结果的数学公式或其它式子统称为表达式。表达式可以是常量也可以是变量或算式，在表达式中又可分为：算术表达式、逻辑表达式和字符串表达式。
1、算术表达式：
算术表达式是最常用的表达式，又称为数值表达式。它是通过算术运算符来进行运算的数学公式。我们先来看Visual Basic中的算术运算符：
算术运算符
运算符表达式 说 明 举 例
* X*Y 求X乘Y的值 6*7=42
/ X/Y 求X除Y的值（浮点数运算） 2.76/1.2=2.3
div X div Y 求X除Y的整数商（对整型数计算） 25\4=5
Mod X mod Y 求X除Y的余数（对整型数运算） 25 mod 4=1
+ X+Y 加法运算 32+2=34
- X-Y 减法运算 48-21=27
由于Visual Basic只能识别按其格式书写的数学表达式，所以必须将我们常用的数学表达式转换成Visual Basic表达式。例如：
数学式 Visual Basic表达式
2、逻辑运算
逻辑运算的结果只有两个：True（眞）和False（假）。Visual Basic提供了六种关系运算符和三种逻辑运算符：
=(等于)、<（小于）、<=（小于等于）、>（大于）、>=（大于等于）、<>（不等于）
NOT（非）、AND（与）、OR（或）
运算关系
p q NOT p p AND q p OR q
True True False True True
True False False False True
False True True False True

False false True False False
例如：5>3 结果为 True, “a”>”b” 结果为False。
3、表达式的运算优先顺序
在进行表达式的转换过程中，必须了解各种运算的优先顺序，使转换后的表达式能满足数学公式的运算要求。运算优先顺序为：
括号→函数→乘方→乘、除→加、减→字符连接运算符→关系运算符→逻辑运算符
如果同级的运算是按从左到右次序进行；多层括号由里向外。
例：
（10+6）*3^2*COS(1)/2*8+7
① ④ ③ ⑤ ② ⑥ ⑦ ⑧
Sqrt(Abs(p/n-1))+1
④ ③ ① ② ⑤

perl

Perl 最初的设计者为拉里·沃尔（Larry Wall），它于1987年12月18日发表。Perl借取了C、sed、awk、shell scripting以及很多其他程序语言的特性。
Perl 一般被称为“实用报表提取语言”（PracticalExtraction andReportLanguage），虽然有时被称做“病态折中垃圾列表器”(PathologicallyEclectic Rubbish Lister)。它是术语，而不仅仅是简写，Perl的创造者，LarryWall提出第一个，但很快又扩展到第二个。那就是为什么“Perl”没有所有字母都大写。没必要争论那一个正确，Larry 两个都认可。
你也可能看到“perl”，所有的字母都是小写的。一般，“Perl”，有大写的 P，是指语言本身，而“perl”，小写的 p，是指程序运行的解释器。
Perl的正式网站是 www.perl.org。
Perl 的特点
Perl的解释程序是开放源码的免费软件，使用Perl不必担心费用。Perl能在绝大多数操作系统运行，可以方便地向不同操作系统迁移。
Perl 是一种能完成任务的语言。从一开始，Perl 就设计成可以把简单工作简单化，同时又不失去处理困难问题能力的语言。它可以很容易操作数字，文本，文件和目录，计算机和网络，特别是程序的语言。这种语言应该很容易运行外部的程序并且扫描这些程序的输出获取感兴趣的东西。而且它还应该很容易能把这些你感兴趣的东西交给其它程序做特殊的处理。当然，这种语言还应该很容易在任何现代的操作系统上可以移植地编译和运行。
Perl 基本语法
标量定义，以$号开头，如：$num =1;
数组定义，以@开头，如：@array = (1,2,3);
数组元素调用 @array[index]，其中index表示数组下标，如上例，@array[0]的值是1

散列定义，以%开头，如：%hash=("a",1,"b",2);
散列调用 %hash，其中key表示键值，如上例，%hash{"b"}的值是1
Perl 的哲学
PERL 追求的是简单, 解决一个一般的问题用它几行代码就完成了. 一个稍复杂一点的问题代码也不会超过一屏! 事实上, 大多数人用PERL写的程序大多都没超过100行.
Perl 最初是当做一种 Unix 的胶水语言设计的，但是她早就移植到大多数其它操作系统里了。因为 Perl 几乎可以在任何地方运行，所以 Perl 可以说是当今最具有移植性的编程环境。要想写可移植的 C/C++ 程序，你得在程序里加上一大堆 #ifdef 标签来区分不同的系统。要想写可移植的 Java 程序，你必须理解每种新的 Java 实现的特质。要想写可移植的
shell，你可能要记住每条命令在每种操作系统上的语法，走运的时候你可能可以找到一些公共的东西。而要想写可移植的 Visual Basic 程序，那么你只需要对“移植”有个更灵活的定义就可以了。
我们很高兴的是 Perl 避免了所有这些问题，同时还保留了这些语言中的许多优点，同时还有一些自己的特色。Perl 的特色来自许多方面：它的特性集的工具，Perl 社区的创造性，以及开源运动的大环境。不过，许多这些特性都是混合的东西；Perl 的身世复杂，它总是把事物看成是优点的不同方面，而不是弱点。Perl 是“背黑锅我来”的语言。如果你觉得自己陷入一团乱麻之中，非常渴望自由，那么请使用 Perl。
Perl 是跨文化的。Perl 的爆炸性增长很大程度上是因为那些前 Unix 系统程序员的渴望，他们希望从他们的“老家”带着尽可能多的东西。对于他们而言，Perl 是可移植的 Unix 文化蒸馏器，是"此路不通"的沙漠中的绿洲。从另外一个角度来看，Perl 还可以从另外一个方向运转：在 Windows 上工作的 web 设计者通常会非常开心地发现他们的 Perl 程序可以不加修改地在 Unix 服务器上跑。
尽管 Perl 在系统程序员和 web 设计师中间非常流行，但这只是因为是他们最早发现 Perl 的，Perl 可以用于更广泛的用途。从 Perl 最早的文本处理语言开始，它已经发展成为一种非常复杂的，通用的编程语言，以及完整的开发环境，包括调试器，调节器，交叉引用，编译器，库，语法提示编辑器，以及所有其它“眞正”的编程语言所具有的所有挂勾，只要你需要。当然这些东西都是让我们可能处理难的问题的东西，而且很多其它语言也可以做到这一点。Perl 之所以成为 Perl 是因为它从来不会因为保持简单事情简单化而丢失其他方面的特性。
因为 Perl 既强大又好用，所以它被广泛地用于日常生活的方方面面，从宇航工程到分子生物学，从数学到语言学，从图形处理到文档处理，从数据库操作到网络管理。很多人用 Perl 进行快速处理那些很难分析或转换的大批量数据，不管你是处理 DNA 序列，网页，还是猪肚皮的未来都无所谓。实际上，在 Perl 社区有一个笑话就是，下次股市大崩盘就很有可能是呢个家伙写的脚本里头有臭虫造成的。（不过，乐观点来看就是，任何还在失业的股票分析师仍然有可以利用的技巧。）
Perl 的成功有许多原因。Perl 早在开源软件的名字出现之前就已经是一个成功的开源项目了。Perl 是自由的，并将永远自由下去。你可以在任何合适的场合使用 Perl，只需要遵守一个非常自由的版权就可以了。如果你在从事商业活动并且还想使用 Perl，那么用就是了。你可以把 Perl 嵌入到你写的商业

软件中而不需要支付任何费用也没有任何限制。如果你碰上一个 Perl 社区解决不了的问题，那你也还有最后的一招：源程序本身。 Perl 社区不会在“升级”的伪装下租给你它们的商业秘密。而且 Perl 社区也不会“停业”，更不会让你孤立无援。

Perl 是自由软件这一点无疑对它是有帮助的。但这一条并不足以解释 Perl 现象，因为许多自由软件包没有能繁荣起来。Perl 不仅自由；而且好玩。人们觉得自己在 Perl 里可以有创造力，因为它们有表达的自由：他们可以选择是为计算机速度优化还是为程序员的速度优化，是冗长还是简洁，是选择可读性还是可维护性，或者选择复用性，移植性，接受性和传授性等等。假如你进入一次模糊的 Perl 比赛，甚至你还可以为模糊性做优化。

Perl 可以给予你所有这些自由，因为它是一门有着分裂人格的语言。Perl 同时是很简单并且很富有的语言。Perl 从其它地方拿来好主意，然后把它们安装到易用的框架里面。对于只是喜欢她的人来说，Perl 是实用抽取和报表语言（Practical Extractoin and Report Language）。对那些热爱她的人而言，她是变态电子垃圾制造者（Pathologically Electric Rubbish Lister）。在少数人眼里，Perl 是毫无意义的重复练习。不过世界需要一点点冗余。精简主义者总是想把事物分隔开。而我们则总是企图把它们合并到一起。

Perl 之所以是简单的语言是有很多原因的。比如你用不着知道什么特殊的指令就可以编译 Perl 程序--只要把它当做批处理或者 shell 脚本执行就可以了。Perl 的类型和结构很容易使用和理解。Perl 对你的数据没有任何限制--你的字串和数组可以要多长就多长（只要你有足够的内存），而且它们都会自动增长。Perl 不会强迫你学习新的语法和语意，Perl 改从许多其它你已经熟悉的语言里（比如 C, awk, BASIC 和 Python, 英文，希腊语等）借来语法。实际上，任何程序员都可以从书写良好的 Perl 代码段中读懂它的含义。

最重要的是，你不用先学习所有 Perl 的东西就可以开始写有用的程序。你可以写很小的 Perl 程序。你也可以象小孩那样写 Perl 程序，我们保证不会笑话你。或者更准确地说是，我们绝不会笑话小孩做事情的创造性。Perl 里的许多观点都是从自然语言中借来的，其中一条最好的观点就是只要你能把自己的意思表述清楚，那么你就可以使用这些语言的一个子集。Perl 文化可以接受任何熟练程度的成员。我们不会在你背后放个语言警察。如果你的老板不炒你，而且你的 Perl 脚本也能完成工作，那么它就是“正确”的。

尽管 Perl 很简单，但它仍然是一种特性很丰富的语言，如果你想用那些特性的话，那你就要学习一些东西。这也是把难题变简单的学费。虽然你要想把所有 Perl 能做的事情吸收还需要一些时间，但到你需要这些功能的时候你就会非常开心地发现 Perl 已经可以做这些事情了。

由于 Perl 的继承性，就算它只是用做数据归纳语言的时候也有丰富的特性，Perl 一开始就设计成可以浏览文件，扫描大量文本并且生成动态数据以及打印出这些数据的良好格式化的报表。不过，随后 Perl 就开始风行，于是它就成了可以操作文件系统，进程管理，数据库管理，进行 C/S 编程和安全编程，web 信息管理，甚至可以进行面向对象和面向功能的编程的语言。而且这些功能并非只是在 Perl 这边，每种新功能都和其它东西交流得很好，别忘了 Perl 从一开始就是设计成胶水语言的。

而且 Perl 并不仅仅只能黏合它自己的特性。Perl 是设计成可以用模块扩展的语言。你可以用 Perl 快速设计，编写，调试和部署 Perl 应用，并且你还可以在需要的时候很方便地扩展这些应用。你可以在其它语言里嵌入 Perl，而且你也可以在 Perl 里嵌入其它语言。通过模块输入机制，你可以把这些外部的扩展当做内置于 Perl 的特性。那些面向对象的外部库在 Perl 内部仍然保持面向对象的特征。

Perl 还在许多其它方面协助你。和严格的每次执行一条命令的命令文件和 shell 脚本不同的是，Perl 先把你的程序快速编译成一种内部格式。和其它任何编译器一样，这个时候还进行各种优化，同时把碰到的任何问题反馈给你。一旦 Perl 的编译器前端对你的程序表示满意了，它就把这些中间代码交给解释器执行（或者是给其它的能生成 C 或者字节码的模块后端）。听起来挺复杂，不过 Perl 的编译器和解释器干这些活效率相当高，我们的编译-运行-修改的过程几乎都是以秒计。再加上 Perl 的许多其他开发特性，这种快速的角色转换很适合做快速原型设计。然后随着你的程序的成熟，你可以逐步拧紧身上的螺母，减少散漫增强记律。如果你做得好，Perl 也能帮你这个忙。

Perl 还可以帮你写更安全的程序。除了其它语言提供的典型的安全接口之外，Perl 还通过一种跟踪数据的机制给你提供预防意外安全错误的保护，这样就可以在灾害发生之前预防其发生。最后，Perl 还可以让你设置一个特殊的防护隔段运行那些来源不明的 Perl 代码，以此来杜绝危险操作。

不过，偏执一点儿说，Perl 帮你的大部分内容和 Perl 本身没有什么关系，而是和使用 Perl 的人有关。坦率地说，Perl 社区的人们可以说是地球上最热心的人了。如果 Perl 运动里面有那么一点点宗教色彩的话，那么这就是它的核心了。Larry 希望 Perl 社区像一小片天堂那样运转，目前看来他的愿望基本上是实现了。我们也请你为此做出自己的努力。

Perl之所以强大, 是因为有CPAN, CPAN上面有无数的开源模块, 从科学计算到桌面应用到网络等等各个方面都有大量的模块! 并且现在世界上也还有无数的人在向上面添加模块! 如果你想要用PERL实现某功能, 不用自己做, 在CPAN上面搜一搜, 多半都会得到已有的结果! CPAN（"the Comprehensive Perl Archive Network"全面的 Perl 存档网络）是查找任何 Perl 有关的东西的中心仓库。它包含从整个 Perl 社区收集来的智慧：成百上千的 Perl 模块和脚本，相当于好几本书的文档，以及整个 Perl 发布。如果有东西是用 Perl 写的，而且这个东西很有用而且是自由的，那么它很有可能就在 CPAN 上。CPAN 在全世界都有镜象，你可以在位于 http://www.perl.com/CPAN 的 CPAN 路牌上找到离你最近的镜象。那块路牌会记住你选择的是哪个镜象并且你以后再访问 http://www.perl.com/CPAN/（注意最后的斜杠）的时候就会自动重新定向到那个镜象。另外，你也可以从 www.cpan.org开始。这个站的界面不同，但是数据是一样的。

Perl 文化

1.时势造英雄

为了理解 Perl 为什么用现在这样的样子定义（或者为什么不定义成其他的样子），我们必须首先明白为什么会有 Perl。所以，让我们先挖掘一下步满尘灰的历史书....

退回到 1986 年，Larry 是一个系统程序员，在做一个多层安全的广域网项目的开发。他负责这么一个系统，这个系统由西海岸的三台 VAX 和三台 sun 机器，通过一条加密了的 1200 波特的串行线路和东

海岸类似配置的系统连接组成的，因为 Larry 的主要工作是支持（他不是该项目的程序员，只是系统专家），所以他就有机会利用他的三种优点（懒惰，不耐心，和狂傲）来开发和提高所有有用的工具——比如 rn，patch，和 warp。（注：正是在这个时候，Larry 被划入了“计算机动物”的范畴，这是以那些人的不可遏止的“再加一个特性”的渴望为基础评判的，因为这种行为几乎成了生物必须。毕竟，如果生活就是太复杂的话，难道程序就不会吗？尤其是想 rn 这样的程序，它真是应该当作一个高级的人工智能项目来看待，因为他们就可以为你阅读新闻。当然，有些人已经在说 patch 程序太复杂了。）

一天，Larry 刚刚把 rn 撕成碎片，把它一片一片地放在他的目录里，大管理员就跑进来说，“Larry，我们需要一个管理配置，用它控制所有六台 VAX 和六台 sun。我们想在一个月里就要它。你做一个吧！”

所以，从不逃避工作的 Larry，开始问自己做一个两个海岸的 CM 系统的最好的方法是什么，它必须不用自己从头开始写，并且还可以查阅两个海岸的问题报告以及核准和控制。他想到的答案只有一个词：B-news。（注：也就是 Usenet 传输软件的第二种实现。）

Larry 着手在这些机器上安装了新闻软件并且增加了两条控制命令：一条“append”命令用于向现有的文章追加内容，和一条“synchronize”命令保持两个海岸的文章数目相同。CM 可以用 RCS （版本控制系统）做，而核准和控制可以用新闻和 rn 来做。到目前挺好。

然后大管理员让他生成报告。新闻是在核心机器里的一个独立的文件里维护的，里面有许多文件间的交叉引用。Larry 的第一个反应是“用 awk。”糟糕的是，那个时候的 awk 无法做到以文件里的信息为基础打开和关闭多个文件。Larry 不想编写一个特殊目的的工具。结果就是产生了一种新的语言。

最初这种新的语言并不叫 Perl。Larry 和他的同事及亲友（Dan Faigin，写这段历史的人，和 Mark Biggar，他的妻弟，在初始设计阶段帮了大忙）交换了一大堆名字。实际上 Larry 考虑并抛弃了字典里的所有三个或四个字母的单词。最早的名字是“Gloria”，以他的宝贝（和老婆）命名。但他很快就发现这样会产生太多家庭混乱。

然后名字就成了“Pearl”，最后它变成了我们现在的“Perl”，部分原因是 Larry 看到另外一种叫 PEARL 的语言的介绍，但最主要的原因是他懒得总要敲五个键。当然，这样 Perl 就可以用做一个四字母的词。（不过，你会注意到，这里有以前首字缩写的残余：“Practical Extraction And Report Language”。）

最早的 Perl 没有今天的 Perl 那么多的特性。那时候有模式匹配和文件句柄，有标量，有格式化，但是很少有函数，没有相关的数组，而且只有一个实现得不怎么样的正则表达式，（从 rn 借来的）。手册页也只有 15 页。但是 Perl 比 sed 和 awk 快，并且开始在该项目的其他应用里使用。

但是其他地方又开始需要 Larry 了。有一天另外一个大经理来了并且说：“Larry，给 R&D 做支持。”并且 Larry 说，好吧。他带上 Perl 并且很快发现它逐渐成为系统管理的好工具。他借来 Henry Spencer 漂亮的正则表达式软件包并且把它变成更有男人味（不过 Henry 可能不会愿意在正餐的时候考虑这些特性。）然后 Larry 增加了大部分他想要的特性，以及一些别人想要的特性。然后它就把 Perl 发布到网络上。（注：更让人吃惊的是，他先后工作于喷气推进实验室（JPL），然后是 NetLabs? 和 Seagate 之后，仍然不断发布新 Perl。现在，其他人做了大部分工作，而 Larry 假装为 O'Reilly & Associates（一个小公司，印刷关于计算机和相关事物的小册子。）其余的就是历史了。（注：而这些

东西，是历史的一个注解。当开始 Perl 的工作的时候，Larry 已经把 rn 分解成碎片，并且准备做一次全面的重写。但因为他开始在 Perl 上干活，所以 Larry 没有再碰 rn。它仍然是碎片。有时候 Larry 说要用 Perl 重写 rn，但是从来没当真。）
然后事情的发展就是这样的：Perl 1.0 在 1987 年十二月十八日发布；有些人仍然很认真地对待 Perl 的生日。Perl 2.0 在 1988 年六月发布，并且 Randal Schwartz 开始了“另外一个 Perl 黑客”的签名的传奇。在 1989 年，Tom Christiansen 在巴尔的摩 Usenix 拿出了第一个公开的 Perl 教程。1989 年十月的 Perl 3.0开始，这门语言第一次以 GNU 公众版权（GPL）发布和分发。
1990 年三月，Larry 写了第一首 Perl 小诗（见下一节）。然后他和 Randal 写了本书的第一版，The Pink Camel；该书在 1991 年早期发行。然后 Perl 4.0 就立即发布了；除 GPL 之外，它还包括了 Artistic License（艺术版权）。
万众期待的 Perl 5 在 1994 年十月发布。这是一个完全重写的 Perl 版本，它包括对象和模块。 Perl 5 的到来甚至连 The Ecomomist 杂志都提到。到了 1995 年，正式向 Perl 社区引入 CPAN。在 1996 年，Jon Orwant 开始出版 The Perl Journal 杂志。在长时间的猜测之后，本书的第二版，The Blue Camel，在那年的年末出版。第一次 O'Reilly Perl 大会（TPC） 1997 年夏季在加州 San Jose 举行。
现在，重大时间几乎是每天都在发生，所以，关于历史的其他部分，请检查 CPAST （Comprehensive Perl Arcana Society Tapestry （history.perl.org））上的 Perl 纪年表。
2. Perl 诗歌
在助手框里的诗歌的仿制品是在1990年的四月一日愚人节张贴到 Usenet 上的。我们不加注释的把它放在这里，只是想表示典型的编程语言的隐喻真的是多么让人作呕。对所有有文学价值的东西大概都是这样的吧。Larry 在最初为 Perl 3 写的那些“Black Perl”到了 Perl 5 不能分析通过之后，真是感觉轻松许多。
不过，Larry 自己的文集很幸运地被 Perl 诗歌的王后，Sharon Hopkins 的光芒所掩盖。她写了相当多的 Perl 诗歌，以及一些她在 1992 年 Usenet 冬季技术大会上拿出来的关于 Perl 诗歌的文章，标题是“Camels and Needles: Computer Poetry Meets the Perl Programming Language”。（这篇文章可以在 CAPN 的 misc/poetry.ps 找到。）除了是最多产的 Perl 诗人之外，Sharon 还是下面这首诗歌的作者，这首诗是发表得最广泛的一首，并且曾经在 Economist 和 Guardian 杂志上刊登：

```
#!/usr/bin/perl
APPEAL:
listen (please, please);
open yourself, wide;
join (you, me),
connect (us,together),
tell me.
```

do something if distressed;
@dawn, dance;
@evening, sing;
read (books,$poems,stories) until peaceful;
study if able;
write me if-you-please;
sort your feelings, reset goals, seek (friends, family, anyone);
do*not*die (like this)
if sin abounds;
keys (hidden), open (locks, doors), tell secrets;
do not, I-beg-you, close them, yet.
accept (yourself, changes),
bind (grief, despair);
require truth, goodness if-you-will, each moment;
select (always), length(of-days)
listen (a perl poem)
Sharon Hopkins
rev. June 19, 1995
Perl Poetry
Article 970 of comp.lang.perl:
Path: jpl-devvax!pl-dexxav!lwall
From: lwall@jpl-dexxav.JPL.NASA.GOV(Larry Wall)
Newsgroups: news.groups,rec.arts.poems,comp.lang.perl
Subject: CALL FOR DISCUSSION: comp.lang.perl.poems
Message-ID: <0401@jpl-devvax.JPL.NASA.GOV>
Date: 1 Apr 90 00:00:00 GMT
Reply-To: lwall@jpl-devvax.JPL.NSAS.GOV(Larry Wall)
Organization: Jet Prepulsion Laboratory, Pasadena, CA
Lines: 61
It has come to my attention that there is a crying need for a place for people to express both their emotional
and technical natures simultaneously. Several people have sent me some items which don't fit into any newsgroup. Perhaps it's because I recently posted to both comp.lang.perl and to rec.arts.poems, but people

seem to be writing poems in Perl, and they're asking me where they should post them. Here is a sampling:
From a graduate student (in finals week), the following haiku:
study, write, study,
do review (each word) if time.
close book. sleep? what's that?
And someone writing from Fort Lauderdale writes:
sleep, close together,
sort of sin each spring & wait;
50% die
A person who wishes to remain anonymous wrote the following example of "Black Perl". (The Pearl poet would have been shocked, no doubt.)
BEFOREHAND: close door, each window & exit; wait until time.
open spellbook, study, read (scan, select, tell us);
write it, print the hex while each watches,
reverse its length, write again;
kill spiders, pop them, chop, split, kill them.
unlink arms, shift, wait & listen (listening, wait),
sort the flock (then, warn the "goats" & kill the "sheep");
kill them, dump qualms, shift moralities,
values aside, each one;
die sheep! die to reverse the system
you accept (reject, respect);
next step,
kill the next sacrifice, each sacrifice,
wait, redo ritual until "all the spirits are pleased";
do it ("as they say").
do it(*everyone***must***participate***in***forbidden**s*e*x*).
return last victim; package body;
exit crypt (time, times & "half a time") & close it,
select (quickly) & warn your next victim;
AFTERWORDS: tell nobody.
wait, wait until time;
wait until next year, next decade;

sleep, sleep, die yourself,
die at last
I tried that, and it actually parses in Perl. It doesn't appear to do anything useful, however. I think I'm glad, actually... I hereby propose the creation of comp.lang.perl.poems as a place for such items, so we don't clutter
the perl or poems newsgroups with things that may be of interest to neither. Or, alternately, we should create rec.arts.poems.perl for items such as those above which merely parse, and don't do anything useful.
(There is precedent in rec.arts.poems, after all.) Then also create comp.lang.perl.poems for poems that actually do something, such as this haiku of my own:
print STDOUT q
Just another Perl hacker,
unless $spring
Larry Wall

Php

目录·PHP
·PHP的特性
·PHP 3与PHP 4
·PHP4的优越性
·数据库方面
·多态
·PHP的高级OOP技术

PHP
PHP，一个嵌套的缩写名称，是英文超级文本预处理语言（PHP:Hypertext Preprocessor）的缩写。PHP 是一种 HTML 内嵌式的语言，PHP与微软的ASP颇有几分相似，都是一种在服务器端执行的嵌入HTML文档的脚本语言，语言的风格有类似于C语言，现在被很多的网站编程人员广泛的运用。PHP 独特的语法混合了 C、Java、Perl 以及 PHP 自创新的语法。它可以比 CGI 或者 Perl 更快速的执行动态网页。用PHP做出的动态页面与其他的编程语言相比，PHP是将程序嵌入到HTML文档中去执行，执行效率比完全生成HTML标记的CGI要高许多；与同样是嵌入HTML文档的脚本语言JavaScript相比，PHP在服务器端执行，充分利用了服务器的性能；PHP执行引擎还会将用户经常访问的PHP程序驻留在内存中，其他用户再一次访问这个程序时就不需要重新编译程序了，只要直接执行内存中的代码就可以了，这也是PHP高效率的体现之一。PHP具有非常强大的功能，所有的CGI或者JavaScript的功能PHP都能实现，而且支持几乎所有流行的数据库以及操作系统。

PHP 最初是1994年Rasmus Lerdorf创建的，刚刚开始只是一个简单的用Perl语言编写的程序，用来统计他自己网站的访问者。后来又用C语言重新编写，包括可以访问数据库。在1995年以Personal Home Page Tools (PHP Tools) 开始对外发表第一个版本，Lerdorf写了一些介绍此程序的文档，并且发布了PHP1.0。在这早期的版本中，提供了访客留言本、访客计数器等简单的功能。以后越来越多的网站使用了PHP，并且强烈要且增加一些特性，比如循环语句和数组变量等等，在新的成员加入开发行列之后，在1995年中，PHP2.0发布了。第二版定名为PHP/FI(Form Interpreter)。PHP/FI加入了对mSQL的支持，从此建立了PHP在动态网页开发上的地位。到了1996年底，有15000个网站使用PHP/FI；时间到了1997年中，使用PHP/FI的网站数字超过五万个。而在1997年中，开始了第三版的开发计划，开发小组加入了 Zeev Suraski 及 Andi Gutmans，而第三版就定名为PHP3。2000年，PHP4.0又问世了，其中增加了许多新的特性。

PHP的特性

PHP的特性包括：

开放的源代码：所有的PHP源代码事实上都可以得到。

PHP是免费的。

基于服务器端：由于PHP是运行在服务器端的脚本,可以运行在UNIX、LINUX、WINDOWS下。

嵌入HTML：因为PHP可以嵌入HTML语言，所以学习起来并不困难。

简单的语言：PHP坚持脚本语言为主，与Java以C++不同。

效率高：PHP消耗相当少的系统资源。

图像处理：用PHP动态创建图像

PHP 3与PHP 4

【PHP3】

PHP3跟Apache服务器紧密结合的特性；加上它不断的更新及加入新的功能；而且几乎支持所有主流与非主流数据库；再以它能高速的执行效率，使得PHP在1999年中的使用站点已经超过了150000万。加上它的源代码完全公开，在 Open Source意识抬头的今天，它更是这方面的中流砥柱。不断地有新的函数库加入，以及不停地更新的活力，使得PHP无论在UNIX、LINUX或是Windows的平台上都可以有更多新的功能。它提供丰富的函数，使得在程序设计方面有着更好的支持。

【PHP4】

PHP4.0整个脚本程序的核心大幅更动，让程序的执行速度，满足更快的要求。在最佳化之后的效率，已较传统CGI或者ASP等程序有更好的表现。而且还有更强的新功能、更丰富的函数库。无论您接不接受，PHP 都将在 Web CGI 的领域上，掀起巅覆性的革命。对于一位专业的Web Master 而言，它将也是必修课程之一。

PHP 4.0是更有效的，更可靠的动态Web页开发工具，在大多数情况运行比 PHP 3.0要快，其脚本描述

更强大并且更复杂, 最显著的特征是速率比的增加。PHP4.0这些优异的性能是PHP 脚本引擎重新设计产生的结果：引擎由 AndiGutmans 和 Zeev Suraski从底层全面重写。PHP4.0 脚本引擎 ——Zend 引擎，使用了一种更有效的编译——执行方式, 而不是PHP 3.0 采用的执行 ——当解析时模型。

PHP4的优越性

PHP4在3.0版的基础上增加或增强了许多有用的特征，主要如下：

（1）别名：在PHP4中，可以利用引用为变量赋值，这给编程带来了很大的灵活性。

（2）扩充了API 模块：PHP 4.0 为扩展的 API 模块的提供了扩展PHP接口模块, 它比旧的 API 版本显著地快。 PHP 模块已有的及最常用的接口多数被转换到使用这个扩展的接口。

（3）自动资源释放：PHP4增加了引用计数功能，这种新技术的引入使PHP4具有了自动内存管理功能,减轻了开发人员的负担。

（4）布尔类型：PHP 4.0 支持布尔类型。

（5）进程生成：在 UNIX 环境下的 PHP 4.0 提供了一个很智能和通用的生成进程, 使用了一种名为基于automake/libtool的系统生成技术。

（6）COM/DCOM 支持：PHP 4.0 提供 COM/DCOM 支持 (仅用于Windows 环境) 可以无缝地存取和访问 COM 对象。

（7）与PHP 3.0 兼容性很好：PHP 4.0 是与 PHP 3.0 代码向后兼容性接近100% 。由于 PHP 4 的改进的体系结构,两者有一些细微的差别，但是大多数人将可能永远不可能遇上这种情况。

（8）配置：PHP4重新设计和增强了PHP。ini文件,这使得用PHP。ini来配置PHP显得极为容易,这个文件可以在运行时被Apache(unix系统)或由Windows 注册(Windows 环境)。

（9）加密支持：PHP4实现了完整的加密, 这些加密功能是一个完整的mycrypt库，并且 PHP 4.0 支持哈希函数。Blowfish，TripleDES,MD5,并且SHA1 也是可使用的一些加密算法。

（10）类型检查：PHP 4.0 支持同一操作符用于评类型检查：===（ 3 等号运算符 ）, 为在两个值和其类型之间作检查。例如, 3 ===3 将视为假 (类型是不同的), 而 3 ==3 (相等判断) 将视为眞。

（11）FTP 支持：PHP 4.0 支持 FTP 。通常, 你会为通过一个调制解调器连接下载一个大文件提供一个接口。然而, 如果你确实有需要，可以使用PHP 。

（12）PHP4新增函数或功能增强函数：PHP 4.0 新增了许多函数,同时也将许多现有的函数功能进行了增强，以下是一些例子。 array_count_values() eval() foreach() nclude() ob_end_clean() ob_end_flush() ob_get_contents() ob_start() strip_tags() unset()

（13）here打印：PHP 4.0 的Here打印是与Perl类似的, 尽管完全不相同。Here是打印大容量文章的一个有用的方法,例如在 HTML文件中，不会漏掉任何一个字符，例如目录标记。

（14）HTTP Session fallback 系统：为 HTTP Session管理的一个 fallback 系统在 PHP 4.0被实现 。缺省情况下，Session标识符由cookies存储。如果没有cookies支持或一项cookies任务失败，Session标识符自动被创建并在 URL 的查询字符串中被携带。

（15）ISAPI 支持：PHP 4.0 能作为一个个性化的 ISAPI 模块作为 IIS插件 。这比 PHP 3.0 更有效, 它

作为一个外部的程序来运行。
（16）内存：PHP 4.0 能更有效的使用内存, 导致较少的内存占用消耗,这主要归功于引用计数技术的实现。
（17）其他类成员函数：在 PHP 4.0 你能在成员函数本身的作用域或全局范围内调用其他类的成员函数。例如,你能用一个子函数覆盖父函数,并在子函数中调用父函数。
（18）多维数组：在 PHP 4.0 ，利用GET，POST,Cookies的进行的数据传输支持多维数组。
（19）个性化的 HTTP Session支持：HTTP Session处理, 包括 fallback 系统管理，在 PHP 4.0被它的新库函数实现 。在版本 3.0 中处理Session要求使用 PHPLIB 和第三方的库函数, 它比把Session直接地由 PHP 支持慢了许多。
（20）个性化的 Java 支持：PHP 4.0 支持和java的交互。这种个性化的Java 支持为PHP 在 Java 对象上创建和使用方法提供一个简单并且有效的工具。
21）对象和数嵌套组：PHP 4.0 实现了功能更加强大的对象, 移去了 PHP 3.0存在的种种句法限制。对象能在数组以内被嵌套并且反过来也如此, 可以根据你的需要实现嵌套。
（22）面向对象的编程：PHP 4.0 为面向对象的编程和构造类及对象提供扩展的功能和新特征。PHP4实现了对象重载,引用技术等新技术。
（23）对象重载支持：对象重载语法允许第三方的基于面向对象的类库使用 PHP4 的面向对象的特征存取他们自身的功能。使用这个特征的一个 COM 模块已经被实现了。
（24）输出缓冲支持：PHP 提供了一个输出缓冲函数集合。输出缓冲支持允许你写包裹函数功能压缩缓冲区。在 PHP4 的输出缓冲支持允许 HTML 头信息存放, 无论 HTML的正文是否输出。头信息((header(), content type, and cookies) 不采用缓冲 。
（25）增加了PCRE 库：PHP 4.0 包括一个 Perl 兼容的正则表达式 (PCRE) 库, 和正常regex库一起与 PHP 绑定。split 和replace PCRE 功能被支持。PCRE 和 Perl 正规表达式之间有一些细微差别。
（26）PHP.ini 文件：PHP.ini文件在 PHP4.0 被重新设计, 使用的 PHP 的配置PHP.ini是更容易并且更有效的。全部文件能被Apache 在运行时间操作 (在 Apache环境 下) 或由 Windows 注册表 (在 Windows 下面) 。被加入PHP.ini文件的配置指令自动地在所有相关的模块中被支持。
（27）引用计数：PHP 4.0 为系统中的每个数值提供了引用计数, 包括资源。一旦一个资源不再被任何变量引用，它自动地被释放以节省内存资源。利用这个特征的最明显的例子一个内置SQL查询的循环语句。在PHP 3.0中 ，每次递归另外的 SQL 结果集合重复申请内存，直到脚本执行完毕,这些结果集合占用的内存才被释放。
（28）支持引用：通过引用可以改变一个变量的值。
（29）函数的运行时绑定：PHP 4.0 的运行时间绑定功能允许你在他们被声明以前调用, 无论声明是否在代码以后或是在运行时间。
（30）类的运行时信息：PHP 4.0 支持在运行时刻存取下列类信息：一个对象的类名，一个对象的父类的类名字，以及对象函数所在的名字。

（31）服务器抽象层：为支持Web服务器提供了增强型 SAPI (服务器 API) 接口,是 PHP 4。0 不可分的一部分。这个服务器抽象层,提供了通用的WEB服务器接口支持,支持多线程WEB服务器,为大多数的WEB服务器提供透明的支持, 这些服务器包括 Apache ,IIS (ISAPI), 以及 AOL 服务器。
（32）语法的点亮显示：PHP 4.0 语法的点亮显示允许开发者看见源代码而不是脚本, 这个功能比PHP 3。0中的更有效。它跑得更快，更执行得更好，并且产生更紧凑的HTML代码。
（33）由引用改变变量的值：PHP 4.0 由引用支持可变的赋值,“关联”的2个变量之中个的任何一个的值被改变,另外的变量的值同样被改变,这类似与C中的指针类型。
（34）在引用字符串中的变量引用：PHP 4.0 增强了在引用字符串中的变量引用。

数据库方面
PHP 在数据库方面的丰富支持，也是它迅速走红的原因之一，它支持下列的数据库或是数据文件：
· Adabas D
· DBA
· dBase
· dbm
· filePro
· Informix
· InterBase
· mSQL
· Microsoft SQL Server
· MySQL
· Solid
· Sybase
· ODBC
· Oracle 8
· Oracle
· PostgreSQL
而在 Internet 上它也支持了相当多的通讯协议 (protocol)，包括了与电子邮件相关的 IMAP, POP3；网管系统 SNMP；网络新闻 NNTP；帐号共用 NIS；全球信息网 HTTP 及 Apache 服务器；目录协议 LDAP 以及其它网络的相关函数。
除此之外，用 PHP 写出来的 Web 后端 CGI 程序，可以很轻易的移植到不同的操作系统上。例如，先以 Linux 架的网站，在系统负荷过高时，可以快速地将整个系统移到 SUN 工作站上，不用重新编译 CGI 程序。面对快速发展的 Internet，这是长期规划的最好选择。
变数类型:

PHP有好多种变数; 主要有这些：
- 数字 (integer - 例: 32)
- 布林值 (boolean - 例: TRUE)
- 字串 (string - 例: 'a string of text')
- NULL
- 资源（resource）
- 阵列 (array - 例: arrayname[2])

语法:
语法有三种:
//comment
/* comment */
comment
基本的 "Control Structures":
* if ... else
if (condition == true) ;
* if ... else then
if (condition == true)
else if (condition2 == true) ;
一个PHP实例:

```
<html>
<head>
<title>First program</title>
</head>
<body>
<?php
echo "hello world";
?>
</body>
</html>
```

请看:
php官方网站：www.php.net
php对面向对象的支持
面向对象编程的概念：
　　不同的作者之间说法可能不一样，但是一个OOP语言必须有以下几方面：

抽象数据类型和信息封装
继承
多态
在PHP中是通过类来完成封装的：

```
<?php
 class Something {
 // 在OOP类中，通常第一个字符为大写
 var $x;
 function setX($v) {
 // 方法开始为小写单词，然后使用大写字母来分隔单词，例如getValueOfArea()
 $this->x=$v;
 }
function getX() {
 return $this->x;
 }
 }
```

当然你可以按自己的喜好进行定义，但最好保持一种标准，这样会更有效。数据成员在类中使用"var"声明来定义，在给数据成员赋值之前，它们是没有类型的。一个数据成员可以是一个整数，一个数组，一个相关数组(associative array)或者是一个对象。方法在类中被定义成函数形式，在方法中访问类成员变量时，你应该使用$this->name，否则对一个方法来说，它只能是局部变量。
使用new操作符来创建一个对象：

```
 $obj=new Something;
```

然后你可以使用成员函数通过：

```
$obj->setX(5);
 $see=$obj->getX();
```

在这个例子中，setX成员函数将5赋值给对象的成员变量x(不是类的)，然后getX返回它的值5。可以象：$obj->x=6那样通过类引用方式来存取数据成员，这不是一个很好的OOP习惯。我强烈建议通过方法来存取成员变量。如果你把成员变量看成是不可处理的，并且只通过对象句柄来使用方法，你将是一个好的OOP程序员。不幸的是，PHP不支持声明私有成员变量，所以不良代码在PHP中也是允许的。继承在PHP中很容易实现，只要使用extend关键字。

```
<?php
 class Another extends Something {
 var $y;
function setY($v) {
```

```
 $this->y=$v;
 }
function getY() {
 return $this->y;
 }
}
```

"Another"类的对象现在拥有了父类(Something)的全部的数据成员及方法，而且还加上了自己的数据成员和方法。

你可以使用

```
 $obj2=new Something;
 $obj2->setX(6);
 $obj2->setY(7);
```

PHP现在还不支持多重继承，所以你不能从两个或两个以上类派生出新的类来。你可以在派生类中重定义一个方法，如果我们在"Another"类中重定义了getX方法，我们就不能使用"Something"中的getX方法了。如果你在派生类中声明了一个与基派同名的数据成员，那么当你处理它时，它将“隐藏”基类的数据成员。

你可以在你的类中定义构造函数。构造函数是一个与类名同名的方法，当你创建一个类的对象时会被调用，例如：

```
<?php
 class Something {
 var $x;
function Something($y) {
 $this->x=$y;
 }
function setX($v) {
 $this->x=$v;
 }
function getX() {
 return $this->x;
 }
}
```

所以你可以创建一个对象，通过：

```
 $obj=new Something(6);
```

构造函数会自动地把6赋值给数据变量x。构造函数和方法都是普通的PHP函数，所以你可以使用

缺省参数。

function Something($x="3",$y="5")

接着：

$obj=new Something(); // x=3 and y=5

$obj=new Something(8); // x=8 and y=5

$obj=new Something(8,9); // x=8 and y=9

缺省参数使用C++的方式，所以你不能忽略Y的值，而给X一个缺省参数，参数是从左到右赋值的，如果传入的参数少于要求的参数时，其作的将使用缺省参数。

当一个派生类的对象被创建时，只有它的构造函数被调用，父类的构造函数没被调用，如果你想调用基类的构造函数，你必须要在派生类的构造函数中显示调用。可以这样做是因为在派生类中所有父类的方法都是可用的。

```
<?php
 function Another() {
 $this->y=5;
 $this->Something();
 //显示调用基类构造函数
}
```

OOP的一个很好的机制是使用抽象类。抽象类是不能实例化，只能提供给派生类一个接口。设计者通常使用抽象类来强迫程序员从基类派生，这样可以确保新的类包含一些期待的功能。在PHP中没有标准的方法，但是：如果你需要这个特性，可以通过定义基类，并在它的构造函数后加上"die" 的调用，这样就可以保证基类是不可实例化的，现在在每一个方法(接口)后面加上"die" 语句，所以，如果一个程序员在派生类中没有覆盖方法，将引发一个错误。而且因为PHP 是无类型的，你可能需要确认一个对象是来自于你的基类的派生类，那么在基类中增加一个方法来实义类的身份(返回某种标识id)，并且在你接收到一个对象参数时校验这个值。当然，如果一个邪恶不好的程序员在派生类中覆盖了这个方法，这种方法就不起作用了，不过一般问题多发现在懒惰的程序员身上，而不是邪恶的程序员。

当然，能够让基类对程序员无法看到是很好的，只要将接口打印出来做他们的工作就可以了。在PHP中没有析构函数。

重载(与覆盖不同)在PHP中不支持。在OOP中，你可以重载一个方法来实现两个或重多的方法具有相同的名字，但是有不同数量或类型的参数(这要看语言)。PHP 是一种松散类型的语言，所以通过类型重载不起作用，然而通过参数的个数不同来重载也不起作用。

有时在OOP中重载构造函数非常好，这样你可以通过不同的方法创建对象(传递不同数量的参数)。在PHP中实现它的技巧是：

```
<?php
class Myclass {
```

```
function Myclass() {
$name="Myclass".func_num_args();
$this->$name();
//注意$this->name()一般是错误的，但是在这里$name是一个将被调用方法的名字
}
function Myclass1($x) {
code;
}
function Myclass2($x,$y) {
code;
}
}
```

通过在类中的额外的处理，使用这个类对用户是透明的：

```
$obj1=new Myclass('1'); //将调用Myclass1
$obj2=new Myclass('1','2'); //将调用Myclass2
```

有时这个非常好用。

多态

多态是对象的一种能力，它可以在运行时刻根据传递的对象参数，决定调用哪一个对象的方法。例如，如果你有一个figure的类，它定义了一个draw的方法。并且派生了circle和rectangle 类，在派生类中你覆盖了draw方法，你可能还有一个函数，它希望使用一个参数x，并且可以调用$x->draw() 。如果你有多态性，调用哪个draw方法就依赖于你传递给这个函数的对象类型。

多态性在象PHP这样的解释语言(想象一下一个C++编译器生成这样的代码，你应该调用哪一个方法？你也不知道你拥有的对象是什么类型的，好，这不是重点)是非常容易和自然的。所以PHP当然支持多态性。

```
<?php
function niceDrawing($x) {
//假设这是Board类的一个方法
$x->draw();
}
$obj=new Circle(3,187);
$obj2=new Rectangle(4,5);
$board->niceDrawing($obj);
//将调用Circle的draw方法
```

```
$board->niceDrawing($obj2);
//将调用Rectangle的draw方法
```

用PHP进行面向对象编程

一些"纯化论者(purists)"可能会说PHP不是一个眞正的面向对象的语言，这是事实。PHP 是一个混合型语言，你可以使用OOP，也可以使用传统的过程化编程。然而，对于大型项目，你可能想/需要在PHP 中使用纯的OOP去声明类，而且在你的项目只用对象和类。

随着项目越来越大，使用OOP可能会有帮助，OOP代码很容易维护，容易理解和重用。这些就是软件工程的基础。在基于web的项目中应用这些概念就成为将来网站成功的关键。

PHP的高级OOP技术

在看过基本的OOP概念后，我就可以向你展示更高级的技术：

序列化(Serializing)

PHP不支持永久对象，在OOP中永久对象是可以在多个应用的引用中保持状态和功能的对象，这意味着拥有将对象保存到一个文件或数据库中的能力，而且可以在以后装入对象。这就是所谓的序列化机制。PHP 拥有序列化方法，它可以通过对象进行调用，序列化方法可以返回对象的字符串表示。然而，序列化只保存了对象的成员数据而不包话方法。

在PHP4中，如果你将对象序列化到字符串$s中，然后释放对象，接着反序列化对象到$obj，你可以继续使用对象的方法！我不建议这样去做，因为(a)文档中没有保证这种行为在以后的版本中仍然可以使用。(b) 这个可能导致一种误解，在你把一个序列化后的版本保存到磁盘并退出脚本时。当以后运行这个脚本时，你不能期待着在反序列化一个对象时，对象的方法也会在那里，因为字符串表示根本就不包括方法。

总而言之，PHP 进行序列化对于保存对象的成员变量非常有用。(你也可以将相关数组和数组序列化到一个文件中)。

例子：

```
<?php
$obj=new Classfoo();
$str=serialize($obj);
//保存$str到磁盘上
//几个月以后
//从磁盘中装入str
$obj2=unserialize($str)
```

你恢复了成员数据，但是不包括方法(根据文档所说)。这导致了只能通过类似于使用$obj2->x来存取成员变量(你没有别的方法！)的唯一办法，所以不要在家里试它。

有一些办法可以解决这个问题，我把它留着，因为对这篇简洁的文章来说，他们太不好。我会很

高兴地欢迎在PHP的后续版本中有全序列化的特性。

使用类进行数据存储PHP和OOP一件非常好的事情就是，你可以很容易地定义一个类来操作某件事情，并且无论何时你想用的时候都可以调用相应的类。假设你有一个HTML表单，用户可以通过选择产品ID号来选择一个产品。在数据库中有产品的信息，你想把产品显示出来，显示它的价格等等。你拥有不同类型的产品，并且同一个动作可能对不同的产品具有不同的意思。例如，显示一个声音可能意味着播放它，但是对于其它种类的产品可能意味着显示一个存在数据库中的图片。你可以使用OOP或PHP来减少编码并提高质量：

定义一个产品的类，定义它应该有的方法(例如：显示)，然后定义对每一种类型的产品的类，从产品类派后出来(SoundItem类，ViewableItem类，等等)，覆盖在产品类中的方法，使它们按你的想法动作。

根据数据库中每一种产品的类型(type)字段给类命名，一个典型的产品表可能有(id, type, price, description, 等等字段)...然后在处理脚本中，你可以从数据库中取出type值，然后实例化一个名为type的对象：

```
<?php
$obj=new $type();
$obj->action();
```

这是PHP的一个非常好的特性，你可以不用考虑对象的类型，调用$obj的显示方法或其它的方法。使用这个技术，你不需要修改脚本去增加一个新类型的对象，只是增加一个处理它的类。

这个功能很强大，只要定义方法，而不去考虑所有对象的类型，在不同的类中按不同的方法实现它们，然后在主脚本中对任意对象使用它们，没有if...else，也不需要两个程序员，只有高兴。

现在你同意编程是容易的，维护是便宜的，可重用是真的吗？

如果你管理一组程序员，分配工作就是很简单的了，每个人可能负责一个类型的对象和处理它的类。

可以通过这个技术实现国际化，根据用户所选的语言字段应用相应的类就可以了，等等。

拷贝和克隆

当你创建一个$obj的对象时，你可以通过$obj2=$obj来拷贝对象，新的对象是$obj的一个拷贝(不是一个引用)，所以它具有$obj在当时的状态。有时候，你不想这样，你只是想生成一个象obj类一样的一个新的对象，可以通过使用new语句来调用类的构造函数。在PHP中也可以通过序列化，和一个基类来实现，但所有的其它类都要从基类派生出来。

进入危险区域

当你序列化一个对象，你会得到某种格式的字符串，如果你感兴趣，你可以调究它，其中，字符串中有类的名字(太好了！)，你可以把它取出来，象：

```
<?php
 $herring=serialize($obj);
```

```
$vec=explode(':',$herring);
$nam=str_replace("\"","",$vec[2]);
```

所以假设你创建了一个"Universe"的类，并且强制所有的类都必须从universe扩展，你可以在universe 中定义一个clone的方法，如下：

```
<?php
class Universe {
function clone() {
$herring=serialize($this);
$vec=explode(':',$herring);
$nam=str_replace("\"","",$vec[2]);
$ret=new $nam;
return $ret;
}
}
//然后
$obj=new Something();
//从Universe扩展
$other=$obj->clone();
```

你所得到的是一个新的Something类的对象，它同使用new方法，调用构造函数创建出的对象一样。我不知道这个对你是否有用，但是Universe类可以知道派生类的名字是一个好的经验。想象是唯一的限制。

php的最新版本是5.2.5(2007-11-08更新)

php的官方网站：http://www.php.net/

PHP：PHP是一种开放源代码的脚本编程语言。主要用于Web服务器的服务器端应用程序，用于动态网页设计，是一种嵌入HTML页面中的脚本语言。

Python

Python(发音:['paiθ(?)n; (US) 'paiθ?n])，是一种面向对象的解释性的计算机程序设计语言，也是一种功能强大而完善的通用型语言，已经具有十多年的发展历史，成熟且稳定。Python 具有脚本语言中最丰富和强大的类库，足以支持绝大多数日常应用。

这种语言具有非常简捷而清晰的语法特点，适合完成各种高层任务，几乎可以在所有的操作系统中运行。

目前，基于这种语言的相关技术正在飞速的发展，用户数量急剧扩大，相关的资源非常多。

Python的Hello World程序

下面是一个在标准输出设备上输出Hello World的简单程序，这种程序通常作为开始学习编程语言时的

第一个程序：

```
#!/usr/bin/env python
print "Hello, world!"
```

Python的历史

Python的创始人为Guido van Rossum。1989年圣诞节期间，在阿姆斯特丹，Guido为了打发圣诞节的无趣，决心开发一个新的脚本解释程序，做为 ABC 语言的一种继承。之所以选中 Python（大蟒蛇的意思）作为程序的名字，是因为他是一个Monty Python的飞行马戏团的爱好者。

ABC是由Guido参加设计的一种教学语言。就Guido本人看来，ABC 这种语言非常优美和强大，是专门为非专业程序员设计的。但是ABC语言并没有成功，究其原因，Guido 认为是非开放造成的。Guido 决心在 Python 中避免这一错误（的确如此，Python 与其它的语言如C、C++和Java结合的非常好）。同时，他还想实现在 ABC 中闪现过但未曾实现的东西。

就这样，Python在Guido手中诞生了。实际上，第一个实现是在Mac机上。可以说，Python是从ABC发展起来，主要受到了Modula-3（另一种相当优美且强大的语言，为小型团体所设计的）的影响。并且结合了Unix shell和C的习惯。

Python在编程语言中的定位

虽然 Python 可能被粗略地分类为"脚本语言（scripting language）", 实际上一些大规模软件开发计划例如 Zope， Mnet 及 BitTorrent. Google也广泛地使用它。 Python 的支持者较喜欢称它为一种高阶动态编程语言 , 原因是"脚本语言" 泛指单用作简单编程任务如 shell scripts ，而Python不能与JavaScript等只能处理简单任务的编程语言相提并论。

Python的特色

可扩充性可说是Python作为一种编程语言的特色。新的内置模块（module）可以用C 或 C++写成。而我们也可为现成的模块加上Python的接口。Python可以使用户避免过分的语法的羁绊而将精力主要集中到所要实现的程序任务上。

Python也被称为是一门清晰的语言。因为它的作者在设计它的时候，总的指导思想是，对于一个特定的问题，只要有一种最好的方法来解决就好了。这在由Tim Peters写的python格言（称为The Zen of Python）里面表述为：

There should be one-- and preferably only one --obvious way to do it.

有意思的是，这正好和Perl语言（另一种功能类似的高级动态语言）的中心思想TMTOWTDI（There's More Than One Way To Do It）完全相反。这似乎是人们常把Perl和Python互相比较的重要原因。

Python语言是一种清晰的语言的另一个意思是，它的作者有意的设计限制性很强的语法，使得不好的编程习惯（例如if语句的下一行不向右缩进）都不能通过编译。这样有意的强制程序员养成良好的编程习惯。其中很重要的一项就是Python的缩进规则。

例如if语句：

```
if age<21:
```

```
print "You cannot buy wine!\n"
print "But you can buy chewing gum.\n"
print "this is outside if\n"
```

一个和其他大多数语言（如C）的区别就是，一个模块的界限，完全是由每行的首字符在这一行的位置来决定的（而C语言是用一对花括号{}来明确的定出模块的边界的，与字符的位置毫无关系）。这一点曾经引起过争议。因为自从C这类的语言诞生后，语言的语法含义与字符的排列方式分离开来，曾经被认为是一种程序语言的进步。不过不可否认的是，通过强制程序员们缩进（包括if，for和函数定义等所有需要使用模块的地方），Python确实使得程序更加清晰和美观。

另外Python在其他部分的设计上也坚持了清晰划一的风格，这使得Python称为一门易读性、易维护性好，并且被大量用户所欢迎的、用途广泛的语言。

Python的局限

虽然Python是一个非常成功的语言，但是也有必要明白它的局限性。

1. 运行效率低下

目前为止，Python可以说是所有主流脚本语言中速度最慢的。这与其脚本引擎的设计思路有关。如果你的应用对于速度有着较高的要求，就要考虑Python是否能满足需要。不过这一点可以通过使用C编写关键模块，然后由Python调用的方式加以部分解决。

2. 多线程支持欠佳

Python支持多线程，但是其运行效率也不高。

3. 独特的语法

这也许不应该被称为局限，但是它用缩进来区分语句关系的方式还是给很多初学者带来了困惑。即便是很有经验的Python程序员，也可能陷入陷阱当中。最常见的情况是tab和空格的混用会导致错误，而这是用肉眼无法分别的。

4. 无类型

作为一种动态语言，随时随地创建和使用变量是Python给我们带来的巨大的便利。但是它也会使得程序不严谨，某些错误只有在运行中才可能出现。所以，使用Python编程的时候，要对类型做到心里有数。这也使得Python的IDE工具无法提供便利的自动完成等功能。

Python的前景

Python在编程领域的占有率一直处于稳步上升之中，根据最新的数据，Python排名第七。前六名分别是Java,

C,VB,C++,PHP和Perl. 作为一个很年轻的语言，Python的位置已经相当令人振奋了。随着微软将Python纳入

.Net 平台，相信Python的将来会更加强劲发展。Python 很可能会成为.Net平台快速开发的主流语言。欲了解这方面情况，请参考Iron Python的相关信息.

著名的搜索引擎 Google 也大量使用Python。

Python的应用
Zope-应用服务器
Plone-内容管理系统
Django-鼓励快速开发的web framework
Twisted - Python Network Application Framework Python的网络应用程序框架
TurboGears - 另一个Web应用快速开发框架
Bit Torrent - 著名的BT下载工具

2006年的Google编程大赛已经将Python作为参赛语言之一

Ruby

Ruby是面向对象的编程语言，她追求的是“简便快捷的面向对象编程”。Ruby是解释型语言，因此不需编译即可快捷地编程。同时Ruby具有类似Perl的强大的文本
处理功能，她可并不只是个玩具，您可以用她来进行实用的编程。此外，您还可以很方便地使用C语言来扩展Ruby的功能，因此可以把她当作各种库的前端来使用。
若您曾经“想要一种简单的面向对象的语言”，或者认为“Perl的功能虽然好用，但它的语法眞让人受不了”，又或者觉得“lisp系列语言的思想不错，但到处都
是括号眞让人讨厌，最起码算式应该按照通常的样式书写”。那么，Ruby或许能让您满意。
归纳以来，Ruby有以下优点。
解释器

Ruby是解释型语言，其程序无需编译即可轻松执行。

变量无类型

Ruby的变量没有类型，因此不必为静态的类型匹配而烦恼。相应地，错误检查功能也变弱了。

不需要变量声明

所有变量均无需声明即可立即使用。另外，从变量名即可判断出是何种变量（局部变量，全局变量，实例变量）。

语法简单

语法比较简单，类似Algol系语法。

不需要内存管理

具有垃圾回收（Garbage Collect，GC）功能，能自动回收不再使用的对象。

一切都是对象

Ruby从一开始就被设计成纯粹的面向对象语言，因此以整数等基本数据类型为首的所有东西都是对象，它们都有发送信息的统一接口。

类，继承，方法

Ruby当然具有面向对象语言的基本功能。

特殊方法

可向某对象添加方法。例如，可以把GUI按钮被按下时的动作作为方法记述下来，还可以用它来进行原型库（prototypebase）的面向对象编程（有人这么干吧）。

用模块进行混合插入（Mixin）

Ruby故意舍弃了多重继承，但拥有混合插入功能。使用模块来超越类的界限来共享数据和方法等。

迭代器

该功能可以将循环抽象化。

闭包

可以将某过程片段对象化。对象化后的该过程片段就称作闭包。

功能强大的字符串操作／正则表达式

以Perl为样板创造出了功能强大的字符串操作和正则表达式检索功能。

拥有超长整数

添加超长整数功能后，可以计算非常大的整数。例如计算400的阶乘也轻而易举。

具有错误处理功能

错误处理功能可以使您编写代码处理出错情况。

可以直接访问OS

Ruby可以使用（UNIX的）绝大部分的系统调用。单独使用Ruby也可以进行系统编程。

动态加载

若OS支持的话，可以在运行时读入对象文件。

但Ruby也有下列缺点。

Ruby On Rails，优点是不像Struts那样需要大量的配置文件，一切都采取默认的配置，包括访问路径，uri等，而这也是它的缺点，不能灵活的配置。

见笑，小弟看了几天，一点见解。

VBScript

VBScript是Visual Basic Script的简称，即 Visual Basic 描述语言，有时也被缩写为VBS。
VBScript是微软开发的一种脚本语言，可以看作是VB语言的简化版，与VBA的关系也非常密切。它具有原语言容易学习的特性。目前这种语言广泛应用于网页和ASP程序制作，同时还可以直接作为一个可执行程序。用于调试简单的VB语句非常方便。
由于VBScript可以通过Windows脚本宿主调用COM，因而可以使用Windows操作系统中可以被使用的程序库，比如它可以使用Microsoft Office的库，尤其是使用Microsoft Access和Microsoft SQL Server的程序库，当然它也可以使用其它程序和操作系统本身的库。
在实践中VBScript一般被用在以下三个方面：
Windows操作系统

VBScript可以被用来自动地完成重复性的Windows操作系统任务。在Windows操作系统中，VBScript可以在Windows Script Host的范围内运行。Windows操作系统可以自动辨认和执行*.VBS和*.WSF两种文件格式，此外Internet Explorer可以执行HTA和CHM文件格式。VBS和WSF文件完全是文字式的，它们只能通过少数几种对话窗口与用户通讯。HTA和CHM文件使用HTML格式，它们的程序码可以象HTML一样被编辑和检查。在WSF、HTA和CHM文件中VBScript和JavaScript的程序码可以任意混合。HTA文件实际上是加有VBS、JavaScript成分的HTML文件。CHM文件是一种在线帮助，用户可以使用专门的编辑程序将HTML程序编辑为CHM。

网页浏览器（客户方的VBS）

网页中的VBS可以用来指挥客户方的网页浏览器（浏览器执行VBS程序）。VBS与JavaScript在这一方面是竞争者，它们可以用来实现动态HTML，甚至可以将整个程序结合到网页中来。

至今为止VBS在客户方面未能占优势，因为它只获得因为它只获得Microsoft Internet Explorer的支持（Mozilla Suite可以通过装置一个外挂来支持VBS）。而JavaScript则受到所有网页浏览器的支持。

在Internet Explorer中VBS和JavaScript使用同样的权限，它们只能有限地使用Windows操作系统中的对象。

网页服务器（服务器方面的VBS）

在网页服务器方面VBS是微软的Active Server Pages的一部分，它与JavaServer Pages和PHP是竞争对手。在这里VBS的程序码直接嵌入到HTML页内，这样的网页以ASP结尾。网页服务器Internet信息服务执行ASP页内的程序部分并将其结果转化为HTML传递给网页浏览器供用户使用。这样服务器可以进行数据库闻讯并将其结果放到HTML网页中。

VBScript主要的优点有：

由于VBScript由网页浏览器解释执行，不需要增大服务器的负担。

易学。

在所有2000 / 98SE以后的Windows版本都可直接使用。

可以使用其它程序和可使用的物件（尤其Microsoft Office）。

缺点有：

现在VBS无法作为电子邮件的附件了。Microsoft Outlook拒绝接受VBS为附件，收信人无法直接使用VBS附件。

VBS的各种编辑程器不受欢迎。

操作系统没有任何特别的保护设施。VBS程序与其它JS、EXE、BAT或CMD程序一样对待。操作系统没有监察恶意功能的能力。

会员留言

李乐

2015-3-18 13:52:00

不懂，深奥的编程，要活到老，学到老

zu

2015-3-4 11:42:00

njvghlkqagfcqwbfhqw8ueg

1 共1页 到第 页 确定

发表留言

您的称呼 联系电话 匿名

为维护健康文明的社区氛围,请不要发表具有谩骂,诽谤,广告,宣传等内容的言论。

输入验证码 F2RX 看不清楚，换一张 提交留言