四种语言Pascal, C, Lisp, Ada的比较

Morteza Marzjarani

1.程序设计语言

引 言

在所有的发明中计算机是最有用的工具，计算机与用户之间的通信渠道是借助“语言”，也就是，（高级）语言是用户和机器之间的翻译。

目前，正在使用着好些种语言。为什么我们需要这许多不同的语言呢？这是个未解决的问题，可能有多种答案。本文试图比较四种语言——PASCAL、C、LISP和Ada——的设计原理、特色和应用，以此来部分地回答这个问题。更明确地说，分别就以下各种特色，诸如语言的设计、语言所接受的数据类型、封装性、类型检查、参数传递、块结构、标识符的引用环境、以及输入/输出等特色进行讨论并加以比较。同时还讨论应用特点，最后对所论结果作出总结。

参数传递

参数传递是显式地传递参数和结果的过程。

Pascal，PASCAL为程序员提供引用调用和传值调用。引用调用是通过形参表中的Var声明获得的。

C，一般情况下，在C中对任何函数的调用都是传值调用。引用调用只在通过指针传递地址的意义下获得。

LISP，在LISP中，函数调用中的实在参数总是表示成表结构的表达式。LISP中的参数传递是传值。LISP有几种传递方法类似于ALGOL中的按名传递。

Ada，在Ada中，通过三种模式处理参数传递，

In，传递值。

Out，传递结果。

In out，传送引用或结果值(取决于实现)，形参和实参的匹配有三种方法。

1.位置标符，第一个实参对应第一个形参。

2.名字相符：每个实参表达式可前缀上〈形参〉⇒。

3.缺省值：如果形参是一个in参数，实参就可以省略。

块结构

在块结构语言中，子程序或程序被组织成嵌套的块（分程序）的集合。按照这个定义，PASCAL、Ada是块结构语言，C和LISP不是。C不是块结构语言，因为函数可以在彼此不相关的块中定义，但变量的定义却按块结构形式。在LISP中，不提供块结构或其它的复合结构。不同函数间的相互作用仅当在调用执行期间发生。

数据类型

数据类型是一类数据对象连同为创建和操纵这些数据对象的操作集合。

Pascal，PASCAL提供一组基本数据类型以及可补充声明程序员定义类型的方法。PASCAL有四种基本数据类型：整型、实型、字符型和布尔型。它还为程序员提供定义新标量类型的机会。此外，还允许定义子界类型。

C，C的数据类型有许多限制域。它的基本数据类型是：整型，（包括长整型、短整型、无符号整型），浮点型和字符型，还可附加机器定义的类型。双精类型可以用来增加精度。可选的typedef（类型定义）看起来象PASCAL的类型声明，但这种选择不提供对新类型的定义，因而意义不同。它只不过是允许程序员建立一个基本C数据类型的同义词。和PASCAL一样，C提供指针设施。

LISP，LISP只有两种数据类型，ATOM（原子）和LIST（表）。

Ada，Ada对算法和数据描述都有很强的设施。

然而，其它语言的算法描述能力也很强。因此Ada与其它语言的最大差别在于数据描述领域。Ada的数据类型可总结如下：

1.预定义类型：有一个预定义值集合和可用的操作集合。

2.派生类型：根据以前定义过的类型来定义派生类型。

3.约束类型：约束一个类型的值集合但并不影响可用的操作集合。

4.匿名类型定义：根据属性定义类型而不是根据以前定义过的类型定义它。类型的等价性决定了匿名类型。

5.子类型：是对父类型的约束，但认为是和父类型一样的类型。

6.属性查询：程序员可用以判定类型的属性。

7.类型的分类：

A.标量类型

Ⅰ.离散类型

a.整型（INTEGER）

b.枚举型

1.BOOLEAN，CHARACTER

2.程序员定义的枚举型。

Ⅱ.实型（FLOAT）

a.浮点型

b.定点型

B.结构类型

Ⅰ.数组和向量

Ⅱ记录

C.访问类型（指针）

封　装

封装是向程序员隐藏信息的过程，例如：PASCAL的类型、子程序和整数的存储表示法。

在LISP中，当引用一个具有全程值的原子时，不查找结合表（A表）而直接检索该值。只要使用原子，在结合表上的任何聚束（binding）都是隐藏的。

在C中，一个函数中声明的变量（如主函数main）不能被其它函数访问（不象PASCAL，主程序中声明的变量可被所有的子程序访问）。要生成一个全程变量，得用external（外部）语句（类似于FORTRAN的COMMON语句）。可是，在C中，一个变量若是全程的仅当它定义于某个函数体之外时。因此，与函数push（压入）、pop（弹出）、clear（清栈）共享的栈和栈指针是在这三个函数之外定义的。但主函数本身并不涉及栈及栈指针。也就是说，它的表示是隐藏的，测试栈顶元素的代码是：

```
push（pop（））；
```

Ada定义和封装新数据类型的设施要比PASCAL、LISP、C、……等语言好得多。子程序和任务的体对用户是隐藏的。私有类型可用于程序包中对用户隐藏局部声明。在PASCAL中，堆栈的任何元素都可以访问（尽管我们只对顶端元素操作）。但在Ada中，用户除了通过push和pop访问栈顶元素外，不能访问栈的其它元素。

引用环境

引用环境是一个操作，这个操作为给定环境中的给定的标识符找到一个合适的结合并保留所结合的数据对象或予程序定义。

Pascal：PASCAL是块结构语言，用静态作用域规则来决定块中非局部引用的名字的含义。其引用环境可划分为：

全程环境，包括子程序、变量等的定义。任何在这个环境中没有定义且在程序中使用的名字，必须在局部环境中声明。PASCAL程序有块结构语言特点的嵌套结构。一个程序由定义主程序的外部块和表示子程序的相互嵌套定义的块组成。

局部环境：局部变量在子程序入口处声明并在出口处删除。

非局部环境：数据可通过块结构和静态作用域在子程序间共享。

C：C中的作用域规则和PASCAL类似，但要在子程序中指明引用的全程变量，有时要求在该子程序内把全程变量当作外部变量声明。若源文件中声明的全程变量和代码文件中的不一样，则需要"extern"声明，否则，是可选择的。

LISP：结合表（Association List简称A表）表示LISP的引用环境。也就是说，LISP的

引用是严格按照最新的结合规则进行的，通过从头到尾查找A表，以查得我们所要找的原子所在的项(该原子是该项的CAR，而CDR是该项的值）。LISP还提供全程公共环境。各系统的实现多少有些不一样。

Ada，Ada是强结构化语言。其引用环境类似于PASCAL。然而，程序员可用块声明建立局部引用环境。而且，程序包对静态使用嵌套程序单元提供一个方案，而数据共享是通过非局部引用达到的（这叫做公共引用环境）。

类型检查

一个存储单元被指派到一个类型则不能用于其它类型。

Pascal，PASCAL是强类型语言。每一变量在它用于程序之前必须先声明其类型，这些类型声明是在每个程序、过程、函数的部首的变量块中作出的。

C，C的类型检查较弱，在这个意义上它比较灵活。但我们在各个变量使用之前仍需声明它的类型。已有的编译程序不提供对数组、下标、参数等的类型检查。如果希望强类型检查，可以使用叫做Lint版本的C编译程序。

LISP，只提供运行时的类型检查。

Ada，同PASCAL一样，Ada是强类型的。几乎所有的类型错误在编译期间均可查出。只有极少数情况下需要在运行时检查类型。Ada非常注意可能在子程序、任务、程序包之间的界面上出现的类型错误。在编译期间要检查所有分割的程序单元之间相互关联时可能发生的错误。当这些分割的程序单元组合到一起时，出现未查出的错误的可能性就很小了。

以下我们总结每种语言的其余特色，并与其它有此特色的语言进行比较。

LISP

1.采用显式的无用单元收集。

2.没有数据结构。

3.取消了显式顺序执行而代之以函数调用。

4.只有两个保留字（T，NIL）。

5.剑桥波兰表示法相当于PASCAL及其它语言的中缀表示法。

6.没有过程，（象C一样）只有函数，相当于PASCAL和Ada中的过程和函数。

7.在交互式环境中运行，相当于其它语言在各自的环境中运行。

8.和PASCAL及C不同，没有通常的主程序形式，但用户在终端上可以以待求值的表达式序列输入主程序。

9.函数全部以表达式形式定义。

10.链表和性态表（以链表的特例表示）组成了基本的数据结构。

11.不象PASCAL、C和Ada，没有声明。

12.解释执行而不是编译执行。

13.包含基本的算术元语PLUS（加）、DEFERENCE（减）、TIMES（乘）、DIVIDE（除）。

14.有推理运算符ZEROP（为零），GREATERP（大于），LESSP（小于）用于数的比较。

15.不象PASCAL和Ada，没有布尔数据类型。

16.提供与PASCAL类似的AND、OR、NOT布尔运算符。

17.LISP不象其它语言那样把直接赋值作为语言规则的核心，许多LISP程序完全没有赋值操作，使用递归和参数传递可间接地达到同样的效果。

18.大多数LISP的输出是自动的。每次函数调用系统都打印出函数名、它的参数和函数的返回值。

19.没有特定的格式输出（即自由格式输出），不象PASCAL那样用户可按预想的格式输出值。

20.除了函数子程序外，无抽象机制。

21.有三类函数：解释性的、编译性的、宏。

22.顺序控制：

借助递归的循环。

借助COND的选择。

注意：PROG特色允许直接为循环编码，但这种编码循环有时变得很复杂。

23.与大多数其它语言不同，LISP函数可以建立和返回任何数据对象。

24.每个表达式用多重括号括起，因此，有时其结果相当的难读、难排错。这个问题若用"("和")"可部分地解决，但是用剑桥波兰表示法使LISP的可读性比PASCAL这类的语言更差。

25.LISP运行时的结构的最有意义的特色是所需要的相似的个数。

26.在许多方面，同PASCAL、C和Ada相比LISP是最不合俗的。

27.在LISP、PASCAL、C和Ada中，LISP的可读性最低并且是唯一的函数程序设计语言。

28.BREAK选择可帮助程序员测试LISP程序。当运行时到达这个选择时，程序被中断并将控制交给程序员。程序员就可以检查值、修改它们并重新启动执行。

29.赋值：例如，SETQ是把值分派到一个存储单元，并返回该值。

30.应用：人工智能。

C语言

1.和LISP一样，没有过程只有函数，相当于PASCAL和Ada的过程和函数。

2.switch（开关）语句相当于PASCAL中的分情形语句（C不同于PASCAL的是，当有一个选择被执行时，其控制还要通过块的其余部分，这个问题可用break语句解决）。

3.有寄存器变量。

4.可以访问到某个结构的内部。

5.语言无输入/输出。

6.C不如PASCAL好读。

7.C比PASCAL灵活，PASCAL比C可靠。

8.C程序员比用PASCAL作程序设计时要做更多的人工检查错误的工作。

9.C允许视见（viewing）表示法（用同一个存储单元表示不同的数据类型）。这在PASCAL中是通过记录实现的。

10.主程序可置于任何地方（不同于PASCAL）。

11.C编译程序不象PASCAL那样计较函数调用时的自变量的个数。

12.若有一个被声明其大小为5的数组，如果试图读第6个元素将不会产生错误信息。这是因为它不会自动地检查数组的下标，这点和PASCAL不同。

13.C和PASCAL有几个控制语句是共同的，例如，if…then 等其处理方法类似，尽管语法略有不同。

14.C和PASCAL都是自由格式语言，两个语言的作用域规则类似。

Ada

1.可靠性好。

2.可维护性好。

3.注重效率。

4.把程序设计看作是人的活动。

5.一个或多个独立的程序组成应用系统。

6.有任务（并行执行的）机制。

7.有异常处理程序。

8.重载（为运算符或已经使用过的子程序名字赋予其它意义）。

9."内定的（built-in）"类型这个概念与其它的语言有些不同。

10.常量和变量的整体处理比PASCAL允许分开的有限种常量定义要雅緻得多。Ada结构使实现能保证所声明的常量不会被修改。

11.不同于PASCAL，枚举型不能直接在变量声明中定义，必须在单独的类型定义中定义。

12.不同于PASCAL，相同的文字名可用于几个枚举类型中（即前面所说的重载）。

13.布尔型操作：AND、OR、XOR、AND THEN、OR ELSE 较特别。

14.不同于PASCAL，对于一维或多维的数组类型的定义，可以不给出下标的范围（即可以使用可变大小的数组）。当许多变

量被声明为这种类型时，其数据对象具体使用的维数可随后给出。

15.不同于PASCAL，变体记录必须有一标志域，称为判别式，它显式地表示在运行时当前存在的是哪个变体。

16.不同于PASCAL定义一个记录数据对象，必须先用一个类型定义来定义记录的结构，再用类型的名字给出变量声明。同样地，如果一个记录的成分本身又是一个记录，那么那个记录必须单有一个类型定义来定义它，第二个记录定义不能嵌套在那个大记录里。

17.访问类型作指针用。

18.同等优先级的操作是从左到右结合。

19.for循环的一个重要特色是子句中给出的<变量-名>不用象PASCAL那样作为通常的局部变量声明，而是在它出现的for子句中隐式地声明，且只能在循环内部引用它。一旦从循环出口，这个变量消失而且再也不能引用。这种结构允许Ada编译程序引入大量重要的循环优化技术。

20.子程序和程序包都有两部分：规格说明和体。

21.断言可用于程序的测试和排错。

22.Ada（以及在某种程度上PASCAL）编译的顺序，例如，子程序的编译必须在它的规格说明和所用到的分散各处的数据的编译之后。

23.Ada同PASCAL一样提供函数和子程序，此外，Ada还提供程序包。

24.应用：Ada被设法用作系统程序设计语言，特别是在异步事件的实时控制领域是必须的。但是，很多人感觉这个语言纯粹是通用语言，而且最终可能仅适合于"成组地嵌入(embedded in banks)"的程序设计系统，就象那些嵌入到军用运载武器的系统那样。

PASCAL

PASCAL语言的大部分特色已经讨论了，以下是它的其余特色：

1.有可供选择的大量的不同的数据类型（类似Ada）：整型、实型、字符型、枚举型、布尔型、数组、记录、顺序文件和有限的集合形式。也提供了在运行时建立新的链接的数据对象的指针和设施。

2.类型定义仅为真正抽象数据类型设施提了个头，但不提供对新类型的数据对象进行操作的子程序集合的成组和封装的类型定义。

3.顺序控制结构是简单的，很少使用语句标号和GOTO。

4.子程序可作为参数传递，但语句标号不行。

5.类似于Ada，在翻译过程中PASCAL提供静态类型检查，因此只需要很少的动态检查。

6.在运行期间，中央栈用于具有堆存储区的子程序活动记录；用于为指针变量直接建立的数据对象。

7.存储管理和输入/出到顺序文件只需要少量运行时刻的支持例程。

8.变体记录允许使用不同类型的记录。

9.表处理允许块中动态分配存储器，然后将相似的队、栈、树和其它动态数据结构链在一起。

10.应用：主要是教学。

总结

正如前边提到的，选择一个语言有许多因素要考虑，其中最主要的是"应用"特征，从这个意义上说，PASCAL是"教学"语言，C是"操作系统"语言，LISP是"人工智能"语言，Ada是"数据描述"语言。

很明显，这种说法存在重叠和折衷，例如，C语言同样可用于数据、文本处理和数据库管理。因此在选择一个语言时，人们应该学习这种语言的所有优点和缺点，并和其它的语言进行比较，以便作出正确的选择。

〈参考文献略〉

〔童中凡，何宝诸 译自Journal of Pascal, Ada, & Modula-2, 1988年1月/2月号，声 钟校〕