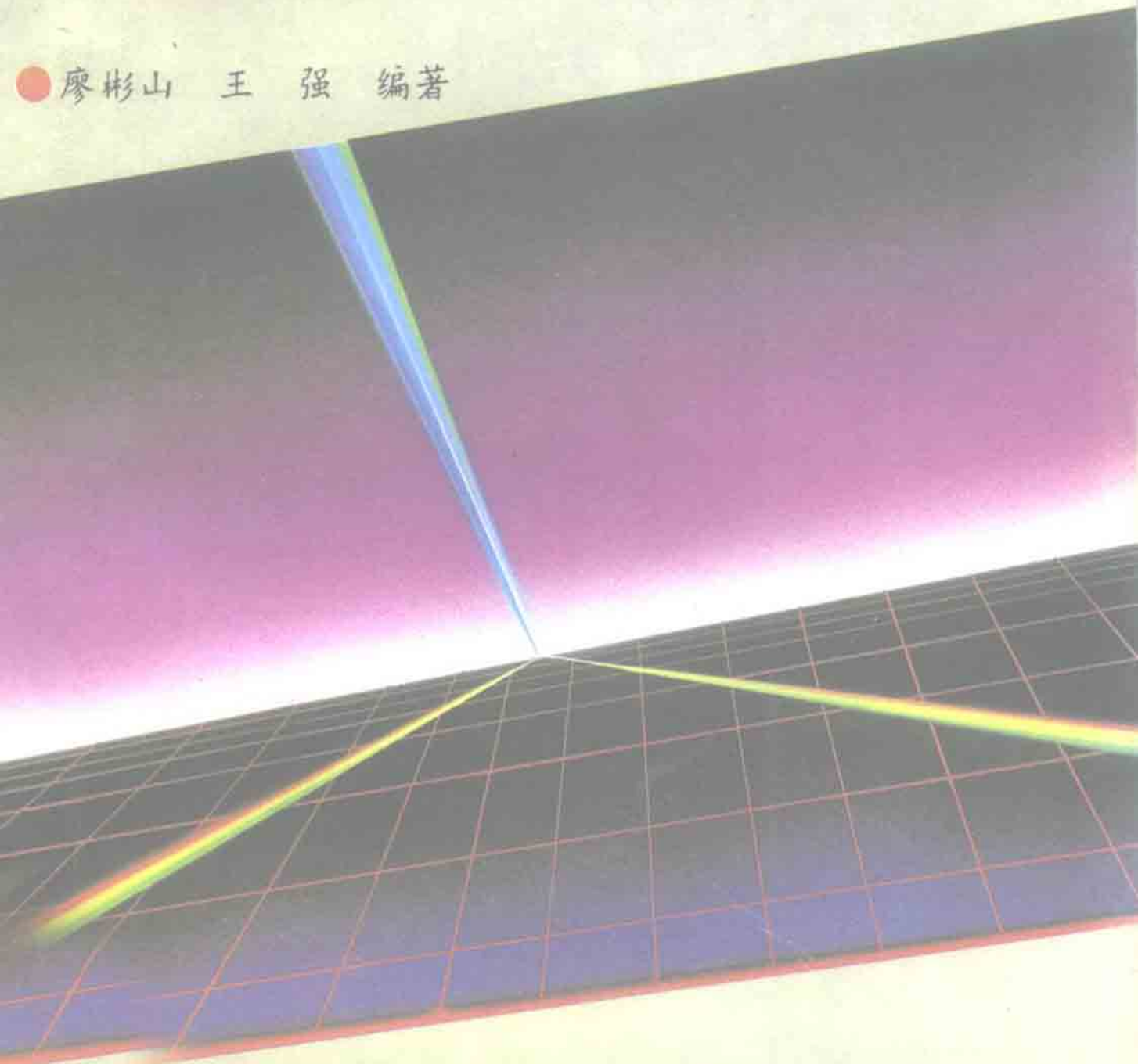
Turbo Pascal 6.0
编程技巧与实例

廖彬山　王　强　编著



電子工業出版社

Turbo Pascal 6.0

编程技巧与实例

电子工业出版社

(京)新登字 055 号

内容提要

本书对运用 Turbo Pascal 进行程序设计作了详细介绍。全书共包括九章，主要内容包括 Turbo Pascal 程序设计基础，覆盖与内存管理技术，内存驻留程序设计与程序段前缀，面向对象程序设计技术，图形程序设计与动画技术，图形输出技术。此外作者还精心设计了大量的编程实例。

本书可适用于不同层次的软件开发人员、大专院校师生、培训班师生阅读。

Turbo Pascal 6.0 编程技巧与实例

廖彬山　王强　编著

庞春立　审校

责任编辑　张丽华

*

电子工业出版社出版

电子工业出版社发行　　各地新华书店经销

北京市顺义县李史山印刷厂印刷

*

开本：787×1092 毫米　1/16　印张：21.50　字数：529 千字

1994 年 4 月第 1 版　　1994 年 4 月第 1 次印刷

印数：1—10100 册　定价：17.50 元

ISBN　7-5053-2229-X/TP・599

前　言

Turbo Pascal 是美国 Borland 公司推出的产品，它具有编译速度快，运行效率高，并可在各种不同的个人计算机上运行的特点。主要有：与国际标准兼容、用户界面好、编程效率高、查错功能强、单元可分别独立编译、支持面向对象程序设计，并提供一整套相应的开发工具。本书对运用 Turbo Pascal 进行程序设计作了详细介绍。为使读者加深对每一部分的理解，作者还精心设计了大量的编程实例，并将这些实例认真调试通过后，整理在软盘中。

本书不同于一般的使用手册，注重编程技巧与实例，集知识性与技术性于一体，是一本不可多得的参考书。

本书可适用于不同层次的软件开发人员、大专院校师生、培训班师生阅读。

本书由北京航空航天大学计算系廖彬山和王强主编，参加本书编写工作的还有：王小林、章为民、高志强、刘岩葵、林海兵、林容生、朱海东、丁建民、刘连生、王伟、连红兵、解晓东。本书的录入排版工作由章群山、苏海东和刘威负责，他们为本书的出版付出了辛勤的劳动，在此对他们表示由衷的感谢。

由于时间仓促，不当之处在所难免，敬请读者批评指正。

编著者

1993 年 7 月

目 录

第一章 Turbo Pascal 程序设计基础

1.1 程序及单元

Turbo Pascal 是一种深受广大用户欢迎的程序设计语言，它除保留了标准 Pascal 语言鲜明的结构特性外，还具有当代程序设计语言的易读、易写、易移植特点，同时拥有功能齐全、效率高的集成开发环境，并在 5.5 以上版本中进行面向对象的扩充，支持面向对象的程序设计。

1.1.1 Turbo Pascal 程序的结构及语法

Turbo Pascal 的程序由程序头、说明部分、主程序体和结束符四个部分组成。Turbo Pascal 程序除了继承标准 Pascal 原有的说明子句外，又增加了一条可选的单元引用说明子句 Uses 子句。下面是 Turbo Pascal 程序的一般结构：

```
PROGRAM <程序名> [<参数表>];
    [Uses 子句]
    [Label 子句]
    [Const 子句]
    [Type 子句]
    [Var 子句]
    [Procedure 定义说明]
    [Function 定义说明子句]
Begin
    [主程序体语句]
End.
```

程序头由关键字 Program 引导，并在程序头中指出本段程序的程序名和参数。

单元引用子句（uses 子句）说明在本段程序中被直接引用和间接引用的单元。程序单元（Unit）是 Turbo Pascal 程序设计的基础，是常量、类型、变量、过程和函数的集合，是一个相对独立的程序段。

在 Turbo Pascal 程序中，程序自动使用 System 单元，System 单元实现了文件 I/O，字符串处理，浮点运算，动态内存分配等低层的例程。除了 System 以外的标准单元（如 Crt，Dos，Printer，Graph 等），Turbo Pascal 程序不能直接引用，因此必须用 Uses 子句说明它们，如：

```
Uses Crt, Dos, Graph;
```

在这条说明语句之后，程序就可以调用单元 Crt、Dos 和 Graph 中提供的常量、变量、类型、过程和函数了。此外，Uses 子句还用于说明被调用的用户自定义单元。由于 uses 子句中单元出现的顺序决定了单元被初始化的顺序，因此，用户单元一般应当写在所有标准单元之后，如：

```
Uses Crt, Dos, UserUnit;
```

其中 UserUnit 假定为用户自定义单元的名称。

Turbo Pascal 程序的其它部分与标准 Pascal 程序相同，下面是 Turbo Pascal 程序的语法框图（图 1.1）：

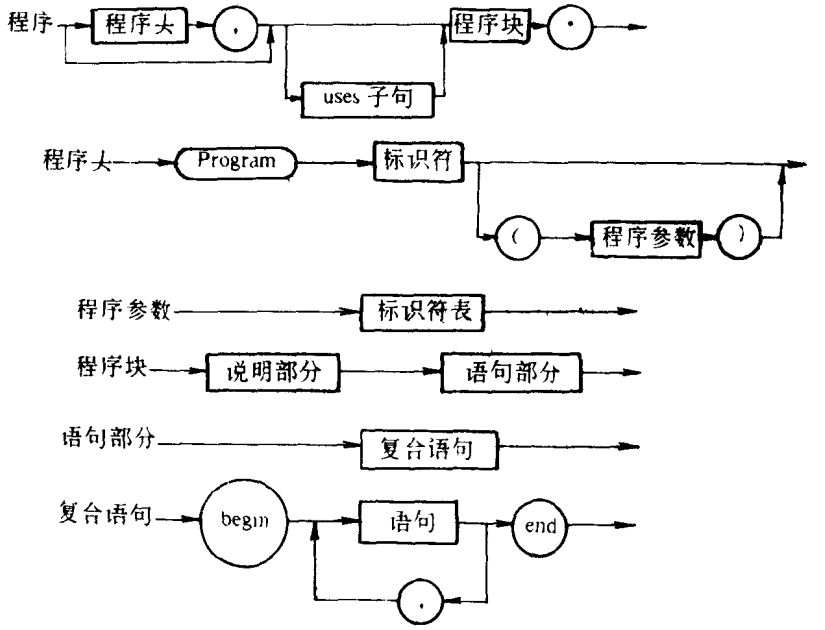


图 1.1 程序语法框图

Uses 子句说明所有被程序直接和间接引用的单元，其语法如图 1.2 所示。

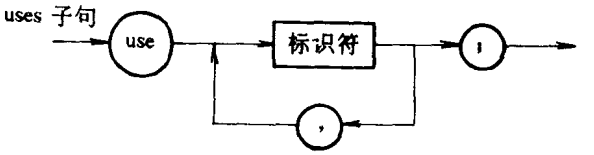


图 1.2 Uses 子句语法框图

Turbo Pascal 程序的说明部分采用过程说明的形式，其语法如图 1.3 所示。

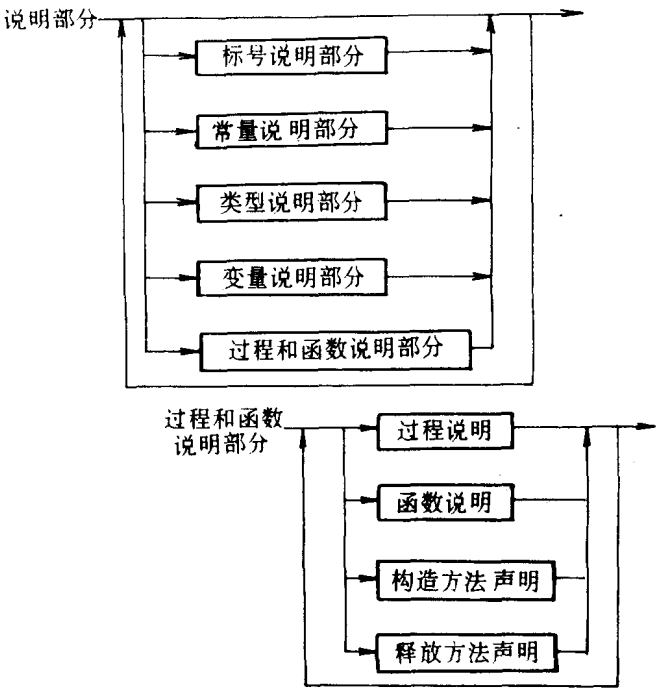


图 1.3 说明部分语法框图

1.1.2 Turbo Pascal 程序单元的结构与语法

单元（Unit）是 Turbo Pascal 模块编程的基础，它用于创建能被许多程序引用而不需要源程序的库，把大程序分成逻辑相关的模块。

Turbo Pascal 提供了大量预定义的常量、数据类型、变量、过程和函数，让用户程序进行存取。这些定义中，一些是 Turbo Pascal 所独有的，另外一些则是 IBM PC 及其兼容机或 MS-DOS 所独有的。一般单个程序不可能用到所有的说明，所以可以把相关的说明分成组，称为单元，而用户程序只使用所需要的单元即可。

单元是常量、数据类型、变量、过程和函数的集合，每个单元像一个独立的 Pascal 程序。有些初始化代码和程序体，程序可以在首部用 Uses 子句调用它。

单元是具有一定功能的一组说明的集合体，例如 CRT 单元包含了与 PC 屏幕子程序有关的子程序和其它说明。

Turbo Pascal 通常提供 8 个标准单元，其中有 6 个单元（System，Overlay，Graph，Crt，Dos 和 Printer 单元）支持通常的 Turbo Pascal 程序，它们存放在 Turbo.TPL 和 Graph.TPL 中，而另外的两个单元（Turbo3 和 Graph3）用于支持 Turbo Pascal 3.0 版的程序和数据，与 6.0 版的兼容。

1. 单元的结构

单元的结构与程序相似，但也有一些重要的区别，下面是单元的结构形式：

Unit<单元名称>;
Interface
[Uses<单元表>;]
[公用说明]
Implementation
[Uses<单元表>]
[私用说明]
[过程和函数的实现部分]
Begin
[初始化代码]
End

(1) 单元头

单元头和关键字以 Unit 开始，后面紧跟着的是单元名称（标识行），这与程序头类似，但没有程序参数表，而且单元的名称与单元的文件名是一致的。如果作为单元名的标识符长度超过 8 个字符，则取前 8 个字符作为单元的文件名。在这一点上与程序名是不同的，因为程序名可以与该程序的文件名不同。

(2) 接口部分

单元的接口部分用关键字 Interface 作为开始，它出现在单元名之后，关键字 Implementation 之前。单元的接口部分决定了引用该单元的其它单元和程序的可见部分，也就是说，其它单元和程序只能直接引用单元接口部分的说明。

Interface 关键字之后的 Uses 子句指明了在该单元接口部分中所使用的其它单元，即在本单元的接口部分可以直接使用这些单元中在接口部分定义的常量和数据类型。

单元接口部分的公用说明包括常量说明、数据类型说明、变量说明、过程和函数说明，并且这些说明可以以任意的顺序出现，而且关键字也可以重复出现。例如：

Type…Var…Procedure<过程名>…Const…Var

单元接口部分说明的过程和函数，可以供其他单元、程序和本单元的其它过程与函数使用，而且其它单元或程序也只能直接调用接口部分指明的过程和函数。在接口部分中出现的仅仅是过程和函数的头部，其实现部分则在 Implementation 之后，公用说明的过程和函数不需要进行超前引用说明，既不需要也不允许使用 Forward 说明。

(3) 实现部分

实现部分以关键字 Implementation 开始。在接口部分所定义的所有说明对象（常量、数据类型、变量、过程和函数），在实现部分都是可见的，可以在实现部分随意引用。不仅如此，在接口部分出现的过程和函数，在实现部分都必须有相应的过程体或函数体。

Implementation 关键字之后可以出现 Uses 子句，由它指明实现部分所用到的其它单元。除 Uses 子句外，实现部分还可以拥有自己独立的私用说明，在这里所定义的标号常量、类型、变量、过程和函数，仅供本单元中的过程和函数调用。对其它的单元和程序是不可见的，而且其它单元和程序不知道它们的存在，不能直接引用或调用它们，但是可以通过接口部分与所定义的过程和函数来间接引用私用部分的说明。

实现部分的过程可以说明为 external(外部)，但必须在单元源文件中出现一个或多个{$I

<文件名>} 指令。

在接口部分说明的过程和函数，必须在实现部分重复出现，且出现时的过程头和函数头必须与接口部分的说明相一致，或者使用省略参数表后的简写形式。

例如，在单元的接口部分定义为：

```
Procefure ISWap (Var V1, V2: Word);
Function IMax (V1, V2: Word): Word;
```

那么，在实现部分可以使用如下形式：

```
Procedure ISwap;
Var
    Temp: Word;
Begin
    Temp: =V1; V1: =V2; V2: =Temp;
End;

Function IMax (V1, V2:: Word): Word;
Begin
    If V1>V2 Then IMax: =V1
    Else IMax : =V2;
End;
```

但在实现部分所定义的私用过程和函数，必须有完整的过程和函数头。

(4) 初始化部分

单元的实现部分通常介于关键字 Implementation 和最后一个 End 之间。如果在实现部分结束之后，最后一个 End 之前增加一个关键字 Begin 也是可以的。同时，这一对 Begin 和 End 之间的复合语句（所有语句）就构成了单元的初始化部分，它与程序的程序体相似。

初始化部分用于在引用单元时对单元的数据结构进行初始化处理，例如，对变量进行赋初值，调用某些过程、函数，或者打开文件以备以后使用等。标准单元 Printer 和初始化部分为所有的打印输出打开文本文件 Lst，然后，在程序的 Write 或 Writeln 语句中直接使用 lst 进行打印输出。

单元的初始化部分仅在引用该单元的程序的程序体开始执行之前调用一次，如果程序中引用多个单元，则按照 Uses 子句中说明单元的顺序逐个调用初始化部分执行。

2. 单元的语法

单元的语法如图 1.4 所示。

图 1.4 单元语法框图

单元头给出单元的名称，其语法如图 1.5 所示。

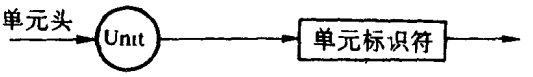


图 1.5 单元头语法框图

单元名在 Uses 子句中使用，此名必须是唯一的，且一条 Uses 子句中不能出现两个同名单元。接口部分是单元提供给其它单元和程序使用的可见说明，其语法如图 1.6 所示。

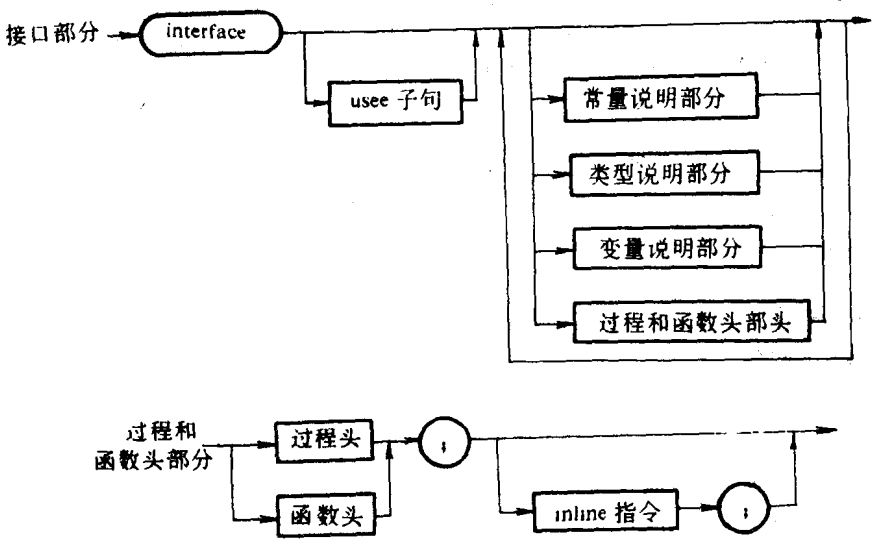


图 1.6 接口部分语法框图

除非过程和函数是 inline 型的，否则接口部分只列出过程和函数头，其程序体在实现部分。实现部分定义所有的公用过程或函数的程序体，此外还有仅供实现部分自身使用的局部标号、常量、类型、变量、过程和函数，其语法如图 1.7 所示。

单元的最后一部分是初始化部分，如果没有初始化部分，则可以省略关键字 Begin，其语法如图 1.8 所示。

1.1.3 Turbo Pascal 单元的使用

程序中使用的单元被编译后，以机器代码而不是 Pascal 源代码的形式存放，它们不是 C 语言中的 Include 文件，而是以扩展名 .TPU 的形式存放在二进制文件中。单元的接口部分用 Turbo Pascal 中使用的特殊二进制符号表的形式存储，而其它的一些标准单元则存放在 Turbo.TPL 文件中，与 Turbo Pascal 的集成环境一起自动加载到内存之中。

如前所述，程序引用单元必须使用 Uses 子句，后面紧跟着使用单元名称，例如：

```
Program MyProgram;
Uses AUnit, BUnit, OtherUnit;
```

当编译器发现 Uses 子句时，自动将 Uses 子句中每个单元的接口部分添加到符号表中，同

时将实现部分的机器码连接到程序中。

Uses 子句中单元引用说明的顺序一般是不重要的，因为编译器自己会考虑单元的连接顺序。尽管如此，我们仍然建议，在 Uses 子句中先指明标准单元（System 单元是不须要说明的，由系统自动引用），其后才是用户自己定义的单元，而且用户自定义单元中，被其它单元引用次数多的单元出现在前。

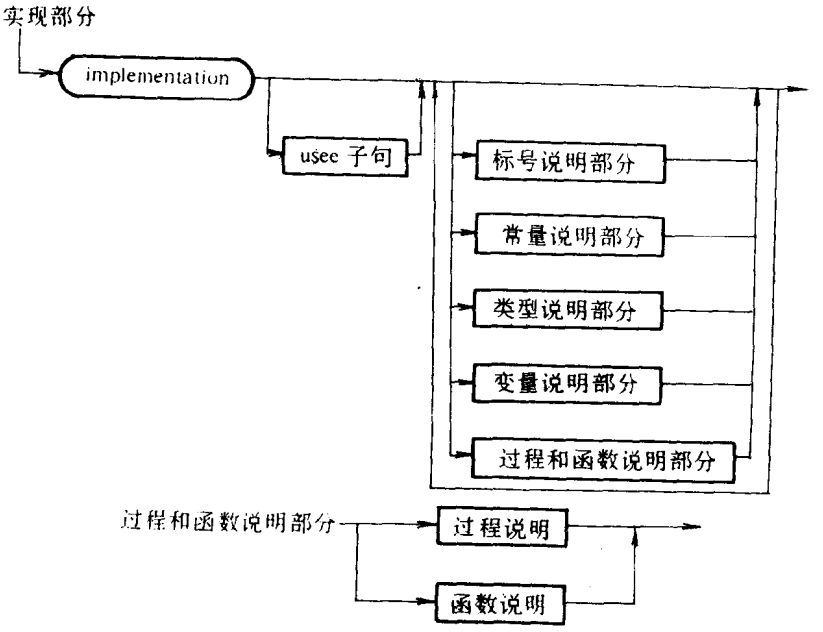


图 1.7 实现部分语法框图

如果在本段程序或者单元中，不直接调用某个单元中的说明，则在 Uses 子句中可以省略该单元的名称，例如，AUnit 单元引用 BUnit，而 MyProgram 只直接使用 AUnit，则可在 My Program 的 Uses 子句中省略 BUnit。隐藏单元 BUnit 中的说明如下：

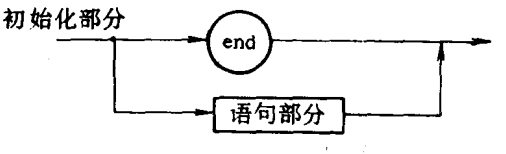


图 1.8 初始化部分语法框图

```
Unit   AUnit;
Uses     Crt,    Dos,.   BUNit;
...
End.
Program   MyProgram;
Uses   AUnit,     OtherUnit;
```

在这个例子中，单元 AUnit 可以直接引用 BUnit 单元接口部分的任何说明，而程序 MyProgram 则不能直接调用单元 BUnit 中的任何过程、函数、常量、变量、数据类型等说明。