工业和信息化普通高等教育“十二五”规划教材立项项目

21世纪高等教育计算机规划教材

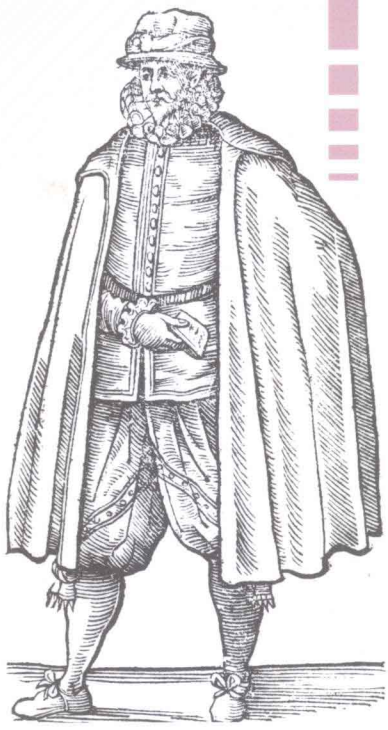
COMPUTER

数据结构（Visual Basic版）

Data Structure in Visual Basic

吴仁群 编著

- 所有算法均使用 Visual Basic 语言实现
- 内容讲述由浅入深，符合初学者计算机语言学习习惯
- 讲述知识点时，辅以图形或具体实例



人民邮电出版社
POSTS & TELECOM PRESS

工业和信息化普通高等教育“十二五”规划教材立项项目
21世纪高等教育计算机规划教材

COMPUTER

数据结构（Visual Basic版）

Data Structure in Visual Basic

吴仁群 编著

人民邮电出版社
北京

图书在版编目（CIP）数据

数据结构 : Visual Basic版 / 吴仁群编著. -- 北京 : 人民邮电出版社, 2012.8
21世纪高等教育计算机规划教材
ISBN 978-7-115-28585-0

Ⅰ. ①数… Ⅱ. ①吴… Ⅲ. ①BASIC语言－程序设计－高等学校－教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2012)第145960号

内 容 提 要

本书是针对数据结构初学者编写的基础教程，书中不仅讲解了数据结构常用的基本理论知识，而且提供了大量应用实例，以帮助初学者加强对知识的理解。全书共分 8 章，包括绪论，线性表，栈和队列，串和数组，树和二叉树，图，查找，排序等。

本书内容实用，结构清晰，实例丰富，可操作性强，可作为高等学校数据结构的教材，也可作为计算机相关专业的培训和自学教材。

21 世纪高等教育计算机规划教材

数据结构（Visual Basic 版）

◆ 编　　著　吴仁群
责任编辑　刘　博

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京昌平百善印刷厂印刷

◆ 开本：787×1092　1/16
印张：14.25　　2012 年 8 月第 1 版
字数：371 千字　　2012 年 8 月北京第 1 次印刷

ISBN 978-7-115-28585-0

定价：32.00 元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前　言

数据结构是计算机相关专业中一门重要的专业基础课程。当用计算机来解决实际问题时，就要涉及数据与数据之间关系的表示及处理，而数据与数据之间关系的表示及处理正是数据结构主要研究的对象。通过数据结构的学习，可以为后续课程尤其是软件方面的课程打下厚实的基础。因此，数据结构在计算机相关专业中具有举足轻重的作用。

作为一本数据结构的基础教材，本书具有以下特点：

（1）内容的讲述由浅入深，符合初学者的计算机语言学习习惯；

（2）在讲述每个知识点时，都辅以图形或具体实例，使读者能够从具体应用中掌握知识，能够很容易地将所学的知识应用于实践；

（3）所有算法均使用 Visual Basic 语言实现，且每章均提供上机验证程序，便于初学者学习和上机操作；

（4）每章后面附有习题，读者可通过习题巩固并掌握所学知识。

全书共有 8 章。第 1 章讲述数据、数据结构、数据类型、抽象数据类型等基本概念及算法和算法描述、算法的性能分析等有关知识。第 2 章介绍线性表的含义及 ADT 描述、线性表的两种顺序存储和链式存储、不同存储方式下基本操作的实现及应用。第 3 章介绍栈和队列的定义及 ADT 描述、栈和队列的存储结构、不同存储结构下基本操作的实现及应用。第 4 章介绍串和数组的定义及 ADT 描述，串的存储结构及应用，数组的存储方式、压缩存储及应用。第 5 章讲述树和二叉树的概念及 ADT 描述、树和二叉树的存储结构、树和二叉树及森林的遍历及应用、树和二叉树及森林的转换、哈夫曼树及应用。第 6 章讲述图的概念及 ADT 描述、图的存储结构、图的遍历、最小生成树、有限无环图及应用等。第 7 章介绍查找的基本概念、静态查找和动态查找的基本方法、哈希表的概念及查找方法等。第 8 章讲述排序的基本概念及插入排序、交换排序、选择排序、归并排序和基数排序等排序的主要方法。

本书由北京印刷学院吴仁群老师编写。在编写过程中，作者参考了本书“参考文献”所列举的图书，并得到了人民邮电出版社的大力支持，在此对“参考文献”中图书的作者及人民邮电出版社表示深深的感谢。

由于时间仓促，书中难免存在一些不足之处，敬请读者批评指正。

编　者

2012 年 1 月

目 录

第 1 章 绪论

本章学习目标

- 了解数据结构的含义及有关概念
- 了解数据结构的逻辑结构和物理结构
- 了解算法的含义、描述方法及重要特性
- 掌握估算算法的时间复杂度和空间复杂度的方法

1.1 学习数据结构的意义

数据结构是计算机科学与技术领域中广泛使用的术语。它用来反映一个数据的内部构成，即一个数据由哪些成分构成，这些成分的构成是什么样的，呈现出什么结构。数据结构作为一门学科，主要研究数据的各种逻辑结构和存储结构，以及对数据的各种操作。因此，数据结构主要有 3 个方面的内容：数据的逻辑结构，数据的物理存储结构及对数据的操作（或算法）。一般来说，算法的设计取决于数据的逻辑结构，算法的实现取决于数据的物理存储结构。

1946 年 2 月 14 日，世界上第一台计算机 ENIAC 在美国宾夕法尼亚大学诞生。在计算机发展初期，计算机主要用于处理科学和工程计算方面的数值计算问题，如求解数值积分，求解线性方程组，求解代数方程，求解微分方程等。这类问题所涉及的运算对象是简单的整型、实型或布尔类型数据，同时对象之间关系较为简单，因此程序设计者在使用计算机处理这类问题时，往往将主要精力集中于程序设计的技巧上，而无须重视数据结构。

随着计算机应用不断深入，非数值计算问题显得越来越重要。例如，人们经常要在表中查找某个对象，或者插入某个对象，或者对有关对象进行排序等。据统计，当今处理非数值计算问题占用了 85%以上的机器时间。相比数值计算问题而言，非数值计算问题所涉及的数据元素之间的关系比较复杂，一般无法用数学方程式加以描述。因此，解决这类问题的关键不再是数学分析和计算方法，而是要设计出合适的数据结构，才能有效地解决问题。下面所列举的就是属于这一类的具体问题。

【例 1-1】 学生成绩查询问题。

编写一个高校学生成绩程序。要求给出任意一个学生的学号，查找出对应学生的成绩。

分析：要解此问题，首先构造一张学生成绩表。每个学生记录包含姓名、学号、班级、成绩等信息，成绩表由所有学生记录信息构成，详见表 1-1。

表 1-1　　学生成绩表

姓名	学号	班级	成绩	……
王一	H101	管 1	80	……
张二	H201	管 2	88	……
胡四	H301	管 3	76	……
……	……	……	……	……

查找算法取决于这张表的结构及存储方式。最简单的方式是将表中记录顺序地存储在计算机中。查找时从头开始依次查对学号，直到找出正确的学号或是找遍整张表均没有找到为止。这种查找算法对于一个不大的单位或许是可行的，但对一个有成千上万私人电话的城市就不实用了。

若这张表是按班级排序，则可另生成一张班级索引表，采用如图 1-1 所示的存储结构。那么查找过程是先在索引表中查对班级，然后根据索引表中的地址到学生成绩表中核查学号，这样查找学生成绩表时就无需查找其他班级的学生。因此，在这种新的结构上产生的查找算法就更为有效。

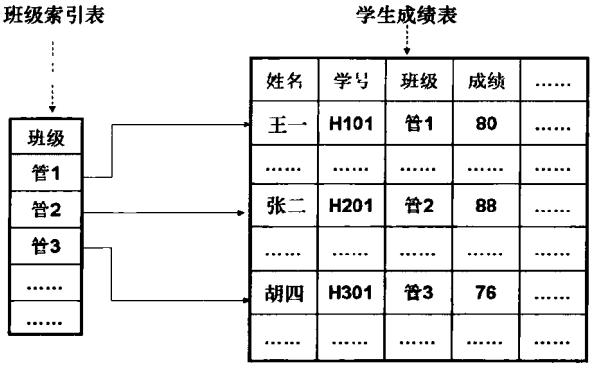


图 1-1　存储结构

【例 1-2】　表中数据插入、删除问题。

假定一个表初始由 n 个元素 a_1，a_2，…，a_n 组成，现实现以下操作：

（1）在表中 a_i 元素前插入元素 e；

（2）删除表中 a_i 元素。

操作实现与表中元素的存储方式有关，下面分顺序存储和链式存储两种方式来讨论。

① 顺序存储。所谓顺序存储就是将表中元素存放在连续的空间中，a_1，a_2，…，a_n 依次存放在存储空间的 n 个连续的位置，如图 1-2 所示。

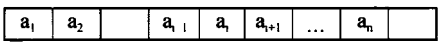


图 1-2　顺序存储

在这种存储方式下，在表中 a_i 元素前插入元素 e 前，先要将 a_n，…a_{i+1}，a_i 等（n−i+1）个元素依次后移一位，然后将 e 放在 a_i 移动前的位置，如图 1-3 所示。

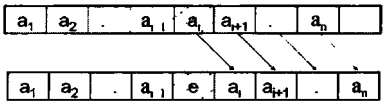


图 1-3　顺序存储时插入元素 e

而在表中删除 a_i 元素，则只需将 a_{i+1}，…，a_n 等（n-i）个元素依次前移一位即可，如图 1-4 所示。

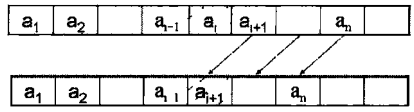


图 1-4　顺序存储时删除元素

② 链式存储。所谓链式存储就是将表中元素存放在非连续的空间中，通过增加辅助的指针信息来反映元素之间的联系，如图 1-5 所示。

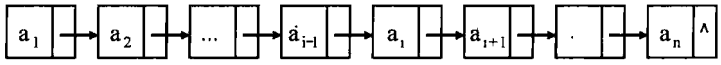


图 1-5　链式存储

在这种存储方式下，在表中 a_i 元素前插入元素 e 时，不需要进行元素移动，而只需修改相关指针内容即可，如图 1-6 所示。

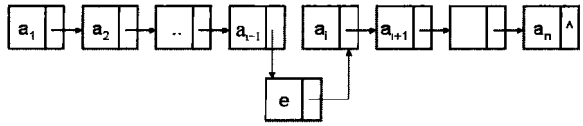


图 1-6　链式存储时插入元素

同样删除元素时也不必移动元素，只需修改有关指针内容即可，如图 1-7 所示。

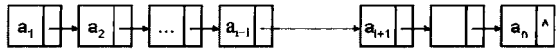


图 1-7　链式存储时删除元素

由以上内容可知，当所处理的非数值问题涉及较多插入、删除等操作时，使用链式存储方式来存储数据元素是比较有效的。

1.2　基 本 概 念

1.2.1　数据和数据结构

1. 数据

数据是对客观事物的符号表示，是由所有能够输入到计算机中并被计算机程序处理的符号（例如，"0"和"1"，"a"、"b"等）的集合。包括字符、文字、表格、图像等都可称为数据。例如，学生信息管理系统所要处理的数据可能是一张如表 1-2 所示的表格。

表 1-2　　学生信息表

姓名	性别	籍贯	出生年月	政治面貌	联系方式
张三	男	湖北	1969-10-1	中共党员	29291230
李四	男	湖南	1969-5-1	群众	29293456
……	……	……	……	……	……

2. 数据元素

数据元素是数据集合中的一个实体，是计算机程序中加工处理的基本单位。数据元素按其组成可分为简单型数据元素和复杂型数据元素。简单型数据元素由一个数据项组成，所谓数据项就是数据中不可再分割的最小单位；复杂型数据元素由多个数据项组成，它通常包含着一个概念的多方面信息。

例如，一个在校大学生的基本信息组成如下：

姓名	性别	籍贯	出生年月	政治面貌	联系方式

在上述信息中，除“出生年月”外，其他都是简单型数据元素。“出生年月”可分为年、月、日 3 个部分，属复杂型数据元素。

出生年月：	年	月	日

3. 数据结构

简单地说，数据结构就是相互之间存在一种或多种特定关系的数据元素的集合。数据结构有逻辑上的数据结构（即逻辑结构）和物理上的数据结构（即物理结构）之分。

数据的逻辑结构是指数据元素之间的逻辑关系。常见的逻辑结构有：集合结构、线性结构、树形结构和图状结构。

集合结构：数据元素之间的关系是“属于同一集合”，如图 1-8（a）所示。

线性结构：数据元素之间存在一对一的关系，如图 1-8（b）所示。

树形结构：数据元素之间存在一对多的关系，如图 1-8（c）所示。

图状结构：数据元素之间存在多对多的关系，如图 1-8（d）所示。

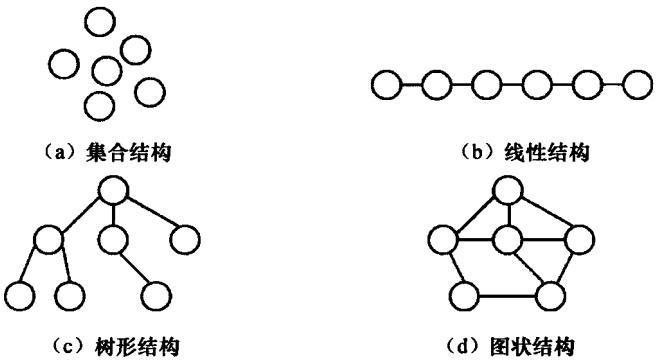


图 1-8　基本逻辑结构

一般来说，一个数据结构（Data Structure，DS）可以表示为一个二元组：

DS=（D，S）

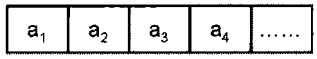
这里 D 是数据元素的集合，S 是定义在 D（或其他集合）上的关系的集合。

【例 1-3】　复数是一个数据结构

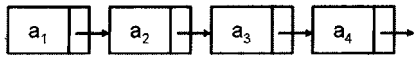
Complex=（C，R）　C=｛c_1，c_2｝　R=｛<c_1，c_2>｝

数据的物理结构，也称存储结构，是指数据结构在计算机存储器中的具体实现，是逻辑结构的存储映像（image）。常见的存储结构有顺序存储结构和链式存储结构。前者是借助于数据元素的相对存储位置来表示数据元素之间的逻辑结构，如图 1-9（a）所示；后者是借助于指示数据元

素地址的指针表示数据元素之间的逻辑结构，如图 1-9（b）所示。



（a）顺序结构



（b）链式结构

图 1-9　基本存储结构

为了叙述上的方便和避免产生混淆，通常我们把数据的逻辑结构统称为数据结构，把数据的物理结构统称为存储结构。

1.2.2　数据类型

在用高级程序语言编写的程序中，必须对程序中出现的每个变量、常量或表达式，明确说明它们所属的数据类型。数据类型是指数据的取值范围及其上可进行的操作的总称。例如，Visual Basic 语言中提供的基本数据类型有：整型，浮点型，双精度型，逻辑型，字符型等。

高级程序设计语言中的数据类型可分为：简单类型和结构类型。简单类型中的值是不可分的，例如整型，实型等。结构类型的值是由若干成分按某种结构组成的，是可分解的，如 Visual Basic 语言中的结构体也是一种结构类型，它是由固定个数的不同类型的数据顺序排列而成。例如：

```
Public Type  STUDENT
     age as Integer
     name as String*20
     score as Float
End Type
stu1 as STUDENT
```

1.2.3　抽象数据类型

抽象数据类型（Abstract Data Type，ADT）是一个数学模型以及定义在该模型上的一组操作。抽象数据类型的定义仅取决于它的一组逻辑特性，而与其在计算机内部如何表示和实现无关，即不论内部结构如何变化，只要它的数学特性不变，都不影响其外部使用。因此抽象数据类型可实现信息隐蔽和数据封装，以及使用与实现相分离。

抽象数据类型可以表示为一个三元组：

ADT=（D，S，P）

这里，D 是数据对象集合，S 是 D 上的关系集合，P 是 D 的基本操作。

实际中，可以按照如下结构来描述：

```
ADT 抽象数据类型名｛
   数据对象：〈数据对象的定义〉
   数据关系：〈数据关系的定义〉
   基本操作：〈基本操作的定义〉
｝//ADT 抽象数据类型名
```

数据对象和数据关系的定义用伪码表示，基本操作定义格式为：

基本操作名（参数表）

初始条件：〈初始条件描述〉

操作结果：〈操作结果描述〉

抽象数据类型定义举例：

```
ADT Triplet {
    数据对象：D={ e1, e2, e3 | e1, e2, e3 ∈ ElemSet }
    数据关系：R1={ <e1, e2>, <e2, e3> }
    基本操作：
    InitTriplet ( &T, v1, v2, v3 )
    操作结果：构造了三元组 T，元素 e1，e2，e3 分别被赋以参数 v1，v2，v3 的值。
    DestroyTriplet ( &T )
    操作结果：三元组被销毁。
    Put ( &T, i, e )
    初始条件：三元组已存在，i ∈ [1,3]
    操作结果：改变 T 中第 i 个元素的值为 e。
    ……
} //ADT Triplet
```

1.2.4 数据结构的符号描述举例

1. 集合结构

【例 1-4】 小组成员组成的数据结构：

Set = (D, R)

D= { 张三，李四，王五，吴一，陈二 }

R = { <张三，李四>，<张三，王五>，<张三，吴一>，<张三，陈二>，<李四，王五>，<李四，吴一>，<李四，陈二>，<王五，吴一>，<王五，陈二>，<吴一，陈二> }

这里关系<a，b>表示 a 和 b 属于同一小组。

小组成员集合示意如图 1-10（a）所示。

2. 线性结构

【例 1-5】 排队购买车票成员组成的数据结构：

List = (D, R)

D= { 张三，李四，王五，吴一，陈二 }

R = { <张三，李四>，<李四，王五>，<王五，吴一>，<吴一，陈二> }

这里关系<a，b>表示在队列中 a 是 b 的直接前驱。

排队购买车票示意如图 1-10（b）所示。

3. 树形结构

【例 1-6】 家庭成员组成的数据结构：

T = (D, R)

D= { 祖父，姑姑，叔叔，父亲，儿子，孙子 }

R = { <祖父，姑姑>，<祖父，叔叔>，<祖父，父亲>，<父亲，儿子>，<儿子，孙子> }

这里关系<a，b>表示在队列中 a 是 b 的父亲。

家庭成员组成示意如图 1-10（c）所示。

4. 图状结构

【例 1-7】　4 个直辖市航空网络的数据结构：

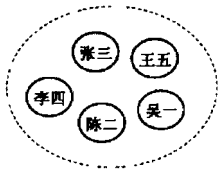
T=（D，R）

D=｛北京，上海，天津，重庆｝

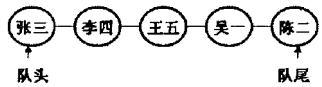
R=｛<北京，上海>，<北京，天津>，<北京，重庆>，<上海，北京>，<上海，天津>，<上海，重庆>，<天津，北京>，<天津，上海>，<天津，重庆>，<重庆，北京>，<重庆，上海>，<重庆，天津｝

这里关系<a，b>表示 a 有直达航班到 b。

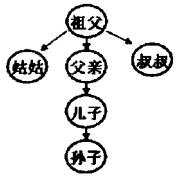
航空网络示意如图 1-10（d）所示。



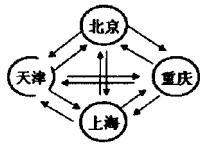
（a）小组成员集合



（b）排队购买车票



（c）家庭成员组成



（d）航空网络

图 1-10　数据结构应用举例

1.3　算法和算法描述

1.3.1　算法概念和特征

1. 算法概念

算法是在有限步骤内求解特定问题所使用的一组定义明确的规则。通俗地说，就是计算机求解特定问题的步骤的描述。特定的问题可以是数值的，也可以是非数值的。解决数值问题的算法叫做数值算法，科学和工程计算方面的算法都属于数值算法，如求解数值积分，求解线性方程组，求解代数方程，求解微分方程等。解决非数值问题的算法叫做非数值算法，数据处理方面的算法都属于非数值算法，如各种排序算法、查找算法、插入算法、删除算法、遍历算法等。数值算法和非数值算法并没有严格的区别。

2. 算法特征

一个算法应该具有以下 5 个重要的特征：有穷性、确定性、可行性、输入和输出。

有穷性是指一个算法必须保证执行有限步之后结束；

确定性是指算法的每一步必须有确切的定义，没有二义性；

可行性是指算法中描述的每一步操作都可以通过已有的基本操作执行有限次实现；

输入是指一个算法有零个或多个输入，以描述运算对象的初始情况，所谓零个输入是指算法本身给出了初始条件；

输出是指一个算法有一个或多个输出，以反映对输入数据加工后的结果，没有输出的算法是毫无意义的。

1.3.2 算法设计要求

评价一个算法的好坏有以下几个标准。

（1）正确性：即算法对于一切合法的输入数据都能产生满足规格说明的结果。

（2）可读性：算法应该易读，易于理解，一般在满足正确性的前提下，算法越简单越好。

（3）健壮性：算法应具有容错处理。当输入非法数据时，算法应对其作出反应，而不是产生莫名其妙的输出结果。

（4）效率与存储量需求。效率是指算法执行的时间；存储量需求是指算法执行过程中所需要的最大存储空间。

在保证满足标准（1）、（2）、（3）的情况下，我们希望算法执行所需时间比较短，所占用存储空间比较小。实际中，要同时满足这两点往往是很困难的，因为时间和空间是彼此冲突的。因此，应视具体情况来权衡时间和空间。

1.3.3 算法描述

算法采取多种方式来进行描述。常见的描述方式有：采用自然语言来描述，采用程序流程图的形式来描述和采用某种具体程序语言来描述等。

1. 采用自然语言来描述

这种方式是使用自然语言来描述问题的求解过程。下面举例说明。

问题 P：判断正整数 N 是否为素数。

使用自然语言来描述的算法如下：

```
Step1：令 i=2；
Step2：判断 i 是否小于等于 N/2，若是，转到 Step3；否则，转到 Step4；
Step3：判断 N 除以 i 的余数 R 是否等于 0：
       若 R 等于 0，转到 Step5；
       否则，i 加 1，转到 Step2；
Step4：输出 N 为素数；
Step5：算法结束。
```

2. 采用程序流程图的形式来描述

这种方式是使用流程图符号来描述问题的求解过程。求解问题 P 所对应的流程如图 1-11 所示。

3. 采用某种具体程序语言来描述

这种方式是使用某种具体语言（如 Visual Basic 语言）来描述问题的求解过程。求解问题 P 所对应的 VB 程序如下：

```
Sub  PrimeNumber（N% ）
```

```
    Dim  i% , j%
    i=2
    Do While （i<=N/2）
       If  （N%i==0）  then exit do
       i=i+1
    Enddo
    if（i>N/2）  then  Print  "N是素数"
End sub
```

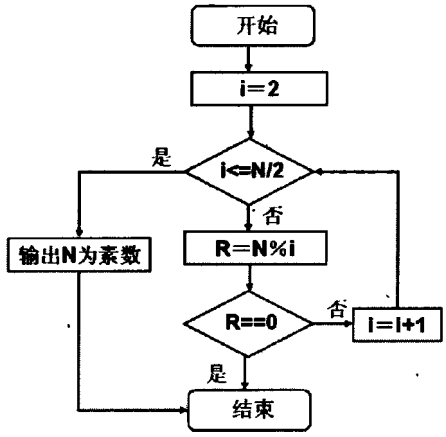


图 1-11　程序流程图

除了上述 3 种方式外，还有利用类语言（如类 C 语言，类 Pascal 等）来描述问题的求解过程，感兴趣者可以参看有关文献。不管采用哪种方式描述算法，都必须能够正确描述求解过程。本书中所有算法均采用 Visual Basic 语言来描述。

4. 设计算法的基本过程

一般来说，针对某个具体设计求解算法的基本过程包括以下 4 个阶段：

首先，通过对问题进行详细分析，抽象出相应的数学模型；

其次，确定使用的数据结构，并在此基础上设计对此数据结构实施各种操作的算法；

再次，选用某种语言将算法转换成程序；

最后，调试并运行这些程序。

1.4　算法的性能分析

算法的性能分析是指对算法质量优劣的评价。除了正确性、可读性和健壮性等特性外，重点要分析算法的时间复杂度和空间复杂度。

1.4.1　时间复杂度

算法的时间复杂度是依据该算法编制的程序在计算机上执行所消耗的时间来度量的。这种度量可采用事后统计和事前估计两种方式。

事后统计就是利用计算机内记时功能，不同算法的程序可以用一组或多组相同的统计数据区分。这种方式缺点在于：必须先运行依据算法编制的程序，所得时间统计量依赖于硬件、软件等环境因素，掩盖算法本身的优劣。

一个高级语言程序在计算机上运行所消耗的时间取决于：

（1）依据的算法选用何种策略；

（2）问题的规模；

（3）程序语言；

（4）编译程序产生机器代码质量；

（5）机器执行指令速度。

同一个算法用不同的语言、不同的编译程序，在不同的计算机上运行，效率均不同，所以使用绝对时间单位衡量算法效率不合适。

实际中，可以撇开那些与计算机硬件软件有关的因素，可以认为一个特定算法的“运行工作量”的大小只依赖于问题的规模（通常用整数量表示），或者说，它是问题规模的函数。

任何一个算法都是由控制结构和若干基本操作组成的。一般情况下，算法中基本操作重复执行的次数是问题规模为 n 的函数，记为 $T(n)$。以下以一个矩阵相乘算法来说明如何计算一个算法中语句执行的次数。

语句	执行次数
Sub matrixMultiply（ A%[]， B%[]， C%[] ）	
Dim i%，j%，k%	
For i= 0 to n-1	n+1
For j= 0 to n-1	n（n+1）
C（i，j）= 0	n^2
For k=0 to n-1	n^2*（n+1）
C（i，j）=C（i，j）+A（i，k）*B（k，j）	n^3
Next k	
Next j	
Next i	
End Sub	

总执行次数 $T(n)$ 为：

$$T(n)=n+1+n(n+1)+n^2+n^2*(n+1)+n^3=2n^3+3n^2+2n+1$$

定义：如果存在一个 $g(n)$，当 $n\to\infty$ 时，有

$$T(n)/g(n)=\text{常数}\neq 0,$$

则称函数 $T(n)$ 与 $g(n)$ 同阶，或者说，$T(n)$ 与 $g(n)$ 同一个数量级，记作

$$T(n)=O(g(n))$$

称上式为算法的时间复杂度，或称该算法的时间复杂度为 $O(g(n))$。其中，n 为问题的规模的量度。

基于高等数学中极限知识可知，当一个算法的执行次数可以表达为如下形式：

$$T(n)=\alpha_m n^{\beta_m}+\alpha_{m-1}n^{\beta_{m-1}}+\cdots+\alpha_0,\beta_i>\beta_{i-1},i=1,2,\cdots,m$$