数据结构与算法

SHUJU JIEGOU YU SUANFA

主　编　贾月乐　刘冬妮　石玉玲
副主编　范　晖　郭少辉　王　亚　孙秀明

中国水利水电出版社
www.waterpub.com.cn

数据结构与算法

SHUJU JIEGOU YU SUANFA

主　编　贾月乐　刘冬妮　石玉玲
副主编　范　晖　郭少辉　王　亚　孙秀明

中国水利水电出版社
www.waterpub.com.cn

内 容 提 要

全书分上、下两篇。上篇主要阐述数据结构的相关内容；下篇主要阐述算法设计的相关内容。具体内容包括：线性表、栈和队列、串、数组与广义表、树与二叉树、图、查找、排序、文件、分治法、动态规划法、贪心法、回溯法、分支限界法等。本书内容丰富，结构严谨，逻辑清晰，可作为高等学校计算机相关专业的教材，也可供从事科研工作的专家学者参考。

图书在版编目(CIP)数据

数据结构与算法 / 贾月乐，刘冬妮，石玉玲主编
.--北京：中国水利水电出版社，2015.6
ISBN 978-7-5170-3360-8

Ⅰ.①数… Ⅱ.①贾… ②刘… ③石… Ⅲ.①数据结构②算法分析 Ⅳ.①TP311.12

中国版本图书馆 CIP 数据核字(2015)第 156007 号

策划编辑：杨庆川 责任编辑：陈 洁 封面设计：崔 蕾

书 名	数据结构与算法
作 者	主 编 贾月乐 刘冬妮 石玉玲 副主编 范 晖 郭少辉 王 亚 孙秀明
出版发行	中国水利水电出版社 (北京市海淀区玉渊潭南路 1 号 D 座 100038) 网址：www.waterpub.com.cn E-mail：mchannel@263.net(万水) sales@waterpub.com.cn 电话：(010)68367658(发行部)、82562819(万水)
经 售	北京科水图书销售中心(零售) 电话：(010)88383994、63202643、68545874 全国各地新华书店和相关出版物销售网点
排 版	北京厚诚则铭印刷科技有限公司
印 刷	三河市佳星印装有限公司
规 格	184mm×260mm 16 开本 24.25 印张 620 千字
版 次	2015 年 11 月第 1 版 2015 年 11 月第 1 次印刷
印 数	0001—2000 册
定 价	82.00 元

凡购买我社图书，如有缺页、倒页、脱页的，本社发行部负责调换

前　言

计算机和通信技术的迅猛发展，不仅形成了融合度最高、潜力最大、增长最快的信息产业，而且成为推动全球经济快速增长和全面变革的关键因素。进入21世纪，我国信息化的发展和信息产业国际竞争能力的提高，迫切需要高素质、创新型的计算机专业人才。计算机在各个领域的应用过程中，都会涉及数据的组织与程序的编排等问题，都会用到各种各样的数据结构，特别是针对各种特殊数据的表示，就更需要学会分析和研究计算机加工对象的特性、选择最合适的数据组织结构及其存储表示方法，以及编制相应实现算法的方法，这是计算机工作者不可缺少的知识。

本书分数据结构和算法设计两部分。在数据结构中，讨论了四大类型数据结构的逻辑特性、存储表示及其应用。在算法设计中着重阐述典型算法的设计与分析。每一章后都配有适量的习题，以供学生练习。

全书分上、下两篇，共15章。前10章为上篇，主要阐述数据结构的相关内容；后5章主要阐述算法设计的相关内容。第1章为绪论；第2章至第5章主要介绍数据结构的基本知识和几种基本的数据结构，即线性表、栈和队列、串、数组和广义表，除广义表外，其他几种数据结构都属于线性数据结构；第6章和第7章叙述非线性数据结构，它们是树和图；第8章和第9章分别介绍数据处理中广泛使用的技术——查找和排序；第10章讨论外存储器上的数据结构——文件；第11章至第14章介绍了分治法、动态规划法、贪心法和回溯法等典型算法及应用；第15章介绍分支限界法的设计与分析。

本书是在计算机科学与技术专业规范和编者多年的教学经验的基础上编写而成的。全书以“语言叙述通俗易懂，讲解由浅入深，算法可读性好，应用性强，易教易学”为指导，并在以下几方面有所突破：

·从应用的角度出发，尽可能地将最基础、最实用的软件技术写入教材，略去一些理论推导和烦琐的数学证明，同时也删掉了平时讲不到或难度较大或应用性差的一些问题，增加了部分更基础、更常用或应用性强的内容。

·在内容的深浅程度上，侧重实用的同时把握理论深度，通过大量的例题、算法和每章后给出的练习题，帮助学生提高数据的抽象能力和程序设计能力。

·部分内容重新进行了组织，使全书内容结构清晰，层次分明。

本书由贾月乐、刘冬妮、石玉玲担任主编，范晖、郭少辉、王亚、孙秀明担任副主编，并由贾月乐、刘冬妮、石玉玲负责统稿，具体分工如下：

第7章～第9章：贾月乐（西南石油大学）；

第5章、第10章、第12章：刘冬妮（甘肃省庆阳林业学校）；

第3章、第6章、第11章：石玉玲（牡丹江大学）；

第2章、第4章：范晖（西京学院）；

第15章：郭少辉（许昌学院）；

第1章、第13章：王亚（许昌学院）；

第 14 章：孙秀明（许昌学院）。

本书在编写过程中参考了大量的文献和著作，在此向相关作者表示感谢。

由于编者水平有限，书中难免存在错误或不妥之处，恳请读者批评指正。

编　者

2015 年 4 月

目　录

上　篇

下 篇

第1章　绪论

1.1　数据结构的基本概念和术语

在本节中，为了与读者就某些概念取得“共识”，将对一些概念和术语赋以确定的含义。这些概念和术语将在后续的章节中多次出现。

数据(data)是信息的载体，是对客观事物的符号表示。它能够被计算机所识别、存储、加工和处理，是计算机程序加工的“原料”。数据是指所有能输入计算机中并被计算机程序处理的符号的总称。例如，一个利用数值分析方法解代数方程的程序，其处理对象是整数和实数；一个编译程序或文字处理程序的处理对象是字符串。因此，对计算机科学而言，数据的含义极为广泛，如图像、声音等都可以通过编码而归之于数据的范畴。

信息是经过计算机加工处理的带有一定意义的结果。如方程式的解、遥感图像、视频信号等等。

数据元素(data element)是数据的基本单位，在计算机程序中通常作为一个整体进行考虑和处理。有时，一个数据元素可由若干个数据项(data item)组成，例如，一本书的书目信息为一个数据元素，而书目信息中的每一项(如书名、作者名等)为一个数据项。数据项是数据的不可分割的最小单位。

数据对象(data object)是性质相同的数据元素的集合，是数据的一个子集。例如，整数数据对象是集合 $N=\{0,\pm1,\pm2,\cdots\}$，字母字符数据对象是集合 $C=\{A',B',\cdots,Z',a',b',\cdots,z'\}$。

简单地说，数据结构(data structure)是相互之间存在一种或多种特定关系的数据元素的集合。在任何问题中，数据元素都不是孤立存在的，而是在它们之间存在着某种关系，这种数据元素相互之间的关系称为结构(structure)。根据数据元素之间关系的不同特性，通常有下列四类基本结构：

1) 集合。结构中的数据元素之间除了“同属于一个集合”的关系外，别无其他关系。

2) 线性结构。结构中的数据元素之间存在一对一的关系。

3) 树形结构。结构中的数据元素之间存在一对多的关系。

4) 图形结构或网状结构。结构中的元素之间存在多对多的关系。

图 1-1 为上述四类基本结构的关系图。由于“集合”是元素之间关系极为松散的一种结构，因此也可用其他结构来表示它。

数据结构的形式定义为：

数据结构是一个二元组 Data－Structure＝(D,S) 其中：D是数据元素的有限集，S是D上关系的有限集。

存储结构(又称映象)是数据结构在计算机中的表示，也称为数据的物理结构。它包括数据元素的表示和关系的表示。在计算机中表示信息的最小单位是二进制数的二位，叫做位(bit)。计算机中，可以用一个由若干位组合起来形成的一个位串表示一个数据元素(如用 16 位二进制数

表示一个整数，用 8 位二进制数表示一个字符等），通常称这个位串为元素（element）或结点（node）。当数据元素由若干数据项组成时，位串中对应于各个数据项的子位串称为数据域（data field）。因此，元素或结点可看成是数据元素在计算机中的映象。

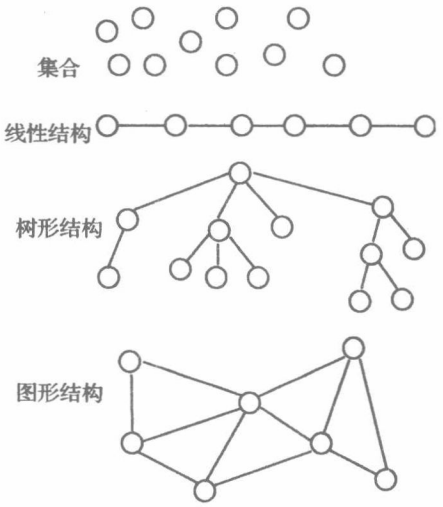

图 1-1　四类基本结构的关系图

数据元素之间的关系在计算机中有两种表示方法：顺序映象和非顺序映象。并由此得到两种不同的存储结构：顺序存储结构和链式存储结构。顺序映象的特点是借助元素在存储器中的相对位置来表示数据元素之间的逻辑关系。例如，假设用两个字节的位串表示一个实数，则可以用地址相邻的四个字节的位串表示一个复数，如图 1-2(a) 为表示复数 $z_1 = 3.0 - 3.2i$ 和 $z_2 = 8.9 - 1.6i$ 的顺序存储结构。

非顺序映象的特点是借助指示元素存储地址的指针（pointer）表示数据元素之间的逻辑关系，如图 1-2(b) 为表示复数 z_0 的链式存储结构，其中实部和虚部之间的关系用值为“0415”的指针来表示（0415 是虚部的存储地址）。数据的逻辑结构和物理结构是密切相关的两个方面，任何一个算法的设计取决于所选定的逻辑结构，而算法的实现依赖于采用的存储结构。那么，在数据结构确立之后如何描述存储结构呢？虽然存储结构涉及数据元素及其关系在存储器中的物理位置，但由于本书是在高级程序语言的层次上讨论数据结构的操作，因此不能如上所谈的那样直接以内存地址来描述存储结构。可以借用高级程序语言中提供的“数据类型”来描述它，例如可以用所有高级程序语言中都有的“一维数组”类型来描述顺序存储结构，以 C 语言提供的“指针”来描述链式存储结构。假如把 C 语言看成是一个执行 C 指令和 C 数据类型的虚拟处理器，那么本书中讨论的存储结构是数据结构在 C 虚拟处理器中的表示，不妨称它为虚拟存储结构。

数据类型（Data type）是和数据结构密切相关的一个概念，用以刻画（程序）操作对象的特性。它最早出现在高级程序语言中。在用高级程序语言编写的程序中，每个变量、常量或表达式都有一个它所属的确定的数据类型。类型明显或隐含地规定了在程序执行期间变量或表达式所有可能取值的范围，以及在这些取值上所允许进行的操作。因此数据类型是一个值的集合和定义在这个值集合上的一组操作的总称。例如 C 语言中的整数类型，其值集为区间[－maxint，maxint]上的整数（maxint 是依赖特定的计算机的最大整数），定义在其上的一组操作为：加、减、乘、整除和取模等。按“值”的不同特性，高级程序语言中的数据类型可分为两类：

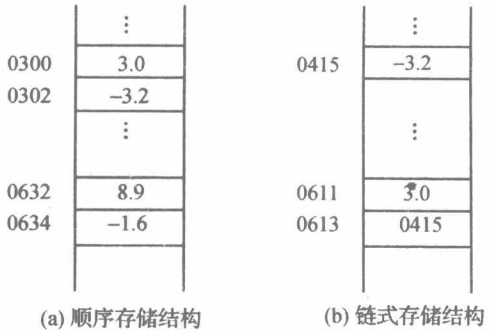

图 1-2　复数的存储结构

一类是非结构的原子类型。原子类型的值是不可分解的，如 C 语言中的标准类型（整型、单精度、双精度、字符型等）。

另一类是结构类型。结构类型的值是由若干成分按某种结构组成的，因此，它是可以分解的。它的成分既可以是非结构的，也可以是结构的。例如数组的值由若干分量组成，每个分量可以是整数，也可以是数组等。

实际上，在计算机中，数据类型的概念并非局限于高级语言中，每个处理器（包括计算机硬件系统、操作系统、高级语言、数据库等）都提供了一组原子类型或结构类型。例如，一个计算机硬件系统通常含有“位”、“字节”、“字”等原子类型。它们的操作通过计算机设计的一套指令系统直接由电路系统完成，而高级程序语言提供的数据类型，其操作需通过编译器或解释器转化成底层语言即汇编语言或机器语言的数据类型来实现。引入“数据类型”的目的，从硬件的角度看，是作为解释计算机内存中信息含义的一种手段，而对使用数据类型的用户来说，实现了信息的隐蔽，即将一切用户不必了解的细节都封装在类型中。例如，用户在使用“整数”类型时，既不需要了解“整数”在计算机内部是如何表示的，也不需要知道其操作是如何实现的。如“两整数求和”，程序设计者注重的仅仅是其“数学上求和”的抽象特性，而不是其硬件的“位”操作如何进行。

抽象数据类型 ADT（Abstract Data Type）是指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。即不论其内部结构如何变化，只要它的数学特性不变，都不影响其外部的使用。

抽象数据类型和数据类型实质上是一个概念。例如，各个计算机都拥有的“整数”类型是一个抽象数据类型，尽管它们在不同处理器上实现的方法可以不同，但由于其定义的数学特性相同，在用户看来是相同的。因此，“抽象”的意义在于数据类型的数学抽象特性。另一方面，抽象数据类型的范畴更广，它不再局限于前述各处理器中已定义并实现的数据类型（也可称这类数据类型为固有数据类型），还包括用户在设计软件系统时自己定义的数据类型。为了提高软件的复用率，在近代程序设计方法学中指出，一个软件系统的框架应建立在数据之上，而不是建立在操作之上（后者是传统的软件设计方法所为）。即在构成软件系统的每个相对独立的模块上，定义一组数据和施于这些数据上的一组操作，并在模块内部给出这些数据的表示及其操作的细节，而在模块外部使用的只是抽象的数据和抽象的操作。显然，所定义的数据类型的抽象层次越高，含有该抽象数据类型的软件模块的复用程度也就越高。

从以上对数据类型的讨论中可见，与数据结构密切相关的是定义在数据结构上的一组操作，操作的种类和数目不同，即使逻辑结构相同，这个数据结构的用途也会大为不同（最典型的例子

是第 3 章所讨论的栈和队列)。

操作的种类是没有限制的,可以根据需要定义。基本的操作主要有以下几种:

1) 插入。在数据结构中的指定位置上增添新的数据元素。

2) 删除。删去数据结构中某个指定的数据元素。

3) 更新。改变数据结构中某个数据元素的值,在概念上等价于删除和插入操作的组合。

4) 查找。在数据结构中寻找满足某个特定要求的数据元素(的位置和值)。

5) 排序。(在线性结构中) 重新安排数据元素之间的逻辑顺序关系,使之按值由小到大或由大到小的顺序排序。

从操作的特性来分,所有的操作可以归结为两类:一类是加工型操作(constructor),此操作改变了(操作之前的) 结构的值;另一类是引用型操作(selector),此操作不改变结构的值,只是查询或求得结构的值。不难看出,前述 5 种操作中除"查找"为引用型操作外,其余均是加工型操作。

1.2 数据类型

数据类型是一组性质相同的值集合以及定义在这个集合上的一组操作的总称。数据类型中定义了两个集合,即该类型的取值范围,以及该类型中可允许使用的一组运算。例如高级语言中的数据类型就是已经实现的数据结构的实例。在高级语言如 C 语言中,整型类型可能的取值范围是 $-32768 \sim +32767$,可用的运算符集合为加、减、乘、除、乘方、取模。

按"值"的不同特性,高级程序语言中的数据类型可分为两大类:

一类是非结构的原子类型。原子类型的值是不可分解的,如 C 语言中的标准类型(整型、实型和字符型) 及指针。

另一类是结构类型,结构类型的值是由若干成分按某种结构组成的,因此是可以分解的,并且它的成分可以是非结构的,也可以是结构的。例如数组的值由若干分量组成,每个分量可以是整数,也可以是数组等。数据类型指由系统定义的、可直接使用且可构造的数据类型。

因此,数据类型不仅是数据对象的集合还有在这些数据对象上的操作集合。

抽象数据类型是指一个数学模型及定义在该数学模型上的一组操作。抽象数据模型的定义取决于它的一组逻辑特性,与其在计算机内部如何表示和实现无关。

抽象数据类型的范畴更加广泛,不局限于已固有的数据类型,还包括了用户自定义的数据类型。因此,在抽象数据定义时定义三元组:D 为数据对象,S 为数据对象间的关系,O 为数据集上所完成的基本操作集 ——(D,S,O)。

在现代软件设计和开发中,大量使用抽象数据类型。

抽象数据类型的定义格式:

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

线性表的抽象数据类型描述:

```
ADT Linear_list  {
```

数据元素所有 a_i 属于同一数据对象，$i=1,2,\cdots,n(n\geqslant 1)$。

逻辑结构所有数据元素 a_i 存在次序关系(a_i,a_{i+1})，a_1 无前驱，a_n 无后继。

线性表的基本操作/* 设 L 为 Linear_list 类型的线性表 */

```
InitList(L);          /* 建立一个空的线性表 L */
Length(L);            /* 求线性表 L 的长度 */
GetElement(L,i);      /* 取线性表 L 中的第 i 个元素 */
Locate(L,x);          /* 确定元素 x 在线性表 L 中的位置 */
Insert(L,i,x);        /* 在线性表 L 中的第 i 个位置处插入数据元素 x */
Delete(L,i);          /* 删除表 L 中第 i 个位置的元素 */
}
```

1.3　算法与算法分析

1.3.1　算法及其特征

算法是对特定问题求解步骤的一种描述，它是指令的有限序列，其中每条指令表示一个或多个操作。算法有以下 5 个重要特征：

1）有穷性。一个算法必须总是（对任何合法的输入值）在执行有穷步之后结束，且每一步都可在有穷时间内完成。

2）确定性。算法中每一条指令必须有确切的含义，不会产生二义性。

3）可行性。一个算法是可行的，即算法中描述的操作都是可以通过已经实现的基本运算的有限次执行来实现。

4）输入性。一个算法有零个或多个的输入。

5）输出性。一个算法有一个或多个的输出。

例 1.1　考虑下列两段描述：

```
(1)void exam1()
   {
       n = 2;
      while(n%2 == 0)
            n = n+2;
      printf("%d",n);
   }
(2)void exam 2()
   {
        y = 2;
        x = 5/y;
        printf("%d,%d\n",x,y);
   }
```

这两段描述均不能满足算法的特征，试问它们违反了哪些特征？

解：1）是一个死循环违反了算法的有穷性特征。

2）包含除零错误，违反了的可行性特征。

1.3.2 算法描述

描述算法的方法很多，有的采用类 PASCAL，有的采用自然语言等。本书采用 C/C++ 语言来描述算法的实现过程。下面总结常用的用于描述算法的 C 语言语句。

(1) 输入语句

scanf(格式控制字符串，输入项表)；

(2) 输出语句

printf(格式控制字符串，输出项表)；

(3) 赋值语句

变量名 = 表达式；

(4) 条件语句

if < 条件 >< 语句 >；

或者

if < 条件 >< 语句 1 > else < 语句 2 >；

(5) 循环语句

• while 循环语句

while 表达式

　　循环体语句；

• do-while 循环语句

do

　　循环体语句；

while 表达式；

• for 循环语句

for(赋初值表达式 1；条件表达式 2；步长表达式 3)

　　循环体语句；

(6) 返回语句

Return(返回表达式)；

(7) 定义函数语句

函数返回值类型(类型名 形参 1，类型名 形参 2，...)

{

　　说明部分；

　　函数语句部分；

}

(8) 调用函数语句

函数名(实参 1，实参 2，...)

在C++语言中，在函数调用时实参和形参的参数传递分为传值和传引用(引用符号为“&”)两种方式。

传值方式是单向的值传递例如，有一个函数 fun1(x,y)(其中 x 和 y 为值形参)，在调用 fun4(a,b)(其中 a 和 b 为实参)时，将 a 的值传给 x，b 的值传给 y，然后执行函数体语句，执行完函数后 x、y 的值不会回传给 a、b。

传引用方式是双向的值传递。例如，有一个函数 fun2(x,y)(其中 x 和 y 为引用形参)，在调用 fun2(a,b)(其中 a 和 b 为实参)时，将 a 的值传给 x，b 的值传给 y 然后执行函数体语句，执行完函数后 x、y 的值分别回传给 a、b。

例如，有如下程序：

```
#include < stdio.h >
void fun1(int x,int y)
{
    y = (x < 0)? - x:x;
}
void fun2(int x,int &y)
{
    y = (x < 0)? - x:x;
}
void main()
{
    int a,b;
    a = - 2;b = 0;
    fun1(a,b);
    printf("fun1:b = %d\n",b);
    a = - 2;b = 0;
    fun2(a,b);
    printf("fun2:b = %d\n",b);
}
```

其中，fun1(x,y)、fun2(x,y)的功能都是计算 y = | x |，fun1 中形参 y 使用传值方式，fun2 中形参 y 使用传引用方式，也就是说，执行 fun1(a,b)时，b 实参的值不会改变，而执行 fun2(a,b)时，b 实参的值可能发生改变。程序执行结果如下：

```
fun1:b=0
fun2:b=2
```

为此，在设计算法(通常设计成 C/C++ 函数)时，若某个形参需要将计算的值回传给对应的实参，则需将其设计为引用传递参数的方式，否则不必使用引用方式。

1.3.3　算法分析

算法分析的两个主要方面是分析算法的时间复杂度和空间复杂度，其目的不是分析算法是否正确或是否容易阅读，主要是考察算法的时间和空间效率，以求改进算法或对不同的算法进行比较。一般情况下，鉴于运算空间(内存)较为充足，所以把算法的时间复杂度作为分析的重点。

算法的执行时间主要与问题规模有关。问题规模是一个和输入有关的量，例如，数组的元素个数、矩阵的阶数等。所谓一个语句的频度，即指该语句在算法中被重复执行的次数。算法中所有语句的频度之和记做 $T(n)$，它是该算法所求解问题规模 n 的函数，当问题的规模 n 趋向无穷大时，$T(n)$ 的数量级称为渐近时间复杂度，简称为时间复杂度，记作 $T(n)=O(f(n))$。

上述表达式中“0”的含义是 $T(n)$ 的数量级，其严格的数学定义是：若 $T(n)$ 和 $f(n)$ 是定义在正整数集合上的两个函数，则存在正的常数 C 和 n_0，使得当 $n \geqslant n_0$ 时，总是满足 $0 \leqslant T(n) \leqslant C \cdot f(n)$。但是应总是考虑在最坏情况下的时间复杂度，以保证算法的运行时间不会比它更长。

另外，由于算法的时间复杂度主要是分析 $T(n)$ 的数量级，而算法中基本运算的频度与 $T(n)$ 同数量级，所以通常采用算法中基本运算的频度来分析算法的时间复杂度，被视为算法基本运算的一般是最深层循环内的语句。

用数量级形式 $O(f(n))$ 表示算法执行时间 $T(n)$ 的时候，函数 $f(n)$ 通常取较简单的形式，如 1、$\log_2 n$、n、$n\log_2 n$、n^2、n^3、2^n 等。在 n 较大的情况下，常见的时间复杂度之间存在下列关系：

$$O(1) < O(\log_2 n) < O(n) < O(n\log_2 n) < O(n^2) < O(n^3) < O(2^n)$$

习题

1. 回答下列问题。

(1) 什么叫数据、数据元素、数据项和数据对象？

(2) 什么叫数据结构？分为哪几类？具体都是什么？

(3) 什么叫数据类型？什么叫抽象的数据类型？

(4) 什么叫算法？算法有哪几个基本特性？

(5) 如何评价一个算法的好与坏？

2. 举例说明常见的时间复杂性有哪些。

3. 写出下列算法总的语句执行次数。

(1)
```
y = 5;x = 1;
while(y <= 10)
  if(x == 5)
  {
      x = 1;
      y += x;
  }
else x++;
```

(2)
```
x = 0;
  for(i = 0;i < 10;i++)
    for(j = 0;j <= i; j++)
        x = x+1;
```

4. 分析如下各种算法的时空性能。

(1) 计算 n 个实数的平均值，并找出其中的最大数和最小数。