（C 语言版）

# 数据结构题集

严蔚敏 吴伟民 编著

清华大学出版社
http://www.tup.tsinghua.edu.cn

# 数据结构题集

## （C语言版）

严蔚敏　吴伟民　编著

清华大学出版社

(京)新登字 158 号

## 内 容 简 介

本题集与清华大学出版社出版的《数据结构》(C 语言版)一书相配套,主要内容有:习题与学习指导、实习题和部分习题的提示或答案三大部分和一个附录["数据结构算法演示系统(类 C 描述语言 3.1 中文版)使用手册",此软件已由清华大学出版社出版]。

其中习题篇的内容和数据结构(C 语言版)一书相对应,也分为 12 章,每一章大致由基本内容、学习要点、算法演示内容及基础知识题和算法设计题五部分组成。实习题分成六组,每一组都有鲜明的主题,围绕 1 至 2 种数据结构,安排 4 至 9 个题,每个题都有明确的练习目的和要求,在每一组中都给出一个实习报告的范例,以供读者参考。

本书内容丰富、程序设计观点新颖,在内容的详尽程度上接近课程辅导材料,不仅可作为大专院校的配套教材,也是广大工程技术人员和自学读者颇有帮助的辅助教材。



**图书在版编目(CIP)数据**

数据结构题集:C 语言版/严蔚敏,吴伟民编著. —北京:清华大学出版社,1999
ISBN 7-302-03314-5

Ⅰ. 数… Ⅱ. ①严… ②吴… Ⅲ. C 语言-程序设计-数据结构-习题 Ⅳ. TP311.12

中国版本图书馆 CIP 数据核字(1999)第 00801 号

**出版者**:清华大学出版社(北京清华大学校内,邮编 100084)
http://www.tup.tsinghua.edu.cn
**印刷者**:北京国马印刷厂
**发行者**:新华书店总店北京发行所
**开 本**:787×1092 1/16 **印张**:15 **字数**:355 千字
**版 次**:1999 年 2 月第 1 版 1999 年 8 月第 3 次印刷
**书 号**:ISBN 7-302-03314-5/TP・1781
**印 数**:32001～63000
**定 价**:16.00 元

# 前　言

数据结构是计算机科学的算法理论基础和软件设计的技术基础，主要研究信息的逻辑结构及其基本操作在计算机中的表示和实现。数据结构不仅是计算机专业的核心课程，而且已成为其他理工科专业的热门选修课。课程的教学要求之一是训练学生进行复杂程序设计的技能和培养良好程序设计的习惯，其重要程度决不亚于知识传授。因此，在数据结构的整个教学过程中，完成习题作业和上机实习是两个至关重要的环节。为了帮助读者学好这门课程，我们编写了这本具有学习指导功能的题集。

目前，出版的数据结构系列教材有C和Pascal两种描述语言的版本。这本题集与《数据结构》(C语言版)(清华大学出版社，1997年出版)配套，习题和实习都是按相同的内容顺序编排的，很多习题涉及教科书上的内容或算法，因此读者手边最好能有这本教科书，以便随时查阅。

习题的作用在于帮助学生深入理解教材内容，巩固基本概念，达到培养良好程序设计能力和习惯的目的。从认知的程度划分，数据结构的习题通常可分为三类：基础知识题、算法设计题和综合实习题。基础知识题主要是检查对概念知识的记忆和理解，一般可作为学生自测题。算法设计题的目的是练习对原理方法的简单应用，多数要求在某种数据存储结构上实现某一操作，是数据结构的基础训练，构成了课外作业的主体。综合实习题则训练知识的综合应用和软件开发能力，主要是针对具体应用问题，选择、设计和实现抽象数据类型(ADT)的可重用模块，并以此为基础开发满足问题要求的小型应用软件，应将其看作软件工程的综合性基础训练的重要一环，并给予足够的重视。

本书第一篇含有全部四百多个习题，组织成12章，分别对应教科书中各章内容，并在每章之前给出该章的内容提要和学习要求。这些习题是作者在多年教学过程中所积累资料的基础上，参考大量国外教材之后精心设计而成的。书中对特别推荐的题目作了标记，并对每道习题的难易程度按五级划分法给出了难度系数，仅供参考。

第二篇分别以抽象数据类型、线性表、栈和队列、串、数组和广义表、树和图以及查找和排序为核心，设置了7组上机实习题，每组有3至9个题目供读者自由选择。希望这些实习题能对习题起到良好的补充作用，使读者受到涉及“从问题到程序”的应用软件设计的完整过程的综合训练，培养合作能力，成为将来进行软件开发和研究工作的“实践演习”。

第三篇安排了部分习题的提示或解答。对于多数有唯一确定解的题给出了答案，而对算法题则有选择地作了示范解答或提示。但是，算法的解答都不是唯一的，我们的解答也不一定是臻于完美的。希望我们的答案和提示能起到抛砖引玉的作用，愿读者开发出更多更好的解法。热忱欢迎读者将这些好的算法寄给我们，在此预先表示感谢。然而，我们仍想特别强调的是，本题集主要是为配合高等院校的教学而编写的，因此，为了培养学生独立思考和解决问题的能力，我们诚恳希望不再出版或编印本题集的更详尽的解答，以免干

扰学校正常的教学和本书的训练意图，敬请谅解。

数据结构是实践性很强的课程，光是“听”和“读”是绝对不够的。在努力提高课堂教学的同时，必须大力加强对作业实践环节的要求和管理。国内外先进院校一般都要求修读数据结构的学生每周应不少于 4 个作业机时，而且有一套严格的作业、实习规范和成绩评定标准，形成行之有效的教学质量保证体系。本题集强调规范化在算法设计基本训练中的重要地位。在习题篇中给出了算法书写规范，在实习题篇中给出了实习步骤和实习报告的规范。教学经验表明，严格实施这些貌似繁琐的规范，对于学生基本程序设计素养的培养和软件工作者工作作风的训练，将能起到显著的促进作用。

数据结构及其算法的教学难点在于它们的抽象性和动态性。虽然在书本教材和课堂授课(板书或投影胶片)中采用图示可以在一定程度上化抽象为直观，但很难有效展现数据结构的瞬间动态特性和算法的作用过程。“数据结构的算法动态模拟辅助教学软件 DS-DEMO”是为学习并掌握数据结构中各类典型算法而开发的一个辅助教学软件，可对教科书中 80 多个典型算法进行动态交互式跟踪演示，在算法执行过程中实现数据结构和算法的动态同步可视化，使读者获得单从教材文字说明中无法获得的直观知识。软件既可用于课堂讲解演示，又能供个人课外反复观察、体会和理解，对提高教学质量和效率有显著效果。为便于读者参考，在习题篇的每一章列举了与该章相关的算法清单，并在附录中提供该软件完整的使用说明。

1987 年出版的旧版题集一直发行至今，曾在计算机和其他理工专业的数据结构教学中得到广泛使用，反映良好。这本 C 语言版题集在力求反映程序设计和软件工程新思想方面作了一些探索，如：算法演示软件、模块化抽象和信息隐蔽、软件工程方法训练等。对于书中存在的谬误和有争议之处，作者诚恳地欢迎广大读者提出批评意见和建议，在此谨向热情的读者致以衷心的感谢。

严蔚敏　　清华大学计算机技术与科学系
吴伟民　　广东工业大学计算机科学与工程一系
1998 年 7 月

# 目　录

# 第一篇　习题与学习指导

## 第0章　本篇提要与作业规范

### 一、本篇提要

本篇内容是按照作者编著的教科书《数据结构》(C语言版)的内容和课程教学要求组织的。各章均由基本内容、学习要点、算法演示内容以及基础知识题和算法设计题五部分组成。其中：

“基本内容”列举了该章的内容提要，提醒读者把握该章主要内容；

“学习要点”指出了该章的教学重点和难点，以供读者在组织教学或自学时选择习题作参考；

“算法演示内容”提供了“数据结构算法演示DSDEMO(类C语言版)”软件中包含的与该章相关的算法清单，通过观察算法执行过程的演示，将有助于深刻理解算法的本质和提高教学效果(在附录中列有类C语言版DSDEMO软件的使用手册)。

数据结构的练习题大致可分为“基础知识题”和“算法设计题”两类。

“基础知识题”主要供读者进行复习自测之用，目的是帮助读者深化理解教科书的内容，澄清基本概念、理解和掌握数据结构中分析问题的基本方法和算法要点，为完成算法设计题做准备。

“算法设计题”则侧重于基本程序设计技能的训练，相对于实习题而言，这类编程习题属于偏重于编写功能单一的“小”程序的基础训练，然而，它是进行复杂程序设计的基础，是本课程习题作业的主体和重点。

各章的题量根据教学内容的多少和重要程度而定，几乎对教科书的每一小节都安排了对应的习题。但对每个读者来说，不必去解全部习题，而只需根据自己的情况从中选择若干求解即可。为了表明题目的难易程度，便于读者选择，我们在每个题的题号之后注了一个难度系数，难度级别从①至⑤逐渐加深，其区别大致如下：难度系数为①和②的习题以基础知识题为主；难度系数为③的习题以程序设计基础训练为主要目的，如强化对“指针”的基本操作的训练等；习题中也收纳了不少难题，其难度系数设为④和⑤，解答这些题可以激起学习潜力较大的读者的兴趣，对广泛开拓思路很有益。但习题的难度系数也只是一个相对量，读者的水平将随学习的进展而不断提高，因此没有必要去比较不同章节的习题的难度系数，此外，该难度系数值的假设是以读者没有参照习题的解答或提示为前提的。

“循序渐进”是最基本的学习原则。读者不应该片面追求难题。对于解难度系数为$i$的

习题不太费力的读者，应试试难度系数为 $i+1$ 的习题，但不要把太多的时间浪费在难度系数为 $i+2$ 的习题上。“少而精”和“举一反三”是实践证明行之有效的。解答习题应注重于“精”，而不要求“多”。为此，编者在一些自认为值得向读者推荐的“好题”题号前加注了标记◆。把握住这些“关键点”，就把握住了数据结构习题、乃至数据结构课程的总脉络。

对算法设计习题的基本要求是使用本章提供的类 C 语言和算法书写规范写出书面作业的算法。需要强调的是“算法的可读性”。算法是为了让人来读的，而不是供机器读的，初学者总是容易忽视这一点。算法的真正意图主要在于提供一种在程序设计者之间交流解决问题方法的手段。因此，可读性具有头等的重要性。不可读的算法是没有用的，由它得到的程序极容易产生很多隐藏很深的错误，且难以调试正确。一般地说，宁要一个可读性好、逻辑清晰简单、但篇幅较长的算法，也不要篇幅较小但晦涩难懂的算法。算法的正确性力求在设计算法的过程中得到保证，然而一开始做不到这一点也没多大关系，可以逐步做到。

算法设计的正确方法是：首先理解问题，明确给定的条件和要求解决的问题，然后按照自顶向下、逐步求精、分而治之的策略逐一地解决子问题，最后严格按照和使用本章后面提供的算法书写规范和类 C 语言完成算法的最后版本。

按照规范书写算法是一个值得高度重视的问题。在基础训练中就贯彻这一规范，不但能够有助于写出“好程序”，避免形成一系列难以纠正且遗害无穷的程序设计坏习惯，而且能够培养软件工作者应有的严谨的科学工作作风。

## 二、类 C 语言语法概要

本书采用的类 C 语言精选了 C 语言的一个核心子集，同时作了若干扩充修改，增强了语言的描述功能。以下对其作简要说明。

(1) 预定义常量和类型：

```
// 函数结果状态代码
#define   TRUE 1
#define   FALSE 0
#define   OK 1
#define   ERROR 0
#define   INFEASIBLE -1
#define   OVERFLOW -2
typedef   int   Status;
        // Status 是函数的类型，其值是函数结果状态代码
typedef int bool;
        // bool 是布尔类型，其值是 TRUE 或 FALSE
```

(2) 数据结构的表示(存储结构)用类型定义(typedef)描述。数据元素类型约定为 ElemType，由用户在使用该数据类型时自行定义。

(3) 基本操作的算法都用以下形式的函数描述：

```
函数类型 函数名(函数参数表){
  // 算法说明
  语句序列
} // 函数名
```

除了函数的参数需要说明类型外，算法中使用的辅助变量可以不作变量说明，必要时对其作用给出注释。一般而言，a,b,c,d,e 等用作数据元素名，i,j,k,l,m,n 等用作整型变量名，p,q,r 等用作指针变量名。当函数返回值为函数结果状态代码时，函数定义为 **Status** 类型。为了便于算法描述，除了值调用方式外，增添了 C++语言的引用调用的参数传递方式。在形参表中，以& 打头的参数即为引用参数。

(4) 赋值语句有

简单赋值　变量名＝表达式；

串联赋值　变量名$_1$＝变量名$_2$＝…＝变量名$_k$＝表达式；

成组赋值　(变量名$_1$,… ,变量名$_k$)＝(表达式$_1$,… ,表达式$_k$)；

　　结构名＝结构名；

　　结构名＝(值$_1$,… ,值$_k$)；

　　变量名[]＝表达式；

　　变量名[起始下标..终止下标]＝变量名[起始下标..终止下标]；

交换赋值　变量名←→变量名；

条件赋值　变量名＝条件表达式？表达式 T ：表达式 F；

(5) 选择语句有

```
条件语句 1  if (表达式) 语句;
条件语句 2  if (表达式) 语句;
            else 语句;
开关语句 1   switch (表达式) {
                case 值1: 语句序列1;  break;
                …
                case 值n: 语句序列n;  break;
                default: 语句序列n+1;
             }
开关语句 2   switch {
                case 条件1: 语句序列1;  break;
                …
                case 条件n: 语句序列n;  break;
                default: 语句序列n+1;
             }
```

(6) 循环语句有

**for** 语句　　**for** (赋初值表达式序列；　条件；　修改表达式序列) 语句；

**while** 语句　　**while**（条件）语句；

**do-while** 语句　**do** {

　　语句序列；

　　} **while**（条件）；

(7) 结束语句有

函数结束语句　**return** 表达式；

　　**return**；

**case** 结束语句　**break**；

异常结束语句　**exit**（异常代码）；

(8) 输入和输出语句有

输入语句　**scanf**([格式串],变量 1,... ,变量 n)；

输出语句　**printf**([格式串],表达式 1,... ,表达式 n)；

通常省略格式串。

(9) 注释

单行注释　　// 文字序列

(10) 基本函数有

求最大值　　**max**（表达式 1,... ,表达式 n)

求最小值　　**min**（表达式 1,... ,表达式 n)

求绝对值　　**abs**（表达式)

求不足整数值　**floor**（表达式)

求进位整数值　**ceil**（表达式)

判定文件结束　**eof**（文件变量）或　**eof**

判定行结束　　**eoln**（文件变量）或　**eoln**

(11) 逻辑运算约定

与运算&&：　对于 A && B，当 A 的值为 0 时，不再对 B 求值。

或运算‖：　对于 A ‖ B，当 A 的值为非 0 时，不再对 B 求值。

## 三、算法书写规范

(1) 算法说明

算法说明，也称为(算法)规格说明，是一个完整算法不可缺少的部分，应该在算法头(即过程或函数首部)之下以注释的形式写明如下内容：指明算法的功能；参数表中各参量的含意和输入输出属性；算法中引用了哪些全局变量或外部定义的变量，它们的作用、入口初值以及应满足哪些限制条件，例如，链表是否带头结点、表中元素是否有序、按递增还是递减方式有序等。必要时，算法说明还可用来陈述算法思想、采用的存储结构等。递归算法的说明特别重要，读者应该力求将它写成算法的严格定义。

算法说明应该在开始设计算法时就写明，可以在算法设计过程中作一些补充和修改，但是切忌最后补写。对于递归算法的情况，这一点尤其重要。这样做也是递归算法设计的正确而有效的途径，在算法设计(即解决一个问题)的过程中，能否利用自身的处理能力来

解决所划分出的一个或几个子问题，全凭检查自身的规格说明而定。书写(递归)算法的规格说明时，应该忽略它如何实现或者假定它能够实现。如何实现的问题正是接下去要做的事。

算法说明书写得不好或不完全时，往往失去了评判一个算法正确与否的标准。书写恰当而又简洁的算法说明是一项具有很强技巧性的活动，通常要经过不断的练习才能达到。在本节的末尾将列出一些规格说明的例子。

(2) 注释与断言

在难懂的语句和关键的语句(段)之后加以注释可以大大提高程序的可读性。注释要恰当，并非越多越好；此外，注释句的抽象程度应略高于语句(段)，例如，应避免用"{i 增加1}"来注释语句"i++;"。

断言是注释的一种特殊写法，是一类特别重要的注释。它是一个逻辑谓词，陈述算法执行到这点时应满足的条件。多写断言式的注释，甚至以断言引导算法段的设计，是提高算法的结构良好性、避免错误和增强可读性的有效手段，是特别值得提倡的。其中最重要的是算法的入口断言和 else 分支断言。注意，正确的算法也只能在输入参数值合法的前提下得出正确的结果。如果算法不含参数合法性检测代码段，书写入口断言是最低限度的要求。

(3) 输入和输出

算法的输入和输出可以通过三种途径实现。第一种是通过 scanf 和 printf 语句实现，其特点是实现了算法与计算环境外部的信息交换；第二种是以算法头中参数表里显式列出的参量作为输入/输出的媒介；第三种是通过全局变量或外部变量隐式地传递信息。后两种方法的特点是实现了一个算法与其调用者之间的信息交换。

如果一个算法是定义在某个数据结构之上的几个操作之一，该数据结构可以不列在算法的参数表中。在其他情况下，应尽量避免使用第三种方法。

(4) 错误处理

尽可能使用函数值返回算法的执行状态(正确/错误，或是错误代码等)，便于调用者处理异常情况，有利于培养良好的程序设计习惯。

(5) 语句选用和算法结构

赋值语句、**if** 分支语句和 **while**( 或 **for**)循环语句是最基本的三种语句，仅用此三种语句就足以对付一切算法的设计了。实际上，不仅是"足够"，而且是"最好"。这样做对于提高结构良好性和可读性、避免逻辑错误是有益和有效的。**switch** 分支语句是广义的 **if** 分支语句，在分支条件复杂时选用可以避免 **if** 语句的多重嵌套，有助于提高算法的可读性，也是一个鼓励使用的语句。一般情况下，不准使用 **goto** 语句，个别的特殊情况除外。

算法设计过程中应尽量避免下列所示的语句结构：

```
do {
   do {
      …
   } while …
   …
```

} **while** …

或者

**if** ( … )

  **if** …

对于第二种情形，如果难以改变，应该对第二个 **if** 语句加上一对语句括号，以便明确条件成立时的作用范围。此外，语句的开/闭括号应对齐。

(6) 基本运算

如果题目中未明确要求用某种数据结构上的基本运算编写算法，不得直接利用教科书中给出的基本运算。如果非用不可，则要求将所用到的所有基本运算同时实现。

(7) 几点建议

· 建议以图说明算法；

· 建议在算法书写完毕后，用边界条件的输入参数值验证一下算法能否正确执行。例如，对于顺序表插入算法，空表是一个边界条件。

(8) 附例：算法规格说明例释

**例 1** 第 2.10 题

```
Status DeleteK(SqList a, int i, int k)
  // 本过程从顺序存储结构的线性表 a 中删除第 i 个元素起的 k 个元素。
```

**例 2** 第 2.39 题

```
float Evaluate( SqPoly pn;  float x )
// 多项式 pn.data[i].coef 存放 a_i，pn.data[i].exp 存放 e_i(i=1,2,…,m)。
// 本算法计算并返回 ∑_{i=1}^{m} a_i x^{e_i} 。不判别溢出。
// 此外，入口时要求 0≤e_1<e_2<…<e_m，算法内不对此再作验证。
```

**例 3** 第 3.15 题的入栈 push 算法

```
void push( TwoWayStack &tws,  int i,  ElemType x,  bool &overflow)
// 两栈(标号 0，1)共享空间 tws.elem[0..m-1]，栈底在两端。Tws.top[0]和
// tws.top[1]为两栈顶指针。tws.data[0..tws.top[0]]为栈 0，
// tws.data[tws.top[1]..m-1]为栈 1。本算法将 x 推入栈 i(=0,1)。
// 若入口时栈满，则不改变栈且返回 overflow 为 TRUE。
```

**例 4** 第 3.30 题的出列 del_queue 算法

```
bool dequeue (ElemType &x)
// 循环队列 squeue[(m+rear-quelen+1) MOD m.(m).rear]之上的元素出列操
// 操作。quelen 为队列长度。当 quelen=0 时，返回 FALSE；否则返回 TRUE，
// 且用 x 返回出列的元素值。
```

# 第 1 章　绪论(预备知识)

## 一、基本内容

数据、数据元素、数据对象、数据结构、存储结构和数据类型等概念术语的确定含义；抽象数据类型的定义、表示和实现方法；描述算法的类 C 语言；算法设计的基本要求以及从时间和空间角度分析算法的方法。

## 二、学习要点

1. 熟悉各名词、术语的含义，掌握基本概念，特别是数据的逻辑结构和存储结构之间的关系。分清哪些是逻辑结构的性质，哪些是存储结构的性质。

2. 了解抽象数据类型的定义、表示和实现方法。

3. 熟悉类 C 语言的书写规范，特别要注意值调用和引用调用的区别，输入、输出的方式以及错误处理方式。

4. 理解算法五个要素的确切含义：①动态有穷性(能执行结束)；②确定性(对于相同的输入执行相同的路径)；③有输入；④有输出；⑤可行性(用以描述算法的操作都是足够基本的)。

5. 掌握计算语句频度和估算算法时间复杂度的方法。

## 三、基础知识题

**1.1①**　简述下列术语：数据、数据元素、数据对象、数据结构、存储结构、数据类型和抽象数据类型。

**1.2②**　试描述数据结构和抽象数据类型的概念与程序设计语言中数据类型概念的区别。

◆**1.3②**　设有数据结构$(D,R)$，其中

$$D=\{d1,d2,d3,d4\},\ R=\{r\},\ r=\{(d1,d2),(d2,d3),(d3,d4)\}。$$

试按图论中图的画法惯例画出其逻辑结构图。

◆**1.4②**　试仿照三元组的抽象数据类型分别写出抽象数据类型复数和有理数的定义(有理数是其分子、分母均为自然数且分母不为零的分数)。

**1.5②**　试画出与下列程序段等价的框图。

```
(1) product=1;  i=1;
    while (i<=n) {
        product *=i;
        i++;
    }
```

```
(2) i=0;
    do {
       i++ ;
    } while ((i!=n) && (a[i]!=x));
(3) switch {
        case x<y: z=y-x; break;
        case x=y: z=abs(x * y); break;   // abs()为取绝对值函数
        default: z=(x-y)/abs(x) * abs(y);
    }
```

◆**1.6**③ 在程序设计中，常用下列三种不同的出错处理方式：

(1) 用 **exit** 语句终止执行并报告错误；

(2) 以函数的返回值区别正确返回或错误返回；

(3) 设置一个整型变量的函数参数以区别正确返回或某种错误返回。

试讨论这三种方法各自的优缺点。

◆**1.7**③ 在程序设计中，可采用下列三种方法实现输出和输入：

(1) 通过 **scanf** 和 **printf** 语句；

(2) 通过函数的参数显式传递；

(3) 通过全局变量隐式传递。

试讨论这三种方法的优缺点。

**1.8**④ 设 $n$ 为正整数。试确定下列各程序段中前置以记号@的语句的频度：

```
(1) i=1;  k=0;                          n-1
    while (i<=n-1) {
       @  k += 10 * i;
          i++;
    }
(2) i=1;  k=0;                          n
    do {
    @  k +=10 * i;
       i++;
    } while(i<=n-1);
(3) i = 1;  k = 0;                      n-1
    while (i<=n-1) {
          i++ ;
       @  k+= 10 * i;
    }
(4) k=0;                                n(n+1)/2
    for( i=1; i<=n; i++) {
        for (j=i ; j<=n; j++)
```

```
        @  k++;
    }
◆(5) for( i=1; i<=n; i++) {
        for (j=1; j<=i; j++) {
          for (k=1; k<=j; k++)
          @  x += delta;
    }
  (6) i=1;  j=0;
    while (i+j<=n) {
    @  if (i>j ) j++ ;
       else i++ ;
    }
◆(7) x=n;  y=0;  //n 是不小于 1 的常数
    while (x>=(y+1)*(y+1)) {
    @  y++;
    }
◆(8) x=91;  y=100;
    while (y>0 ) {
    @  if (x>100 ) { x -= 10;  y--; }
    else x++;
    }
```

**1.9**③ 假设 $n$ 为 2 的乘幂，并且 $n>2$，试求下列算法的时间复杂度及变量 count 的值(以 $n$ 的函数形式表示)。

```
int Time ( int n ) {
  count=0;  x=2;
  while (x<n/2 ) {
    x *=2;  count++;
  }
  return (count)
    } //Time
```

**1.10**② 按增长率由小至大的顺序排列下列各函数：

$2^{100}$，$(3/2)^n$，$(2/3)^n$，$(4/3)^n$，$n^n$，$n^{3/2}$，$n^{2/3}$，$\sqrt{n}$，$n!$，$n$，$\log_2 n$，$n/\log_n 2$，$\log_2^2 n$，$\log_2(\log_2 n)$，$n\log_2 n$，$n^{\log_2 n}$。

**1.11**③ 已知有实现同一功能的两个算法，其时间复杂度分别为 $O(2^n)$ 和 $O(n^{10})$，假设现实计算机可连续运算的时间为 $10^7$ 秒(100 多天)，又每秒可执行基本操作(根据这些操作来估算算法时间复杂度) $10^5$ 次。试问在此条件下，这两个算法可解问题的规模(即 $n$ 值的范围)各为多少？哪个算法更适宜？请说明理由。

**1.12③** 设有以下三个函数：

$f(n)=21n^4+n^2+1000$，　$g(n)=15n^4+500n^3$，　$h(n)=5000n^{3.5}+n\log n$

请判断以下断言正确与否：

(1) $f(n)$是$O(g(n))$

(2) $h(n)$是$O(f(n))$

(3) $g(n)$是$O(h(n))$

(4) $h(n)$是$O(n^{3.5})$

(5) $h(n)$是$O(n\log n)$

**1.13③** 试设定若干 $n$ 值，比较两函数 $n^2$ 和 $50n\log_2 n$ 的增长趋势，并确定 $n$ 在什么范围内，函数 $n^2$ 的值大于 $50n\log_2 n$ 的值。

**1.14③** 判断下列各对函数 $f(n)$和 $g(n)$，当 $n\to\infty$时，哪个函数增长更快？

(1) $f(n)=10^2+\ln(n!+10^{n^3})$　　$g(n)=2n^4+n+7$

(2) $f(n)=(\ln(n!)+5)^2$　　$g(n)=13n^{2.5}$

(3) $f(n)=n^{2.1}+\sqrt{n^4+1}$　　$g(n)=(\ln(n!))^2+n$

(4) $f(n)=2^{(n^3)}+(2^n)^2$　　$g(n)=n^{(n^2)}+n^5$

**1.15③** 试用数学归纳法证明：

(1) $\sum_{i=1}^{n} i^2 = n(n+1)(2n+1)/6$　　$(n\geqslant 0)$

(2) $\sum_{i=0}^{n} x^i = (x^{n+1}-1)/(x-1)$　　$(x\neq 1, n\geqslant 0)$

(3) $\sum_{i=1}^{n} 2^{i-1} = 2^n - 1$　　$(n\geqslant 1)$

(4) $\sum_{i=1}^{n} (2i-1) = n^2$　　$(n\geqslant 1)$

## 四、算法设计题

◆**1.16②** 试写一算法，自大至小依次输出顺序读入的三个整数 $X$，$Y$ 和 $Z$ 的值。

**1.17③** 已知 $k$ 阶裴波那契序列的定义为

$$f_0=0,\quad f_1=0,\quad \cdots,\quad f_{k-2}=0,\quad f_{k-1}=1;$$

$$f_n=f_{n-1}+f_{n-2}+\cdots+f_{n-k},\quad n=k,k+1,\cdots$$

试编写求 $k$ 阶裴波那契序列的第 $m$ 项值的函数算法，$k$ 和 $m$ 均以值调用的形式在函数参数表中出现。

**1.18③** 假设有 A,B,C,D,E 五个高等院校进行田径对抗赛，各院校的单项成绩均已存入计算机，并构成一张表，表中每一行的形式为

| 项目名称 | 性　别 | 校　名 | 成　绩 | 得　分 |
|---|---|---|---|---|

编写算法，处理上述表格，以统计各院校的男、女总分和团体总分，并输出。

◆**1.19④** 试编写算法，计算$i!\cdot 2^i$的值并存入数组 a [0..arrsize−1]的第 $i-1$ 个分量

中 ($i=1,2,\cdots,n$)。假设计算机中允许的整数最大值为 $maxint$，则当 $n>arrsize$ 或对某个 $k(1\leqslant k\leqslant n)$使 $k!\cdot 2^k>maxint$ 时，应按出错处理。注意选择你认为较好的出错处理方法。

◆ **1.20④** 试编写算法求一元多项式的值 $P_n(x)=\sum_{i=0}^{n}a_ix^i$ 的值 $P_n(x_0)$，并确定算法中每一语句的执行次数和整个算法的时间复杂度。注意选择你认为较好的输入和输出方法。本题的输入为 $a_i(i=0,1,\cdots,n)$，$x_0$ 和 $n$，输出为 $P_n(x_0)$。