计算机科学丛书

# 程序设计基础

管纪文 著

科学出版社

计算机科学丛书

# 程序设计基础

管纪文 著



科学出版社
1985

## 内 容 简 介

本书阐述程序设计的基本理论和技术. 内容包括: 算法的设计及其分析; 汇编语言 MIXAL; 子程序, 共行程序, 解释程序, MIX 模拟程序, 跟踪程序和输入输出; 还有浮点算术, 高精度算术, 有理数算术, 多项式算术和幂级数运算等等.

本书可作为计算机软件和数学专业的教师、研究生和大学生的参考书, 也可供有关专业的科技工作者参考.



计算机科学丛书
程序设计艺术
管纪文 著
责任编辑 杨家福 那莉莉
科学出版社出版
北京朝阳门内大街137号
中国科学院印刷厂印刷
新华书店北京发行所发行 各地新华书店经售

1985年5月第一版 开本: 850×1168 1/32
1985年5月第一次印刷 印张: 10 1/2
印数: 00001—28,000 字数: 276,000

统一书号: 15031·650
本社书号: 3952·15—8

定价: 3.00 元

# 计算机科学丛书

## 编 委 会

# 前　言

在计算机科学中，程序设计日益显示其重要性．本书阐述程序设计的基本理论和技术．

计算机与数学有深刻的联系．从一组基本的指令来编制一个计算机程序，非常类似于从一组公理来构造一个数学证明．至于计算机与数学中的数值分析、数论、概率统计诸方面的联系，则是众所周知的．总之，掌握程序设计的基础首先要求具有坚实的数学基础．本书对于数学基础的要求是严格的．书中将用严格而强有力的数学工具，来精确描述和深入分析计算机算法，阐述程序设计的基本理论和技术，同时，也阐述应用程序设计的基本理论和技术来设计程序．

在进行程序设计或描述计算机算法时，需要有形式的精确的语言，可以采用代数语言，如 ALGOL 或 FORTRAN 等等，也可以采用机器语言．这里，我们赞同 Knuth D. E. 的意见，采用机器语言．一是因为前者更适合于数值问题，后者更适合于非数值问题，而本书则主要讨论非数值问题．所谓非数值问题，就是逻辑判断处于主要地位，而算术运算相对地处于次要地位的问题．例如分类、检索、翻译语言、证明定理、智能模拟这样一些问题．二是因为机器语言更有效，更接近实际．三是因为这里讲的程序都是精心编制的，所以用机器语言编写都不长，也不难了解．四是因为机器语言本身也是需要的，有时甚至是必要的．诚然，高级语言更易于编写和校正，但是实际的机器语言则效率更高，而本书所讨论的程序都是最基本、最常用的，所以应特别强调效率．

既然采用了机器语言，那么采用哪一种机器语言呢？采用130机的语言是不行的．因为这样会把本书局限于100系列机，而130机本身也存在进一步发展的问题．所以我们采用 Knuth 的 MIX

机．这是一种“超通用”计算机，即抽象的计算机，它具有现存计算机的共同性质，MIX 程序可为大多数实际的计算机所接受，或在大多数计算机上进行模拟．要模拟执行 MIX 程序，就需要有一个 MIX 模拟程序．比方说要在 130 机上模拟执行 MIX 程序，就需要有一个用 130 机器语言编写的 MIX 模拟程序．本书要讲一个用 MIX 机器语言本身编写的 MIX 模拟程序，读者可以参照着来编写自己所用的具体计算机的 MIX 模拟程序．MIX 可为当今的通用计算机所通用，故称之为超通用计算机．

本书共分四章：第一章介绍算法，包括算法的概念、例子和分析等．第二章介绍超通用 MIX 机，包括 MIX 指令系统和汇编语言 MIXAL．第三章介绍程序，包括子程序、共行程序、解释程序、MIX 模拟程序、跟踪程序、输入和输出等．第四章介绍算术，包括浮点算术、高精度算术、进制转换、有理数算术、多项式算术、幂级数操作等等．

本书的原稿曾作为讲义讲授过，在此基础上，根据计算机科学丛书的宗旨，北京大学的杨芙清和邹悦两同志审阅了全稿，并提出了很多宝贵的意见，据此，作者对原稿进行了较大的修改．作者谨向这两位同志表示衷心的谢意．

限于水平，书中的错误在所难免，尚请读者批评指正．

# 目　录

# 第一章 算 法

## 1.1 算法的概念

设计计算机程序，就是要在计算机上实现某种算法．算法就是用自然语言编写的程序，程序就是用计算机所能接受的语言编写的算法．因此，对于计算机程序设计而言，最基本的是算法，所以我们先讲算法．最著名的是 Euclid 算法．

**算法 E** (Euclid 算法) 给定两个正整数 $m$ 和 $n$，要求它们的最大公约数(即同时整除 $m$ 和 $n$ 两数的最大正整数)．

**E1**. [求余数]以 $n$ 除 $m$ 并令 $r$ 为所得余数 $(0 \leqslant r < n)$。

**E2**. [余数为 0?] 若 $r = 0$，则本算法结束；$n$ 是答案。

**E3**. [互换]置 $m \leftarrow n$，$n \leftarrow r$，并返回步骤 E1。

这里，我们来解释算法的写法，也就是来约定算法的书写格式．

1.算法都有一个标识字母，如 Euclid 算法的 E．今后引用这个算法时，就写做算法 E，而算法中的每一步骤则记成 E1，E2，等等．

2.算法每一步骤开头都有带方括号的短句，综述该步的主要内容．这样就可以直接得出一个框图，来帮助直观地了解这个算法．例如图 1.1 就是算法 E 的框图．

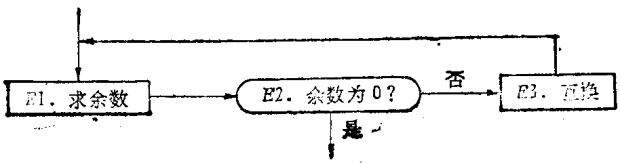


图 1.1 算法 E 的框图

框图的画法中，方框表示执行框，圆框表示判断框。

3. 算法的每一步骤在带方括号的短句之后，说明该步骤所要执行的动作或所要进行的判断，有时还有带圆括号的注释，但这种注释只为阅读算法提供方便，完全不影响该步骤的动作。

4. 步骤 E3 中的"$\leftarrow$"是最重要的替代或赋值运算. "$m \leftarrow n$"表示变量 $m$ 的值换成了变量 $n$ 的当前值，也就是以变量 $n$ 的当前值赋给变量 $m$. 例如，$n$ 增加 1 的运算就可以表示成"$n \leftarrow n+1$"(注意，不要写成"$n \rightarrow n+1$"). 注意步骤 E3 中动作的次序很重要，"置 $m \leftarrow n$, $n \leftarrow r$"完全不同于"置 $n \leftarrow r$, $m \leftarrow n$". 后者根本就是"置 $n \leftarrow r$, $m \leftarrow r$". 今后，我们约定把"$n \leftarrow r$, $m \leftarrow r$"写成"$n \leftarrow m \leftarrow r$". 而交换两个变量的值则写成"交换 $m \longleftrightarrow n$"，这等于"置 $t \leftarrow m$, $m \leftarrow n$, $n \leftarrow t$".

5. 算法从步骤 1 开始，然后顺序地执行. 如算法 E，执行到 E3，即返回执行 E1. 在步骤 E2，条件"若 $r=0$"是执行该步骤动作的前提. 言下之意，若 $r \neq 0$，则执行下一步骤，即 E3.

6. ▮表示本算法已写完.

7. 今后在算法中，我们还把带下标的量，例如 $v_j$ 和 $a_{ij}$ 分别记成 $v[j]$ 和 $a[i, j]$. 我们也用 PRIME [$K$] 来表示第 $K$ 个质数，等等.

以上只讲了算法的书写格式，那么怎样来了解一个算法呢？唯一的办法就是通过实例进行试验，然后才能有理性认识，从理论上证明其正确性. 例如算法 E，我们试以 $m=119$, $n=544$ 为例试验，作辗转相除如表 1.1.

总而言之，算法就是一个解决问题的办法，更确切地说，就是一组(有限个)规则，它们给出了一系列解决特定问题的操作. 算法有五大特性：

1. 有限性. 算法必须在执行有限步后结束. 例如算法 E，E1 中 $r<n$，故若 $r \neq 0$，则经 E3 置 $n \leftarrow r$ 后，$n$ 已经减小. 所以对于任给的初始值 $n$，E1 中的 $n$ 是逐次减小的，而正整数的严格递减序列必然终止于有限步.

表 1.1　算法 E 的工作例子

| E | $\frac{m}{n}$ | $m$ | $n$ | $r$ |
|---|---|---|---|---|
| | | 119 | 544 | |
| E1 | $\frac{119}{544}$ | | | 119 |
| E2 | | | | |
| E3 | | 544 | 119 | |
| E1 | $\frac{544}{119}=4\frac{68}{119}$ | | | 68 |
| E2 | | | | |
| E3 | | 119 | 68 | |
| E1 | $\frac{119}{68}=1\frac{51}{68}$ | | | 51 |
| E2 | | | | |
| E3 | | 68 | 51 | |
| E1 | $\frac{68}{51}=1\frac{17}{51}$ | | | 17 |
| E2 | | | | |
| E3 | | 51 | 17 | |
| E1 | $\frac{51}{17}=3$ | | (119,544) = 17 | 0 |
| E2 | | | | |

2. 确定性. 算法的每一步骤都是确定的，待执行的动作对每一情况都有严格确切的规定. 以后我们还要用形式定义的计算机语言来书写算法，以避免用自然语言书写时可能带来的二义性. 例如算法 E，在执行 E1 时，就必须确切地知道如何做 $n$ 除 $m$ 并求余数 $r$，所以就必须保证 $m$ 和 $n$ 总是正整数. 事实上，这在开始时已假定如此，到下次执行 E1 时，必从 E3 转来，但 E3 中的 $m \leftarrow n$ $(n \neq 0)$，$n \leftarrow r$，$r \neq 0$. 可见 E1 中的 $m, n$ 总是正整数.

3. 输入. 算法有 0 个或多个输入，即在算法开始之前给予算法的量. 这些输入取自确定的对象集合. 例如算法 E 有两个输入 $m$ 和 $n$，均取自正整数集合.

4. 输出. 算法有一个或多个输出，即与输入有确定关系的量. 例如算法 E 有一个输出，E2 的 $n$，是两个输入的最大公约数. 事实上，经步骤 E1 后，

$$m = qn + r,\ q\text{整} \geqslant 0,\ 0 \leqslant r < n$$

若 $r = 0$，则 $n \mid m$，$n = \gcd(m, n)$，即E2的 $n$ 是所求的最大公约数．若 $r \neq 0$，则由上式知

$m, n$ 之公约数 $\mid r, n$

$r, n$ 之公约数 $\mid m, n$

亦即

$m, n$ 之公约数是 $r, n$ 之公约数

$r, n$ 之公约数是 $m, n$ 之公约数

或

$m, n$ 之公约数集合 $= r, n$ 之公约数集合

特别

$m, n$ 之最大公约数 $= r, n$ 之最大公约数

因此，E3并不改变原问题的答案．这样，由E3返回E1，经E1后又得到

$$m = qn + r,\ q\text{整} \geqslant 0,\ 0 \leqslant r < n$$

如此经有限次循环后，必得 $r = 0$，算法最终到E2结束，而得 $n = \gcd(m, n)$．总之，E2的 $n$ 是两个输入的最大公约数．

5．能行性．算法中有待实施的操作都十分基本，用手工也能在有限长的时间内做完．例如算法E用到两正整数相除，测试一整数是否为零，赋值，这些都是能行的．非能行操作的例子，例如要计算不尽小数，要测试“$x^n + y^n = z^n (n \geqslant 2)$ 有无正整数解”等等．

这就是算法的五大特性．这里要重新回过头来说一下有限性．如果这组规则具有五大特性中的其它四大特性，唯独没有有限性，则称为计算方法．即使是算法，其有限性也有强弱的问题．就实用而言，我们不仅要求算法结束于有限的步数，而且还要求很有限的步数，我们需要好的算法，即省时间、速度快的算法，这就需要进行分析，计算一算法的运行速度．算法的分析后面再讨论．

前面只对算法做了非形式的描述．现在我们来给出算法的形式定义．

**定义 1** 一个计算方法就是一个自动机，即四元组 $\mathscr{A}=\langle S, X, Y, f\rangle$，其中 $S$，$X$ 和 $Y$ 分别是计算的状态集合、输入集合、输出集合，$X$ 和 $Y$ 都是 $S$ 的子集，而 $f$ 是 $S$ 的变换(计算规则)，$f$ 不变 $Y$ 中的每个元素，即 $f(y)=y$，对于 $y\in Y$.

集合 $X$ 中的每个输入 $x$ 确定一个**计算序列** $x_0, x_1, x_2, \cdots$ 如下：

$$x_0=x, \text{ 而 } x_{k+1}=f(x_k) \text{ 对于 } k\geqslant 0$$

此计算序列**终止于 $k$ 步**，如果 $k$ 是使 $x_k\in Y$ 的最小正整数. 这时就说**输入 $x$ 产生输出 $x_k$** (于是 $x_k=x_{k+1}=x_{k+2}\cdots\in Y$).

一个计算方法说是一个**算法**，如果对于所有的 $x\in X$，$x$ 所确定的计算序列都终止于有限步.

【例 1】 对于算法 E，$X=\{(m, n)\mid m, n, \text{正整数}\}$，$Y=\{(n)\mid n \text{ 正整数}\}$，$S=X\cup Y\cup\{(m, n, r, \mathrm{E1}), (m, n, r, \mathrm{E2}), (m, n, p, \mathrm{E3})\mid m, n, p \text{ 正整数}, r \text{ 非负整数}\}$. 而 $f$ 应定义为

$f(m, n)=(m, n, 0, \mathrm{E1})\cdots$表示算法 E 以初始值 $m, n$ 进入 E1

$f(m, n, r, \mathrm{E1})=\left(m, n, \dfrac{m}{n} \text{ 之余数}, \mathrm{E2}\right)\cdots$执行 E1 后进入 E2

$$f(m, n, r, \mathrm{E2})=\left\{\begin{matrix}(n) & \text{若 } r=0\\ (m, n, r, \mathrm{E3}) & \text{若 } r\neq 0\end{matrix}\right\}\cdots\text{执行 E2}$$

$f(n)=(n)\cdots$E2 中的 $n$ 是答案，若 $r=0$

$f(m, n, p, \mathrm{E3})=(n, p, p, \mathrm{E1})\cdots$执行 E3：置 $m\leftarrow n$，$n\leftarrow p$，返 E1

前面给出的形式定义太广了，它已不具备“能行性”特性. 这里的 $S$ 中可以有手工无法计算的无穷序列，$f$ 也可能包括手工不能实施的操作. 为了达到“能行性”要求，可以对自动机 $\mathscr{A}$ 再加上种种限制. 例如 A. A. Марков **算法**可以定义如下：$\mathscr{A}=(S_N, X, Y, f)$，$N$ 是非负整数，$S_N=\{(\alpha, j)\mid \alpha\in A^*, 0\leqslant j\leqslant N\}$，其中 $A$ 是有限的字母集合，$A^*$ 表示 $A$ 上所有的串 $\alpha=a_1a_2\cdots a_n$ ($n\geqslant 0$，诸 $a_i\in A$) 的集合；$X=\{(\alpha, 0)\mid \alpha\in A^*\}$，$Y=\{(\alpha, N)\mid \alpha\in A^*\}$. 通过两组串 $\theta_j$，$\varphi_j$ 和两组非负整数 $j'$，$j''(0\leqslant j', j''\leqslant$

$N$)，其中 $j=0,1,\cdots,N-1$，定义 $f$ 如下：对于 $j=0,1,\cdots,N-1$，

$$f(\alpha,j)=\begin{cases}(\alpha,j') & \text{若 } \theta_j \text{ 不出现于 } \alpha\cdots\text{执行步骤 } j \text{ 后，转步骤 } j'\\ (\alpha[\theta_j\leftarrow\varphi_j],j'') & \text{其中 } \alpha[\theta_j\leftarrow\varphi_j] \text{ 表示以 } \varphi_j \text{ 去替换 } \theta_j \text{ 在 } \alpha \text{ 中的最早出现，若 } \theta_j \text{ 出现于 } \alpha\cdots\text{执行步骤 } j \text{ 后转步骤 } j''\end{cases}$$

而

$$f(\alpha,N)=(\alpha,N)\cdots\text{输出 }(\alpha,N),\text{结束于步骤 } N$$

如此定义的计算方法是能行的，因为这种替换能用手工实行，而且 $S_N$ 中的每个元素也能手工计算．经验证明这也足够广了，已经包括了一般的能行算法．

此外，还可以利用 Turing 机来形式化能行的计算方法．

## 1.2 算法举例

本节要列举一些基本的算法，计有数学归纳法、扩充算法 E、质数算法、置换的算法等等．

数学归纳法原理可以写成一个作证明的算法．下面我们举例说明如何用它来证明一个算法是有效的．但我们并不证明它本身，所以特标明为原理．

**算法 I**（数学归纳法原理） 给定一正整数 $n$，本算法将输出命题 $P(n)$ 为真的一个证明．

I1. [$k\leftarrow 1$ 证 $P(k)$] 置 $k\leftarrow 1$，输出 $P(1)$ 的一个证明．

I2. [$k=n$?] 若 $k=n$，则结束本算法；所求的证明已输出．

I3. [证 $P(k+1)$] 输出“若 $P(1),\cdots,P(k)$ 皆真，则 $P(k+1)$ 为真”之一证明．并输出“既已证明了 $P(1),\cdots,P(k)$；故 $P(k+1)$ 为真”．

I4. [$k$ 增 1] $k$ 加 1 并转步骤 I2. ∎

I3 中的第二个输出是由第一个输出直接推出来的．

算法 I 的框图如图 1.2 所示．

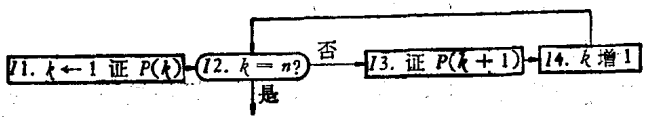


图 1.2 算法 I：数学归纳法

我们不妨以 $n = 5$ 为例，来看一看算法 I 的工作过程，见表 1.2。

表 1.2 算法 I 的工作实例

| | |
|---|---|
| I1 | $k \leftarrow 1$，$P(1)$ |
| I2 | $1 < n$ |
| I3 | “若 $P(1)$，则 $P(2)$”；“既 $P(1)$，故 $P(2)$” |
| I4 | $k \leftarrow 2$ |
| I2 | $2 < n$ |
| I3 | “若 $P(1)$，$P(2)$，则 $P(3)$”；“既 $P(1)$，$P(2)$，故 (3)” |
| I4 | $k \leftarrow 3$ |
| ⋮ | ⋮ |
| I2 | $4 < n$ |
| I3 | “若 $P(1)$，$P(2)$，$P(3)$，$P(4)$，则 $P(5)$”；“既 $P(1)$，$P(2)$，$P(3)$，$P(4)$，故 $P(5)$” |
| I4 | $k \leftarrow 5$ |
| I2 | $k = 5$，前已有 $P(5)$，这 $P(5)$ 来自前面的“既 $P(1)$，$P(2)$，$P(3)$，$P(4)$，故 $P(5)$” |

实施算法 I，关键在于：

1. 证明“$P(1)$”（I1 的输出）；

2. 证明“对于任意的正整数 $k$，若 $P(1), P(2), P(3), \cdots, P(k)$ 皆真，则 $P(k+1)$ 为真”（I3 的第一个输出）。

算法 E 可以扩充如下：

**算法 E**（扩充的 Euclid 算法） 给定两个正整数 $m$ 和 $n$，要求其最大公约数 $d$ 以及两整数 $a$ 和 $b$，使得 $am + bn = d$。

E1. [初始化] 置 $a' \leftarrow b \leftarrow 1$，$a \leftarrow b' \leftarrow 0$，$c \leftarrow m$，$d \leftarrow n$。

**E2.** [除]命 $q$，$r$ 分别是 $c$ 除以 $d$ 之商和余数（$c = qd + r$，$0 \leqslant r < d$）.

**E3.** [余数为 0?] 如果 $r = 0$，则本算法结束；这时已求得 $am + bn = d$.

**E4.** [再循环]置 $c \leftarrow d$，$d \leftarrow r$，$t \leftarrow a'$，$a' \leftarrow a$，$a \leftarrow t - qa$，$t \leftarrow b'$，$b' \leftarrow b$，$b \leftarrow t - qb$，并返回 E2. ∎

本算法共用了九个变量：$a, b, a', b', c, d, q, r, t$. 只要从本算法中删去变量 $a, b, a'$ 和 $b'$，并仍用 $m, n$ 来代替 $c, d$，就得到 1.1 节中的算法 E. 如此为本算法添枝加叶，无非是还要确定系数 $a, b$. 为便于了解，我们把新扩充的部分都用黑体字标出.

先以 $m = 1769$，$n = 551$ 为例来运行算法 E，如表 1.3，我们看到：

$$am + bn = 5 \times 1769 - 16 \times 551 = 29,\quad d = 29$$

答案是对的，即如此求得之 $a, b$，确使 $am + bn = d$，其中 $d = \gcd(m, n)$，即 $m$ 和 $n$ 的最大公约数. 这在 1.1 节算法 E 的讨论中已经证明.

**表 1.3 算法 E 的工作实例**

| | $a'$ | $a$ | $b'$ | $b$ | $\frac{c}{d}$ | $c$ | $d$ | $q$ | $r$ |
|---|---|---|---|---|---|---|---|---|---|
| E1 | 1 | 0 | 0 | 1 | | 1769 | 551 | | |
| E2 | | | | | $\frac{1769}{551} = 3\,\frac{116}{551}$ | | | 3 | 116 |
| E3 | | | | | | | | | |
| E4 | 0 | 1 | 1 | −3 | | 551 | 116 | | |
| E2 | | | | | $\frac{551}{116} = 4\,\frac{87}{116}$ | | | 4 | 87 |
| E3 | | | | | | | | | |
| E4 | 1 | −4 | −3 | 13 | | 116 | 87 | | |
| E2 | | | | | $\frac{116}{87} = 1\,\frac{29}{87}$ | | | 1 | 29 |
| E3 | | | | | | | | | |
| E4 | −4 | 5 | 13 | −16 | | 87 | 29 | | |
| E2 | | | | | $\frac{87}{29} = 3$ | | | 3 | 0 |
| E3 | | | | | | | | | |

现在，我们需要证明本算法E是有效的，即对于任何正整数$m$和$n$，本算法所求得的$a$和$b$确使$am+bn=d=\gcd(m,n)$．我们用算法I来证明．

首先我们证明，每当进入E2时，总有

$$a'm+b'n=c \tag{1}$$

$$am+bn=d \tag{2}$$

为使用算法I，命

$P(k)$＝“在第$k$次进入E2时，有(1)，(2)两式成立．”

1.证明“$P(1)$”，即在第一次进入E2时，有(1)，(2)两式成立．事实上，在第一次进入E2时，由E1，$a'\leftarrow b\leftarrow 1$，$a\leftarrow b'\leftarrow 0$，$c\leftarrow m$，$d\leftarrow n$．故有

$$a'm+b'n=1m+0n=m=c$$

$$am+bn=0m+1n=n=d$$

2.证明：对于任意的正整数$k$，若$P(1),P(2),\cdots,P(k)$皆真，则$P(k+1)$为真．即，对于任意正整数$k$，设在第1，2…，$k$次进入E2时，有(1)，(2)两式成立，要证在第$k+1$次进入E2时，有(1)，(2)两式成立，事实上，在第$k+1$次进入E2时，必从E4转来，不难说明E4并不改变(1)，(2)两式的正确性．因为，既设在第$k$次进入E2时，已有(1)，(2)两式成立．兹经E4后，$c\leftarrow d$，$d\leftarrow r$，$t\leftarrow a'$，$a'\leftarrow a$，$a\leftarrow t-qa$，$t\leftarrow b'$，$b'\leftarrow b$，$b\leftarrow t-qb$，所以

$$\bar{a}'m+\bar{b}'n=am+bn\overset{(2)}{=}d=\bar{c}$$

即(1)式成立；又

$$\bar{a}m+\bar{b}n=(a'-qa)m+(b'-qb)n=(a'm+b'n)$$

$$-q(am+bn)\overset{(1)}{\underset{(2)}{=}}c-qd\overset{\text{E2}}{=}r=\bar{d}$$

即(2)式成立(其中$\bar{a}',\bar{b}',\bar{a},\bar{b},\bar{c},\bar{d}$分别是经E4后且进入E2时的$a',b',a,b,c,d$的新值)．

于是，用算法I我们证明了每当进入E2时，总有(1)，(2)两

式成立．由此推知，算法E结束时必有 $am + bn = d$，因为E2和E3都不改变 $a$, $b$, $m$, $n$, $d$ 诸值．

现在我们再用算法I一举证明本算法E是有效的．今既已证得算法E结束时，必有 $am + bn = d$，所以只要证明算法E结束时，$d = \gcd(m, n)$．为此命

$P(n)$="对 $n$ 和所有正整数 $m$，算法E结束时 $d=\gcd(m, n)$"

1. 证明"$P(1)$"，即当 $n = 1$ 时，对所有 $m$，算法E结束时 $d = \gcd(m, 1)$．事实上，因 $n = 1$，必 $n \mid m$．故算法E在第一次的E3结束，而 $d = n = 1 = \gcd(m, 1)$．

2. 证明：对于任意的正整数 $n>1$，设每当 $1 \leqslant k < n$ 时 $P(k)$ 皆真，则 $P(n)$ 为真．即，对于任意的整数 $n > 1$，设每当 $1 \leqslant k < n$ 时，对于 $k$ 和所有的正整数 $m$，算法E结束时 $d = \gcd(m, k)$，要证对于 $n$ 和所有的正整数 $m$，算法E结束时 $d = \gcd(m, n)$．

从第1个证明中看到，如果 $n \mid m$，则算法E在第一次的E3结束，而 $d = n = \gcd(m, n)$．所以剩下只须考虑 $n \nmid m$ 的情况．这时，经第一次执行E4后，$c \leftarrow d \leftarrow n$，$d \leftarrow r$，$r < n$，按归纳法假设，即前面黑体字的那段话，对于 $r$ 和 $n$，算法E结束时，$d = \gcd(n, r)$．同时经第一次执行E之E2后有 $m = qn + r$，按1.1节所证，$\gcd(n, r) = \gcd(m, n)$．所以证明了算法E结束时 $d = \gcd(m, n)$，即 $P(n)$ 为真．

总之，两次应用算法I，我们证明了算法E是有效的．

在1.1节关于算法之输出的讨论中，我们已证明了算法E结束时的 $d$ 是最大公约数，这里则用算法I作出严格的证明．

下一个例子还是数论方面的，即枚举质数的算法．要设计一算法，打印出前500个质数，排成50行10列的一张表，即每列打50个质数，共打成10列．具体地说，此表有如下形式：

FIRST FIVE HUNDRED PRIMES

0002 0233 0547 0877 1229 1597 1993 2371 2749 3187
0003 0239 0557 0881 1231 1601 1997 2377 2753 3191
0005 0241 0563 0883 1237 1607 1999 2381 2767 3203