第7章　简单算法

“程序＝算法＋数据结构”。在这一章中，我们将介绍简单的算法。早在公元前1世纪的《周髀算经》中，“算法”的概念已经有所提及。经过长时间的发展，算法艺术可谓是凝结了人类智慧的精华。只有有了算法这个灵魂，我们的程序才会有强大的力量。接下来，我们将通过“高精度处理”、“枚举”、“模拟”、“简单动态规划”、“递归回溯”这几个算法体会一下算法的奥妙。

7.1　什么是算法

算法是解题方案的准确而完整的描述，是一系列解决问题的清晰指令，算法代表着用系统的方法描述解决问题的策略机制。也就是说，算法就是解决问题的办法，能够对一定规范的输入，在有限时间内获得所要求的输出。常见的算法有递推法、递归法、穷举法、贪心算法、分治法、动态规划法、回溯法等。

算法具有五个特征：

（1）有穷性：一个算法必须能够对任何合法的输入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）确切性：算法中每一条指令必须有确切的含义，读者理解时不会产生歧义。并且，在任何条件下，算法只有唯一的一条执行路径，即对于相同的输入只能得出相同的输出。

（3）可行性：算法中描述的操作都是可以通过已经实现的基本操作执行有限次来实现。

（4）输入：一个算法有零个或多个输入，这些输入取自于某个特定的对象的集合。

（5）输出：一个算法有一个或多个输出，这些输出是与输入有着某些特定关系的量。

接下来我们会通过介绍秦九韶算法的中心思想与实现方法，来对这五个特征加以直观体会。

秦九韶算法是中国南宋时期的数学家秦九韶提出的一种多项式简化算法。

对于一个一元 n 次多项式

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

我们在代入一个特定的 x 进行 f(x) 整体求值时，若用朴素方法处理，则需要经过 $\frac{n(n+1)}{2}$ 次乘法操作和 n 次加法操作。而使用秦九韶算法计算，通过去除冗余的计算步骤，只需 n 次乘法操作和 n 次加法操作就可以计算出 f(x)。

它的中心思想是，将上面的 n 次多项式改写成以下形式：

$$\begin{aligned} f(x) &= a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 \\ &= (a_n x^{n-1} + a_{n-1} x^{n-2} + \ldots + a_1) x + a_0 \\ &= ((a_n x^{n-2} + a_{n-1} x^{n-3} + \ldots + a_2) x + a_1) x + a_0 \\ &= \ldots \\ &= ((\ldots(a_n x + a_{n-1}) x + a_{n-2}) x + \ldots) x + a_1) x + a_0 \end{aligned}$$

用朴素方法处理时，我们在计算 x，x^2，…，x^{n-1}，x^n 等 x 的次幂时会造成很多的无效操作，而秦九韶算法通过多项式的层层嵌套，巧妙地避开了不必要的计算，从而减少计算操作的次数。秦九韶算法描述如下：

第一步：输入 f(x) 的系数 a_0，a_1，…，a_n，输入 x 的值。

第二步：答案 Ans 初始化为 a_n。

第三步：对于 i=1，2，…，n，循环执行 Ans=Ans*x+a_{n-i}。

第四步：输出 Ans。

它的算法流程图如图 7.1 所示。

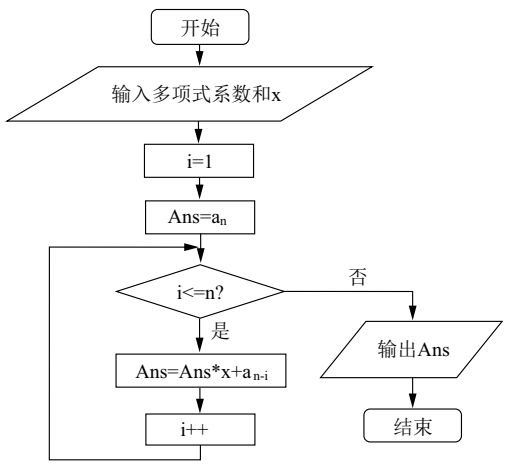

图 7.1　秦九韶算法流程图

算法是程序设计的核心，是灵魂，有了算法就可以结合数据结构转变成程序。秦九韶算法对应的程序如下：

```
#include <iostream>
using namespace std;
int n,a[20],x,ans;
int main()
{
    cin>>n;
    for(int i=0;i<=n;i++)cin>>a[i];
    cin>>x;
    ans=a[n];
    for(int i=1;i<=n;i++)ans=ans*x+a[n-i];
    cout<<ans<<endl;
    return 0;
}
```

7.2 高精度数值处理

高精度数值处理是采用模拟算法对位数达上百位甚至更多位数的数字进行各种运算，包括加法、减法、乘法、除法等基础运算。

在C++中，数值的加、减、乘、除运算都已经在系统内部被定义好了，我们可以很方便地对两个变量进行简单的运算。然而其中变量的取值范围，以整数为例，最大的是long long类型，范围是$[-2^{63},2^{63})$。假如我们要对两个范围更大，比如在$[-2^{1000},2^{1000})$范围内的数进行简单基础运算，就不能用C++的内部运算器了。同理，在使用实数类型计算时，由于精确度有限，也不能精确计算小数点后的数百位的数值。那么我们该如何对两个大整数进行运算呢？

回想一下我们小学数学课上的加法“竖式”运算。

首先把加数与被加数的个位对齐，然后个位对个位、十位对十位、百位对百位，位位对应进行加法操作，有进位的要相应地进行处理。

对应地，我们不妨对每一个数位开一个整数变量进行存储。那么两个位分别相加，进位这些问题我们都可以用程序表示出来。而在实践中，对进行运算的两个数分别用两个数组进行储存会使问题变得十分方便。

以上便是对“高精度加法”算法思路的简单描述。

如图7.2所示，我们要计算出537+543的值，不妨设数组A={7,3,5}，