合(least squares fit)，我们发现，psum1 和 psum2 的运行时间(用时钟周期为单位)分别近似于等式 368+9.0n 和 368+6.0n。这两个等式表明对代码计时和初始化过程、准备循环以及完成过程的开销为 368 个周期加上每个元素 6.0 或 9.0 周期的线性因子。对于较大的 n 的值(比如说大于 200)，运行时间就会主要由线性因子来决定。这些项中的系数称为每元素的周期数(简称 CPE)的有效值。注意，我们更愿意用每个元素的周期数而不是每次循环的周期数来度量，这是因为像循环展开这样的技术使得我们能够用较少的循环完成计算，而我们最终关心的是，对于给定的向量长度，程序运行的速度如何。我们将精力集中在减小计算的 CPE 上。根据这种度量标准，psum2 的 CPE 为 6.0，优于 CPE 为 9.0 的 psum1。

旁注　什么是最小二乘拟合

对于一个数据点(x_1, y_1)，…，(x_n, y_n)的集合，我们常常试图画一条线，它能最接近于这些数据代表的 X−Y 趋势。使用最小二乘拟合，寻找一条形如 $y=mx+b$ 的线，使得下面这个误差度量最小：

$$E(m,b) = \sum_{i=1,n} (mx_i + b - y_i)^2$$

将 $E(m, b)$分别对 m 和 b 求导，把两个导数函数设置为 0，进行推导就能得出计算 m 和 b 的算法。

练习题 5.2　在本章后面，我们会从一个函数开始，生成许多不同的变种，这些变种保持函数的行为，又具有不同的性能特性。对于其中三个变种，我们发现运行时间(以时钟周期为单位)可以用下面的函数近似地估计：

版本 1：60+35n

版本 2：136+4n

版本 3：157+1.25n

每个版本在 n 取什么值时是三个版本中最快的？记住，n 总是整数。

5.3　程序示例

为了说明一个抽象的程序是如何被系统地转换成更有效的代码的，我们将使用一个基于图 5-3 所示向量数据结构的运行示例。向量由两个内存块表示：头部和数据数组。头部是一个声明如下的结构：

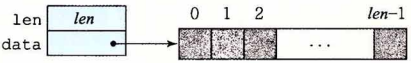

图 5-3　向量的抽象数据类型。向量由头信息加上指定长度的数组来表示

code/opt/vec.h

```
1  /* Create abstract data type for vector */
2  typedef struct {
3      long len;
4      data_t *data;
5  } vec_rec, *vec_ptr;
```

code/opt/vec.h

这个声明用 data_t 来表示基本元素的数据类型。在测试中，我们度量代码对于整数(C 语言的 int 和 long)和浮点数(C 语言的 float 和 double)数据的性能。为此，我们会分别为不同的类型声明编译和运行程序，就像下面这个例子对数据类型 long 一样：