我们说像 sumvec 这样顺序访问一个向量每个元素的函数，具有步长为 1 的引用模式(stride-1 reference pattern)(相对于元素的大小)。有时我们称步长为 1 的引用模式为顺序引用模式(sequential reference pattern)。一个连续向量中，每隔 k 个元素进行访问，就称为步长为 k 的引用模式(stride-k reference pattern)。步长为 1 的引用模式是程序中空间局部性常见和重要的来源。一般而言，随着步长的增加，空间局部性下降。

对于引用多维数组的程序来说，步长也是一个很重要的问题。例如，考虑图 6-18a 中的函数 sumarrayrows，它对一个二维数组的元素求和。双重嵌套循环按照行优先顺序(row-major order)读数组的元素。也就是，内层循环读第一行的元素，然后读第二行，依此类推。函数 sumarrayrows 具有良好的空间局部性，因为它按照数组被存储的行优先顺序来访问这个数组(图 6-18b)。其结果是得到一个很好的步长为 1 的引用模式，具有良好的空间局部性。

```
1  int sumarrayrows(int a[M][N])
2  {
3      int i, j, sum = 0;
4
5      for (i = 0; i < M; i++)
6          for (j = 0; j < N; j++)
7              sum += a[i][j];
8      return sum;
9  }
```

a) 另一个具有良好局部性的程序

地址	0	4	8	12	16	20
内容	a_{00}	a_{01}	a_{02}	a_{10}	a_{11}	a_{12}
访问顺序	1	2	3	4	5	6

b) 数组a的引用模式 ($M=2$, $N=3$)

图 6-18　有良好的空间局部性，是因为数组是按照与它存储在内存中一样的行优先顺序来被访问的

一些看上去很小的对程序的改动能够对它的局部性有很大的影响。例如，图 6-19a 中的函数 sumarraycols 计算的结果和图 6-18a 中函数 sumarrayrows 的一样。唯一的区别是我们交换了 i 和 j 的循环。这样交换循环对它的局部性有何影响？函数 sumarraycols 的空间局部性很差，因为它按照列顺序来扫描数组，而不是按照行顺序。因为 C 数组在内存中是按照行顺序来存放的，结果就得到步长为 N 的引用模式，如图 6-19b 所示。

```
1  int sumarraycols(int a[M][N])
2  {
3      int i, j, sum = 0;
4
5      for (j = 0; j < N; j++)
6          for (i = 0; i < M; i++)
7              sum += a[i][j];
8      return sum;
9  }
```

a) 一个空间局部性很差的程序

地址	0	4	8	12	16	20
内容	a_{00}	a_{01}	a_{02}	a_{10}	a_{11}	a_{12}
访问顺序	1	3	5	2	4	6

b) 数组a的引用模式 ($M=2$, $N=3$)

图 6-19　函数的空间局部性很差，这是因为它使用步长为 N 的引用模式来扫描

6.2.2　取指令的局部性

因为程序指令是存放在内存中的，CPU 必须取出(读出)这些指令，所以我们也能够评价一个程序关于取指令的局部性。例如，图 6-17 中 for 循环体里的指令是按照连续的内存顺序执行的，因此循环有良好的空间局部性。因为循环体会被执行多次，所以它也有很好的时间局部性。