为什么要使用数组

我们已经可以编写程序来解决许多问题了，但是当需要处理的数据比较多时，仅依靠前面的知识是不够的，即使简单的问题也可能需要很复杂的程序来处理。

例如：输入 50 个学生的课程成绩，输出低于平均分的学生序号与成绩。

可以通过一个变量来累加读入 50 个成绩求出学生的总分，进而求出平均分。

但因为只有读入最后一个学生的分数后才能求得平均分，并且要求打印出低于平均分的学生序号和成绩，所以必须把 50 个学生的成绩都保留起来，然后逐个和平均分比较，把低于平均分的成绩打印出来。

所以要用 50 个变量 a_1，a_2,…，a_{50} 存储这些数据：

```
1. cin>>a1>>a2>>…>>a10>>…>>a41>>a42>>…>>a50;
```

如果可以把大量相同性质的数据组合成一个新类型的变量，就可以用简单的程序（比如循环 50 次）对这个新变量的各个分量进行相同的处理。

第一个数组程序

```cpp
#include<iostream>
using namespace std;
int main()
{
    //声明一个大小为 50 的整型数组，存储每个学生的成绩
    int a[50];
    //存储 50 个学生的总分
    int total = 0;
    //循环读入每一个学生的成绩，并把它累加到总分中
    for (int i = 0; i < 50; i++)
    {
        cin >> a[i];
        total += a[i];
    }
    //计算平均分
    double average = total / 50.0;
    //循环检查每一个学生的成绩，并输出小于平均分的学生的学号及成绩
    for (int i = 0; i < 50; i++)
    {
        if (a[i] < average)
        {
            cout << "No. " << (i + 1) << " " << a[i];
        }
    }
    return 0;
}
```

在 C++语言中，数组用来存储一系列相同类型的数据。

上面的程序使用了带下标的变量 a[i]来代替 a_1，a_2，…，a_{50}。其中每个分量变量 a[i]叫做数组元素，方括号中的 i 称为下标:

- 当 i=0 时，a[i]就是 a[0];
- 当 i=1 时，a[i]就是 a[1]……;
- 当 i=49 时，a[i]就是 a[49]。

一维数组的声明

数组的声明并不是声明一个个单独的变量，比如 a0、a1、…、a99，而是声明一个数组变量，比如 a，然后使用 a[0]、a[1]、…、a[99] 来代表每个变量。

当数组中的每个元素只带有一个下标时，我们称这样的数组为一维数组。

【声明格式】

一维数组的声明格式如下：

```
1. 类型标识符 数组名[数组元素个数];
```

其中：

- 数组名：命名规则与变量名的命名规则一致。
- 数组元素个数：必须是常量，可以是符号常量或常量表达式，但不能是变量。

【声明样例】

下面是一个一维数组声明的实际例子：

```
1. int a[10];
```

这句代码声明了一个 10 个元素的整型数组，其中

- a 是一维数组的数组名，该数组有 10 个元素。
- 数组中的每个元素都是整型变量。
- a 数组最小下标为 0，最大下标 9。
- 在内存中 10 个数组元素共占 10 个连续的存储单元。

a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9]

【参考程序】

```
1. #include<iostream>
2. using namespace std;
3. int main()
```

```
{
    int a[10]; //大小为 10 的整型数组
    bool b[3]; //大小为 3 的布尔数组
    char c[8]; //大小为 8 的字符数组
    double d[5]; //大小为 5 的双精度实数数组
    return 0;
}
```

【不好的声明方式】

```
#include<iostream>
using namespace std;
int main()
{
    int n;
    cin >> n;
    //不符合 C++的标准，是非常不好的声明方式
    int a[n];
    return 0;
}
```

一维数组元素的引用

通过数组名称和元素在数组中的位置编号（即下标），程序就可以引用这个数组中的任何一个元素。一维数组元素的引用格式：

```
数组名[下标]
```

在数组元素位置编号在数组定义的下标值范围内的前提下，若 i 和 j 都是整型变量，则：

- 下标可以是值为整型的常量。比如 a[0]、a[99]、a['a']。
- 下标可以是整型的变量。比如 a[i]。通过对下标变量值的灵活控制，达到灵活处理数组元素的目的。
- 下标可以是值为整型的表达式。表达式里可以包含变量和函数调用。比如 a[i+j]等。

数组元素可以像同类型的普通变量那样进行赋值和运算的操作。

在使用数组时，只能逐个引用数组元素，而不能一次引用整个数组。

【参考程序】

```
#include<iostream>
using namespace std;
int main()
{
    int a[100]; //大小为 100 的整型数组
    a[0] = 0; //第 1 个元素赋值为 0
    a[99] = 99; //第 100 个元素赋值为 99
    a['a'] = 'a'; //a[97]=97
    int i = 10;
    a[i] = 10; //a[10]=10
    int j = 5;
    a[i+j] = i+j; //a[15]=15
    cin >> a[6];
    cout << a[6];
    cout << a; //这是错误的
    return 0;
}
```

一维数组的初始化

【一维数组初始化的方式】

数组的初始化可以在声明时一并完成。格式如下：

```
类型标识符 数组名[常量表达式]={值1，值2，…};
```

在初值列表中可以写出全部数组元素的值，例如：

```
int a[5]={1, 2, 3, 4, 5};
```

在初值列表中可也可以写出部分。例如，以下方式仅对数组的前 2 个元素进行初始化，其余值为 0。

```
int b[5]={1, 2};
```

对数组元素全部初始化为 0，可以简写为{}。例如，以下方式将数组 c 的 5 个元素都初始化为 0。

```
int c[5]={};
```

【问题描述】

请观察下面程序中数组变量 a、b、c、d、e 的初值情况。

```
#include<iostream>
using namespace std;
int e[5];
int main()
{
    int a[5] = {1, 2, 3, 4, 5};
    int b[5] = {1, 2};
    int c[5] = {};
    int d[5];
    for (int i = 0; i < 5; i++)
    {
        cout << a[i] << " ";
    }
    cout << endl;
    for (int i = 0; i < 5; i++)
    {
        cout << b[i] << " ";
```

```
18.     }
19.     cout << endl;
20.     for (int i = 0; i < 5; i++)
21.     {
22.         cout << c[i] << " ";
23.     }
24.     cout << endl;
25.     for (int i = 0; i < 5; i++)
26.     {
27.         cout << d[i] << " ";
28.     }
29.     cout << endl;
30.     for (int i = 0; i < 5; i++)
31.     {
32.         cout << e[i] << " ";
33.     }
34.     cout << endl;
35.     return 0;
36. }
```

【运行结果】

```
1 2 3 4 5
1 2 0 0 0
0 0 0 0 0
40 0 1 0 0
0 0 0 0 0
```

【说明】

数组 a 在 int main()之内定义，其初始值是分别是 1、2、3、4、5。

数组 b 在 int main()之内定义，只给 a[0]、a[1]赋初值，a[2] ~ a[4]自动赋 0 值。

数组 c 在 int main()之内定义，其初始值是 0 值。

数组 d 在 int main()之内定义，其初始值是随机的。

数组 c 在 int main()之外定义，其初始值是 0 值。

一维数组的逆序输出

【问题描述】

输入 n（n≤100）个数，按照输入时的逆序把这 n 个数打印出来。

【分析】

定义数组 a 存放输入的 n 个数，然后将数组 a 中的内容逆序输出。

【参考程序】

```
#include<iostream>
using namespace std;
const int N = 100;
int a[N+1]; //从 1 下标开始最大容量为 N 个数的数组
int main()
{
    int n; //输入数的个数
    cin >> n;
    for (int i = 1; i <= n; i++)
    {
        cin >> a[i];
    }
    //逆序输出
    for (int i = n; i >= 1; i--)
    {
        cout << a[i] << " ";
    }
    return 0;
}
```

【说明】

第 6 行：声明了一个包含 100 个整型变量的数组，它们是：a[0]，a[1]，a[2]，…，a[99]。注意，没有 a[100]。

逆序输出奇数位置的数

【问题描述】

输入 n 个数，逆序将奇数位置的数字打印出来。

最后 1 个数字的位置为 1，倒数第 2 个数字位置为 2，依次类推。

【输入格式】

第 1 行为一个正整数 n，第 2 行为 n 个小于 100 的正整数。

【输出格式】

1 行逆序输出的若干个正整数，中间以 1 个空格隔开。

【数据范围】

$$1 \leq n \leq 100$$

【输入样例】

```
6
3 4 5 6 7 8
```

【输出格式】

```
8 6 4
```

【参考程序】

```
#include<iostream>
using namespace std;
const int N = 100;
int a[N];
int main()
{
    int n;
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        cin >> a[i];
    }
```

```
    for (int i = n - 1; i >= 0; i -= 2)
    {
        cout << a[i] << " ";
    }
    return 0;
}
```