

第五讲 数组与字符串

— 数组（数值型）

潘建瑜@MATH.ECNU

目录页

Contents

1 一维数组

2 二维数组

3 数组作为函数的参数

1 一维数组

- 怎么定义数组
- 怎么使用数组
- 数组在内存中是怎么存放的
- 数组的赋值和初始化

一维数组的声明

数组： 具有一定顺序关系的同类型数据的集合

一维数组的声明

类型说明符 变量名[n];

- 声明一个长度为 n 的一维数组
- 类型说明符：数组元素的数据类型
- n：数组的长度，即元素的个数

```
int x[5]  // 声明一个长度为 5 的一维数组
```

一维数组的引用

变量名[k]



注意：下标 k 的取值范围为 0 到 n-1

一维数组举例

ex05_array_01.cpp

```
#include<iostream>
using namespace std;

int main()
{
    int i, x[5];

    for (i = 0; i < 5; i++)
        x[i] = 2*i;

    for (i = 0; i < 5; i++)
        cout << "x[" << i << "]=" << x[i] << endl;

}
```

注意：数组的下标不能越界，否则会引起严重的后果！

注意事项

类型说明符 变量名[n];

几点注记

- 数组长度必须是正整数，可以是表达式，但值必须是正整数

```
int n = 5;
int x[n];
```

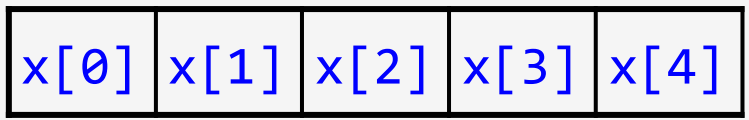
```
int m = 2, n = 3;
int x[m*n+2];
```

```
int n;
cin << n;
int x[n];
```

ex05_array_01.cpp

- 只能逐个引用数组元素（循环），而不能一次引用整个数组
- 数组名代表数组存放在内存中的首地址
- 数组元素在内存中顺序存放，它们的地址是连续的

例：x[5] 在内存中的存放顺序是



一维数组初始化

- 初始化： 在声明的同时赋初值

```
int x[5] = {0,2,4,6,8};
```

- 可以只给部分元素赋初值

```
int x[5] = {0,2,4};   // 从前往后依次初始化，剩下的赋值 0
```

- 全部初始化时可以不指定数组长度

```
int x[] = {0,2,4,6,8}; // 根据所赋初值的个数自动确定数组长度
```

注意：只能对数组元素赋值，不能对数组名赋值！

```
int x[5];
x[0] = 1;   // OK
x = 6;    // ERROR!
```

2

二维数组

- 怎么定义，怎么使用
- 与一维数组的关系
- 在内存中是怎么存放的
- 赋值和初始化，多维数组

二维数组的声明

类型说明符 变量名[m][n];

- 声明一个 m x n 的二维数组

二维数组的引用

变量名[i][j]

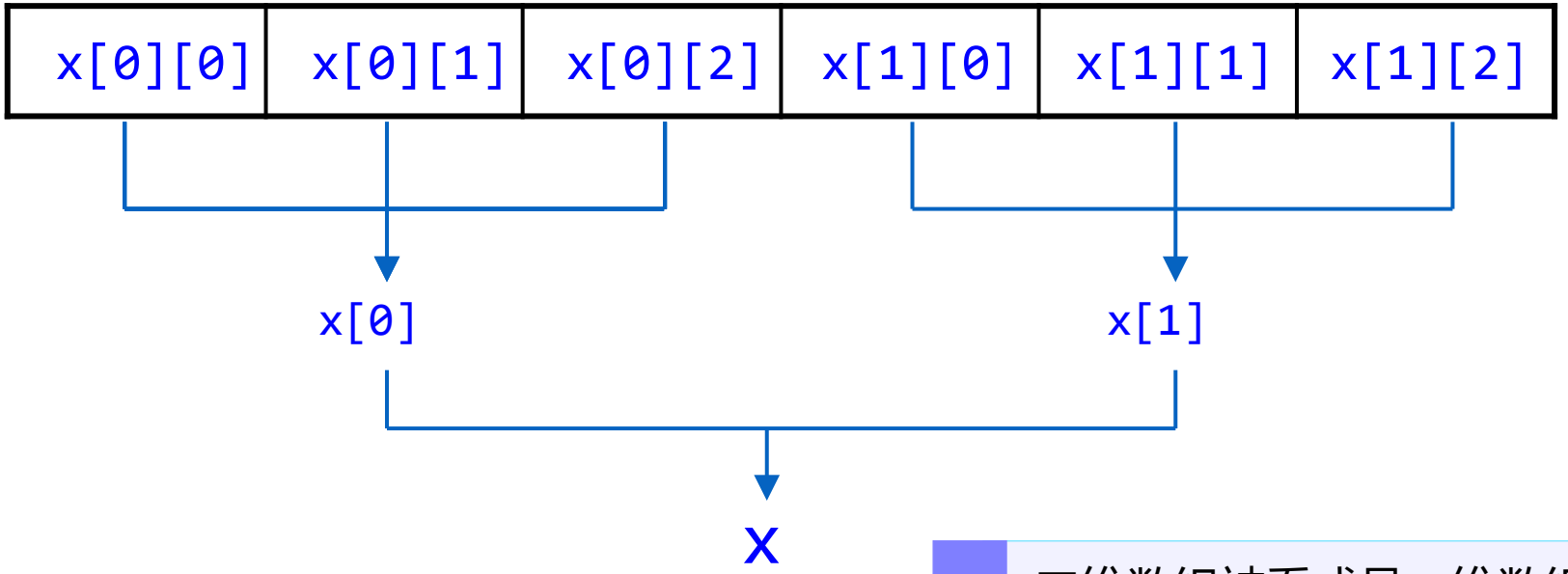
- 注意下标 的取值范围，不要越界！

```
int x[2][3];
x[0][0] = 1; x[0][1] = 3; x[0][2] = 5;
for (j = 0; j < 3; j++)
   x[1][j] = 2*x[0][j];
```

二维数组存储方式

二维数组的存储： 按行存储

例：x[2][3] 在内存中的存放顺序是



二维数组被看成是一维数组的数组

二维数组初始化

- 全部初始化

```
int x[2][3] = {1,3,5,2,6,10};
```

```
int x[][3] = {1,3,5,2,6,10}; // 注：只能省第一维的长度!
```

- 分组初始化

```
int x[2][3] = {{1,3,5}, {2,6,10}};
```

- 部分初始化

```
int x[2][3] = {{1}, {2,6}};
```

二维数组举例

ex05_array_hilb.cpp

例：计算 n 阶 Hilbert 矩阵与全是 1 的 向量的乘积

$$H = [h_{ij}], \qquad h_{ij} = \frac{1}{i + j - 1}$$

$$y = Hx = \sum_{i=1}^{n} \sum_{j=1}^{n} h_{ij} x_j$$

高维数组

类型说明符 变量名[n1][n2][n3]...

- 高维数组的声明、赋值、引用、初始化与二维数组类似。

3 数组作为函数的参数

- 数组中的单个元素作实参

  —— 传递单个元素，值传递

- 数组名作为参数

  —— 传递整个数组，地址传递

数组元素作为参数

传递单个数组元素

- 形参是普通变量，实参是数组元素，值传递

```
void my_swap(int a, int b)
{
   int t;
   t = a; a = b; b = t;
}
int main()
{
   int x[2] = {1,3};
   my_swap(x[0], x[1]);
   cout << "x[0] = " << x[0] << ", x[1] = " << x[1] << endl;
}
```

数组名作为参数

传递整个数组

- 形参和实参都是数组名，类型一样
- 形参后面要加中括号
- 传递的是数组首地址，即形参和实参代表同一个数组

> 在函数中对形参数组的任何改变都会直接影响到实参数组！

数组举例

ex05_array_swap.cpp

```
void my_swap(int a[], int b[], int n)
{
  int t, i;
  for (i = 0; i < n; i++)
  { t = a[i]; a[i] = b[i]; b[i] = t; }
}
int main()
{ const int n = 3;
  int i, x[n] = {1,2,3}, y[n] = {2,4,6};
  my_swap(x,y,n);
}
```

用一维数组作形参，可以省略长度，但中括号不能省！

为增加灵活性，数组名作形参时一般不指定长度；但此时通常需要加一个参数，用来传递实参数组的长度（也可以通过全局变量实现）。

数组名不传递数组长度



注意：数组名作为函数参数，传递数组首地址，但**不会传递数组长度**！

ex05_array_sizeof.cpp

```
void my_swap(int a[], int b[], int n)
{
  cout << "sizeof(a) = " << sizeof(a) << endl; // 输出结果不是数组的长度!
  int t, i;
  for (i = 0; i < n; i++)
  { t = a[i]; a[i] = b[i]; b[i] = t; }
}
```

数组举例

例：计算矩阵各列的和

ex05_array_fun.cpp

```
const int m = 3, n = 4; // 常量，矩阵维数
void sum_col(double A[][n], double s[])
{   int i, j;
    for(j = 0; j < n; j++)  s[j] = 0.0; // 赋初值
    for(j = 0; j < n; j++)
    for(i = 0; i < m; i++) s[j] = s[j] + A[i][j];
}
int main()
{   double H[m][n], s[n];
    for(int i = 0; i < m; i++)
    for(int j = 0; j < n; j++)  H[i][j] = 1.0/(i+j+1);
    sum_col(H, s);
    cout << "s[0] = " << s[0] << ", s[n-1] = " << s[n-1] << endl;
    return 0;
}
```

二维数组名作为形参，只能省略第一个维的大小，即行数

数组名作为参数须指定大小

- 在定义函数时，如果形参是数组名，则须指定数组的大小（常量表达式）

```
void sum_col(double A[10][10], double s[10])
```

可以省略第 1 维的大小

```
void sum_col(double A[][10], double s[])
```

若数组的大小中含有变量，则必须是常量（全局）

```
const int n = 10;
void sum_col(double A[][n], double s[])
```

含数组形参的函数调用

- 函数调用时，只需输入数组名即可

```
sum_col(A, s)
```

第五讲上机作业（数组）

1、计算均值和标准偏差：

给定一组数 $x_1, x_1, \ldots, x_n$，其均值和标准偏差分别定义为：

$$\text{mean} = \frac{x_1 + x_2 + \cdots + x_n}{n}, \qquad \text{deviation} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \text{mean})^2}{n-1}}$$

编写程序，生成 100 个随机双精度数，计算他们的均值和标准偏差。

要求：**(1)** 编写两个函数：`mean` 和 `deviation`，分别计算一个数组的均值和标准偏差；

(2) 在主函数中生成一个长度为 100 的随机双精度数组 (元素的值在 [-10,10] 内)，调用以上函数计算均值和标准偏差（程序取名 **hw05_01.cpp**）

```
double mean(double x[], int n)
double deviation(double x[], int n)
```

3、反转数组：编写函数，反转一个数组，并在 **main** 函数中生成一个长度为 **10** 的随机整数数组(元素的值在 **0** 到 **99** 之间)，反转该数组 (反转后的数组仍存放在原数组中)，要求输出反转前和反转后的数组，以便比较。

（程序取名 **hw05_03.cpp**）

```
void reverse(int x[], int n);
```

第五讲上机作业（数组）

4、矩阵乘积：编写函数，计算两个 5 阶矩阵的乘积 Z=X*Y。在主函数中生成两个 5 阶的随机矩阵，其元素为在 0 到 9 之间的正整数，然后计算它们的乘积，并在主函数中将这三个矩阵输出。

```
const int N = 5;
void matrix_prod(int X[][N], int Y[][N], int Z[][N])
```

（程序取名 hw05_04.cpp）

5、找最小值所在位置：编写函数，找出给定一维数组的最小值所在下标。

在主函数中以数组 [34, 91, 85, 59, 29, 93, 56, 12, 88, 72] 为例，输出最小数所在的下标。

（若有多个最小数，返回第一个的下标即可，程序取名 hw05_05.cpp）

```
int findmin(int a[], int n);
```

6、编写函数 insert，实现下面功能：*n* 个整数，已经从小到大排列，在主函数中输入一个整数，调用 insert 函数，把输入的整数插入到原有数列中，保持大小顺序，并将被挤出的最大数（有可能就是被插入的数）返回给主函数输出。（以 [12, 29, 34, 56, 59, 72, 85, 88, 91, 93] 为例，要求输出原数列、插入后的新数列，以及被挤出的数，程序取名 hw05_06.cpp）

```
int insert(int a[], int n, int x);
```

第五讲上机作业（数组）

7、储物柜问题

学校有 100 个储物柜，100 个学生。开学第一天所有储物柜都是关闭的，第一个学生到校后将所有储物柜打开；第二个学生到校后，从第二个储物柜开始，每隔 1 个储物柜，将它们关闭；第三个学生到校后，从第三个储物柜开始，每隔 2 个，将它们的状态改变，即如果是开着的则将其关闭，如果是关闭的则将其打开。依此类推，直至第 100 个学生到校后将第 100 个储物柜的状态改变。问：当所有学生完成这个过程后，有哪些储物柜是开着的？编写程序求解该问题（程序取名 **hw05_07.cpp** ）

提示：使用一个数组，保存每个储物柜的状态改变次数，如果一个储物柜的状态改变次数为奇数，则该储物柜是开着的。

8、找最小值所在位置：编写函数，找出给定二维数组的最小值所在下标。

在主函数中以一个 8 阶随机矩阵 (元素[-1,1] 中的随机小数) 为例，在屏幕上输出最小数及其下标。(若有多个最小数，返回第一个的下标即可 (按行排序)，程序取名 **hw05_08.cpp**)

```
void findmin2D(double A[M][N], int idx[2]);
```