程序设计与算法(一)

C语言程序设计

郭 炜

微信公众号

微博：http://weibo.com/guoweiofpku

学会程序和算法，走遍天下都不怕!

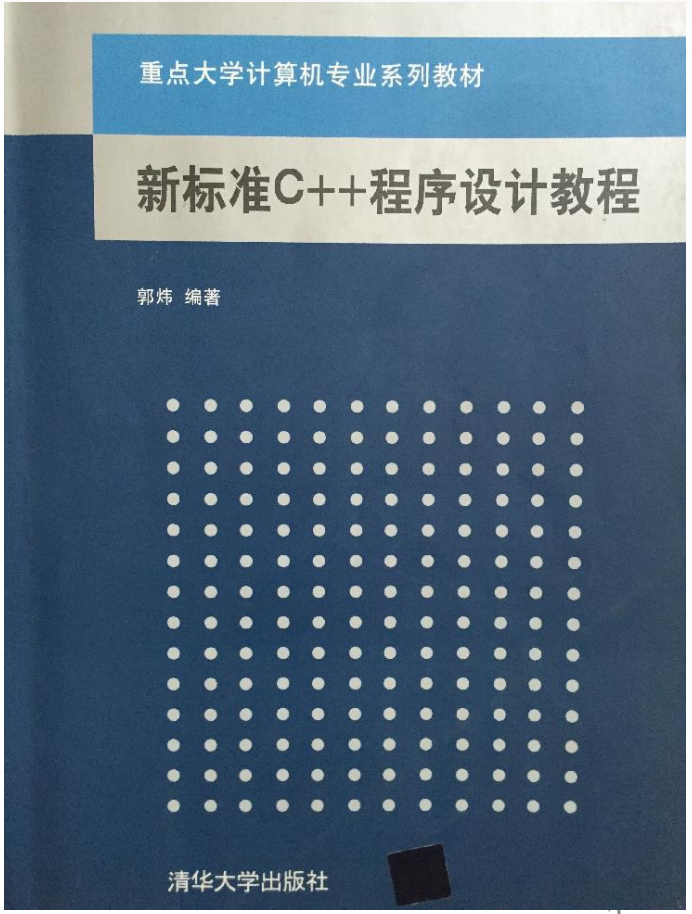
讲义照片均为郭炜拍摄

指定教材：

《新标准C++程序设计教程》

郭炜 编著

清华大学出版社



数 组

数组的概念

瑞士少女峰

倒序问题

- 接收键盘输入的100个整数，然后将它们按和原顺序相反的顺序输出。

倒序问题

- 接收键盘输入的100个整数，然后将它们按和原顺序相反的顺序输出。

- 如何存放这100个整数?

倒序问题

- 接收键盘输入的100个整数，然后将它们按和原顺序相反的顺序输出。
- 如何存放这100个整数?
- 定义100个int型变量，n1, n2, n3 ……n100，用来存放这100个整数???!!!!

倒序问题

- 接收键盘输入的100个整数，然后将它们按和原顺序相反的顺序输出。
- 如何存放这100个整数?
- 定义100个int型变量，n1, n2, n3 ……n100，用来存放这100个整数???!!!!
- 使用数组！！

数组

- 可以用来表达类型相同的元素的集合，集合的名字就是数组名。

数组

●可以用来表达类型相同的元素的集合，集合的名字就是数组名。

●数组里的元素都有编号，元素的编号叫下标。通过数组名和下标，就能访问元素。

数组

●可以用来表达类型相同的元素的集合，集合的名字就是数组名。

●数组里的元素都有编号，元素的编号叫下标。通过数组名和下标，就能访问元素。

●一维数组的定义方法如下：

类型名 数组名[元素个数];

数组

●可以用来表达类型相同的元素的集合，集合的名字就是数组名。

●数组里的元素都有编号，元素的编号叫下标。通过数组名和下标，就能访问元素。

●一维数组的定义方法如下：

类型名 数组名[元素个数];

●其中“元素个数”必须是常量或常量表达式，不能是变量，而且其值必须是正整数。元素个数也称作“数组的长度”。

数组

- int a[100];

  名字为a的数组，有100个元素，每个元素都是一个int型变量。

数组

- int a[100];

  名字为a的数组，有100个元素，每个元素都是一个int型变量。

- **T a[N];** //T为类型名，如char，double，int等。
  //N为正整数或值为正整数的常量表达式。

数组a有N个元素，每个元素都是一个类型为T的变量。
N个元素在内存里是一个挨一个连续存放的。
a数组占用大小总共为 N × sizeof(T)字节的存储空间。

数组

- int a[100];

  名字为a的数组，有100个元素，每个元素都是一个int型变量。

- **T a[N];** //T为类型名，如char，double，int等。
  //N为正整数或值为正整数的常量表达式。

数组a有N个元素，每个元素都是一个类型为T的变量。
N个元素在内存里是一个挨一个连续存放的。
a数组占用大小总共为 N × sizeof(T)字节的存储空间。

- 表达式“sizeof(a)”的值就是整个数组的体积，即N × sizeof(T)。

数组

- `int a[100];`

数组

- `int a[100];`

- 数组下标从0开始，N个元素的数组，下标从0 至 N-1

数组

- `int a[100];`

- 数组下标从0开始，N个元素的数组，下标从0 至 N-1
- 数组名a代表数组的地址，假设为p，则变量a[i]的地址就是p+i*sizeof(int)

倒序问题

- 接收键盘输入的100个整数，然后将它们按和原顺序相反的顺序输出

```
#include <iostream>
using namespace std;
#define NUM  100   //使用符号常量，便于修改
int a[NUM]; //数组一般不要定义在main里面，尤其是大数组
int main()   {
      for(int i = 0;i < NUM; ++i)
            cin >> a[i];
      for(int i = NUM-1;i >= 0; --i)
            cout << a[i] << " ";
      return 0;
}
```

例题：筛法求素数

瑞士日内瓦湖

筛法求n以内素数

- 判断一个数n是不是素数，可以用2到$\sqrt{n}$之间的所有整数去除n，看能否整除。如果都不能整除，那么n是素数(慢) 。

筛法求n以内素数

- 判断一个数n是不是素数，可以用2到$\sqrt{n}$之间的所有整数去除n，看能否整除。如果都不能整除，那么n是素数(慢) 。

- 筛法求素数：把2到n中所有的数都列出来，然后从2开始，先划掉n内所有2的倍数，然后每次从下一个剩下的数(必然是素数)开始，划掉其n内的所有倍数。最后剩下的数，就都是素数。

筛法求n以内素数

- 判断一个数n是不是素数，可以用2到$\sqrt{n}$之间的所有整数去除n，看能否整除。如果都不能整除，那么n是素数(慢) 。

- 筛法求素数：把2到n中所有的数都列出来，然后从2开始，先划掉n内所有2的倍数，然后每次从下一个剩下的数(必然是素数)开始，划掉其n内的所有倍数。最后剩下的数，就都是素数。

2	3	4	5	6	7	8	9	10

筛法求n以内素数

- 判断一个数n是不是素数，可以用2到$\sqrt{n}$之间的所有整数去除n，看能否整除。如果都不能整除，那么n是素数(慢) 。

- 筛法求素数：把2到n中所有的数都列出来，然后从2开始，先划掉n内所有2的倍数，然后每次从下一个剩下的数(必然是素数)开始，划掉其n内的所有倍数。最后剩下的数，就都是素数。

2	3	4	5	6	7	8	9	10

- 空间换时间，加快了计算速度

筛法求n以内素数(变形)

- 设置一个标志数组 isPrime，isPrime[i]的值是1就表示i是素数。开始数组元素值全部为1
- 划掉k的倍数，就是把isPrime[2*k]，isPrime[3*k] ….置成0
- 最后检查isPrime数组，输出isPrime[i]为1的那些i

```cpp
#include <iostream> //筛法求素数
#include <cmath>
using namespace std;
#define MAX_NUM  10000000
bool isPrime[MAX_NUM + 10]; //最终如果isPrime[i]为1，则表示i是素数
int main()
{
        for( int i = 2;i <= MAX_NUM; ++i) //开始假设所有数都是素数
                isPrime[i] = true;
        for( int i = 2;i <= MAX_NUM; ++i) { //每次将一个素数的所有倍数标记为非素数
                if( isPrime[i])  //只标记素数的倍数
                        for( int j = 2 * i; j <= MAX_NUM; j += i)
                                isPrime[j] = false; //将素数 i 的倍数标记为非素数
        }
        for( int i = 2;i <= MAX_NUM; ++i)
                if( isPrime[i])
                        cout << i << endl;
        return 0;
}
```

数组的初始化

德国国王湖

数组的初始化

- 在定义一个一维数组的同时，就可以给数组中的元素赋初值：

  类型名 数组名[常量表达式]={值，值……值}；

{ }中的各数据值即为各元素的初值，值之间用逗号间隔

数组的初始化

```
int a[10]={ 0,1,2,3,4,5,6,7,8,9 };
```

效果： `a[0]=0;a[1]=1...a[9]=9;`

数组的初始化

●数组初始化时，{ }中值的个数可以少于元素个数。相当于只给前面部分元素赋值，而后面的元素，其存储空间里的每个字节都被写入二进制数0：

```
int a[10]={0,1,2,3,4};
```

只给a[0]～a[4]5个元素赋值，而后5个元素自动赋0值

用数组取代复杂分支结构

罗马古城遗址

用数组取代复杂分支结构

- 有时会用一个数组存放一些固定不变的值，以取代复杂的程序分支结构。

用数组取代复杂分支结构

- 有时会用一个数组存放一些固定不变的值，以取代复杂的程序分支结构。

- 例：接受一个整数作为输入，如果输入1，则输出“Monday”，输入2，则输出“Tuesday”……输入7,则输出“Sunday”，输入其他数，则输出“Illegal”。

用数组取代复杂分支结构

```
#include <iostream>
#include <string> //使用string须包含此“头文件”
using namespace std;
string weekdays[] = {    //string是字符串类型。可存放字符串常量
"Monday","Tuesday","Wednesday","Thursday",
            "Friday","Saturday","Sunday" }; //字符串数组
int main()
{
        int n;
        cin >> n;
        if( n > 7 || n < 1 )
                cout << "Illegal";
        else
                cout << weekdays[n-1];
        return 0;
}
```

用数组取代复杂分支结构

●例题： 已知2012年1月25日是星期三，编写一个程序，输入用“年 月 日”表示的一个2012年1月25日以后的期，输出该日期是星期几(星期天输出0)。

Sample Input

```
2015 11 02
```

Sample Output

```
1
```

用数组取代复杂分支结构

●例题： 已知2012年1月25日是星期三，编写一个程序，输入用“年 月 日”表示的一个2012年1月25日以后的期，输出该日期是星期几(星期天输出0)。

```
Sample Input
2015 11 02
Sample Output
1
```

●**思路：2012年1月22日是星期天。算出给定日期是从该天起过了x天，然后输出x%7**

```
#include <iostream>
using namespace std;
int monthDays[13] = {-1,31,28,31,30,31,30,31,31,30,31,30,31};
int main()
{
	int year,month,date;
	int days = 0; //从2012-01-22开始过了多少天
	cin >> year >> month >> date;
	for(int y = 2012; y < year; ++y) {
		if( y%4 ==0 && y%100!= 0 || y%400 == 0)
			days += 366;
		else
			days += 365;
	}
	if( year%4 ==0 && year%100!= 0 || year%400 == 0)
		monthDays[2] = 29;
	for(int m = 1; m < month; ++m)
		days += monthDays[m];
```

```
        days += date;
        days -= 22;  //2012年1月22日是星期天
        cout << days % 7 << endl;
        return 0;
}
```

数组越界

古罗马大斗兽场

数组越界

●数组元素的下标，可以是任何整数，可以是负数，也可以大于数组的元素个数。不会导致编译错误：

```
int a[10];
a[-2] = 5;
a[200] = 10;
a[10] = 20;
int m = a[30];
```

数组越界

●数组元素的下标，可以是任何整数，可以是负数，也可以大于数组的元素个数。不会导致编译错误：

```
int a[10];
a[-2] = 5;
a[200] = 10;
a[10] = 20;
int m = a[30];
```

但运行时很可能会出错！！！

数组越界

- `int a[10];`

- `a[-2] = 10; a[11] = 100;` **均可能导致程序运行出错！！！**
  因为可能写入了别的变量的内存空间，或者写入指令的内存空间

数组越界

- 用变量作为数组下标时，不小心会导致数组越界（变量下标值变为负数，或者太大）

数组越界

- 用变量作为数组下标时，不小心会导致数组越界（变量下标值变为负数，或者太大）

- 可能引起意外修改其他变量的值，导致程序运行结果不正确

数组越界

- 用变量作为数组下标时，不小心会导致数组越界（变量下标值变为负数，或者太大）

- 可能引起意外修改其他变量的值，导致程序运行结果不正确

- 可能试图访问不该访问的内存区域，导致程序崩溃

数组越界

- 用变量作为数组下标时，不小心会导致数组越界（变量下标值变为负数，或者太大）

- 可能引起意外修改其他变量的值，导致程序运行结果不正确

- 可能试图访问不该访问的内存区域，导致程序崩溃

- 数组越界的程序，用某些编译器编译后可能可以正确运行，换一个编译器编译后就运行错误

例题：矩阵乘法

威尼斯

矩阵乘法

编程求两个矩阵相乘的结果。输入第一行是整数m,n，
表示第一个矩阵是m行n列的。接下来时一个m×n的矩阵。
再下一行的输入是整数p,q，表示下一个矩阵是p行q列（n=p）
再接下来就是一个p行q列的矩阵。
要求输出两个矩阵相乘的结果矩阵(1 < m,n,p,q <= 8)。

输入样例:

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

输出样例:

```
10 19 30
4 8 16
```

矩阵乘法

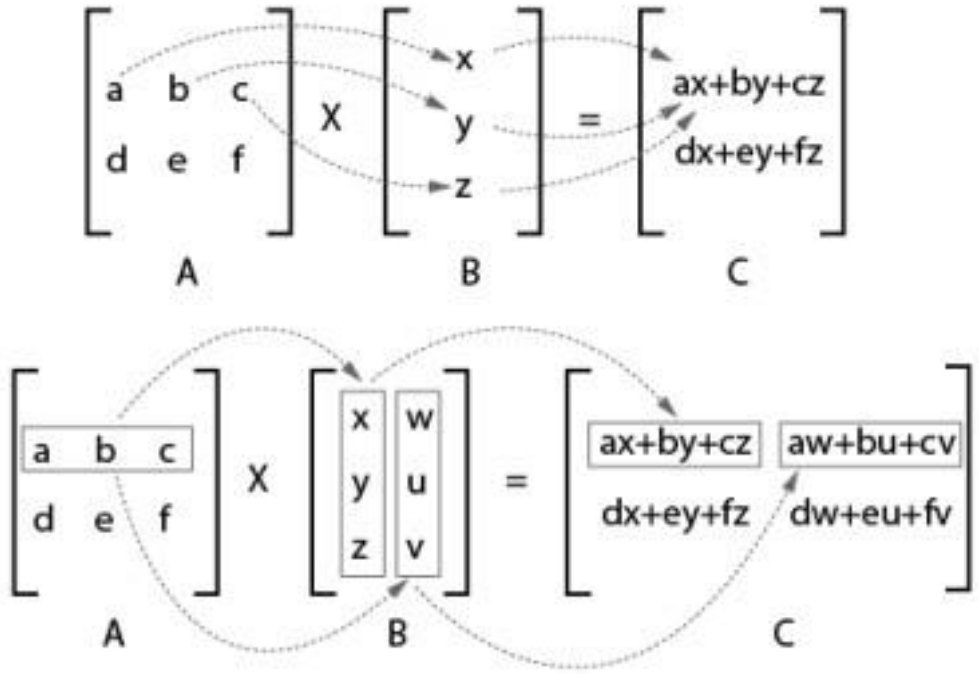
m行n列的矩阵

X

n行k列的矩阵

=

m行k列的矩阵



矩阵的乘法

矩阵乘法

编程求两个矩阵相乘的结果。输入第一行是整数m,n，
表示第一个矩阵是m行n列的。接下来时一个m×n的矩阵。
再下一行的输入是整数p,q，表示下一个矩阵是p行q列的（n=p）。
再接下来就是一个p行q列的矩阵。
要求输出两个矩阵相乘的结果矩阵(1 < m,n,p,q <= 8)。

输入样例:

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

输出样例:

```
10 19 30
4 8 16
```

用什么存放矩阵?

二维数组

瑞士苏黎世湖

二维数组

- 定义N行M列的二维数组：

```
T a[N][M];    // T :类型名，如char , double, int等。
              // M、N : 正整数，或值为正整数的常量表达式
```

二维数组

- 定义N行M列的二维数组：

```
T a[N][M];    // T :类型名，如char , double, int等。
              // M、N : 正整数，或值为正整数的常量表达式
```

- 每个元素都是一个类型为T的变量

二维数组

- 定义N行M列的二维数组：

```
T a[N][M];    // T :类型名，如char , double, int等。
              // M、N : 正整数，或值为正整数的常量表达式
```

- 每个元素都是一个类型为T的变量

- N×M个元素在内存里是一个挨一个连续存放的。

二维数组

- 定义N行M列的二维数组：

```
T a[N][M];    // T :类型名，如char , double, int等。
              // M、N : 正整数，或值为正整数的常量表达式
```

- 每个元素都是一个类型为T的变量
- N×M个元素在内存里是一个挨一个连续存放的。
- 数组占用了一片连续的、大小总共为 N×M×sizeof(T)字节的存储空间。

二维数组

- 定义N行M列的二维数组：

```
T a[N][M];    // T :类型名，如char , double, int等。
              // M、N : 正整数，或值为正整数的常量表达式
```

- 每个元素都是一个类型为T的变量
- N×M个元素在内存里是一个挨一个连续存放的。
- 数组占用了一片连续的、大小总共为 N×M×sizeof(T)字节的存储空间。
- 表达式“sizeof(a)”的值就是整个数组的体积，即N×M×sizeof(T)。

二维数组

- 访问数组元素的方法：

  数组名[行下标][列下标]

  例如: a[i][j]

二维数组

- 访问数组元素的方法：

  数组名[行下标][列下标]

  例如: a[i][j]

- 行下标和列下标都从0开始

二维数组的存放方式

- 数组T a[N][M] 每一行都有M个元素

二维数组的存放方式

- 数组T a[N][M] 每一行都有M个元素

- 第i行的元素就是a[i][0]、a[i][1]……a[i][M-1]。
  同一行的元素，在内存中是连续存放的。

二维数组的存放方式

- 数组T a[N][M] 每一行都有M个元素

- 第i行的元素就是a[i][0]、a[i][1]……a[i][M-1]。
  同一行的元素，在内存中是连续存放的。

- 第j列的元素的元素，就是a[0][j]、a[1][j]……a[N-1][j]。

二维数组的存放方式

●数组T a[N][M] 每一行都有M个元素

●第i行的元素就是a[i][0]、a[i][1]……a[i][M-1]。
同一行的元素，在内存中是连续存放的。

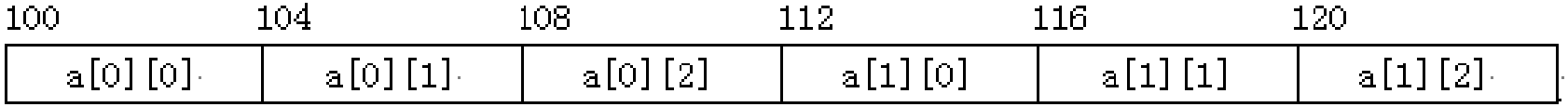
●第j列的元素的元素，就是a[0][j]、a[1][j]……a[N-1][j]。

●a[0][0]是数组中地址最小的元素。如果a[0][0]存放在地址n，则a[i][j]存放的地址就是

$$n + i \times M \times sizeof(T) + j \times sizeof(T)$$

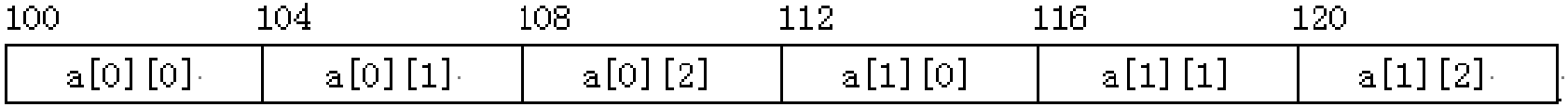
二维数组的存放方式

- int a[2][3] 在内存中的存放方式:



二维数组的存放方式

- int a[2][3] 在内存中的存放方式:



- 二维数组的每一行，实际上都是一个一维数组。

a[0]，a[1]都可以看作是一个一维数组的名字，可以直接当一维数组使用。

二维数组的初始化

```
int a[5][3]={{80,75,92},{61,65},{59,63,70},{85,90},{76,77,85}};
```

每个内层的{}，初始化数组中的一行。

二维数组的初始化

```
int a[5][3]={{80,75,92},{61,65},{59,63,70},{85,90},{76,77,85}};
```

每个内层的{}，初始化数组中的一行。

- 二维数组初始化时，如果对每行都进行了初始化，则也可以不给出行数:

  int a[][3]={ {80,75,92},{61,65} };

  a 是一个2行3列的数组，a[1][2]被初始化成0。

遍历二维数组

遍历一个二维数组，将其所有元素逐行依次输出：

```
#define ROW 20
#define COL 30
int a[ROW][COL];
for( int i = 0; i < ROW ; ++i) {
        for( int j = 0; j < COL ; ++j )
                cout << a[i][j] << " ";
        cout << endl;
}
```

矩阵乘法

编程求两个矩阵相乘的结果。输入第一行是整数m,n，
表示第一个矩阵是m行n列的。接下来时一个m×n的矩阵。
再下一行的输入是整数p,q，表示下一个矩阵是p行q列（n=p）
再接下来就是一个p行q列的矩阵。
要求输出两个矩阵相乘的结果矩阵(1 < m,n,p,q <= 8)。

输入样例:

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

输出样例:

```
10 19 30
4 8 16
```

矩阵乘法

```
#include <iostream>
using namespace std;
#define ROWS 8
#define COLS 8
int a[ROWS][COLS];
int b[ROWS][COLS];
int c[ROWS][COLS]; //结果
int main()
{
        int m,n,p,q;
        cin >> m >> n;
        for(int i = 0;i<m; ++i) //读入a矩阵
                for(int j = 0; j < n; ++j)
                        cin >> a[i][j];
        cin >> p >> q;
        for(int i = 0;i<p; ++i) //读入b矩阵
                for(int j = 0; j < q; ++j)
                        cin >> b[i][j];
```

矩阵乘法

```
        for(int i = 0; i < m; ++i) {
                for(int j = 0; j < q; ++j)  {
                        c[i][j] = 0;
                        for(int k = 0; k < n; ++k)
                                c[i][j] += a[i][k] * b[k][j];
                }
        }
        for(int i = 0;i<m;++i){
                for(int j = 0; j < q; ++j) {
                        cout << c[i][j] << " ";
                }
                cout << endl;
        }
        return 0;
}
```

```
2 4 5
2 1 3

1 1 1
2 3 2
0 1 4
```