C++大学基础教程

第5章　数组

北京科技大学

信息基础科学系

资料整理
火龙果软件
荐

第5章 数组

5.1 数组基本概念

5.2 数组元素的下标

5.3 数组初始化

5.4 数组的大小和数组越界

5.5 字符数组

5.6 向函数传递数组

5.7 多维数组

资料整理
火龙果软件
荐

本章教学要求

- 掌握数组定义、数组元素的引用以及初始化数组的知识和方法
- 学会把数组用作函数参数
- 学会二维数组的使用
- 初步掌握使用数组实现一些典型算法：排序、矩阵操作、用字符数组处理字符串等。

资料整理
火龙果软件
荐

5.1　数组基本概念

- 数组是具有一定顺序关系的若干相同类型变量的集合。
- 组成数组的变量称为该数组元素。
- 数组元素都有相同的变量名（数组名），但是有不同的下标。
- 数组属于构造类型。

资料整理
火龙果软件
荐

一维数组的定义与引用

- 一维数组的定义

  类型说明符　　数组名[常量表达式];

数组名的构成方法与一般变量名相同，必须是合法的标识符。

例如：　int　a[10];

1）a为整型数组名；

2）a数组有10个元素，每个元素都是整数数据：a[0]...a[9]

上C++课程的学生成绩(180个学生)

float scoresCPlus[180];

scoresCPlus[179]

资料整理 火龙果软件 荐

一维数组的存储顺序

数组元素在内存中顺序存放，它们的地址是连续的。

Eg. int a[10];

具有10个元素的数组 a，在内存中的存放次序如下：

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

数组名字是数组首元素的内存地址。

数组名是一个常量，不能被赋值。

一维数组的定义与引用

- 引用

1. 数组必须先定义，后使用。
2. 只能逐个引用数组元素，而不能一次引用整个数组
3. 数组元素的引用是通过下标变量实现的。

元素的引用形式为：

数组名[下标表达式]

例如1：int a[10];

a[0]=a[5]+a[7]-a[2*3];

2：int salaries [6];

int a=5; salaries[a]=900;

资料整理 火龙果软件 荐

注意

◆ 在使用数组元素时需要注意：

▸数组元素的下标表达式其结果必须为自然数（≥0）。

▸数组元素的下标值从0开始，不得超过声明时所定义的上界。

资料整理
火龙果软件
荐

数组元素的下标

- 数组元素的下标是数组元素到数组开始地址的偏移量。
- 第1个元素的下标为0，其地址是数组的首地址，第2个元素的下标为1，偏移量距离首地址是1个数组元素大小，依次类推。
- 因此，数组元素是一系列大小相同的连续项，每项到公共基点（数组起始地址）的偏移量是固定的。

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

资料整理
火龙果软件
荐

错误的数组定义语句

- void VoidArray[10];
  //void不可以做数组类型
- int a=9; float floatArray[a];
  //数组的长度不可以是变量
- char charArray[1.5+1.5];
  //3.0不是整数

资料整理
火龙果软件
荐

例5.1 定义一个10个整数的数组

解：以ARRAY命名的数组，用100至109对数组元素赋值，并对其求和。需要在一个循环中使用数组。这是使用数组最经常的方式。

```
void main()
{  int ARRAY [10];
   int sum=0;
   for (int i=0;i<10;i++)
   {   ARRAY[i]=100+i;
       sum+= ARRAY[i];
   }
}
```

资料整理 火龙果软件 荐

注意

- 不能直接把一个数组赋给另一个数组。假设要将数组total_sales的值拷贝到数组saved_sales中，使用下面的赋值方法就是错误的：

```
saved_sales =total_sales;        //error
```

- 应使用一个循环语句将total_sales中的元素的值逐个赋给数组saved_sales中的每个元素。如下面的代码所示：

```
for (int i=0;i<ARRAY_SIZE;i++)
    saved_sales[i] =total_sales[i];
```

资料整理
火龙果软件
荐

一维数组的初始化

可以使数组得到初值：

- 在声明数组时对数组元素赋以初值。
  例如：int a[10]={0,1,2,3,4,5,6,7,8,9};
- 可以只给一部分元素赋初值。其他元素自动赋值为0
  例如：int a[10]={0,1,2,3,4};
- 在对全部数组元素赋初值时，可以不指定数组长度。长度为初值的个数。

例如：int a[ ]={1,2,3,4,5}　等价于

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

资料整理
火龙果软件
荐

大括号的使用

- 通过使用大括号，可初始化任何一种类型的数组。

  例如，要记录前三年的销售总额，则可以如下定义并初始化一个数组：

```
double sales[]= {4323.43，122355.32，343324.96}
```

- 注意：上面这种使用大括号来初始化数组的方法只能在定义数组时使用。在数组定义之后，就不能用这种方法了，而只能逐个元素地赋值。

```
double sales[3];
sales= {4323.43，122355.32，343324.96}；
```

资料整理 火龙果软件 荐

全局数组初始化

- C++自动将全局数组变量中的所有元素初始化为0或null。
  - 如果未赋初值，那么一个全局的字符数组中的所有元素将为null(空)，一个全局的数字数组中的所有元素将为0。
- 程序中应尽量限制全局数组变量的使用。如果要用，也最好在程序中用语句将其显式初始化为0，以明确表示编程者的意图。

资料整理
火龙果软件
荐

错误例子

例如下面的代码对数组进行初始化是错误的：

- int array1[5]={0,1,2,3,4,5};
  //error初始化值个数多于数组元素个数
- int array2[5]＝{, , 1, 2, 3 };
  //error不能加入“,”来跳过不赋值的元素。
- int array2[5]={0,,2,3,4};
  //error不能加入“,”来跳过不赋值的元素。
- int array3[5]={ };
  //error语法格式错误

资料整理
火龙果软件
荐

例5.2初始化全局和局部数组

```
#include <iostream>
using namespace std;
int array1[5]={1,2,3};
static int array2[5]={1};
void main()
{ int arr1[5]={2};
  static int arr2[5]={1,2};
  int n;
  cout <<"global:";
  for(n=0; n<5; n++)
    cout <<" " <<array1[n];
  cout <<"\nglobal static:";
  for(n=0; n<5; n++)
    cout <<"  " <<array2[n];
  cout <<"\nlocal:\n";
  for(n=0; n<5; n++)
    cout <<"  " <<arr1[n];
  cout <<"\nlocal static:";
  for(n=0; n<5; n++)
    cout <<"  " <<arr2[n];
  cout <<endl;}
```

运行结果为：

```
global:    1 2 3 0 0          //全局数组初始化的结果
global static:  1  0  0  0  0  //全局静态数组初始化的结果
local:  2  0  0  0  0          //局部数组初始化的结果
local static:  1  2  0  0  0   ////局部静态数组初始化的结果
```

资料整理
火龙果软件
荐

例5.3 将5个温度值赋给数组

```
#include "stdafx.h"
#include<iostream.h>
void main()
{   float temp[5];
    temp[0]=31.3;
    temp[1]=28.7;
    temp[2]=32.2;
    temp[3]=34.5;
    temp[4]=19.7;
    cout<<"Daily temperature for 5 days\n";
    for (int i=0;i< 5 ;i++)
          cout<<temp[i]<<"\n";
}
```

资料整理
火龙果软件
荐

数组的大小和数组越界

- 定义数组时，编译器必须知道数组的大小。
- 如果数组定义时省略了大小，由初始化的值的个数来决定数组大小。
- 在程序中怎么知道数组的大小呢？

  sizeof操作解决了该问题。

资料整理
火龙果软件
荐

sizeof

- sizeof()能够返回传递给它的数据类型所占用内存的字节数。
- 如果将一个数组变量传给sizeof()，则它将返回系统为此数组在内存中预留的字节数。
- 因此，数组的大小可用用以下公式来计算：

  数组大小 = sizeof(数组名) / sizeof(数组类型)

资料整理
火龙果软件
荐

例5.7 用sizeof确定数组的大小

```
#include <iostream>
using namespace std;
void main()
{  int a[ ]={1,2,4,8,16};
  for(int i=0; i<(sizeof(a)/sizeof(int)); i++)
  cout <<a[i] <<" ";
  cout <<endl;
}
```

运行结果为：

1 2 4 8 16

资料整理
火龙果软件
荐

数组越界

- 数组的大小是固定的，使用数组时，要注意不能越界。

  如：char sal_codes[5]={‘a’,’b’,’c’,’d’,’e’};

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

- c++编译器对于数组越界是不报错的。

  如程序中有语句：exemptions[6]=65;
  不会出现编译错误，但此操作可能造成不可预料的错误。

- 防止数组越界操作的责任就落在编程者的身上。

资料整理
火龙果软件
荐

一维数组的举例

例: 5个整数排列顺序，从小到大排列输出。

1)定义存放5个数的变量

```
int a[5];
```

2)先从5个数中找到最小，放到a[0]中。

```
for (i=1;i<5;i++)
if(a[0]>a[i])  swap(a[0],a[i]);
```

3)再从4个数中找到最小，放到a[1]中。

```
for (i=2;i<5;i++)
if(a[1]>a[i])  swap(a[1],a[i]);
```

资料整理
火龙果软件
荐

一维数组的举例

例: 5个整数排列顺序，从小到大排列输出。

算法的实现使用二重循环:

```
for (j=0;j<4;j++)
    for (i = j+1;i<5;i++)
        if(a[j]>a[i]) swap(a[i],a[j]);
```

资料整理
火龙果软件 荐

理解多维数组(5.7)

- 一维数组是多个数值的单列表示，而多维数组则是数值的表格，甚至多表格表示，它具有多个下标值，最常用的表格是二维表格(具有两个下标)。

资料整理
火龙果软件
荐

例子

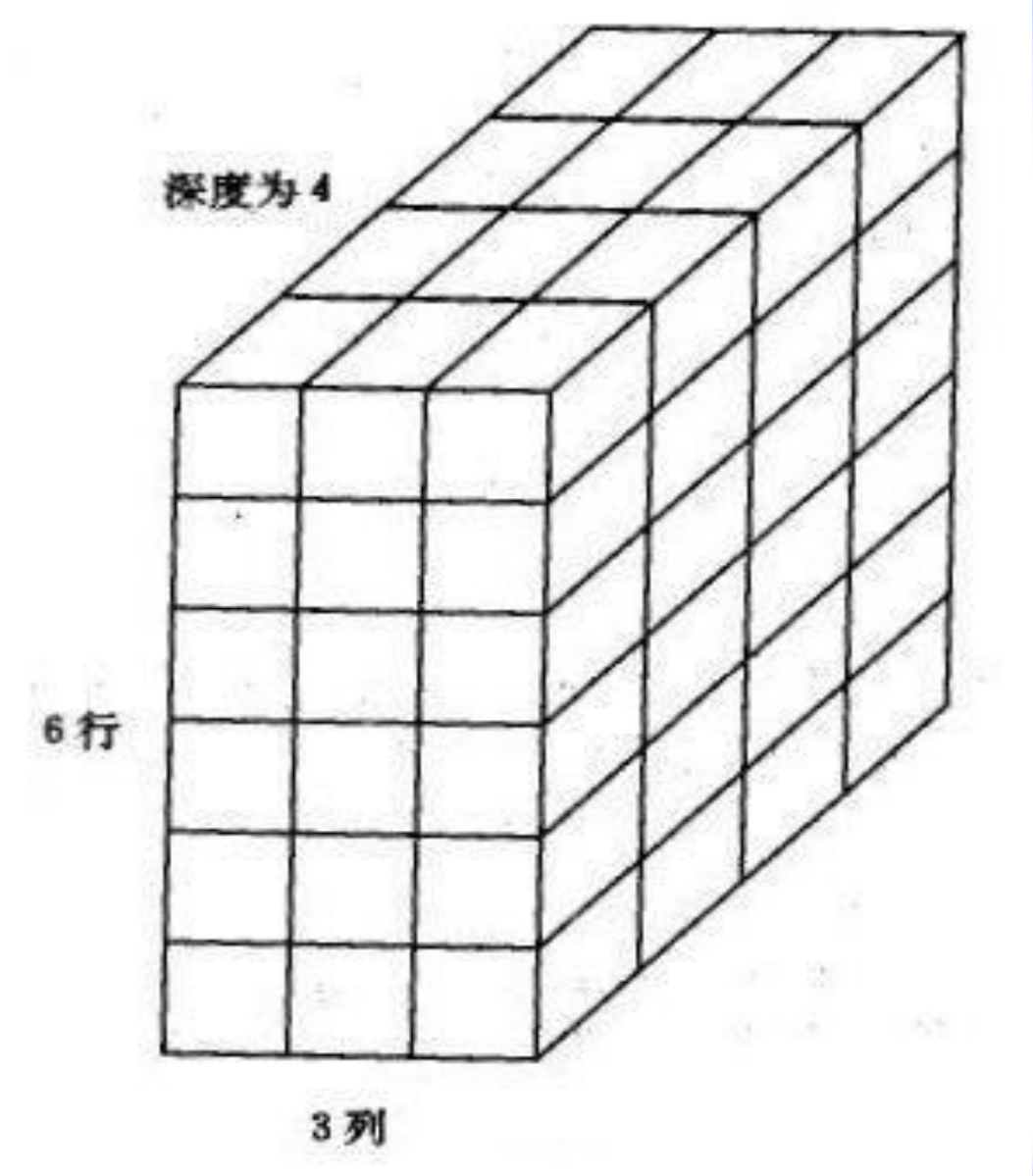
- 假定要记录一个垒球队中每个队员的击球数。队中有6个队员，进行了3场比赛。表中所示为击球记录。

队员姓名	1	2	3
张大明	2	1	2
李方春	3	3	3
林志松	2	2	2
崔明东	1	1	1
刘屈武	0	0	0
安度璧	1	1	1

资料整理
火龙果软件
荐

三维表的图示

- C++提供存储多维数据的能力，尽管现实世界很少碰到三维以上的情况。



资料整理
火龙果软件
荐

二维数组的定义及引用

- 二维数组的声明

  类型说明符 数组名[常量表达式][常量表达式]

  例如：float a[3]□;

  可以理解为：

  a
  - a[0]——a_{00} a_{01} a_{02} a_{03}
  - a[1]——a_{10} a_{11} a_{12} a_{13}
  - a[2]——a_{20} a_{21} a_{22} a_{23}

- 存储顺序

  按行存放，上例中数组a的存储顺序为：

  a_{00} a_{01} a_{02} a_{03} a_{10} a_{11} a_{12} a_{13} a_{20} a_{21} a_{22} a_{23}

- 引用

  例如：b[1][2]=a[2][3]/2

下标不要越界

资料整理 火龙果软件 荐

表5.2球队数组的下标

- team表具有18个元素，

  定义： int team[6][3]

元素下标值如下表

[0][0]	[0][1]	[0][2]
[1][0]	[1][1]	[1][2]
[2][0]	[2][1]	[2][2]
[3][0]	[3][1]	[3][2]
[4][0]	[4][1]	[4][2]
[5][0]	[5][1]	[5][2]

资料整理
火龙果软件
荐

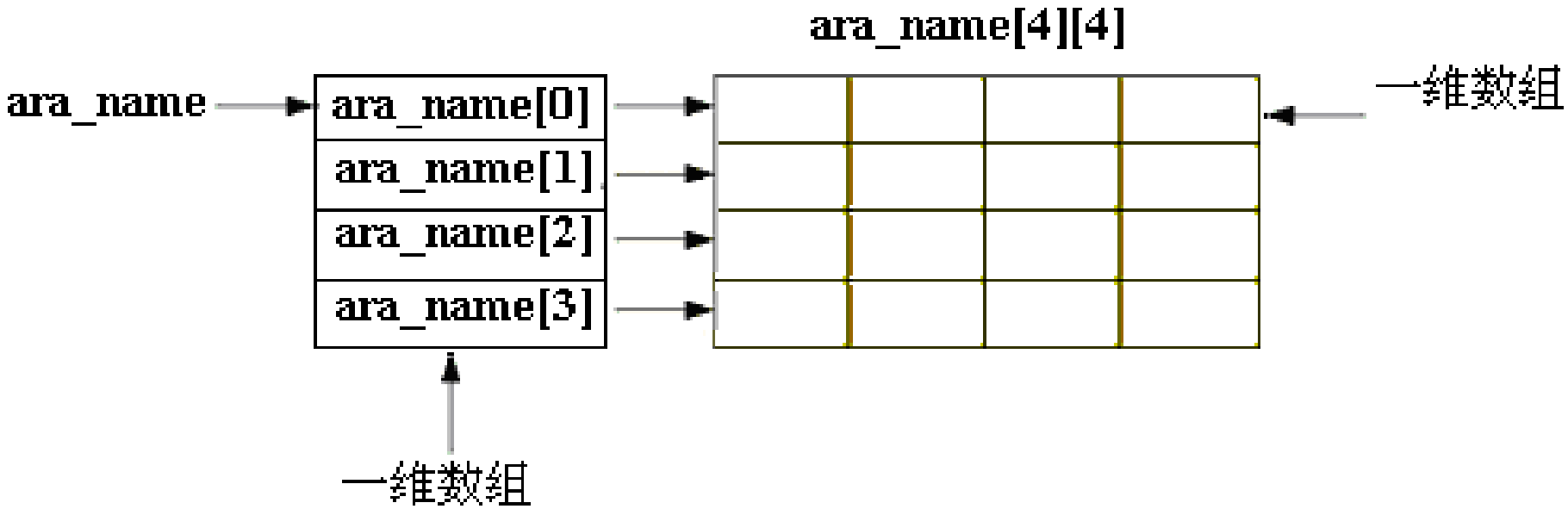
C++按行存储多维数组

- 二维数组实际上是“数组的数组”，它以行和列的形式出现，实际上还是一个一维数组，只不过数组的每个元素的类型不是整型，浮点型或字符型，而是另外一个数组。
- 数组元素也是连续存储的，按行存储，即最右边的下标变化最快，最左边的下标变化最慢。

资料整理
火龙果软件
荐

二维数组看作是一维数组的一维数组

- 一维数组的4个元素是ara_name[0]~ara_name[3]。每一个元素则是其对应的一维数组的首地址。



资料整理
火龙果软件
荐

二维数组的初始化

- 分行给二维数组赋初值

  例如：int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};

- 将所有数据写在一个{}内，按顺序赋值

  例如：int a[3][4]={1,2,3,4,5,6,7,8,9,10,11,12};

- 可以对部分元素赋初值

  例如：int a[3][4]={{1},{0,6},{0,0,11}};

  int a[3][4]={0};

资料整理
火龙果软件
荐

5.7.5表格与for循环

- 嵌套循环适用于多维数组
- 注意到for循环的循环次数与数组的下标数目相同，外层循环代表第一个下标(行下标)；内层循环代表第二个下标(列下标)。
- 嵌套for循环将遍历表中的每一个元素。

资料整理
火龙果软件
荐

例5.13嵌套for循环处理多维表

```
int main()
{for (int  row=0;row<2;row++)
    {for (int  col=0;col<3;col++)
          cout<<row<<" "<<col<<"\n";
    }
}
```

将产生如下输出结果

```
0  0
0  1
0  2
1  0
1  1
1  2
```

资料整理
火龙果软件
荐

例5.15 输出带标题的二维表格

```
void main()
{   float disk[2][4];                    //存放磁盘价格表
    int row,col;
    disk[0][0]=2.30;                     //第一行第一列
    disk[0][1]=2.75;
    disk[0][2]=3.20;
    disk[0][3]=3.50;
    disk[1][0]=1.75;
    disk[1][1]=2.10;
    disk[1][2]=2.60;
    disk[1][3]=2.95;
```

资料整理
火龙果软件
荐

```
    cout<<"\tSingle-Side,\tDouble-Side"
        <<"\tSingle-Side,\tDouble-Side\n";
    cout<<"\tDouble-density,\tDouble-density"
        <<"\tHigh-density,\tHigh-density\n";
    for (row=0;row<2;row++)    //打印表格
    {   if (row==0)
            cout<<"3 inch\t";
        else
            cout<<"5 inch\t";
        for (col=0;col<4;col++)
            cout<<"$"<<setprecision(2)
                <<disk[row][col]<<"\t\t";
        cout<<"\n";
    }
}
```

资料整理
火龙果软件
荐

程序运行结果

- 输出带标题的二维表格

	Single-Side Double-density,	Double-Side Double-density	Single-Side High-density	Double-Side High-density
3 inch	$2.30	$2.75	$3.20	$3.50
5 inch	$1.75	$2.10	$2.60	$2.95

资料整理 火龙果软件 荐

mai

{

程序

运行结果：

```
array a:
        1       2       3
        4       5       6

array b:
        1       4
        2       5
        3       6
```

}

资料整理
火龙果软

5.5 字符数组

- **字符串常量**:是用一对双引号括起来的字符序列，每个字符占一个字节，并在末尾添加’\0’作为结尾标记。例如：″china″

- 没有字符串变量，用字符数组来存储和处理字符串

资料整理
火龙果软件
荐

字符数组的定义和引用

格式： char 数组名[常量表达式]；

Eg. 字符数组：

char a [5]；

存储该数组占5个字节。每个元素的类型是字符型。数组下标从0开始，分别是a[0]，a[1]，a[2]，a[3]，a[4]。

资料整理
火龙果软件 荐

用字符数组存储和处理字符串

字符数组的初始化

对数组进行初始化赋值时，在末尾放置一个’\0′，便构成了C++字符串。

例：

```
static char str[8]={112,114,111,103,114,97,109,0};
static char str[8]={'p','r','o','g','r','a','m','\0'};
static char str[8]="program";
static char str[  ]="program";
```

static char str[7]
可否?

例：

```
static char str[7]={'p','r','o','g','r','a','m'};
```

资料整理
火龙果软件
荐

例子

```
#include<iostream>
using namespace std;
void main()
{
    char a[6]="hello";
   char b[]="program";
   char c[8]={'p','r','o','g','r','a','m','\0'};
   char d[7]={'p','r','o','g','r','a','m'};
   cout<<a<<b<<endl
       <<c<<d<<endl;
   for(int i=0;i<7;i++)   cout<<d[i];
   cout<<endl;
}
```

资料整理
火龙果软件
荐

注意

- 不要忘记为最后的’\0’分配空间。如果要初始化一个字符串“hello”，那为它定义的数组至少有6个数组元素。

- 例如，下面的代码给数组初始化，但会引起不可预料的错误：
  - ~~char array[5]=“hello”；~~
- 该代码不会引起编译错误，但由于改写了数组空间以外的内存单元，所以是危险的。

资料整理
火龙果软件
荐

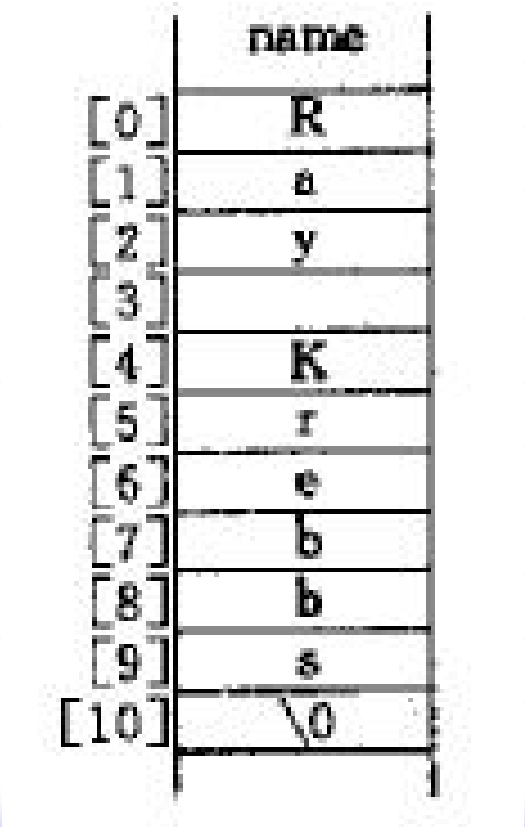
例5.9 字符数组的初始化和使用。

```
#include <iostream.h>
void main()
{   char str1[8]={112,114,111,103,114,97,109,0 };
    char str2[8]={'P','r','o','g','r','a','m','\0' };
    char str3[8]="program";
    char str4[ ]="program";
    for (int i=0;i<8;i++)
      cout<<str1[i];
    cout<<endl;
    cout<<str2<<endl;
    cout<<str3<<endl;
    cout<<str4<<endl;
}
```

资料整理
火龙果软件
荐

数组在内存中的存放

- 数组元素都是存放在一片连续的内存单元中的。
- 下图显示 char name[]="Ray Krebbs"的内存情况



资料整理
火龙果软件
荐

字符串的输入/输出

- 方法
  - 逐个字符输入输出
  - 将整个字符串一次输入或输出
    例：char c[ ]="China";
    cout<<c;
- 注意
  - 输出字符不包括 '\0'
  - 输出字符串时，输出项是字符数组名，输出时遇到'\0'结束。
  - 输入多个字符串时，以空格分隔；输入单个字符串时其中不能有空格。

资料整理
火龙果软件
荐

例如：

程序中有下列语句：

```
char str1[5],str2[5],str3[5];
cin>>str1>>str2>>str3;
```

运行时输入数据：

How are you?

内存中变量状态如下：

str1:	H	o	w	\0	
str2:	a	r	e	\0	
str3:	y	o	u	?	\0

资料整理 火龙果软件 荐

若改为：

```
static char str[13];
cin>>str;
```

运行时输入数据：

How are you?

内存中变量 str 内容如下：

str:	H	o	w	\0									

资料整理
火龙果软件
荐

字符串处理函数

strcat（连接），strcpy（复制），
strcmp（比较），strlen（求长度），
strlwr(转换为小写），
strupr（转换为大写）

头文件<string.h>

资料整理
火龙果软件
荐

字符串处理函数

```
#include <iostream.h>
#include <string.h>
void main()
{
        char str1[ ]="China";
        char str2[100];
        strcpy(str2,str1);  cout << "str2:" <<str2 << endl;
        strcat(str2,"hello"); cout << "str2:" <<str2 << endl;
        cout << "strlen(str2):" << strlen(str2) << endl;
        cout << "sizeof(str2):" << sizeof(str2) << endl;
}
```

```
China
Chinahello
10
100
```

例5.8

资料整理 火龙果软件 荐

例5.10字符串的输入和输出

输入一个字符串，输出这个字符串及字符数目。

```
#include <iostream>
using namespace std;
void main()
{ int a=0;
  char chArray[30];
  cin>>chArray;
  for(int i=0; chArray[i]!='\0'; i++)
  {cout <<chArray[i];
   a++;
  }
  cout <<endl;
  cout<<"输入的字符数是："<<a<<endl;
}
```

资料整理 火龙果软件 荐

数组作为函数参数

- 数组元素作实参，与单个变量一样。
- 数组名作参数，形、实参数都应是数组名，类型要一样，传送的是数组首地址。对形参数组的改变会直接影响到实参数组。

资料整理
火龙果软件
荐

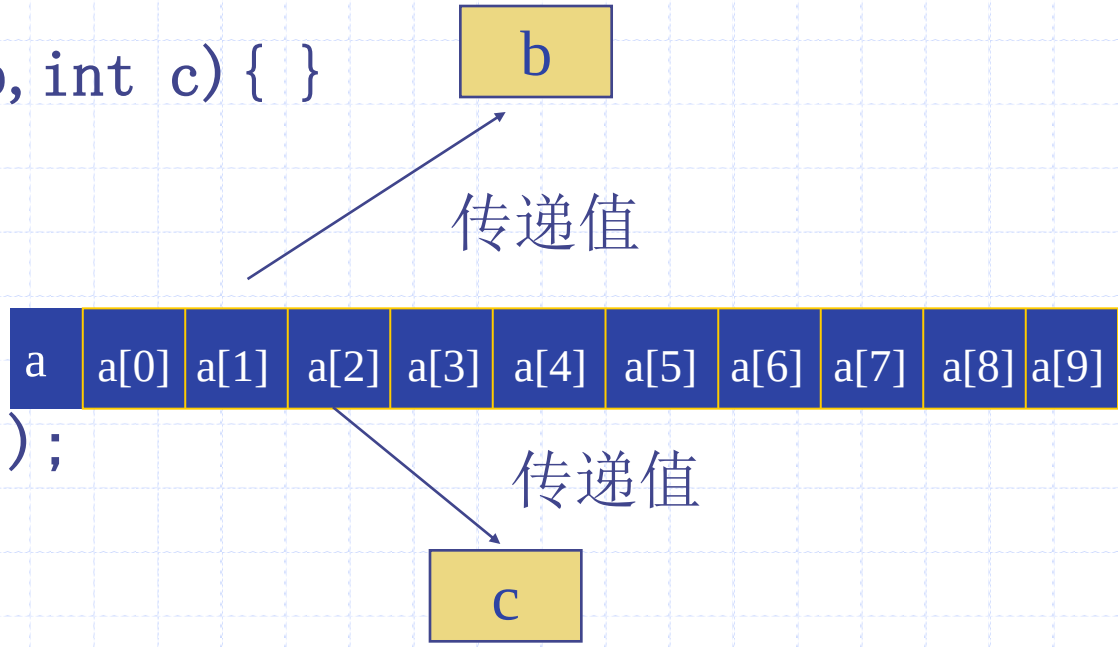
使用数组元素作为函数参数

- 定义

```
int swap( int b,int c){ }
```

- 调用

```
int a[10];
  ....
  swap(a[1],a[2]);
```



资料整理
火龙果软件
荐

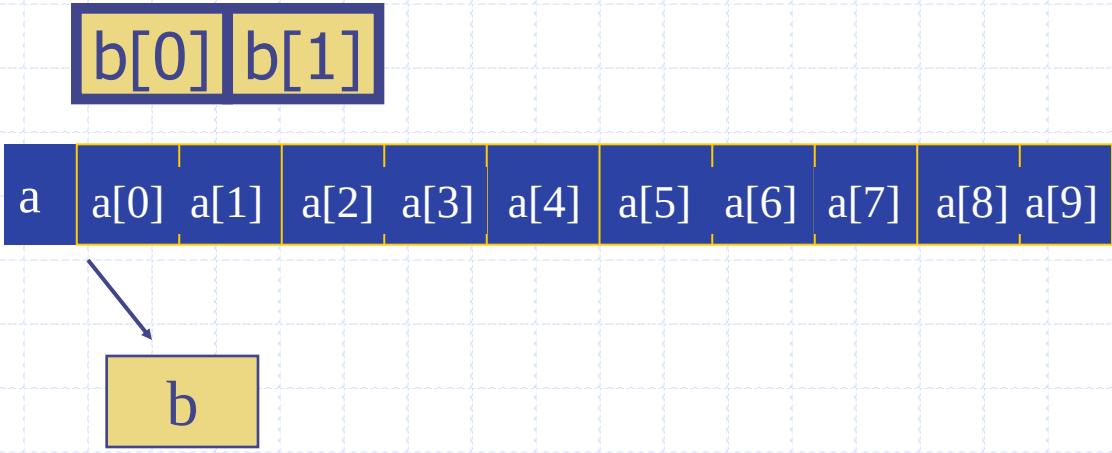
使用数组名作为函数参数

- 定义

```
int sort( int b[10]){ }
int sort (int b[], int number){ }
```

- 调用

```
int a[10];
  ....
sort(a);
sort(a,10);
```



资料整理
火龙果软件
荐

5.6 向函数传递数组

- 无论何时，将数组作为参数传给函数，实际上只是把数组的地址传给函数。
- 物理上，把整个数组放在活动记录中是不合理的，因为存储活动记录的空间大小是一定且有限的。
- 如果把传送给函数的整个数组都放在栈中(内存的大块复制)，则很快会把栈空间用光。

资料整理
火龙果软件
荐

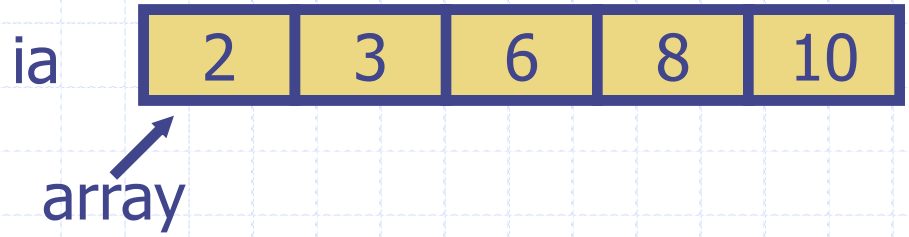
例5.12 数组名作为函数的形参

```
#include <iostream>
using namespace std;
int sum(int [ ], int);
void main()
{   static int ia[5]={2,3,6,8,10};
    int sumOfArray;
    sumOfArray= sum(ia, 5)  ;
    cout <<"数组元素的和等于:"
        <<sumOfArray <<endl;
}
```

运行结果为：

数组元素的和等于: 29

```
int sum(int array[], int len)
{   int iSum=0;
    for(int i=0; i<len; i++)
        iSum+=array[i];
    return iSum;
}
```



资料整理 火龙果软件 荐

```
#include <iostream>
using namespace std;
int sort(int [ ], int);
void main()
{  int array[5] = {8,4,6,7,3};
   for (int i = 0; i < 5; i++)
       cout << array[i] << "   ";
       cout << endl;
   sort(array);
     for (int i = 0; i < 5; i++)
       cout << array[i] << "   ";
       cout << endl;
   }
```

资料整理
火龙果软件
荐

```
Void  sort(int b[ ])
{for (int j=0;j<4;j++)
    for (int i = j+1;i<5;i++)
        if(b[j]>b[i]) {int t,t=b[j], b[j]= b[i], b[i]= t};
}
```

资料整理
火龙果软件
荐

注意

- sum()函数以整数数组作为第一个参数，以整数作为第二个参数。
- 由于传递数组名实际上传递的是地址，所以函数原型中，数组参数的书写形式无须在方括号中写明数组大小。
- 如果写明了数组大小，编译器将忽略之。数组形参的全方括号只是告诉函数，该参数是个数组的起始地址。

资料整理
火龙果软件
荐

例5-4 使用数组名作为函数参数

- 主函数中初始化一个矩阵并将每个元素都输出，然后调用子函数，分别计算每一行的元素之和，将和直接存放在每行的第一个元素中，并输出各行元素的和。

```
int Table[3][4] = {{1,2,3,4},{2,3,4,5},{3,4,5,6}};
```

资料整理
火龙果软件
荐

```
void main(void)
{ int Table[3][4] =
  {{1,2,3,4},{2,3,4,5},{3,4,5,6}};

  for (int i = 0; i < 3; i++)
  {   for (int j = 0; j < 4; j++)
          cout << Table[i][j] << "    ";
      cout << endl;
  }

  RowSum(Table,3);
}
```

资料整理
火龙果软件
荐

```
#include <iostream.h>

void RowSum(int A[ ][4], int nrow)
{ int sum;

  for (int i = 0; i < nrow; i++)
  {
      sum = 0;
      for(int j = 0; j < 4; j++)
            sum += A[i][j];
      A[i][0] = sum;
      cout << "Sum of row " << i
              << " is " << sum << endl;
  }
}
```

资料整理
火龙果软件
荐

运行结果：

```
1    2    3    4
2    3    4    5
3    4    5    6
Sum of row 0 is 10
Sum of row 1 is 14
Sum of row 2 is 18
```

资料整理
火龙果软件
荐

本章小结

- 本章介绍了如何定义和初始化一个数组。数组的初始化可以在声明部分进行，也可以在程序体中进行。
- 现在我们已经学习过了如何定义，初始化及处理多维数组，尽管并非所有数据适用于表格的紧缩格式，但它确实在很多时候非常有用。使用嵌套for循环可以遍历多维数组。
- 通过本章的学习，我们知道，通过引入数组，处理数组元素比处理相同个数的单个变量要简单得多。

资料整理
火龙果软件
荐

作业

- 第5章习题：4，11，12，15

资料整理
火龙果软件
荐