

第5章 数组

5.1 数组的概念

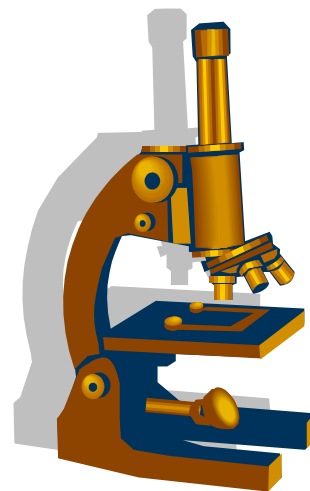
5.2 一维数组的定义和引用

5.3 二维数组的定义和引用

5.4 用数组名作函数参数

5.5 字符数组

5.6 字符串类与字符串变量



5.1 数组的概念

- 构造数据类型之一

- 数组:有序数据的集合,用数组名标识

- 元素:属于同一数据类型,用数组名和下标确定



5.2 一维数组的定义和引用

◆ 一维数组的定义

❖ 定义方式: **数据类型 数组名[常量表达式];**

[]: 数组运算符
单目运算符
优先级(2)
左结合
不能用()

例 int a[6];

合法标识符

表示元素个数
下标从0开始

数组名表示内存首地址,
是地址常量

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

编译时分配连续内存
内存字节数=数组维数*
sizeof(元素数据类型)

```
例  int i=15;  
      int data[i];
```

 (×不能用变量定义数组维数)

```
例  int data[5];  
      data[5]=10;  //C++对数组不作越界检查，使用时要注意
```

◆ 一维数组的引用

- ❖ 数组必须**先定义，后使用**
- ❖ 只能逐个引用数组元素，不能一次引用整个数组
- ❖ 数组元素表示形式：**数组名[下标]**
其中：下标可以是常量或整型表达式

```
例      int a[10];  
          cout<<a<<endl;      (×)  
必须  for(j=0;j<10;j++)  
          cout<<a[j]<<" ";      (✓)
```

◆ 一维数组的初始化

在定义数组时，为数组元素赋初值(在编译阶段使之得到初值)

❖ 初始化方式

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

等价于：a[0]=1; a[1]=2; a[2]=3; a[3]=4; a[4]=5;

❖ 说明：

- 数组不初始化，其元素值为随机数
- 只给部分数组元素赋初值

如 `int a[5]={6,2,3};`

等价于：a[0]=6; a[1]=2; a[2]=3; a[3]=0; a[4]=0;

如 `int a[3]={6,2,3,5,1};` (×)

- 当全部数组元素赋初值时，可不指定数组长度

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

编译系统根据初值个数确定数组维数

◆ 程序举例

例1 读10个整数存入数组，找出其中最大值和最小值

步骤:

1. 输入:for循环输入10个整数

2. 处理:

(a) 先令 $\text{max}=\text{min}=\text{x}[0]$

(b) 依次用 $\text{x}[i]$ 和 max,min 比较(循环)

若 $\text{max}<\text{x}[i]$,令 $\text{max}=\text{x}[i]$

若 $\text{min}>\text{x}[i]$,令 $\text{min}=\text{x}[i]$

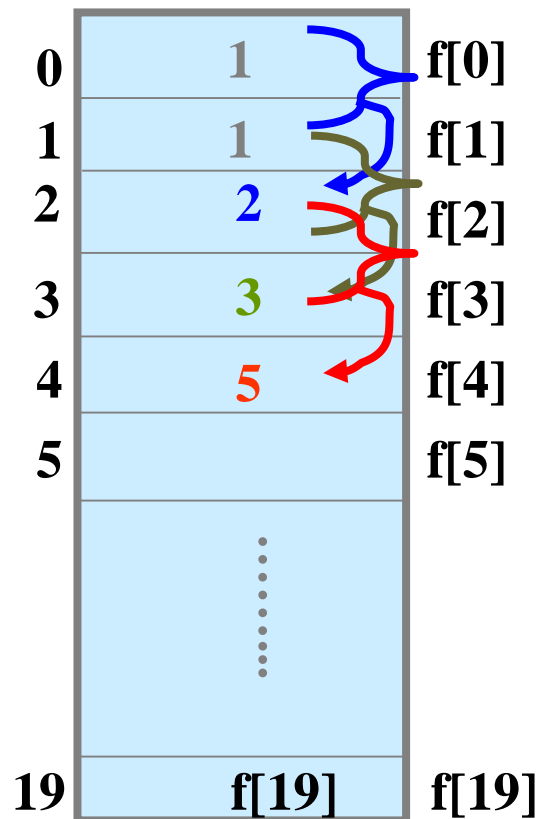
3. 输出: max 和 min

```
#include <iostream>
#define SIZE 10
using namespace std;
int main()
{   int x[SIZE],i,max,min;
    cout<<"Enter 10 integers:\n";
    for(i=0;i<SIZE;i++)   cin>>x[i];
    max=min=x[0];
    for(i=1;i<SIZE;i++)
    {   if(max<x[i]) max=x[i];
        if(min>x[i]) min=x[i];
    }
    cout<<"Maximum value is" <<max<<endl;
    cout<<"Minimum value is" <<min<<endl;
    return 0;
}
```

例 用数组求Fibonacci数列前20个数

```
#include <iostream>
using namespace std;
int main()
{   int i;
    int f[20]={1,1};
    for(i=2;i<20;i++)
        f[i]=f[i-2]+f[i-1];
    for(i=0;i<20;i++)
    {   if(i%5==0) cout<<endl;
        cout<<f[i]<<" ";
    }
}
```

$$\begin{aligned} F_1 &= 1 & (n=1) \\ F_2 &= 1 & (n=2) \\ F_n &= F_{n-1} + F_{n-2} & (n \geq 3) \end{aligned}$$



5.3 二维数组及多维数组

◆ 二维数组的定义

❖ 定义方式:

数据类型 数组名[常量表达式][常量表达式];



行数



列数

```
例 int a[3][4];  
    float b[2][5];  
    int c[2][3][4];  
    int a[3,4];        (×)
```

5.3 二维数组及多维数组

◆ 二维数组的定义

❖ 定义方式:

❖ 数组元素的存放顺序

- 原因:内存是一维的
- 二维数组: 按行序优先
- 多维数组: 最右下标变化最快

int c[2][3][4]

0	c[0][0][0]
1	c[0][0][1]
2	c[0][0][2]
3	c[0][0][3]
4	c[0][1][0]
5	c[0][1][1]
6	c[0][1][2]
7	c[0][1][3]
...	c[0][2][0]
...	c[0][2][1]
...	c[0][2][2]
...	c[0][2][3]
...	c[1][0][0]
	c[1][0][1]
	c[1][0][2]
	c[1][0][3]
	c[1][1][0]
	c[1][1][1]
	c[1][1][2]
	c[1][1][3]
20	c[1][2][0]
21	c[1][2][1]
22	c[1][2][2]
23	c[1][2][3]

int a[3][2]

a[0][0]	a[0][1]
a[1][0]	a[1][1]
a[2][0]	a[2][1]

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

❖ 二维数组理解

二维数组a是由3个元素组成

例 int a[3][4];

a[0]	2000 a[0][0] 1	2004 a[0][1] 3	2008 a[0][2] 5	200 C a[0][3] 7
a[1]	2010 a[1][0] 9	2014 a[1][1] 11	2018 a[1][2] 13	201 C a[1][3] 15
a[2]	2020 a[2][0] 17	2024 a[2][1] 19	2028 a[2][2] 21	202 C a[2][3] 23

行名

每个元素a[i]由包含4个元素的一维数组组成

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

★ 二维数组元素的引用

形式: **数组名[下标][下标]**

★ 二维数组元素的初始化

- 分行初始化:
- 按元素排列顺序初始化

第一维长度省略初始化

全部初始化

例 `int a[][3]={{1},{4,5}};`

1	0	0	4	5	0
<code>a[0][0]</code>	<code>a[0][1]</code>	<code>a[0][2]</code>	<code>a[1][0]</code>	<code>a[1][1]</code>	<code>a[1][2]</code>

例 求二维数组中最大元素值及其行列号

```
#include <iostream.h>
using namespace std;
int main()
{ int a[3][4]={{1,2,3,4}, {9,8,7,6}, {-10,10,-5,2}};
  int i,j,row=0,column=0,max;
  max=a[0][0];
  for(i=0;i<=2;i++)
    for(j=0;j<=3;j++)
      if(a[i][j]>max)
        { max=a[i][j];
          row=i;  column=j;
        }
  cout<<"max="<<max<<"row="<< row<<"column="<<column<<endl;
  return 0;
}
```

5.4 数组作为函数参数

◆ 数组元素作函数实参——**值传递**

◆ 数组名作函数参数

● **地址传递**

● 在主调函数与被调函数分别定义数组,且类型应一致

● 形参数组大小(多维数组第一维)可不指定

● 形参数组名是**地址变量**

例 求学生的平均成绩

```
#include <iostream>
using namespace std;
float average(int stu[10], int n);
int main()
{
    int score[10], i;
    float av;
    cout<<"Input 10 scores: \n";
    for( i=0; i<10; i++ )
        cin>>score[i];
    av=average(score,10);
    cout<<"Average is: " <<av<<endl;
    return 0;
}
```

形参用数组定义, \Leftrightarrow int stu[]

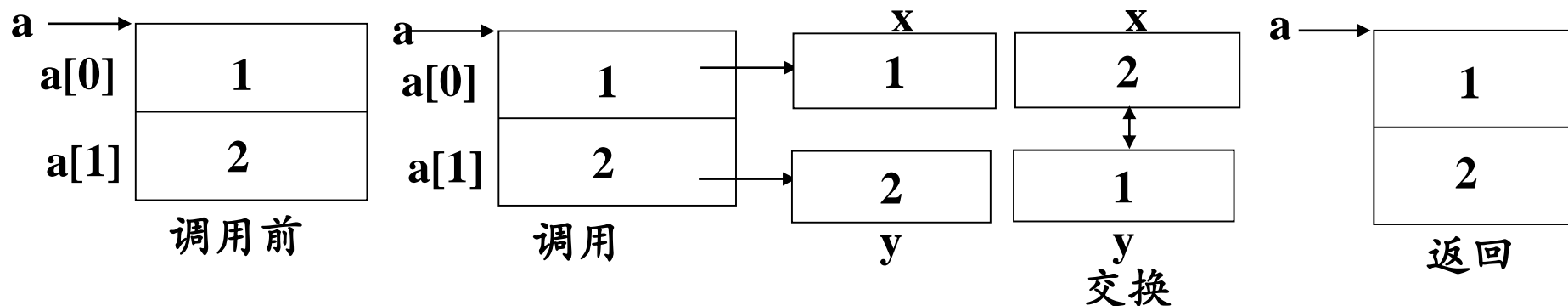
```
float average(int stu[10], int n)
{ int i;
  float av,total=0;
  for( i=0; i<n; i++ )
      total += stu[i];
  av = total/n;
  return av;
}
```

实参用数组名

例 数组元素与
数组名作函数
参数比较

值传递

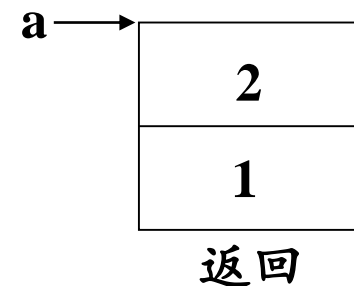
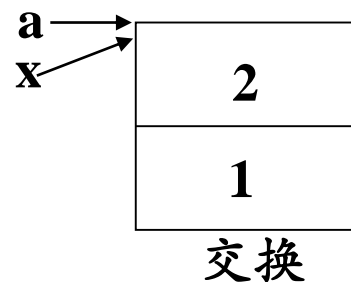
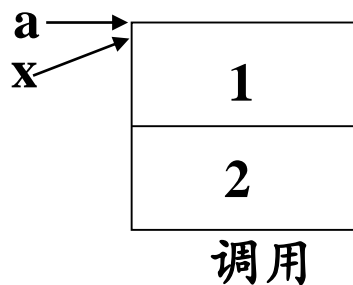
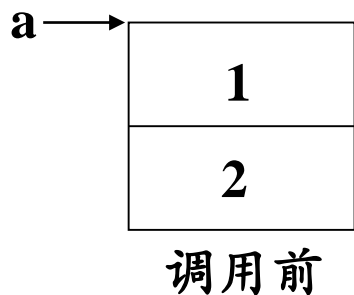
```
#include <iostream>
using namespace std;
void swap2(int x, int y)
{ int z; z=x; x=y; y=z; }
void main( )
{ int a[2]={1,2};
  swap2(a[0],a[1]);
  cout<<"a[0]="<< a[0]<< " \na[1]="<< a[1];
}
```



例 数组元素与
数组名作函数
参数比较

地址传递

```
#include <iostream>
using namespace std;
void swap2(int x[])
{ int z; z=x[0]; x[0]=x[1]; x[1]=z; }
void main( )
{ int a[2]={1,2};
  swap2(a);
  cout<<"a[0]="<< a[0]<< " \na[1]="<< a[1];
}
```



5.3 字符数组和字符串

★ 字符数组

❖ 定义 例: `char c[10], ch[3][4];`

❖ 字符数组的初始化

- 逐个字符赋值
- 用字符串常量

用字符串常量


例 `char ch[6]={“Hello”};`
`char ch[6]=“Hello”;`
`char ch[]=“Hello”;`

H	e	l	l	o	\0
ch[0]	ch[1]	ch[2]	ch[3]	ch[4]	ch[5]

二维字符数组初始化

例 `char diamond[][5]={{'.','!','*'},{'!','*','!','*'},
{'!','*','!','*'},{'*','!','!','!','*'},{'!','*','!','*'},{'!','!','*'}};`

diamond[0]	.	.	*	\0	\0
diamond[1]	.	*	.	*	\0
diamond[2]	*	.	.	.	*
diamond[3]	.	*	.	*	\0
diamond[4]	.	.	*	\0	\0



二维字符数组初始化

例 `char fruit[][7]={“Apple”,“Orange”,
“Grape”,“Pear”,“Peach”};`

fruit[0]	A	p	p	l	e	\0	\0
fruit[1]	O	r	a	n	g	e	\0
fruit[2]	G	r	a	p	e	\0	\0
fruit[3]	P	e	a	r	\0	\0	\0
fruit[4]	P	e	a	c	h	\0	\0

```
#include <iostream>
using namespace std;
void main()
{ char diamond[][5]={{' ',' ','*'},{' ','*',' ','*'},
                     {'*',' ',' ',' ','*'},{' ','*',' ','*'},{' ',' ','*'}},i;
  for(i=0;i<5;i++)
  {
    for(int j=0;j<5;j++)
      cout<<diamond[i][j];
    cout<<endl;
  }
}
```

◆ 字符串

- 无字符串变量，用字符数组处理字符串
- 字符串结束标志

例 “hello”共5个字符，在内存占6个字节 字符串长度5

h	e	l	l	o	\0
104	101	108	108	111	0

内存存放字符ASCII码

❖ 字符串的输入输出

- 逐个字符输入输出
- 整个字符串一次输入或输出

例 逐个字符

```
void main()
{   char str[5];
    int i;
    for(i=0;i<5;i++)
        cin>>str[i];
    for(i=0;i<5;i++)
        cout<< str[i];
}
```

例 整个字符串

```
void main()
{   char str[5];
    cin>> str;
    cout<< str;
}
```

用字符数组名
输入串长度<数组维数
遇空格或回车结束
自动加 '\0'

用字符数组名,
遇 '\0' 结束

```
例 void main( )  
    { char a[5]={‘H’,’e’,’l’,’l’,’o’};  
      cout<<a;  
    }
```

0	1	2	3	4
h	e	l	l	o

结果: Hello#-=*

输出时，遇 ‘\0’ 结束

```
例 void main( )  
    { char a[ ]=“Hello”;  
      cout<<a;  
    }
```

结果: Hello

例

```
void main()
{
    char a[]={'h','e','l','\0','l','o','\0'};
    cout<<a;
}
```

输出：hel

h	e	l	\0	l	o	\0
---	---	---	----	---	---	----

数组中有多个 '\0' 时,
遇第一个结束

◆ 常用的字符串处理函数 包含在头文件string中

● 字符串连接函数strcat

格式: `strcat(字符数组1, const 字符数组2)`

功能: 把字符数组2连到字符数组1后面

返回值: 返回字符数组1的首地址

说明: ① 字符数组1必须足够大

② 连接前,两串均以 '\0' 结束;

连接后,串1的 '\0' 取消,新串最后加 '\0'

```
例: char a1[50]="Hello", a2[ ]=" World!";  
    strcat(a1,a2); -----> a1="Hello World!"
```

◆ 字符串拷贝函数strcpy

- 格式: **strcpy**(字符数组1,const 字符数组2)
- 功能: 将字符串2, 拷贝到字符数组1中去
- 返回值: 返回字符数组1的首地址
- 说明:
 - ◆ 字符数组1必须足够大
 - ◆ 拷贝时 ‘\0’ 一同拷贝
 - ◆ 不能使用赋值语句为一个字符数组赋值

```
例 strcpy(str1,"China");  
char str1[20],str2[20];  
str1={"Hello!"};           (×)  
str2=str1;                  (×)
```

◆ 字符串比较函数strcmp

- 格式: `strcmp(字符串1, 字符串2)`
- 功能: 比较两个字符串
- 比较规则: 对两串从左向右逐个字符比较 (ASCII码), 直到遇到不同字符或 '`\0`' 为止
- 返回值: 返回int型整数, a. 若字符串1 < 字符串2, 返回负整数
b. 若字符串1 > 字符串2, 返回正整数
c. 若字符串1 == 字符串2, 返回零
- 说明: 字符串比较不能用 "`==`", 必须用strcmp

```
if(str1>str2) cout<<"yes";           // X
```

```
if(strcmp(str1,str2)>0) cout<<"yes"; // ✓
```

◆ 字符串长度函数strlen

格式: strlen(字符数组)

功能: 计算字符串长度

返回值: 返回字符串实际长度,

不包括 '\0' 在内

```
void main()
{   char  str[5]="qwe";
    cout<< sizeof(str)<<endl;
    cout<<strlen(str )<<endl;
}
```

运行结果:

5 // (字节数包括 '\0')

3 // 字符个数

例 对于以下字符串, strlen(s)的值为:

(1) char s[10]={ 'A','\0','B','C','\0','D'};

(2) char s[]="\\t\\v\\\\0will\\n";

(3) char s[]="\\x69\\082\\n";

答案: 1 3 1

◆ 应用举例 输入一行字符，统计其中有多少个单词

```
#include <iostream>
using namespace std;
void main()
{   char string[81];
    int i, num=0, word=0;   char c;
    gets(string);
    for(i=0;(c=string[i])!='\0';i++)
        if(c==' ') word=0;
        else if(word==0)
            {   word=1; num++;   }
    cout<<"There are"<< num <<"words\n";
}
```

5.6 C++处理字符串的方法——字符串类与字符串变量

◆ 字符串类型(string类型)

● C++标准库中声明的字符串类，可定义字符串对象。

● 定义字符串变量

- ◆ 字符串变量必须先定义后使用
- ◆ 定义字符串变量要用类名string。
- ◆ 使用string类功能时，须包含C++标准库string头文件
- ◆ 不需指定长度，长度随其中的字符串长度而改变。

```
string str1;  
string str2="OK";
```

5.6 C++处理字符串的方法——字符串类与字符串变量

◆ 字符串类型(string类型)

● 对字符串变量的赋值

- 可以用赋值语句对它赋予一个字符串常量
- 可以用一个字符串变量给另一个字符串变量赋值

● 字符串变量的输入输出

```
string str1;  
str1="Canada"; ✓  
char str2[10];  
str2="Hello!"; // ✗
```

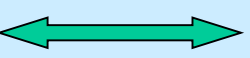
```
cin>>str1;  
cout<<str1;
```


● 字符串变量的运算(string类对象)

- ◆ 字符串复制用赋值号
- ◆ 字符串连接用加号
- ◆ 字符串比较直接用关系运算符

✚ 直接用 ==、>、<、!=、>=、<= 等关系运算符来进行字符串的比较

例如：

```
string string1=" C++" ;  
string string2=" Language" ;  
string1=string2;  strcpy(string1,string2);  
  
string1=string1 + string2; // " C++ Language"
```

◆ 归纳：

✚ C++对字符串的处理有两种方法：

✚ 一种是用字符数组的方法，一般称为C string方法；

✚ 一种是用string类定义字符串变量，称为string方法。

✚ string方法概念清楚，使用方便，最好采用这种方法。

C++保留C-string方法主要是为了与C兼容，使以前用C写的程序能用于C++环境。

作业：

P_{151} 4, 5

提示：可以定义数组时进行初始化
如： `int a[5]={8,6,5,4,1};`