



清华大学

腾讯会议 ID: 894 461 851

Tsinghua University

计算机程序设计基础

第12讲 指针和数组

沈瑜 (010-62782951)

shenyu@tsinghua.edu.cn

清华大学电机系

2022.11.29





主要内容

- 指针的算术运算和数组处理
- 多维数组和多维指针
- 指针与数组的应用举例
 - 应用1：指针用于数组处理
 - 应用2：用数组名作为函数参数

参考教材： 第8章、第9.3、9.4节





回顾

- 数组名是数组首元素地址

```
char a[]={ 'y', 'b', 'c', 'd' };  
if( a== &a[0] )  
    printf("a = %p\n", a);  
printf("a[0] = %c\n", *a);
```

```
a = 00D3F7F8  
a[0] = y  
请按任意键继续. . .
```

数组名可以看作是指针





回顾

• 数组名与常用指针区别（1）

```
void main()
{
    int a1[] = {1, 2, 3, 4};
    int a2[] = {1, 2, 3, 4};
    int*p;
    a2 = a1; // 错误用法!
    p = a1;
    p = a2;
}
```

错误列表

2 个错误

0 个警告

0 个消息

说明

2 IntelliSense: 表达式必须是可修改的左值

1 error C2106: "=": 左操作数必须为左值

数组名是常量

不可以给a2赋值，以上程序将不能通过编译。



回顾

• 数组名与常用指针区别（2）

```
void main()
{
    int a[] = {1, 2, 3, 4};
    int*p = a;

    printf( "a size is %2d\n", sizeof(a) );
    printf( "p size is %2d\n", sizeof(p) );
}
```

sizeof结果不一样

sizeof(a), 求得的是为数组a分配的存储空间

sizeof(p), 求得的是为变量p分配的存储空间

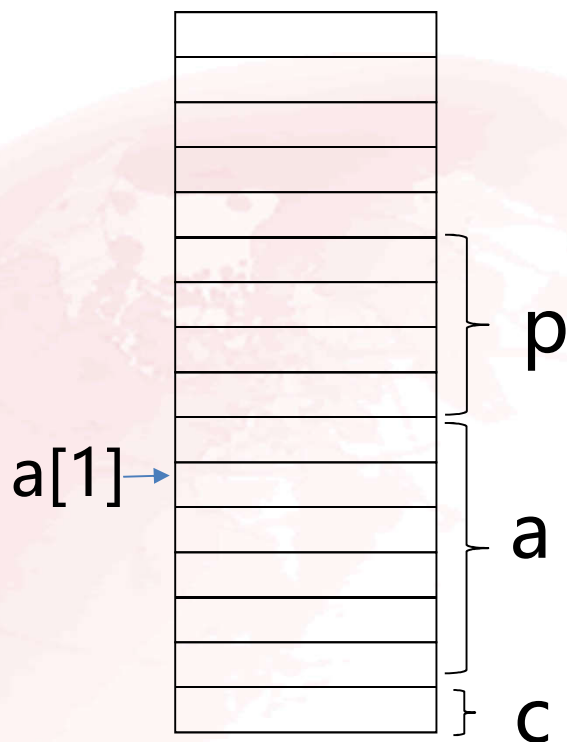


12.1 指针的算术运算和数组处理

1. 回顾: p指向c

```
char c, a[6];  
char *p = &c;
```

程序



内存分配

- p指向c

- p是指针变量
- p中存储了c的地址

含义

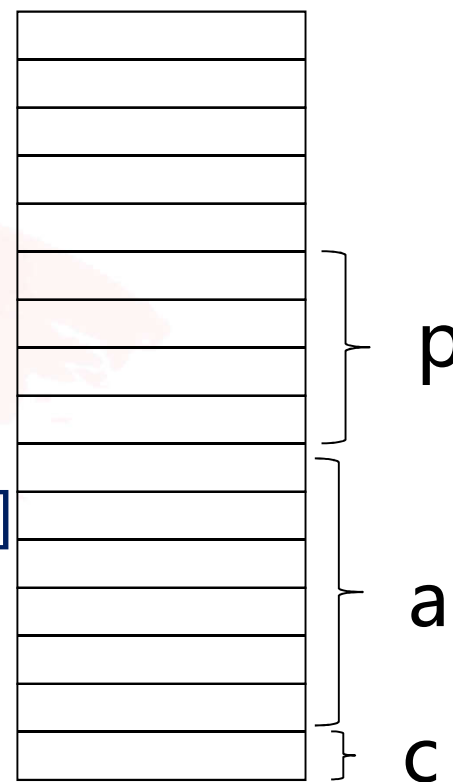
p可以指向a[1]吗?

比较: p指向a[1]

```
char c, a[6];  
char *p = &c;  
  
p = &a[1];
```

程序

a[1]



内存分配

可指向a[5]吗?



```
void main() {  
    char c, a[6];  
    char *p = &c;  
  
    p = &a[5];  
}
```

p指向a[5]

- 数组中元素数目可能非常多。
- 指向数组的任一元素
- 都要依靠取地址来实现吗？



也可通过指针前后移动来实现

- 指针往后移动：
 - 指针与整数相加
 - 例： $p+1$ ，指针往后挪动一个元素（表明：指针指向数组下一元素）
- 指针往前移动
 - 指针与整数相减
 - 例： $p-1$ ，指针往前挪动一个单位（表明：指针指向数组上一元素）

指针与整数相加/减，
不是对地址加减！



例1: p指向a[5]

通过取地址实现

```
void main() {  
    char c, a[6];  
    char *p = &c;  
  
    p = &a[5];  
}
```

通过前后移动实现

```
void main() {  
    char c, a[6];  
    char *p = &c;  
  
    p = &a[1];    指向a[1]  
    p = p + 4;    再后移4个元素  
}
```



指针前后移动：指针算术运算

- 指针后移：与整数相加
 - $p+n$;
 - $p+n$ 指向？
- 指针前移：与整数相减
 - $p-n$;
 - $p-n$ 指向？

指针可以做乘法和除法吗？

不可以。
做乘法和除法没有明确的物理含义。

指针还可以做哪些算术运算呢？



p++, p--

- p++
p = p + 1
- p--
p = p - 1

Watch一下吧!
p, 5

*p--, 什么含义?

*p++, 什么含义?

```
void main( ) {  
    int a[] = { 1, 2, 3, 4, 5 };  
    int *p = a;  
  
    p++;  
    printf("%3d\n", *p );  
    p--;  
    printf("%3d\n", *p );  
    *p--;  
    printf("%3d\n", *p );  
}
```

```
2  
1  
-858993460  
请按任意键继续.
```

监视 1		
名称	值	类型
p, 5	0x0067f768	int *
[0]	2	int
[1]	3	int
[2]	4	int
[3]	5	int
[4]	-858993460	int


***p++**

运算符结合次序：**自右向左**


相当于：*(p++)

(*p)++

- 即：
- 表达式的值：***p**
- 运算后, 指针加**1**

 **p = p+1**

- 即：
- 表达式的值：***p**
- 运算后, 指针指向内容加**1**

 **(*p) = (*p) + 1**



指针相减：指针算术运算

例2：p2-p1值是？

- 指针相减：

✚ p2 - p1

✚ 求 p2 与 p1 之间相隔了多少个元素

- 如果p2在p1后面，
p2 - p1值是正还是负？
值为正

```
typedef struct {  
    double len[10], w;  
} ST;
```

```
void main() {  
    ST a[6];  
    ST *p1, *p2;  
  
    p1 = &a[1];  
    p2 = p1 + 4;  
    printf("p1 = %p\n", p1);  
    printf("p2 = %p\n", p2);  
    printf("p2-p1 = %d\n", p2-p1);  
}
```

```
p1 = 00DBF9C0  
p2 = 00DBFB20  
p2-p1 = 4
```

请按任意键继续



小结——算术运算三种形式

- 与整数相加
 - $p+n$
 - 指针后移 n 个元素
- 与整数相减（指针前移）
 - $p-n$
 - 指针前移 n 个元素
- 指针相减（指针间距离）
 - $p2-p1$
 - 指针间相隔多少个元素

1. 针对数组
2. 单位为元素个数



例3：程序输出结果是？

```
void main() {  
    char a[6];  
    char *p1, *p2;  
  
    p1 = &a[1];  
    p2 = &a[5];  
    printf("%d", p2-p1);  
}
```

```
void main() {  
    char a[6];  
    char *p1, *p2;  
  
    p1 = &a[1];  
    p2 = p1 + 4;  
    printf("%d", p2-p1);  
}
```

```
typedef struct {  
    double len[10],w;  
} ST;
```

```
void main() {  
    ST a[6];  
    ST *p1, *p2;  
  
    p1 = &a[1];  
    p2 = &a[5];  
    printf("%d", p2-p1);  
}
```

```
void main() {  
    int a[6];  
    int *p1, *p2;  
  
    p1 = &a[1];  
    p2 = &a[5];  
    printf("%d", p2-p1);  
}
```

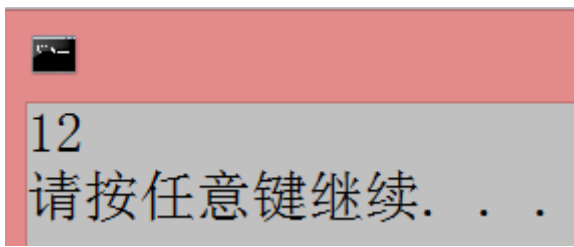

```
void main() {  
    int a[6];  
    int *p1, *p2;  
  
    p1 = &a[1];  
    p2 = p1 + 4;  
    printf("%d", p2-p1);  
}
```



例4：程序输出结果是？

```
void main()
{
    int a[10] = {11, 12, 13};
    int*p = a;

    p=p+2;
    p=p-1;
    printf( "%d\n", *p );
}
```



12
请按任意键继续. . .

这样可以吗?

- `p++;`
- `p--;`
- `p+=n;`
- `p-=n;`
- `p[i];`
- `p = p+1;`
- `p = p - 1;`
- `p = p + n;`
- `p = p - n;`
- `*(p+i)`



除算术运算外，还可通过比较运算判断前后关系

- $>$, \geq

- ✚ $p2 > p1$

- ✚ 若成立，表明 $p2$ 指向元素的地址位于 $p1$ 指向元素后面

- $<$, \leq

- $==$, $!=$

- 判断两个指针是否指向同一位置



例5：什么含义?这段代码是否有错？

```
void main( ) {  
    FILE *stream = fopen( "data", "r" );  
  
    if ( stream == NULL ) // 比较运算  
        printf( "The file 'data' was not opened\n" );  
    else  
        printf( "The file 'data' was opened\n" );  
  
    fclose(stream);  
}
```

错误：若文件未打开，指针为空，
也会执行调用fclose函数，**导致运行错误**



正确写法

```
void main( ) {  
    FILE *stream = fopen( "data", "r" );  
  
    if ( stream == NULL ) // 比较运算  
        printf( "The file 'data' was not opened\n" );  
    else {  
        printf( "The file 'data' was opened\n" );  
        fclose(stream);  
    }  
}
```

不足之处:

异常情况与正常情况
混在一起, 可读性不好



推荐写法

```
// 最佳实践
void main( ) {
    FILE *stream = fopen( "data", "r" );

    if ( stream == NULL ) { // 先判断是否有异常
        printf( "The file 'data' was not opened\n" );
        return;           // 出了异常立刻返回
    }

    // 主要代码
    printf( "The file 'data' was opened\n" );
    fclose(stream);
}
```

例6：思考

- 若指针指向的是变量，而不是数组元素，
可以对其进行前后移动的操作吗？
- 语法上可以
但运行时会导致未可知的结果





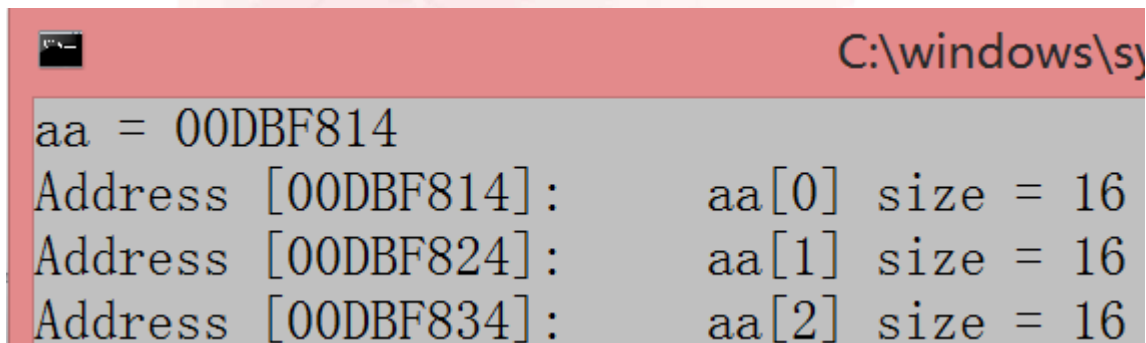
12.2 多维数组和多维指针

1. 学习目标：理解下表

名称	例子	含义
二维数组	<code>int aa[3][4]</code>	一个数组，其每一个元素又是一个数组
二维指针	<code>int **pp</code>	一个指针，指向另一个整型指针
数组指针	<code>int (*p)[4]</code>	一个指针，指向长为4的数组
指针数组	<code>int *p[3]</code>	一个数组，其每一个元素又都是一个整型指针

2. 详解：理解aa[i]

```
void main( ) {  
    int aa[3][4];  
  
    printf("aa = %p\n", aa);  
    for (int i=0; i<3; i++)  
    {  
        printf("Address [%p]:\t", &aa[i][0]);  
        printf("aa[%1d] size = %2d\n",  
            i, sizeof(aa[i]) );  
    }  
}
```



```
C:\windows\system32\cmd.exe  
aa = 00DBF814  
Address [00DBF814]:      aa[0] size = 16  
Address [00DBF824]:      aa[1] size = 16  
Address [00DBF834]:      aa[2] size = 16
```

- **aa[i]**，实质：
长为4的数组
 - 有4个元素
 - 每个元素都是整数
 - aa[i] 为这个数组首元素的地址
 - &aa[i][0]
 - &aa[i]

3. 详解：理解 &p0

```
void main( ) {  
    int aa[3][4];  
  
    int *p0 = aa[0] ;  
    int *p1 = aa[1] ;  
    int *p2 = aa[2] ;  
  
    printf("&aa = %p \n", aa);  
    printf("&p0 = %p \n", &p0);  
}
```

- p0 是指针变量
 - 指向整数 aa[0][0]
- &p0 是指针变量的地址
- 若将 &p0 记录在变量 pp 中，则：
 - 需通过两次间接寻址才能找到原始变量 aa[0][0]
 - 先找到 pp
 - 通过 pp 找到 p0
 - 通过 p0 找到 aa[0][0]

4. 详解：理解 `int **pp`

```
void main( ) {  
    int aa[3][4];  
  
    int *p0 = aa[0] ;  
    int *p1 = aa[1] ;  
    int *p2 = aa[2] ;  
  
    int **pp = &p0;  
    printf("pp = %p", pp);  
}
```

- `pp` 是一个指向指针变量的指针
 - 通过两次间接访问才能找到原始变量 `aa[0][0]`
 - 称`pp`为二维指针
 - 用两个星号作为标记

两个星号，表明二维指针
表明需要进行二次间接寻址

5. 详解：理解 `int aa[3][4]`

```
void main( ) {  
    int aa[3][4];  
  
    printf("aa = %p\n", aa);  
    for (int i=0; i<3; i++)  
        printf("aa[%1d] = %p\n",  
            i, aa[i]);  
}
```

- `aa` 是一个长为3的数组
 - `aa`有三个元素
 - 每个元素又都是一个数组
 - `aa[0],aa[1],aa[2]`
 - `aa[0]`：指针



6. 详解：理解 `int (*pp)[4]`

```
void main( ) {  
    int aa[3][4];  
  
    int *p0 = aa[0] ;  
    int *p1 = aa[1] ;  
    int *p2 = aa[2] ;  
  
    int (*pp)[4] = aa;  
    printf("pp = %p", pp);  
}
```

- 根据结合律，**pp**是一个指针
- 这个指针指向什么呢？
 - 将 ***pp** 用变量**a**代替，得到？
 - **int a[4]**
- **pp**指向一个长为**4** 的整数数组
- **aa** 正是长为**3**的数组的首元素的地址。
 - 首元素是长为**4**的整数数组
- 可以将**aa**赋给**pp**

指向数组的指针



6. 详解：理解 `int *pp[3]`

```
void main( ) {  
    int aa[3][4];  
    int *pp[3];  
  
    pp[0] = aa[0] ;  
    pp[1] = aa[1] ;  
    pp[2] = aa[2] ;  
}
```

- **pp**是一个数组
 - **pp**的长度为3
 - **pp[i]**:存储整数变量的地址
- **aa[i]**, 代表整数变量 **aa[i][0]** 的地址
- 可将**aa[i]** 赋给 **pp[i]**





12.3 指针与数组 应用举例

- 应用1：指针用于数组处理
 - 可通过指针移动指向数组任意元素
 - 因此，可通过指针对于数组元素进行处理
- 应用2：数组名作为函数参数
 - 数组名就是指针
 - 因此，数组名作为函数参数，相当于以指针作为函数参数



1.应用1： 指针用于数组处理

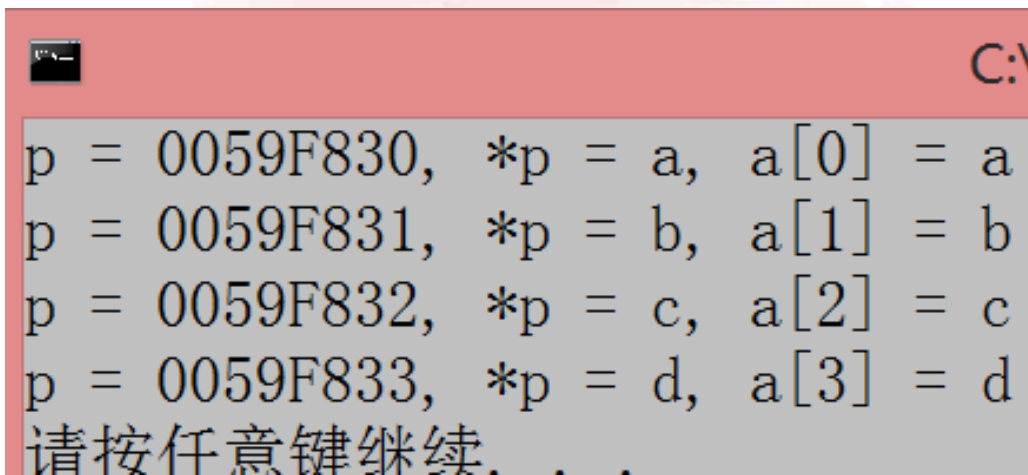
- Question:
- p 是指针， 那么
 - ✚ $p+i$ 相当于 $\&p[i]$?
 - ✚ $*(p+i)$ 相当于 $p[i]$?

$p+i$ 相当于 $\&p[i]$
 $*(p+i)$ 相当于 $p[i]$



例1：程序输出是？

```
void main( ) {  
    char a[] = { 'a', 'b', 'c', 'd', 'e', 'f' };  
    char *p = &a[0];  
  
    for (int i=0; i <4; i++) {  
        printf("p = %p, *p = %c, \  
a[%1d] = %c \n",  
            p+i, *(p+i), i, a[i] );  
    }  
}
```



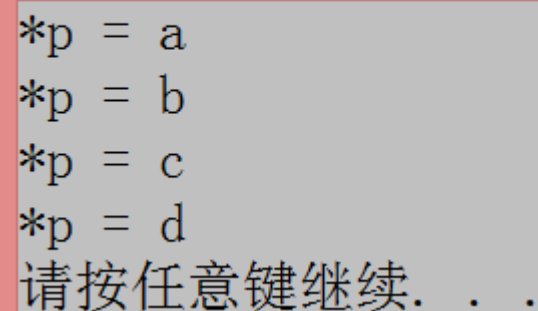
```
C:\  
p = 0059F830, *p = a, a[0] = a  
p = 0059F831, *p = b, a[1] = b  
p = 0059F832, *p = c, a[2] = c  
p = 0059F833, *p = d, a[3] = d  
请按任意键继续.
```

- p?
 - 指向a的首元素
- p+i ?
- *(p+i) ?
- 程序输出是？

由于可通过指针移动
指向数组任意元素，
因此，也可通过指针
对数组元素进行处理

例2：程序输出是？

```
void main( ) {  
    char a[] = {'a', 'b', 'c', 'd'};  
    char *p = &a[0];  
  
    for (int i=0; i <4; i++) {  
        printf("*p = %c \n", *p);  
        p++;  
    }  
}
```



```
*p = a  
*p = b  
*p = c  
*p = d  
请按任意键继续. . .
```

1.应用2：用数组名作为函数参数

- 要点：与用指针作为函数参数一样

例1：求整个数组最大元素

```
int findMax(int *p, int len)
{
    int k = 0; //
    for ( int i=1; i<len; i++ )
        if ( p[i] > p[k] ) k=i;
    return p[k];
}

void main()
{
#ifdef DEBUG
    int a[] = { 1, 3, 9, 10, 21, 35, 99, 10 };
    int max = findMax(a, sizeof(a)/sizeof(a[0]));
    assert ( max == 99);
#endif
}
```

详解1：数组名
作为参数

a是指针，因此
对应参数p也定
义为指针

不建议写成：

```
int findMax(int p[],int len)
这样定义的p依然是指针
```

详解1：指针的算术运算

```
#define DEBUG

int findMax(int *p, int len)
{
    int k = 0; //
    for ( int i=1; i<len; i++ )
        if ( p[i] > p[k] ) k=i;
    return p[k];
}

void main()
{
    #ifdef DEBUG
        int a[] = { 1, 3, 9, 10, 21, 35, 99, 10 };
        int max = findMax(a, sizeof(a)/sizeof(a[0]));
        assert ( max == 99);
    #endif
}
```

回顾：

p是指针，p[k] 即相当于 *(p+k)

详解2：数组长度的处理

```
#define DEBUG

int findMax(int *p, int len)
{
    int k = 0; //
    for ( int i=1; i<len; i++ )
        if ( p[i] > p[k] ) k=i;
    return p[k];
}

void main()
{
    #ifdef DEBUG
        int a[] = { 1, 3, 9, 10, 21, 35, 99, 10 };
        int max = findMax(a, sizeof(a)/sizeof(a[0]));
        assert ( max == 99);
    #endif
}
```

数组长度可以不传过来吗？

这样对吗？

```
int findMax(int *p )
{
    int len = sizeof(p)/sizeof(p[0]);
    int k = 0; //
    for ( int i=1; i<len; i++ )
        if ( p[i] > p[k] ) k=i;
    return p[k];
}

void main()
{
#ifdef DEBUG
    int a[] = { 1, 3, 9, 10, 21, 35, 99, 10 };
    int max = findMax( a );
    assert ( max == 99);
#endif
}
```

也不可以

p是指针，不是数组。

因而sizeof(p)的值为4.

不管使用哪种方式传递数组，都不能在函数内部直接求得数组长度！

有变通方法吗？

- 有
- 可以在数组中加入特殊元素
 - 从数组首元素，依次往后进行处理
 - 当碰到特殊元素时，即表明数组元素全部处理完毕
- 对字符数组，为不将数组长度作为参数，一般使用 ‘\0’ 作为特殊元素
- 以 ‘\0’ 结束的字符数组，称为字符串





12.4 函数指针

1. 指向函数的指针

- 函数名，表示函数的地址

- 指向函数的指针

例如：

```
float fun(int, char);
```

```
float (* p)();
```

```
p = fun;
```



2.用函数指针变量调用函数

可用函数指针来调用函数.

其形式为:

(*函数指针变量名)(<实参表>)

例如:

```
int i=5;  
char ch='a';  
float fun(int, char), (*p)( );  
p=fun;  
(*p)(i, ch); //或(p)(i, ch);  
...
```



```
int i=5;  
char ch='a';  
float fun(int, char);  
fun(i, ch);
```



例：批量测试数学函数

```
void main()
{
    double x=0.5, y;
    int i;

    double (*pFunctions[7])(double) =
    { sin, cos, tan, exp, log, log10, sqrt };
    char strFunctions[][20] =
    { "sin", "cos", "tan", "exp", "log", "log10", "sqrt" };
    int nNum = sizeof(pFunctions)/sizeof(double*);

    for( i=0; i< nNum; i++ )
    {
        y = (pFunctions[i])( x );
        printf( "%s(x) = %lf\n", strFunctions[i], y );
    }
    return;
}
```

```
sin(x) = 0.479426
cos(x) = 0.877583
tan(x) = 0.546302
exp(x) = 1.648721
log(x) = -0.693147
log10(x) = -0.301030
sqrt(x) = 0.707107
请按任意键继续. . .
```

