福州大学至诚学院
FUZHOU UNIVERSITY ZHICHENG COLLEGE

高级语言程序设计（C语言与数据结构）

杨雄

83789047@qq.com

第八章 指针

8.1 指针和指针变量的概述

8.2 指向变量的指针变量

8.3 指针与数组

学习目的

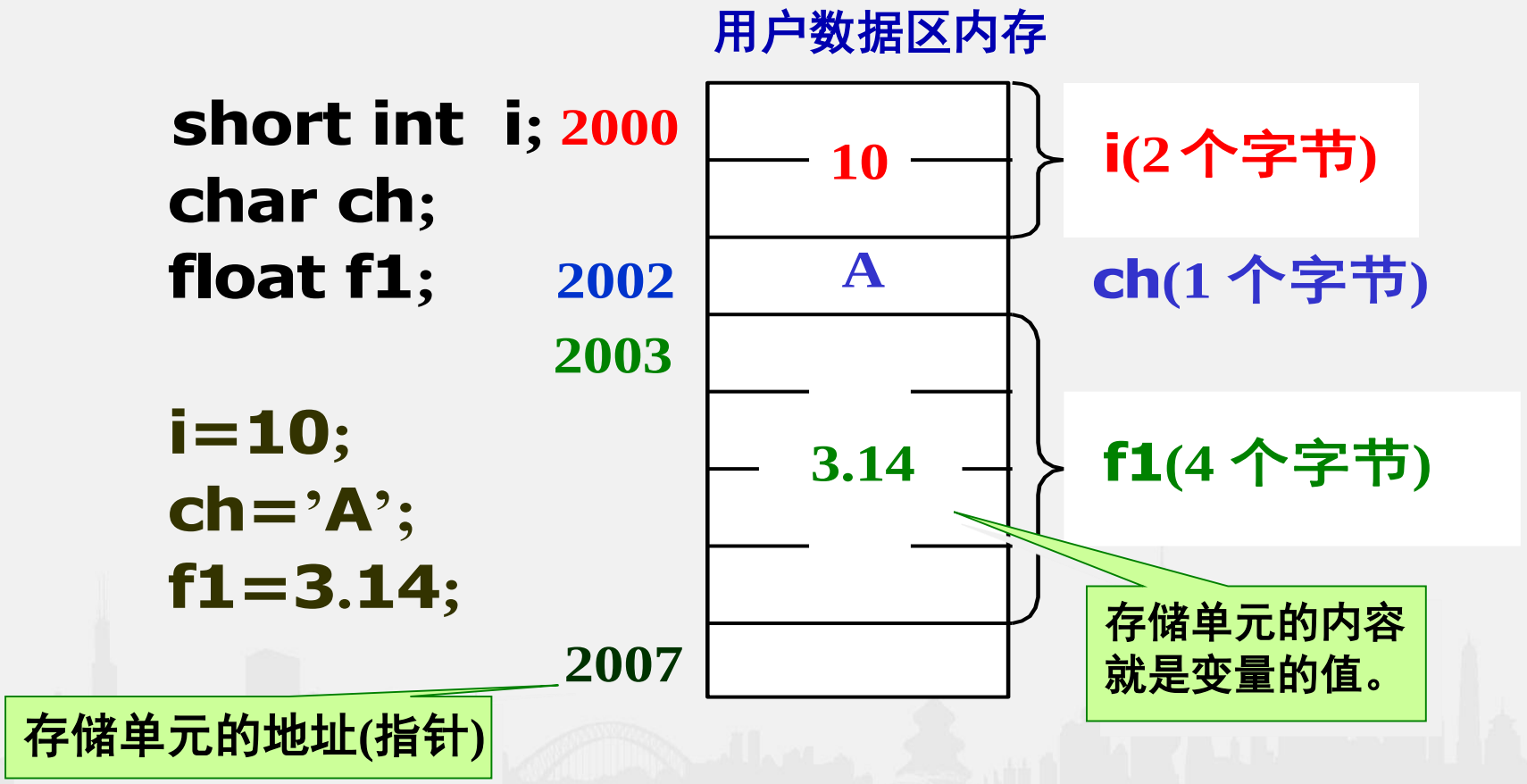
- **掌握指针和指针变量的概念、表达方式和区别**
- **掌握指针作为函数参数的使用**
- **掌握指针在数组中的应用**

Part.1

8.1 指针和指针变量的概述

8.1 指针和指针变量的概述

在程序中定义了一个变量，该变量在内存中就要占一定的存储单元，这个空间的大小由变量的类型决定。

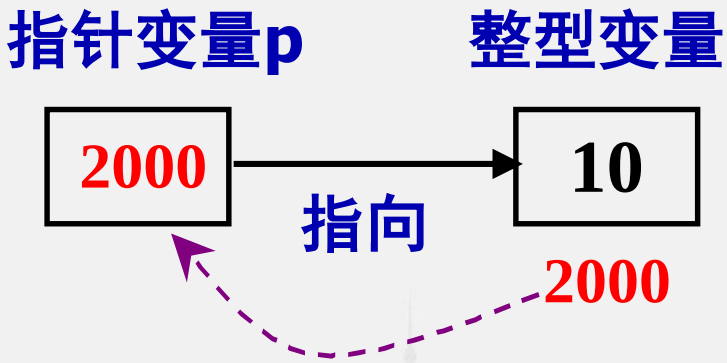


8.1 指针和指针变量的概述

- **指针**：就是变量的**内存地址**，是一个常量。
- 指针变量：就是**存放另一变量内存地址（指针）的变量**。

当把**变量 i 的地址存入指针变量 p** 后，我们就说这个**指针变量p指向变量 i**。

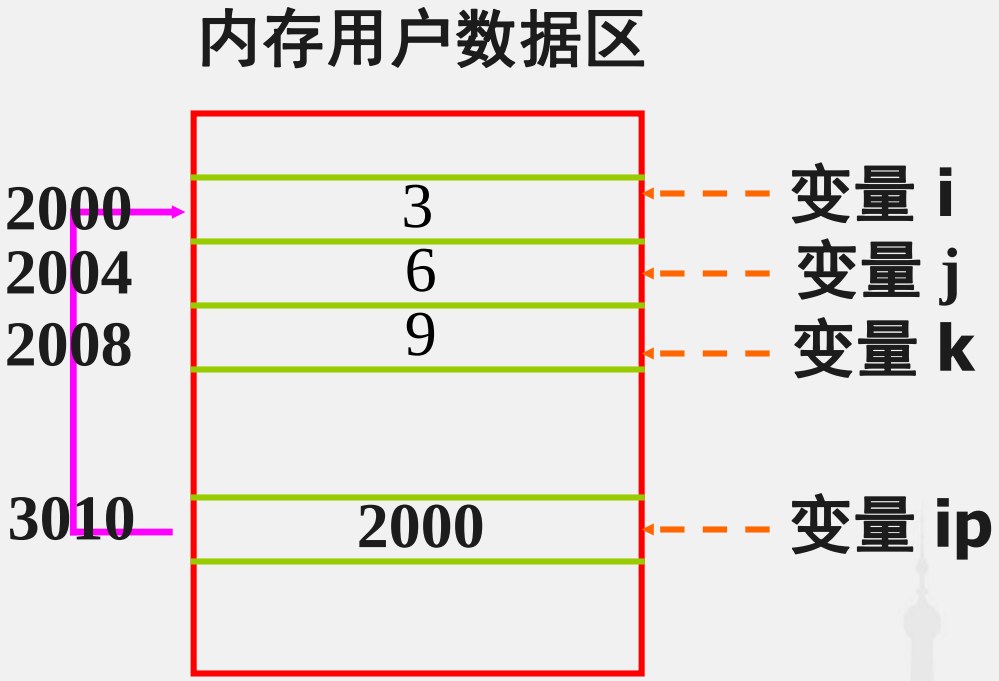
```
int  i=10,*p ;
p=&i ;
```



- 指针变量的值是某个变量的内存地址。

8.1 指针和指针变量的概述

```
int  i=3, j=6, k=9, *ip;
ip=&i ;
```

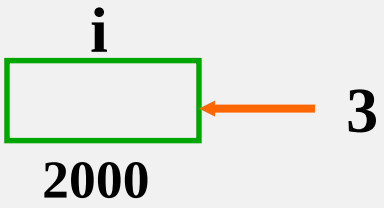


8.1 指针和指针变量的概述

◆ 对内存单元的访问方式:

- **直接访问**:
  - 直接通过**变量名**存取变量的值

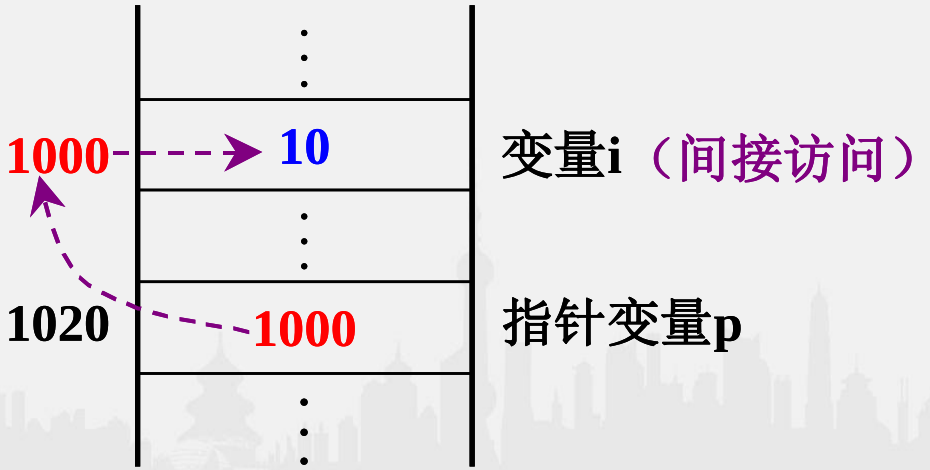
  如: i=3



- **间接访问**:
  - 通过指向某变量的**指针变量**访问

```
int  i=10, x, *p;

x=*p;
```



Part.2

8.2 指向变量的指针变量

8.2 指向变量的指针变量

8.2.1 指针变量的定义

8.2.2 指针变量的引用

8.2.3 指针变量的初始化

8.2.4 指针变量作为函数参数

8.2.1 指针变量的定义

- **格式:**

  类型说明符　*指针变量名；

  如:

  int *p1;　//定义p1为指向整型变量的指针变量

  char *p2; //定义p2为指向字符变量的指针变量

- **指针变量的类型, 是指针所指向的变量的类型，而不是自身的类型。**

8.2.1 指针变量的定义

说明:

1) **在指针定义中, 一个*号只表示定义一个指针。**

定义多个指针变量时, 每个变量前都必须有 *。

```
int  *p1, *p2 ;
```

2) **指针变量中只能存放地址 (指针), 不能和整型变量混淆。如:**

```
int  *ip;
ip = 1000;      ×
```

指针变量可取值为0(NULL), 表示该指针变量不指向任何变量。

8.2.1 指针变量的定义

说明:

3) **指针变量的类型与它所指向变量的类型必须一致。**

```
如: void main( )
  { int *p1 ;              //定义p1应指向一个整型变量
    float x=3.33333 ;
×   p1=&x ;               //将一个实型变量的地址赋给p1
    printf( "%f\n" , *p1) ;
  }
```

p1只能指向整型变量, 不能指向实型和字符型变量。

8.2.2 指针变量的引用

在C语言中，指针变量可以通过一对**互逆**的运算符进行引用。

- **取地址运算符**—— “&”
- **引用运算符** —— “*”

8.2.2 指针变量的引用

1. & 一 取地址运算符

- **作用：获取变量或数组元素的地址**

  例： `int i, *p1, *p2, a[5];`

  `p1=&i ;  p2=&a[4];`

- **注意：不能对常量、表达式进行“&”运算。**

  例： `p1=&68;` ×

  `p1=&(i+1);`

2. * — 指针运算符（间接访问运算符）

- 作用：获得该指针所指向变量的值。

  如：

```
int  i=100, *pi;
pi=&i;
printf( “%d\n” , *pi );
```

- 注意：非指针变量不能使用指针运算符，*只能作用于地址。如：printf(“%d” , *i); ×

8.2.2 指针变量的引用

可以看出：printf（“%d\n”，*pi);

等同于　　printf(“%d\n”，i);

*pi ⟺ i ，　pi ⟺ &i

因此, 可以通过pi来改变i的原始值。

如：　*pi=150;　等同于　i=150;

8.2.2 指针变量的引用

例8-1

输出结果:
*p=50
a=100
19CA 19CA 19CA 19CA

```
#include <stdio.h>
 void  main( )
{ int  a=50, *p;                    //声明整型指针变量p
   p=&a;                            //赋值给指针变量p,让p指向变量a
   printf( “*p=%d\n” , *p);   //取指针p所指地址的内容
   *p=100;
   printf( “a=%d\n” , a);
  printf( “%X %X %X %X\n” , &a, p, &*p, *&p);
}
```

*和 & 是互反的

8.2.3 指针变量的初始化

- **格式:**

  类型说明符 *指针变量名1=初始地址值，…;

如: `int i=26 ;`

`int *ip=&i;` //初始化为整型地址

不能写成: `int *ip=i ;`

`*ip= &i ;`

8.2.3 指针变量的初始化

说明:

- 指针变量在使用前必须在声明或赋值语句中初始化, 使指针变量指向一个确定的内存单元, 未经初始化的指针变量禁止使用。
- 指针变量可以被初始化为0、 NULL或某个地址。

如:

```
int  i, *ip ;
*ip=58 ;          // !
```

  - 如果使用一个没有确定值的指针变量, 不仅可能破坏你的程序, 而且可能导致系统的破坏。

8.2.3 指针变量的初始化

说明:

- 可以把一个指针的值赋给另一个指针

```
int  n ;
int  *pn=&n ;
int  *p1=pn ;
```

- 不能用常量或表达式来给指针变量赋初值

```
int  i=10, *p ;
p=&67;   p=&(i+5) ;   ×
```

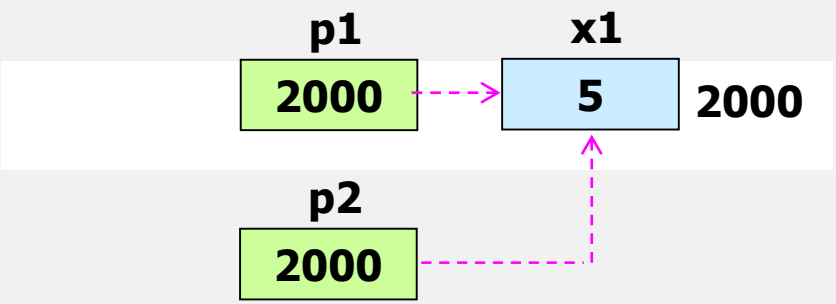
8.2.3 指针变量的初始化

例8.2

```
#include <stdio.h>
void  main( )
{ int   x1, x2, *p1, *p2= &x2 ;
  scanf( “%d%d” , &x1, p2 );
   &x2。
  p1 = &x1;
  printf( “*p1=%d,*p2=%d\n” , *p1, *p2 );
  p2 = p1 ;
 printf( “*p1=%d,*p2=%d” , *p1, *p2 );
}
```

运行结果：

```
5 4 ↙
*p1=5,*p2=4
*p1=5,*p2=5
```

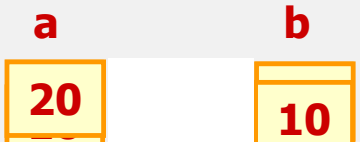


8.2.4 指针变量作为函数参数

- 特点:
  - 将一个变量的地址作为实参传送到被调用函数中。
  - 采用“单向值传递”方式（特殊的地址值）
  - 被调用函数不能改变实参指针变量的值，但可以改变实参指针变量所向指变量的值。
  - 可以得到多个返回值

8.2.4 指针变量作为函数参数

3. swap( )函数: 交换形参*x、*y的值。即a, b的值。




```
swap( int  *x, int  *y )
{ int   t;  //形参为同类型的指针变量
   t=*x;   *x=*y;   *y=t;
 }  //交换形参指针变量所指向变量的值
 void main( )
{ int  a=10, b=20;
   printf( "1)a=%d, b=%d\n" , a, b );
   swap(&a, &b);      //实参为变量的地址(指针), 地址值传递
   printf( "2)a=%d, b=%d\n" ,a,b);
}
```

8.2.4 指针变量作为函数参数

例8.8_2交换两个元素的值

```
swap( int  *x, int  *y )
{ int  *p;
   *p=*x;          //此语句有问题
   *x=*y;     *y=*p;
}
 void main( )
{ int  a = 10, b = 20;
   swap( &a, &b);
   printf( “a=%d, b=%d\n” ,  a, b);
}
```

Part.3

8.3 指针和数组

8.3 指针和数组

8.3.1 指针变量的运算

8.3.2 一维数组的指针

8.3.1 指针变量的计算

- 实质: 地址的运算
- 指针可以进行以下运算:
  - 赋值运算：给指针变量赋值
  - 算术运算:
    - 指针加(减)一个整数
    - 两个指针相减
  - 关系运算：两个指针之间的比较运算

8.3.1 指针变量的计算

2. 指针的算术运算

- 自增自减（++, --）
- 加上一个整数（+, +=, -, -=）
- 两个指针相减
- 只有当指针指向数组时，指针的加减运算才有意义。
- 指针加（减）一个整数n的意义，是使指针向前或向后移动一个或几个存储单元。
- 对于不同类型的指针变量移动的字节数是不一样的，指针移动是以它指向的数据类型所占的字节数为移动单位。

指针的算术运算

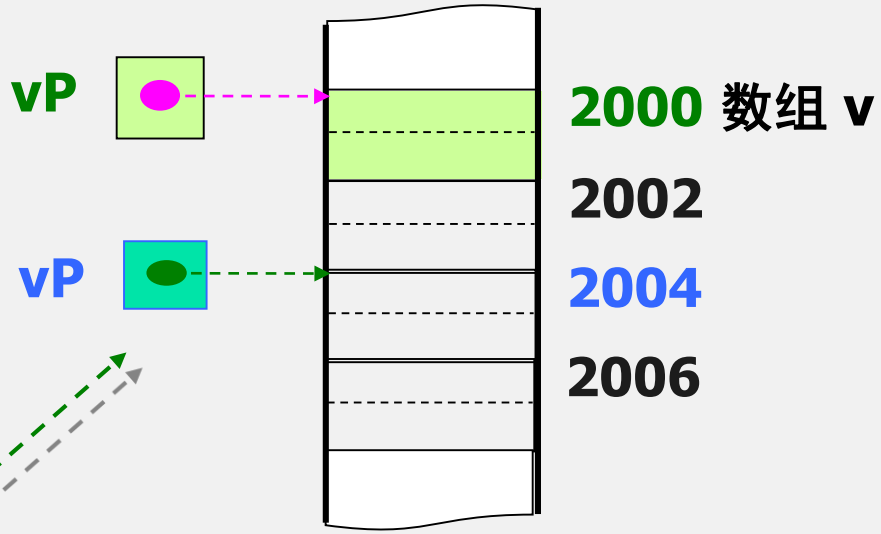
```
short v[4],*vP;
vP = v;
vP = vP + 2;
```

? vP+2

2000 + 2 = 2002

2000+2*2 = 2004

类型的长度



- 整型指针+1移动四个字节;
- 单精度指针+1移动四个字节;
- 字符型指针+1移动一个字节;

8.3.1 指针变量的计算

☞只有加法和减法可以用于指针运算

如:

```
int a[5];
int *ip=&a[1];
ip--;                    //指向a[0]
*ip=3;
ip--;                    //指向a[-1]！
*ip=6;                   // !!!
```

8.3.1 指针变量的计算

例: 移动指针变量访问数组元素

```
#include <stdio.h>
void  main( )
{ int  a[10]={1, 3, 5, 7, 9, 11, 13, 15, 17, 19};
   int  n=6, *p1, *p2 ;
   p1=p2=&a[0] ;                    //赋值给指针变量, 让p1、p2指向a数组
   printf( "1) *p1=%d, *p2=%d\n" , *p1, *p2 ) ;
   p1=p1+n ;   p2++ ;      //指针变量加/减一个整数
   printf( "2) *p1=%d, *p2=%d\n" , *p1, *p2);
   ++p1;  p1=p1-3 ;   p2=p2+7 ;
   printf( "3) *p1=%d, *p2=%d\n" , *p1, *p2) ;
}
```

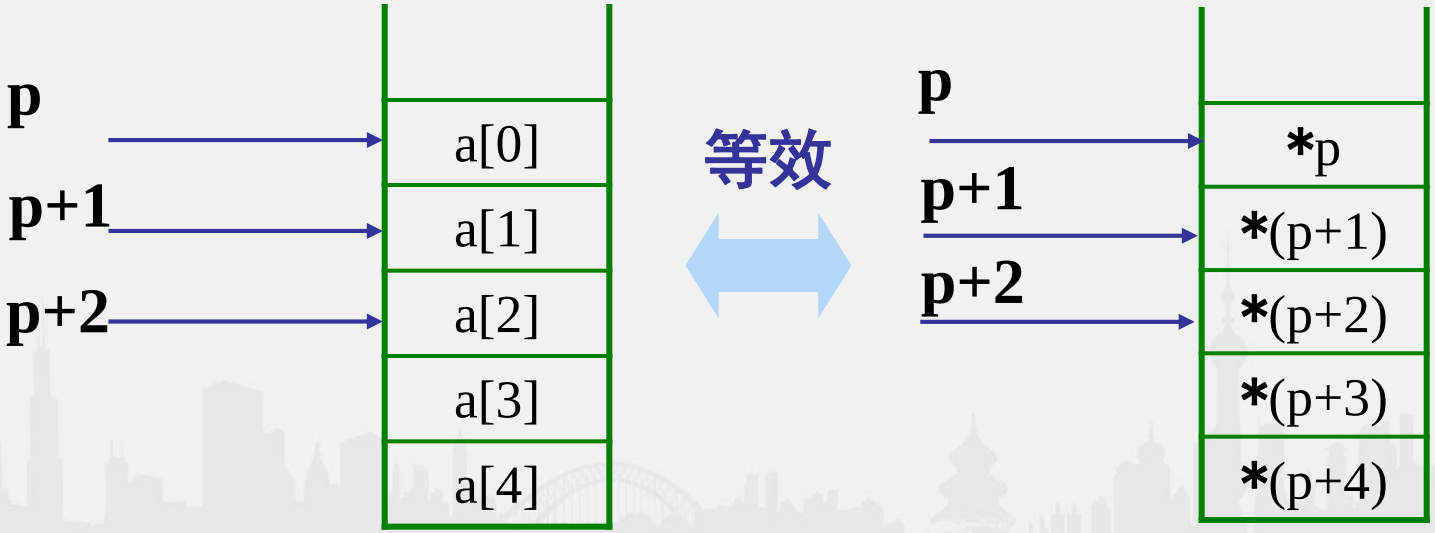
程序运行结果:

1) *p1=1, *p2=1
2) *p1=13, *p2=3
3) *p1=9, *p2=17

8.3.1 指针变量的计算

说明

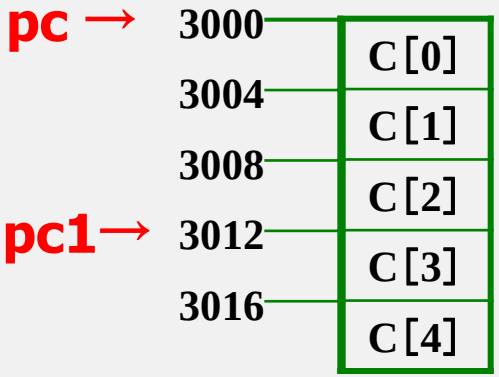
- **p=a; 这里数组名作为数组的起始地址, 即a[0]的地址。因此 p=a 等效于 p=&a[0];**
- **p=p+1; 如p指向a[0], 则p=p+1之后, p指向a[1]**
- **如果p=a 等效于 p=&a[0]; 则 p=a+4 等效于 p=&a[4];**



8.3.1 指针变量的计算

两个指针变量相减

- 当两个指针指向同一数组时，两个指针相减其差值为两个指针相隔的元素个数。




```
#include <stdio.h>
void main( )
 { int  k, c[5]={2,4,6,8,10};
    int  *pc=&c[0], *pc1= &c[3];
    k=pc1-pc;
    printf("*pc=%d,*pc1=%d\n", *pc, *pc1);
    printf("k=%d\n", k);
 }
```

输出结果：
*pc=2, *pc1=8
k=3

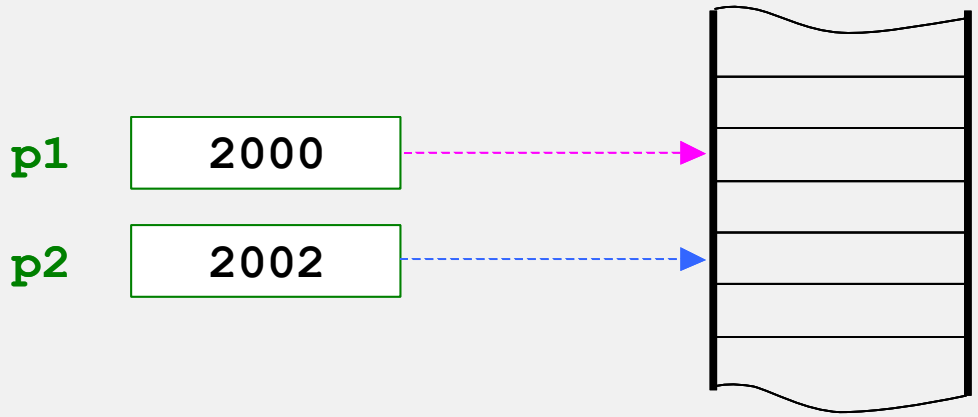
pc1和pc之间元素的个数

8.3.1 指针变量的计算

3. 两个指针变量的比较

- 比较两个指针变量包含的内存地址

```
p1 < p2
p1 > p2
p1 == p2
p1 != p2
```



比较运算常用于数组，判定两个指针变量所指向的数组元素的位置先后。

8.3.1 指针变量的计算

例1:

```
int  x=4, y, z ;
int  *px=&x ;
y=*px*2 ;                          y=8
printf( “%d\n” , *px);             4
z=sqrt((double)*px) ;              z=2
*px=0 ;                            x=0
*px+=1;                            x+=1; x=1
(*px)++ ;                          x++;  x=2
```

8.3.1 指针变量的计算

2. 下面这些关于指针的**定义、赋值和运算**的语句是否有错, 若有, 找出并改正。

```
int  *pm, *pn, *pk, *p1, *p2, m=5, n, data[8];
pm=&m ;
pn= &n ;
p1= data ;
p2=&data[7] ;
p1+p2 ;  ×
p2-p1 ;
scanf( “%d” , pn);
scanf( “%d” , pm) ;
```

8.3.1 指针变量的计算

3. 经过语句 int i, a[10], *p ; 定义后, 下列语句中合法的是:

(a) p=100 ;　　(b) p=a[5] ;

(c) p=a[2]+2 ;　　(d) p=&(i+2) ;

✓(e) p=a+2 ;

4. 若已定义x为int型变量, 下列说明指针变量p的正确语句是:

A) int p=&x ;　　B) int *p=x ;

✓C) int *p=&x ;　　D) *p= &x ;

8.3.1 指针变量的计算

5. 已知p、p1为指针变量，a为数组名，i为整型变量， 下列赋值语句中不正确的是_____ 。

A) p=&i ;　　B) p=a ;

C) p=&a[i] ;　　D) p=10 ;

6. 两个指针变量不可以 _____ 。

A) 相加　　B) 比较

C) 相减　　D) 指向同一地址

8.3.2 一维数组的指针

- **数组的指针：数组的首地址**
  - 数组名是**常量指针**
- C语言规定：数组名代表数组**第一个元素**的地址

因此，对于数组a，有：**a 等于 &a[0]**

若：　int a[10], *p ;

p=a;　等价于　p=&a[0];

//指针变量p为指向数组a的指针变量

8.3.2 一维数组的指针

- 一维数组的首地址加上偏移量 i 可以得到其他元素的地址。

  如: int a[10], *p=a;

  &a[1] 表示数组元素a[1]的地址

  等价于 a+1 或 p+1

  &a[i] 表示数组元素a[i]的地址

  等价于 a+i 或 p+i

- 注意: p+i 指向数组的a[i]元素，不是简单地使指针变量p的值加 i，其所代表的地址实际上是 p+i×size。（size为数组元素的数据类型所占的字节数）。

8.3.2 一维数组的指针

在C语言中,对数组元素的访问可用以下三种形式:

- **下标法**: 用**a[i]**的形式存取数组元素
- **地址法**: 用 ***(a+i)** 或 ***(p+i)** 的形式存取数组元素
- **指针法**: 用指针变量指向数组的首地址, 然后通过移动指针***p++**存取数组元素。

若访问下标为 i 的元素:

a[i] 等价于 *(a+i) 等价于 *(p+i) 等价于 p[i]

8.3.2 一维数组的指针

例8.5　设数组a有10个元素，试用以上三种形式存取数组中的所有数组元素。

①用下标法存取数组元素

```
#include <stdio.h>
void main( )
{ int   a[10], i , *p=a ;
  for( i=0; i<10; i++)
     scanf( "%d", &p[i] ) ;
 for( i=0; i<10; i++)
     printf( "%d ", p[i] ) ;
  printf( "\n" );
}
```

②用地址法输入输出数组各元素 (数组名+偏移量)

```
#include <stdio.h>
void main( )
{ int  a[10], i , *p=a ;
   for( i=0; i<10; i++)
        scanf( “%d”, p+i ) ;
   for( i=0; i<10; i++)
        printf( “%d ”, *(p+i) ) ;
   printf( “\n” );
}
```

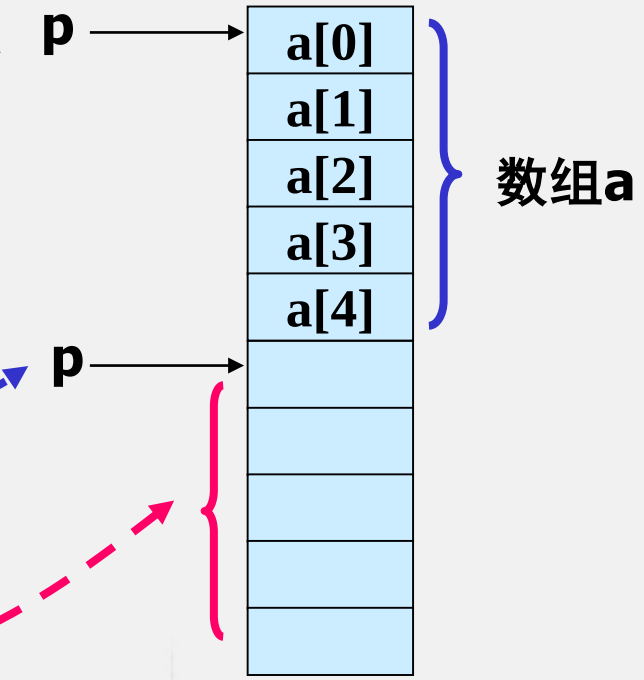
③用指针法输入输出数组各元素（移动指针）

```
#include <stdio.h>
void main( )
{  int  a[10] , i, *p;
    for( p=a;  p<a+10;  p++)
          scanf( “%d” , p );
    for( p=a;  p<(a+10);  p++ )
          printf( “%d ” , *p ) ;
    printf( “\n” );
}
```

8.3.2 一维数组的指针

例: 8_51.c

```
void  main( )
{ int  a[5], i, *p;
  p=a;
  for( i=0;  i<5;  i++)
     scanf( “%d” , p++);
  printf( “\n” ); p=a;
  for( i=0;  i<5;  i++)
     printf( “%d ” , *p++);
}
```



错

要注意指针变量的当前值。

8.3.2 一维数组的指针

- 这三种方法中，**下标法**比较直观易读。前两种方法的程序运行效率是一样的，因为[ ]实际上是变址运算符，编译时，编译系统是将数组元素**a[i]**转换为***(a+i)**处理的。

- 第3种方法是**指针移动法**，利用指针变量的自加操作，使指针顺序指向下一元素，不必每次重新计算地址，因此**执行效率高于前两种**，且编程方便。

数组和指针互换使用的注意事项：

- **数组名是一个常量指针，不允许重新赋值。**

  a+=1;　a++ ;　×

- **指针变量是一个变量，可以重新赋值。**

- **a+i ⟹ p+i 均表示a[i]的地址，均指向a[i]。**

- ***(a+i) ⟹ *(p+i)**

  均表示p+i和a+i所指对象的内容，即a[i]。

8.3.2 一维数组的指针

注意：如果先使p指向数组 a，即p=a；则:

1) **p++** (或 **p+=1**)，**p**指向下一元素，即 **p=&a[1]**。
   ***p** 取出**a[1]**的值

2) ***p++** 等价于 ***(p++)**，
   作用是：先得到***p**，再使**p=p+1**。

3) ***(++p)** 作用是：先使**p+1**赋给**p**，再取***p**。
   若**p=a;** 输出***p++**，得**a[0]**的值，
   若**p=a;** 输出***(++p)**，得**a[1]**的值。

注意：

4) **(*p)++:** 表示p所指向的变量(元素)的值+1

即等价于a[i]++

例：

```
int  a[5]={0, 2, 4, 6, 8}, *p;
p=a+2 ;
(*p)++;          //即a[2]+1
printf( “%d\n” , *p);
```

程序段输出为：**5**

注意：是元素值+1，而不是指针值+1。

注意：

5) 指向数组元素的指针变量也可以带下标，

如 *(p+i) 可以表示成 p[i]。

- 注意：如果p不指向a[0]，则p[i]和a[i]是不一样的。
- 例如： int a[10], *p ;

  p=a+3 ; //p指向a[3]

  p[3]=5 ; //p[3]相当于a[6]而不是a[3]。
- 这种方式容易出错，一般不提倡使用。

数组元素的访问小结

```
int  i, a[10], *p=a;
```

表　现　形　式	含　义
&a[0] ⇒ a ⇒ p	一维数组首地址
&a[i] ⇒ a+i ⇒ p+i	一维数组下标为 i 的元素地址
a[0] ⇒ *a ⇒ *p	一维数组下标为0的元素的值
a[i] ⇒ *(a+i) ⇒ *(p+i)	一维数组下标为 i 的元素的值

Part.4

总结

总 结

1.指针（Pointer）就是内存的地址。

C语言允许用一个变量来存放指针，这种变量称为指针变量。指针变量可以存放基本类型数据的地址，也可以存放数组以及其他指针变量的地址。

2. 指针变量可以进行加减运算。

例如p++、p+i、p-=i。指针变量的加减运算并不是简单的加上或减去一个整数，而是跟指针指向的数据类型有关。

总 结

3. 给指针变量赋值时，要将一份数据的地址赋给它，不能直接赋给一个整数。

例如int *p = 1000;是没有意义的，使用过程中一般会导致程序崩溃。

4.使用指针变量之前一定要初始化。

否则就不能确定指针指向哪里，如果它指向的内存没有使用权限，程序就崩溃了。

对于暂时没有指向的指针，建议赋值NULL。

总 结

5. 两个指针变量可以相减。

如果两个指针变量指向同一个数组中的某个元素，那么相减的结果就是两个指针之间相差的元素个数。

6. 表达式中的数组名会被转换为一个指向数组的指针。

习 题

1. 以下程序的输出结果是

```
#include <stdio.h>
 void  main( )
{ int  a[12]={1,2,3,4,5,6,7,8,9,10,11,12};
   int  *p1, *p2, i=1 ;
    p1=&a[i++*2] ;
    p2=&a[i++*2] ;
    printf( “%d,%d\n” , *p1, p2[2] );
 }
```

A) 输出项不合法　　B) 2,6　　C) 3,7　　D) 3,5

习 题

2. 若有说明 int i, j=7,*p=&i; 则与i=j;等价的语句是:

a) i=*p; b) *p=*&j; c) i=&j; d) i=**P;

习 题

3. 若定义 int a[ ]={1,2,3,4,5,6}, p=a; 则表达式 ++(*++p)的值是【3】。

4. 若有以下说明，则 a[*(a+a[3])]的值为

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

a) 8　　b) 3　　c) 0　　d) 不合法

习 题

5. 若有以下定义和语句, 则能正确表示a数组元素地址的表达式是:

```
double  a[5],*p1;
 p1=a;
```

✓a) a　　b)p1+5　　c) *p1　　d)&a[5]

习 题

6. 以下程序运行后, 输出结果是:

```
#include <stdio.h>
 void main( )
{  char  ch[5]={ “693” }, *p1;
   int  j, s=0;
   p1=&ch[0];
   for( j=0; p1[j]>= ‘0’  && p1[j]<= ‘9’ ; j+=2)
         s=10*s+p1[j];
   printf( “%d\n” , s );
}
```

A) 693　　B) 63　　C) 591　　D) 693825