

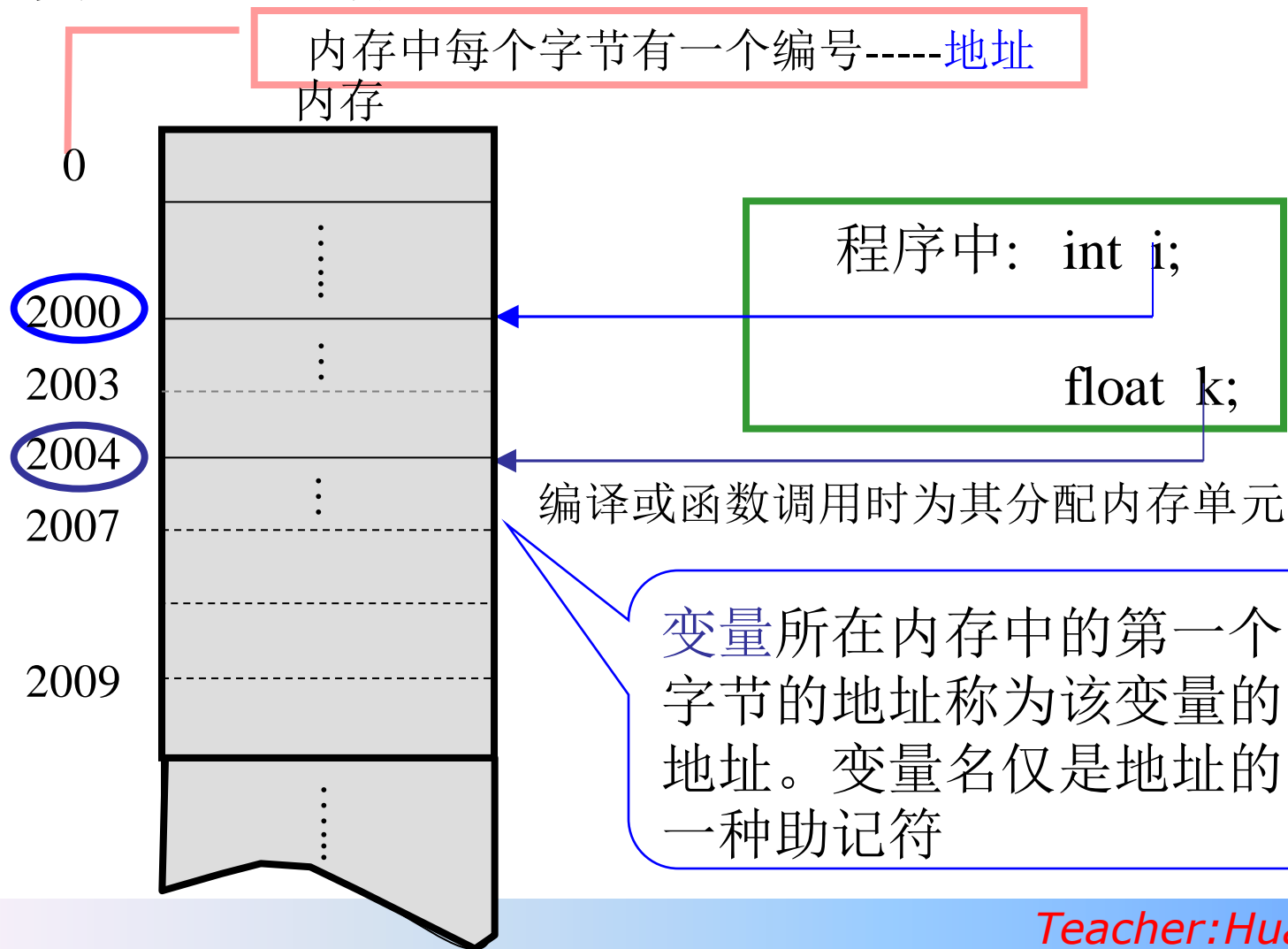
第八章

指 针



8.1 指针是什么？

➤ 变量与地址的关系



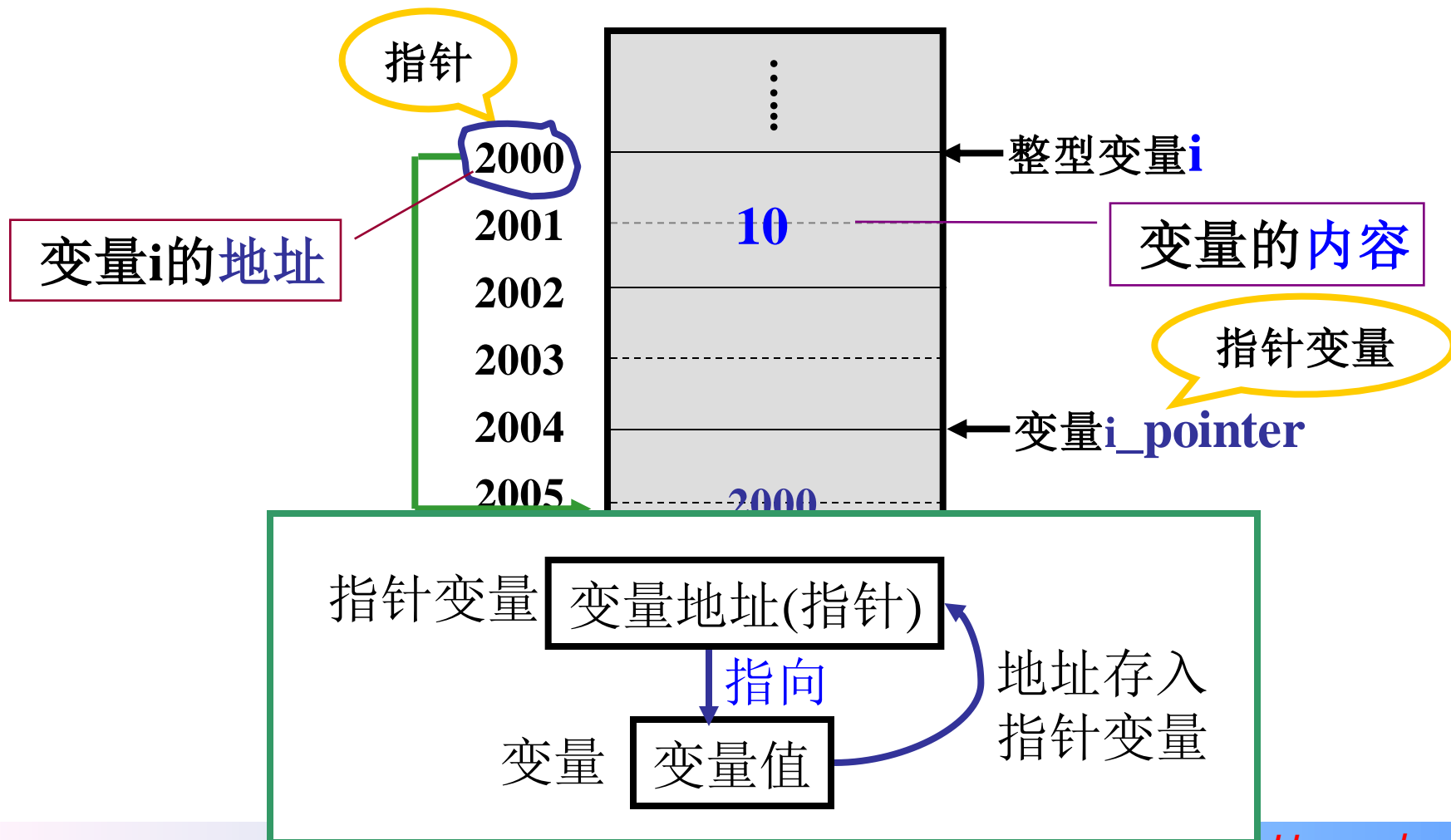
8.1 指针是什么？

➤ 指针与指针变量

- 指针：即一个变量的地址。这个地址被形象的称为指针。
- 指针变量，如果一个变量专门被用来存储另一份数据的地址（即指针），则称该变量为**指针变量**。（数据包括：普通变量、函数、数组、指针变量）。

8.1 指针是什么？

➤ 指针与指针变量



8.2 指针变量

1、如何定义指针变量

指针变量具有普通变量的三要素：名字、类型、值

❖一般形式：数据类型 *指针名；

指针变量定义的*是指针变量类型标识符
不是 float 运算符
`char *name;`

注意：

- 1、区分 `int *p1,*p;` 与 `int *p1,p;`
- 2、指针变量名是 `p1,p`, 不是 `*p1,*p`;
- 3、指针变量只能指向定义时所规定类型的变量;
- 4、指针变量在定义后，其变量值不确定，因此使用前必须先赋值;

8.2 指针变量

2、如何引用指针变量

- 要熟练掌握两个有关的运算符：

(1) & 取地址运算符。

&a是变量**a**的地址

(2) * 指针运算符（“间接访问”运算符）

如果：**p**指向变量**a**，则***p**就代表**a**。

k=*p; (把**a**的值赋给**k**)

***p=1;** (把**1**赋给**a**)

8.2 指针变量

2、如何引用指针变量

- 在引用指针变量时，可能有三种情况：
 - 给指针变量赋值。如：`p=&a;`
 - 引用指针变量所指向的变量。如有
`p=&a; *p=1;`
则执行`printf("%d",*p);`将输出1
 - 引用指针变量的值。如：`printf("%x",p);`

***p相当于a**

以16进制输出a的地址

8.2 指针变量

3、指针变量的初始化（即赋值）

一般形式：数据类型 *指针名=初始化的地址值；

例 int i;
 int *p=&i;

赋给指针变量，

例 int i;
例 int *p=&i;
 int *q=p;

×

用已初始化的指针变量来
初始化另一个指针变量

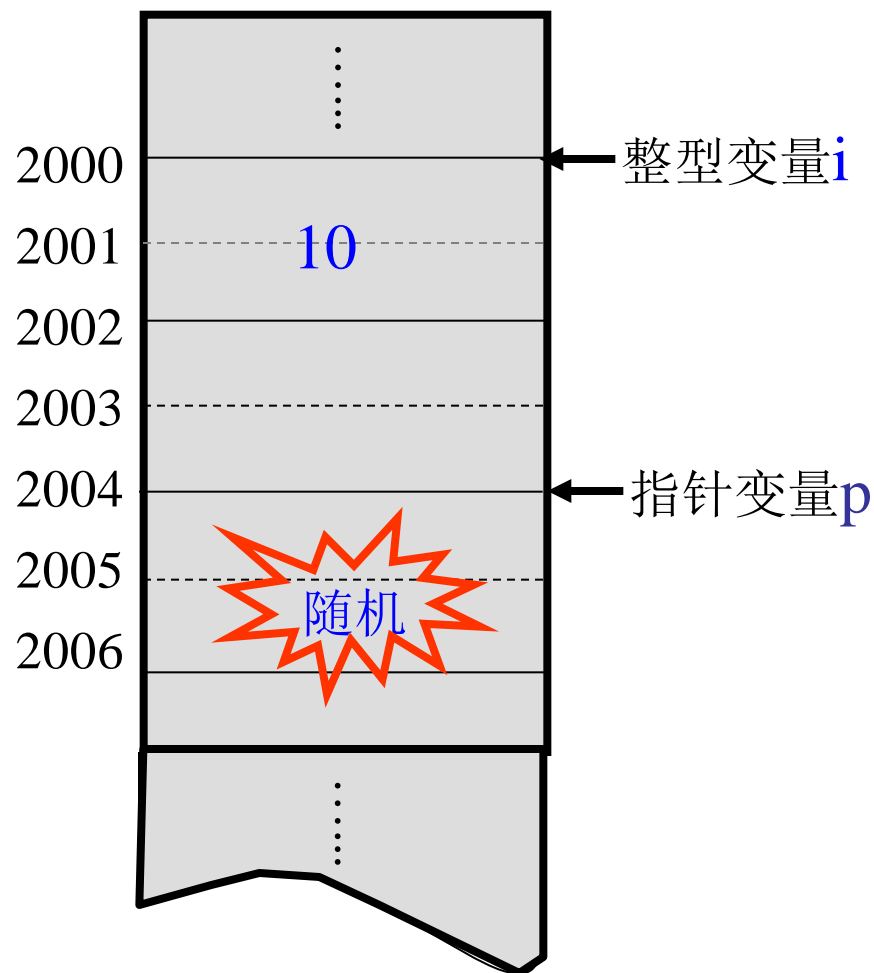
第8章 指针

```
例 void main( )  
{   int i=10;  
    int *p;  
    *p=i;  
    printf(“%d”,*p);  
}
```

危险!

```
例 void main( )  
{   int i=10,k;  
    int *p;  
    p=&k;  
    *p=i;  
    printf(“%d”,*p);  
}
```

指针变量必须先赋值, 再使用!



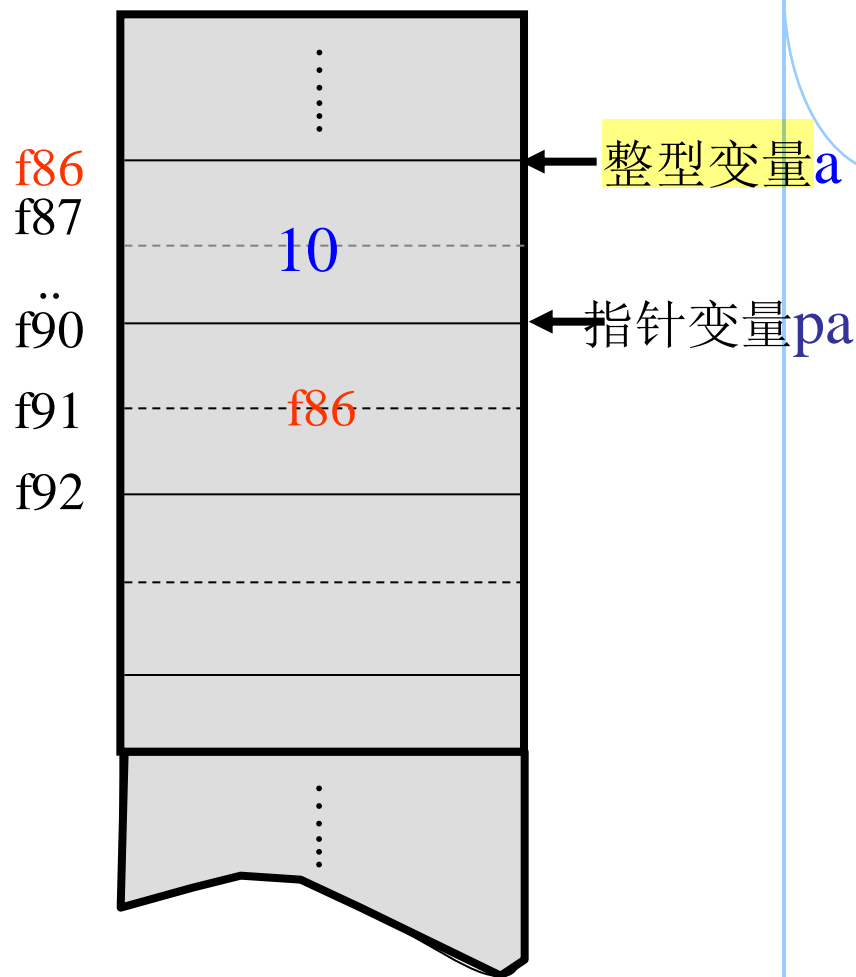
第8章 指针

例 指针的概念举例

```
Void main()
{   int a;
    int *pa=&a;
    a=10;
    printf("a:%d\n",a);
    printf("*pa:%d\n",*pa);
    printf("&a:%x(hex)\n",&a);
    printf("pa:%x(hex)\n",pa);
    printf("&pa:%x(hex)\n",&pa);
}
```

运行结果:

```
a:10
*pa:10
&a:f86(hex)
pa:f86(hex)
&pa:f90(hex)
```



思考题：

- 如果已经执行了 “**pointer_1=&a;**” 的语句，则 **&*pointer_1** 代表什么含义？
- ***&a** 又代表的含义是什么？

8.2 指针变量

4、指针变量作为函数参数

问题引入：请编写一个函数**swap()**，通过函数调用的方式实现交换两个普通变量的值。

第8章 指针

方法一：编写void swap(int x, int y)函数

```
void swap(int x, int y)
```

```
{
```

```
    int t;
```

```
    t = x;  x = y;  y = t;
```

```
}
```

```
int main()
```

```
{
```

```
    int a = 3, b = 5;
```

```
    swap( a, b );
```

```
    printf("%d %d\n",a,b);
```

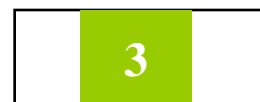
```
    return 0;
```

```
}
```

x

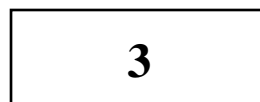


y

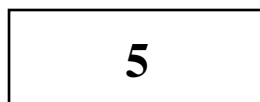


不能完成交换。
形参变量值的改变不会影响实参。

a



b



运行结果：

3 5

第8章 指针

方法二：用指针编写swap函数

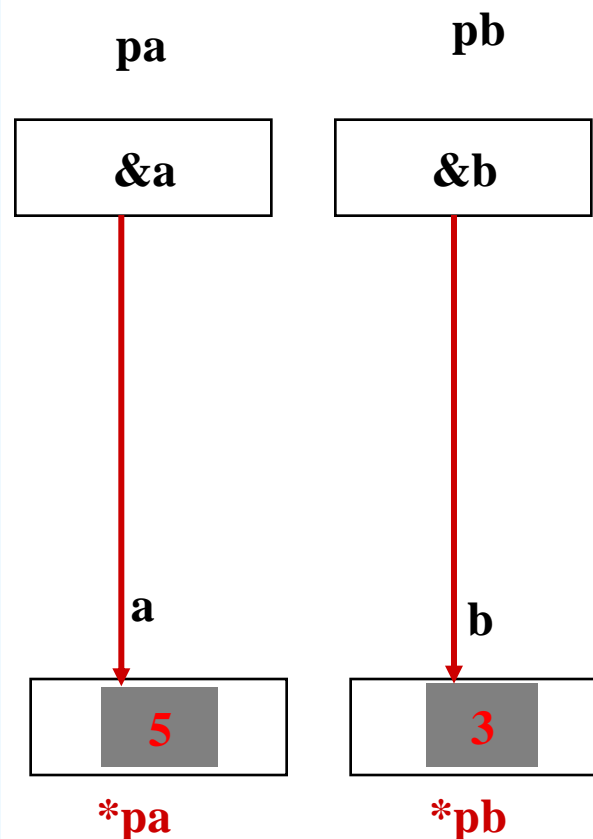
```
void swap( int *pa, int *pb)
{
    int t;
    t = *pa; *pa = *pb; *pb=t;
}

int main()
{
    int a = 3, b = 5;
    swap(&a, &b);
    printf("%d %d\n",a,b);
    return 0;
}
```

方法二成功

*pa是对a的引用

*pb是对b的引用



运行结果：
5 3

第8章 指针

总结: 被调函数如何访问主调函数中的变量

- 要通过函数调用来改变主调函数中某个变量的值:
 - (1) 在主调函数中, 将该变量的地址作为实参;
eg: `swap(&a, &b);`
 - (2) 在被调函数中, 用指针类型的形参来接受该变量的指针;
eg: `void swap(int *pa, int *pb)`
 - (3) 在被调函数中, 用形参指针间接访问主调函数中的变量。

```
void swap( int *pa, int *pb)
{
    int t;
    t = *pa; *pa = *pb; *pb=t;
}
```

8.3 数组与指针

1、指向数组元素的指针

定义一个指向数组元素的指针变量的方法，与以前介绍的指向普通变量的指针变量相同。

例如： `i n t a [1 0] ;`

(定义 a 为包含 1 0 个整型数据的数组)

`i n t * p ;`

(定义 p 为指向整型变量的指针变量)

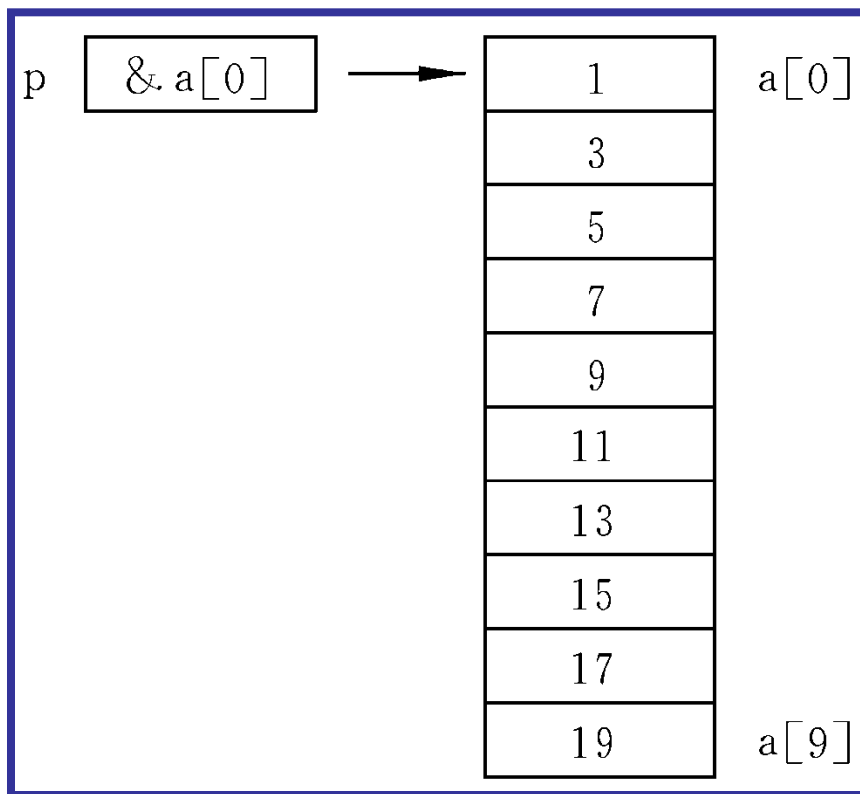
应当注意，如果数组为 `i n t` 型，则指针变量的基类型亦应为 `i n t` 型。

第8章 指针

对该指针变量赋值：

$p = \& a [0] ;$

把 $a [0]$ 元素的地址赋给指针变量 p 。也就是使 p 指向 a 数组的第 0 号元素，如图：



8.3 数组与指针

2、通过指针引用数组元素

- 在使用指针引用数组元素时，允许以下运算：
 - 加一个整数(用+或+=)，如 $p+1$
 - 减一个整数(用-或-=)，如 $p-1$
 - 自加运算，如 $p++$ ， $++p$
 - 自减运算，如 $p--$ ， $--p$
 - 两个指针相减，如 $p1-p2$ (只有 $p1$ 和 $p2$ 都指向同一数组中的元素时才有意义)

8.3 数组与指针

2、通过指针引用数组元素

注意事项:

(1) C规定, 如果指针变量 p 已指向数组中的一个元素, 则 $p+1$ 指向同一数组中的下一个元素, $p-1$ 指向同一数组中的上一个元素。

`float a[10], *p=a;`

假设 $a[0]$ 的地址为2000, 则

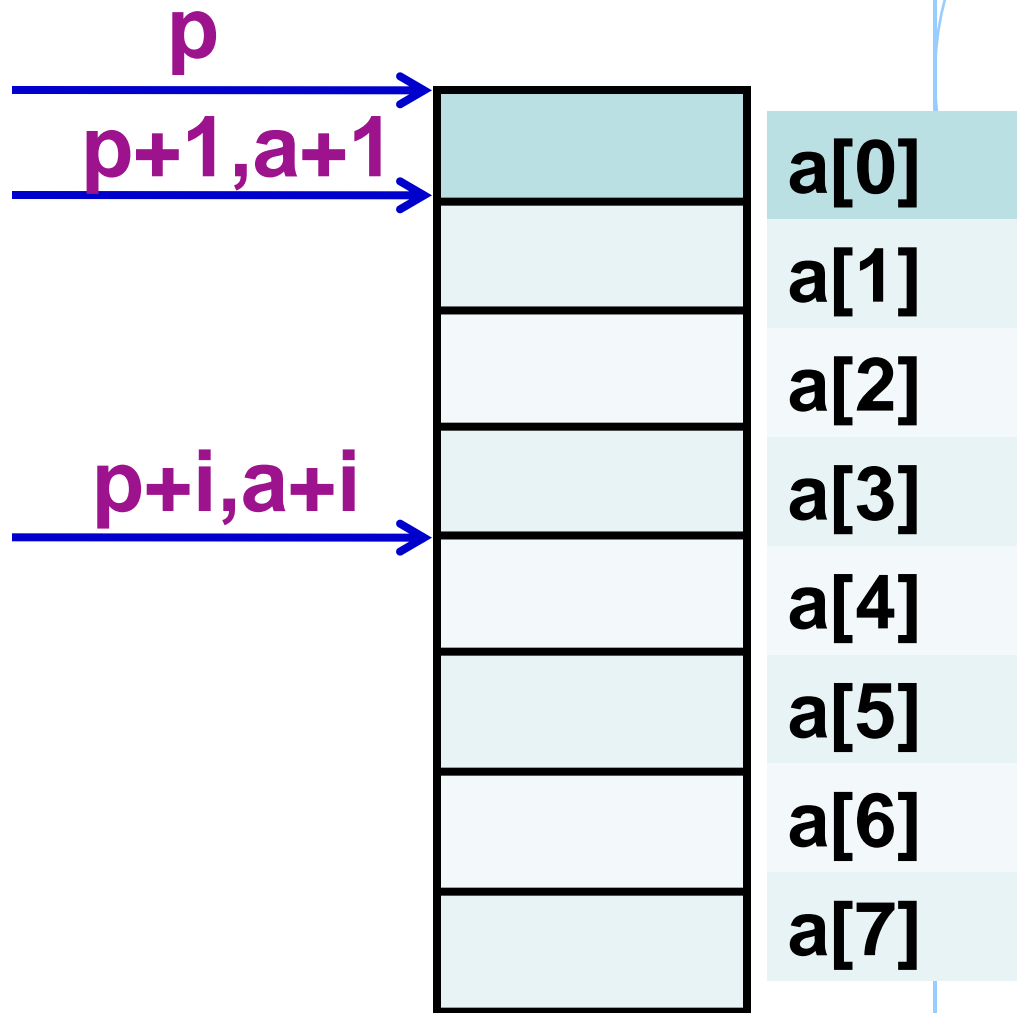
- p 的值为2000
- $p+1$ 的值为2004
- $p-1$ 的值为1996

越界

8.3 数组与指针

2、通过指针引用数组元素

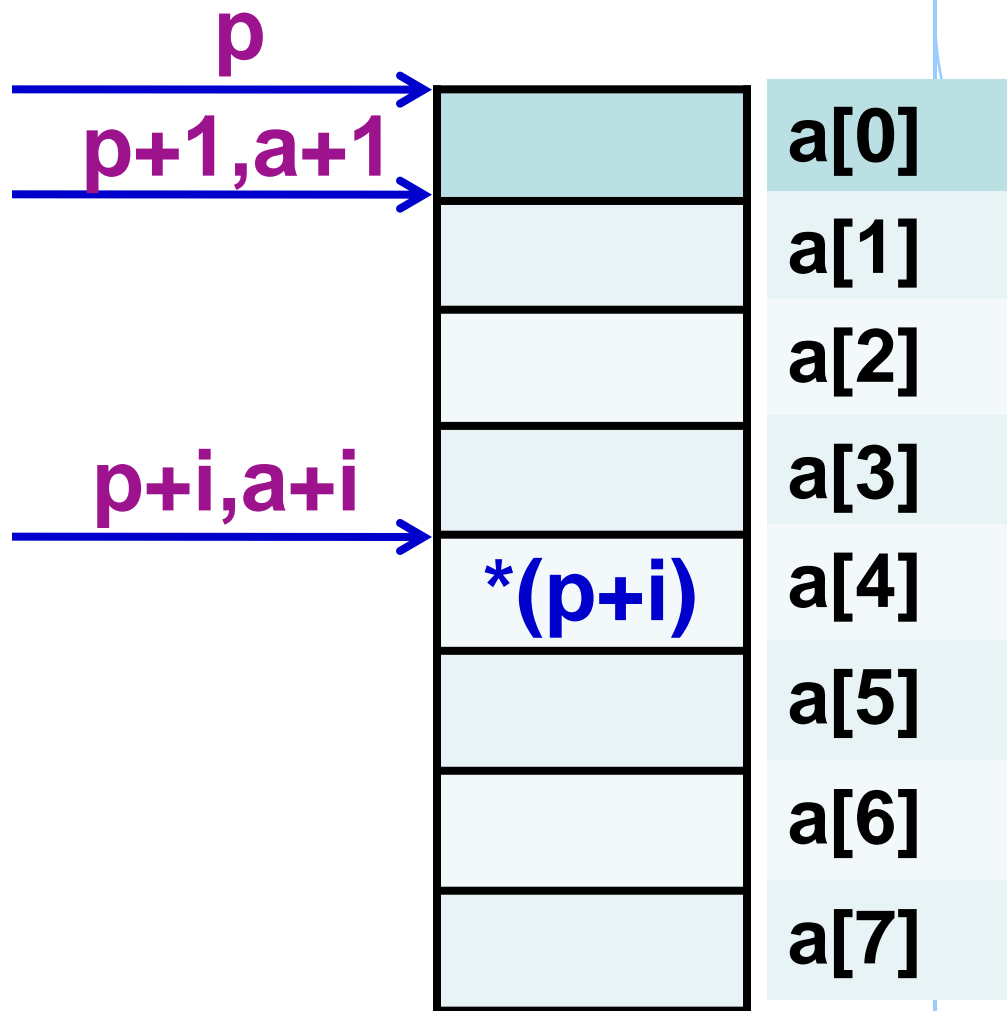
(2) 如果 p 的初值为 $\&a[0]$ ，则 $p+i$ 和 $a+i$ 就是数组元素 $a[i]$ 的地址



8.3 数组与指针

2、通过指针引用数组元素

(3) $*(p+i)$ 或 $*(a+i)$ 是 $p+i$ 或 $a+i$ 所指向的数组元素，即 $a[i]$ 。



8.3 数组与指针

2、通过指针引用数组元素

- 使用指针引用一个数组元素，可用下面两种方法：
 - （1）下标法，如 $\mathbf{a[i]}$ 形式
 - （2）指针法，如 $\mathbf{*(a+i)}$ 或 $\mathbf{*(p+i)}$

其中 \mathbf{a} 是数组名， \mathbf{p} 是指向数组元素的指针变量，其初值
 $\mathbf{p=a}$

例 有一个整型数组 \mathbf{a} ，有10个元素，要求输出数组中的全部元素。

8.3 数组与指针

2、通过指针引用数组元素

(1)下标法

```
#include <stdio.h>
```

```
int main ( )
```

```
{ int a [ 1 0 ] ;
```

```
    int    i ;
```

```
    for ( i = 0 ; i < 1 0 ; i ++ )
```

```
        scanf ( "% d " , & a [ i ] ) ;
```

```
        printf ( " \ n " ) ;
```

```
    for ( i = 0 ; i < 1 0 ; i ++ )
```

```
        printf ( "% d " , a [ i ] ) ;
```

```
}
```

8.3 数组与指针

2、通过指针引用数组元素

(2) 用指针变量指向数组元素

```
#include <stdio.h>
```

```
int main ( )
```

```
{ int a [ 1 0 ] ;
```

```
    int * p , i ;
```

```
    for ( i = 0 ; i < 1 0 ; i ++ )
```

```
        scanf ( "% d " , & a [ i ] ) ;
```

```
    printf ( " \ n " ) ;
```

```
    for ( p = a ; p < ( a + 1 0 ) ; p ++ )
```

```
        printf ( "% d " , * p ) ;
```

```
}
```


8.3 数组与指针

3、数组名作为函数参数

- 用数组名作函数参数时，因为实参数组名代表该数组首元素的地址，形参应是一个指针变量
- **C**编译器都是将形参数组作为指针变量来处理的

8.3 数组与指针

3、数组名作为函数参数

```
int main()
{ void fun(int arr[],int n);
  int array[10];    :
  :
  fun (array,10);
  return 0;
}
void fun(int arr[ ],int n)
{ : }
```

看起来像数组，本质上是
指针

fun(int *arr,int n)

8.3 数组与指针

3、数组名作为函数参数

例 使用函数调用实现对数组元素求和

看起来像数组，本质上是指针

```
int main()
{   int i;
    int b[5] = {1, 4, 5, 7, 9};
    printf("%d\n", sum(b, 5) );
    return 0;
}
```

```
int sum (int *arr , int n)
{
    int i, s = 0;
    for(i=0; i<n; i++)
        s += arr[i];
    return(s);
}
```

8.3 数组与指针

3、数组名作为函数参数

总结：

1. 数组名作函数的实参，实参的值就是**首元素的地址**，调用时传给形参（指针变量），因此，形参指向实参数组的首元素。
2. 如果改变形参所指向单元的值，就是改变实参数组元素的值。

8.3 数组与指针

3、数组名作为函数参数

例 编写函数**reverse(int p[], int n)**实现将数组元素逆序存放。

8.3 数组与指针

3、数组名作为函数参数

```
int main(void)
{
    int i, a[10], n;
    scanf("%d", &n);
    for(i = 0; i < n; i++)
        scanf("%d", &a[i]);
    reverse(a, n);
    for(i = 0; i < n; i++)
        printf("%d\t", a[i]);
    return 0;
}
```

```
void reverse(int p[ ], int n)
{
    int i, j, t;
    for(i=0, j=n-1; i<j; i++, j--)
        {t = p[i] ; p[i] = p[j] ; p[j] =
          t;}
}
```

```
void reverse(int *p, int n)
{
    int *pj, t;
    for(pj = p+n-1; p<pj; p++, pj--)
        {t = *p ; *p = *pj ; *pj = t;}
}
```


8.4 指针与字符串

- 字符串是一种特殊的一维数组，字符串的特殊性在于：**字符串的末尾是结束标志 '\0'**，所以访问字符串时常用结束标志进行判断。
- 1、引用一个字符串，可以使用以下两种方法。
 - (1) 用字符数组存放一个字符串，可以通过数组名和格式声明“%s”输出该字符串，也可以通过数组名和下标引用字符串中的一个字符。
 - (2) 用字符指针变量指向一个字符串常量，通过字符指针变量引用字符串常量。

8.4 指针与字符串

例 定义一个字符数组，在其中存放字符串 “I love China!”，输出该字符串和第8个字符。

```
#include <stdio.h>
int main()
{ char string[]="I love China!";
  printf("%s\n",string);
  printf("%c\n",string[7]);
  return 0;
}
```



```
I love China!
C
```


8.4 指针与字符串

例 定义一个字符指针变量来指向一个字符串，并通过字符指针变量输出该字符串。

```
#include <stdio.h>
int main()
{ char *string="I love China!";
  printf("%s\n", string);
  return 0;
}
```

string



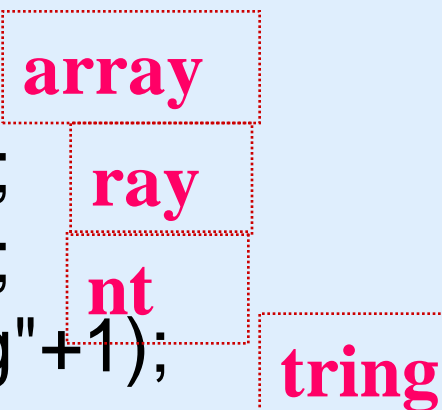
I love China!

**char *string;
string=" I love China!";**

8.4 指针与字符串

- 课堂讨论: 分析程序段输出结果

```
char sa[ ] = "array";  
char *sp = "point";  
printf("%s\n", sa);  
printf("%s\n", sa+2);  
printf("%s\n", sp+3);  
printf("%s\n", "string"+1);
```



The diagram illustrates the output of the printf statements. The first line prints 'array'. The second line prints 'ray' (starting from the third character of 'array'). The third line prints 'nt' (starting from the third character of 'point'). The fourth line prints 'tring' (starting from the second character of 'string').

8.4 指针与字符串

- 2、字符指针作为函数参数
 - 如果想把一个字符串从一个函数“传递”到另一个函数，可以用地址传递的办法，即可以用字符数组作参数，也可以用字符指针变量作参数。
 - 在被调用的函数中可以改变字符串的内容

例 自定义一个函数，用函数调用方法实现字符串的复制

8.4 指针与字符串

(1) 用字符数组作为函数参数

```
#include <stdio.h>
```

```
int main()
```

```
{ void copy_string(char from[],char to[]);
```

```
  char a[]="I am a teacher.";
```

```
  char b[]="you are a student.";
```

```
  printf("a=%s\nb=%s\n",a,b);
```

```
  printf("copy string a to string b:\n");
```

```
  copy_string(a,b);
```

```
  printf("a=%s\nb=%s\n",a,b);
```

```
  return 0;
```

```
}
```

8.4 指针与字符串

```
void copy_string(char from[], char to[])
{ int i=0;
  while(from[i]!='\0')
  {  to[i]=from[i];
    i++;
  }
  to[i]='\0';
}
```

```
a=I am a teacher.
b=You are a student.
copy string a to string b:
a=I am a teacher.
b=I am a teacher.
```

8.4 指针与字符串

(2)用字符型指针变量作参数

```
#include <stdio.h>
```

```
int main()
```

```
{void copy_string(char *from, char *to);
```

```
char *a="I am a teacher.";
```

```
char b[]="You are a student.";
```

```
char *p=b;
```

```
printf("a=%s\nb=%s\n",a,b);
```

```
printf("\ncopy string a to string b:\n");
```

```
copy_string(a,p);
```

```
printf("a=%s\nb=%s\n",a,b);
```

```
return 0;
```

```
}
```

8.4 指针与字符串

(2)用字符型指针变量作参数

```
void copy_string(char *from, char *to)
{ for( ;*from!='\0'; from++,to++)
    { *to=*from; }
  *to='\0';
}
```

8.5 指针与函数

1、什么是函数指针

- 如果在程序中定义了一个函数，在编译时会为函数分配一段存储空间，这段存储空间的起始地址，称为这个函数的指针或地址。**函数名即为起始地址。**
- 我们可以定义一个指向函数的指针变量，用来存放某一个函数的起始地址，这就意味着此指针变量指向了该函数。

8.5 指针与函数

1、什么是函数指针

- 定义指向函数的指针变量的一般形式为：
数据类型 (*指针变量名)(函数参数表列);
如 `int (*p)(int,int);` //定义了一个函数指针
`p=isprime;` 对,将一个函数名赋值给函数指针变量
`p=isprime(b);` 错

例 自定义一个函数，请使用函数指针求整数**a**和**b**中的大者。

8.5 指针与函数

1、什么是函数指针

通过指针变量访问它所指向的函数

```
#include <stdio.h>
```

```
int main()
```

```
{ int max(int,int);
```

```
    int (*p)(int,int); int a,b,c;
```

```
    p=max;
```

```
    printf("please enter a and b:");
```

```
    scanf("%d,%d",&a,&b);
```

```
    c=(*p)(a,b);
```

```
    printf("%d,%d,max=%d\n",a,b,c);
```

```
    return 0;
```

```
}
```

只能指向函数返回值为整型且有两个整型参数的函数

必须先指向，若写成
p=max(a,b); 错

通过函数指针变量来调用指向的函数

8.5 指针与函数

2、什么是指针函数

- 一个函数可以返回一个整型值、字符值、实型值等，也可以返回指针型的数据，即地址。其概念与以前的类似，只是返回值的类型是指针类型而已。其他操作与普通函数完全相同
- 定义返回值是指针的函数的一般形式为
- 类型名 *函数名(参数表列);

例 定义一个函数，求出两个字符串中较长串。

8.5 指针与函数

2、什么是指针函数

```
#include <stdio.h>
#include <string.h>
char *largestring(char *str1, char *str2)
{
    if(strlen(str1) >= strlen(str2)){
        return str1;
    }else{
        return str2;
    }
}
int main()
{
    char str1[30], str2[30], *str;
    gets(str1);
    gets(str2);
    str = largestring(str1, str2);
    printf("Longer string: %s\n", str);
}
```