第8章 指针（1）

S.IST@XMU

第五（三）次上机实验小结

	我班（名单上所有同学）	全年级	所占比例
完成6题	12	29	41.4%
完成5题	5	25	20%
完成4题	1	23	4.3%
完成3题	9	37	24.3%
完成2题	7	66	10.6%
完成1题	6	43	14.0%
完成0题	0	29	0%
未统计	2	?	N/A
总　　计	42	258+	16.3%-

- 主要问题（此外还有很多其他问题……）
  - 数组边界的判断
  - 代码风格混乱
  - 定义静态数组大小时未使用常数
  - 超出时限的问题

复习回顾

- 上次课的内容：
  - 变量的作用域
  - 变量的生存期
  - 声明与定义
  - 内部函数
  - 外部函数
  - 你开始习惯写函数了吗？

2012是如何实现的？

假定造成世界末日的上帝是一个程序员，作为一名合格的程序员，他绝不应该写出类似于“摧毁地球”这样的程序，而应该写一个“摧毁(行星)”的函数，然后把地球作为参数传进去!

传说中的指针

“指针是C语言的重点”

“指针是C语言的灵魂”

“指针是C语言的难点”

“指针是C语言的精髓”

“指针是区分C语言新手和老手的重要标志”

“指针是一扇窗，推开它，
你将看到别样的风景；
指针是一扇门，打开它，
你才能进入遍布荆棘和
无限可能的编程领域！”

欢迎来到内存管理的雷区！

内存的概念

- 从物理上讲内存就是一块半导体存储材料
- 任何需要被计算机执行的程序，包括**指令和数据**都要先从外部存储器（包括硬盘光盘等）调入内存后才能被执行
- CPU需要精准地从内存中找到需要的数据参与运算

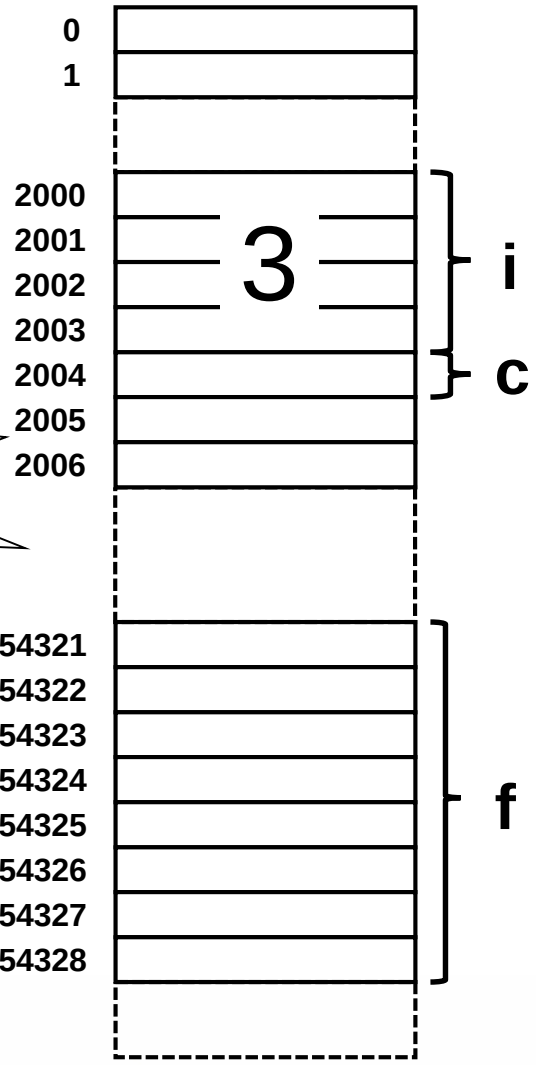
◆CPU如何找到想要的数据？

C语言编译器眼里的内存

- 随着以下C语句的编译运行，编译器做了哪些事情，内存会发生什么变化？

```
int i;
char c;
double f;
i = 3;
```

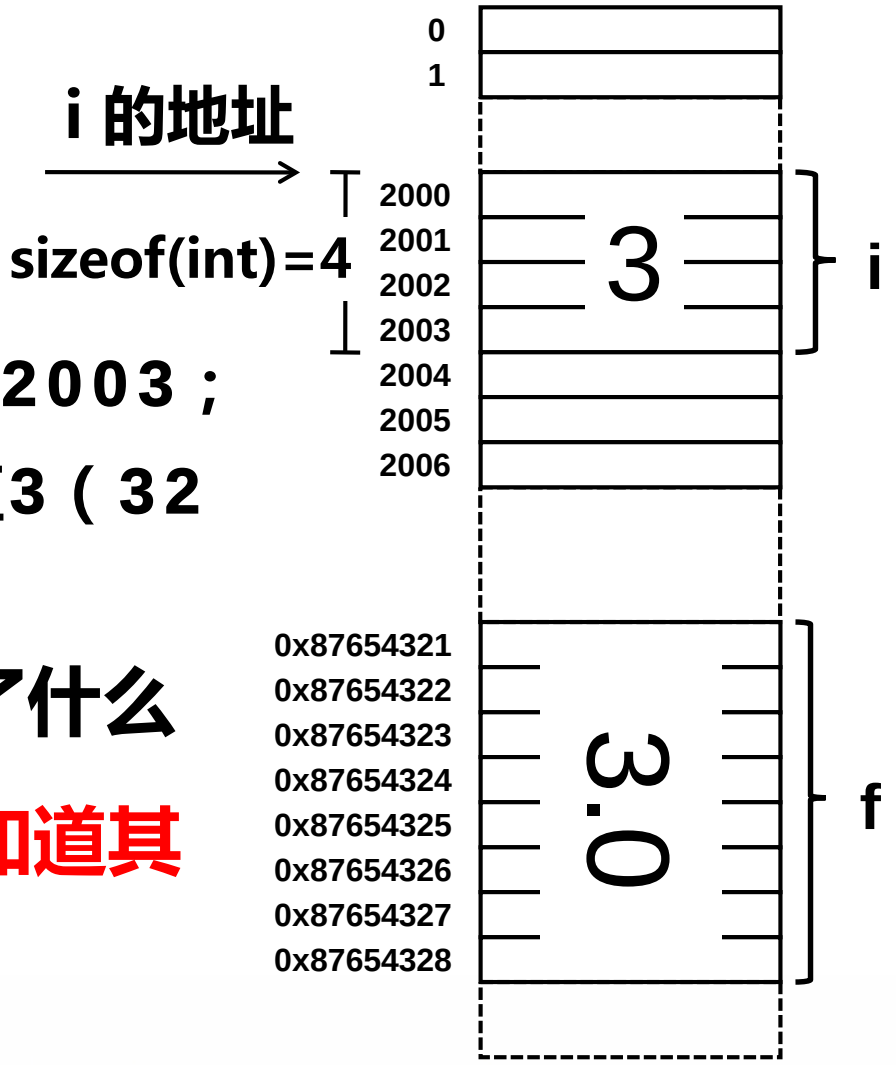
变量名和地址
建立了对应关系



符号表			
变量名	i	c	f
地址	2000	2004	0x87654321
数据类型	int	char	double

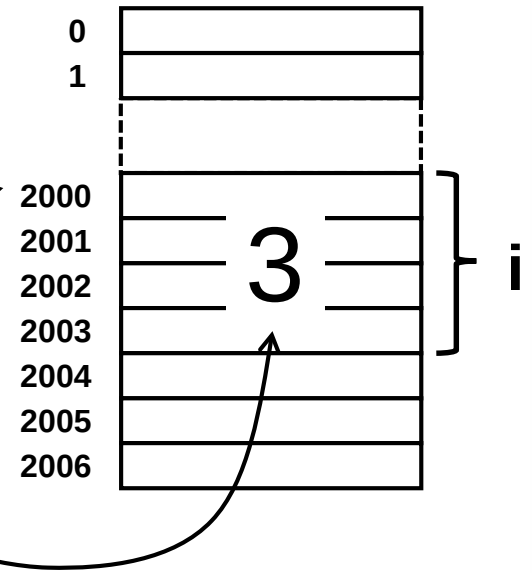
内存的管理方法与地址

- 执行"i=3;"会发生什么？
  1. 找到 i 的地址2000；
  2. 确定 i 所占的字节2000～2003；
  3. 往这四个字节写入整型数值3（32位）。
- 接下来，分析"f=i;"发生了什么
- 由此可见，访问变量必须知道其地址 和 类型！



存储单元内容和地址的区别

- 务必弄清楚存储单元的**地址**和存储单元的**内容**这两个概念的区别
  - **地址**：存储单元的编号
  - **内容**：存储单元里存放的值



指针是什么

“指针就是地址！”

- 内存区的每一个字节有一个 “地址” 。
- 数据存放在地址所标识的内存单元中
- 由于通过地址能找到所需的内存单元，我们可以说，地址指向该内存单元。
- 所以我们将地址形象化地称为 “指针”

怎样定义指针变量

- 定义指针变量的一般形式为：

  类型 * 指针变量名;

  如：int *p;

  - ◆int是为指针变量指定的“基类型”
  - ◆基类型指定指针变量可以指向的变量类型
    - ●如 p 可以指向整型变量，但不能指向浮点型变量

取地址操作符&

- 声明一个变量时，**编译器自动为变量开辟内存空间**，我们无需关心变量在内存中的位置。
- 当程序需要用到某个变量的地址信息，就需要取地址符号&。
  - 例如，int num=0; &num就是num变量的地址

指针变量定义举例

- 我们可以将变量地址赋值给同类型的指针

```
int  a, b;
int  *ip=&a, *iq=&b;
float  *fp;
char  *cp;
```

ip = &b; 正确

*ip = &b; 错误，注意此处的*前面没有数据类型!

指针变量定义注意事项

- 下列定义实现了怎样的功能？

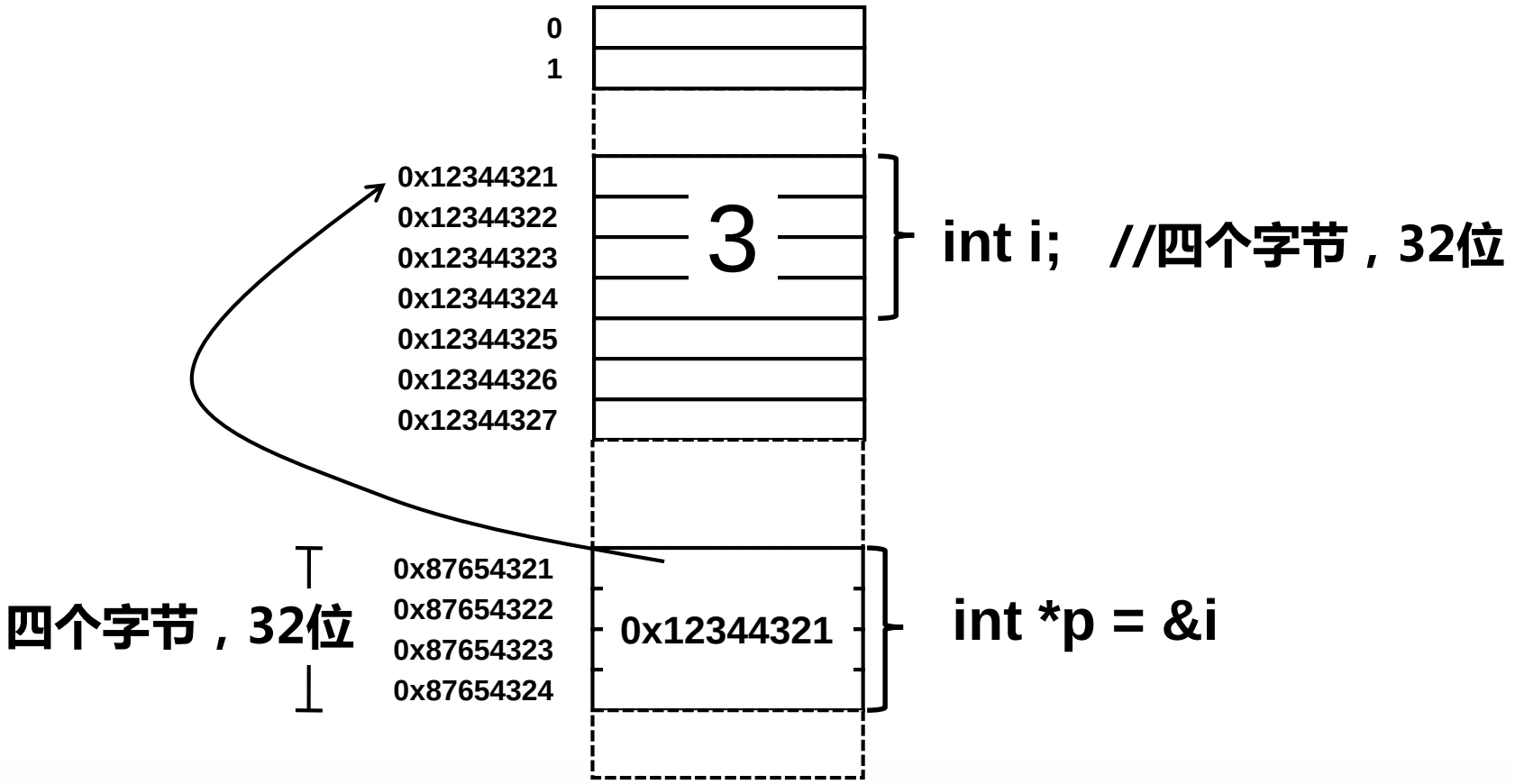
```
int * a, b, c;
```

- 结果是：定义了一个指针变量a，两个整型变量b, c；
- 如果想将a,b,c都定义成指针变量，需要：

```
int *a,*b,*c;
```

或 int *a; int *b; int *c;

图解指针变量


0
1
0x12344321
0x12344322
0x12344323
0x12344324
0x12344325
0x12344326
0x12344327
3
int i; //四个字节，32位
四个字节，32位
0x87654321
0x87654322
0x87654323
0x87654324
0x12344321
int *p = &i

定义指针变量必须指定其类型

- 某类型指针变量只能存放该类型变量的地址

```
int  a, b;
int  *p1 = &a, *p2 = &b;
float  *p3;
p3 = &a;
```

错误，float类型指针不能指向整型变量

指针变量仅能存放变量的地址

- 将一个整数复制给指针变量是错误的
- 变量的地址必须通过一个已经被分配了存储空间的变量加上取地址符“&”来获得，而不能直接指定一个具体的地址
  - 如 int *a; a=0x5050; 错误
  - 但是可以用常量0给指针赋值，表示不指向任何变量的空指针变量，如a=0; 正确

指针变量的初始化

- 在声明一个指针变量后，若没有经过初始化，指针变量的值是不确定的，直接使用未加初始化的指针变量可能会给程序带来各种内存错误。因此**指针变量的初始化很重要**。
- 如果一开始不知该将此指针指向何处，**最简单的方式是利用NULL将指针置0**
  - ◆如：`int *pInt = NULL;`

指针变量值的含义

- 在32位计算机系统中，所有类型的指针的值都是32位的整数。
- “一个指针的值是A”
  - 即“该指针指向了以A为首地址的一片内存区域”
- “一个指针指向了某内存区域”
  - 即“该指针的值是这块内存区域的首地址”

显示指针变量所占空间的实例

```
pChar size=4
pInt size=4
pDouble size=4
```

```
1. #include <stdio.h>
2. int main()
3. {
4.     char * pChar;
5.     int * pInt;
6.     double * pDouble;
7.     //以下sizeof关键字可以用来计算括号内的变量占的字节数
8.     printf("pChar size=%d\n",sizeof(pChar));
9.     printf("pInt size=%d\n",sizeof(pInt));
10.    printf("pDouble size=%d\n", sizeof(pDouble));
11.    return 0;
12.}
```

和指针有关的运算符

- 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）

若有p=&a，则*p就等价于a

k=*p; (表示把a的值赋给k)

*p=1; (表示把1赋给a)

指针运算符代码实例

```
double num = 3;
double * pNum;
pNum = &num;
```

- pNum：指向double类型的指针变量，其值是num的地址
- &num：返回变量num的地址，与pNum等价
- *pNum：返回pNum所指的变量，与num等价
- &(*pNum)：与&num（即pNum)等价
- *(&num)：与*pNum（即num）等价

(&*p)和(*&p)的区别

- “*”和“&”具有相同的优先级，按照从右向左的方向结合
  - ◆对于&*p而言：首先进行一次*p运算，再取地址，相当于&(*p)，此情况p必须是一个指针变量！
  - ◆对于*&p：相当于*(&p)，就是先取地址再做”*”运算，结果等价于p，此情况p可以是任何类型变量。
- 可见，* 与 & 互为逆运算！

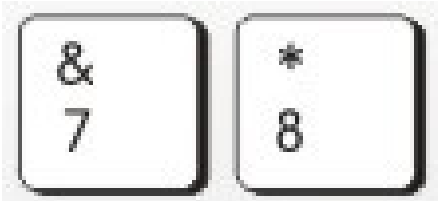
指针运算符与牛肉干

C语言上机课，某女同学边写代码边偷偷吃起牛肉干

有一粒牛肉干掉到了键盘上，卡在7和8键之间……

女同学左手搭在shift键上，右手就在键盘上抠啊、抠啊……

于是程序里某一行代码变成这个样子：




```
int *pa=&*&*&*&*&*&*&*&*&a;
```

女同学没有注意，继续完成程序，结果呢……

——程序顺利通过编译，运行结果完全正确！

指针变量综合举例

```
1. #include <stdio.h>
2. int main()
3. {
4.     unsigned int a = 5;
5.     unsigned int * pint = NULL;
6.     printf("&a = %p\n a = %d\n", &a, a);
7.     printf(" &pint=%p\n pint=%p\n &(*pint)=%p\n",
8.             &pint,     pint,     &(*pint));

9.     pint = &a;
10.    printf("\n&a=%p\n a=%d\n", &a, a);
11.    printf(" &pint=%p\n pint=%p\n &(*pint)=%p\n",
12.            &pint,     pint,     &(*pint));
13.    printf(" *pint=%d\n", *pint);

14.    *pint = 10;
15.    printf("&a=%p\n a=%d\n", &a, a);
16.    printf(" &pint=%p\n pint=%p\n &(*pint)=%p\n",
17.            &pint,     pint,     &(*pint));
18.    printf(" *pint=%d\n", *pint);
19.    return 0;
20.}
```

```
&a = 0022FF4C
 a = 5
 &pint=0022FF48
 pint=00000000
 &(*pint)=00000000
```

```
&a=0022FF4C
 a=5
 &pint=0022FF48
 pint=0022FF4C
 &(*pint)=0022FF4C
 *pint=5
```

```
&a=0022FF4C
 a=10
 &pint=0022FF48
 pint=0022FF4C
 &(*pint)=0022FF4C
 *pint=10
```

不考虑指针，数据只能这样访问

➢ 直接访问方式

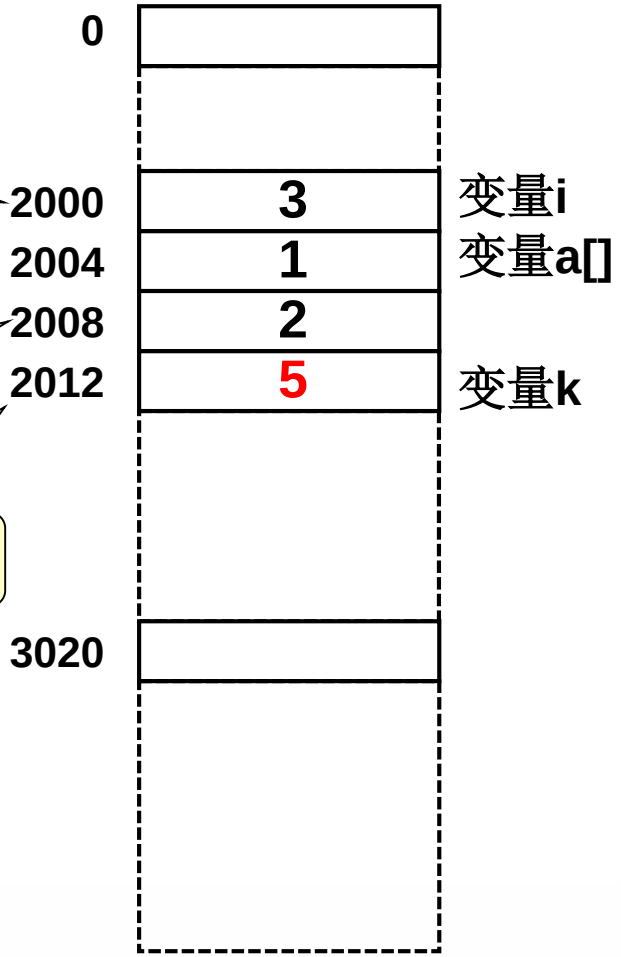
◆通过符号表获得变量名i对应的地址 直接进行访问

```
int i=3,k;
int a[2]={1,2};
k = i + a[1];
```

从这里取3

从这里取2

将5送到这里



符号表			
变量名	i	a[]	k
地址	2000	2004	2008
数据类型	int	int[2]	int

指针带来了新的数据访问方式

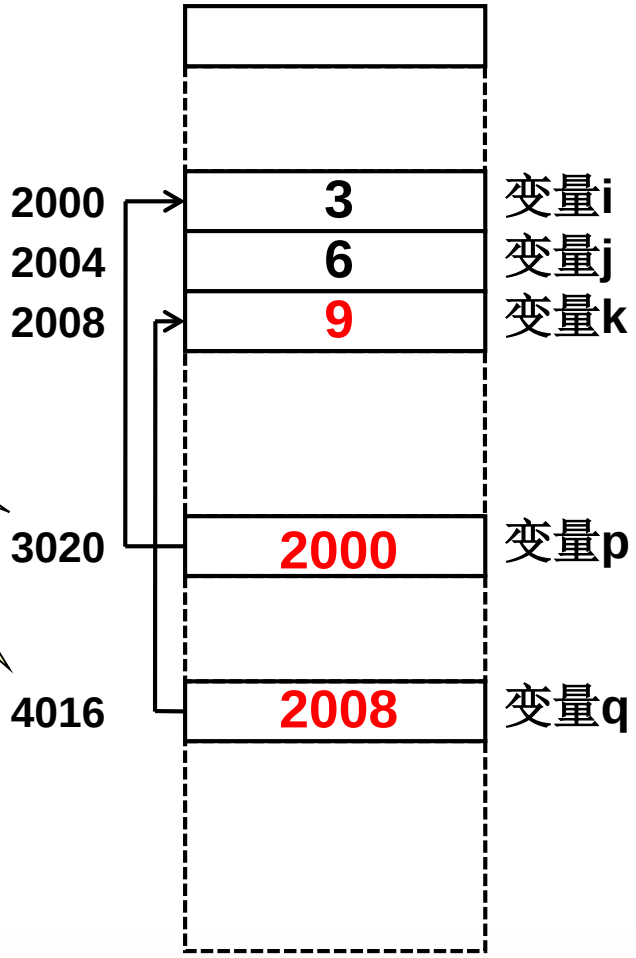
➢ 间接访问方式

◆通过变量p的值获得其所指向的变量的地址，再通过这个地址进行访问

```
int i=3,j=6,k,*p,*q;
p = &i;
q = &k;
*q = (*p)+j;
```

将i的地址存到这里

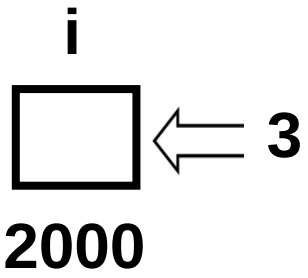
将k的地址存到这里



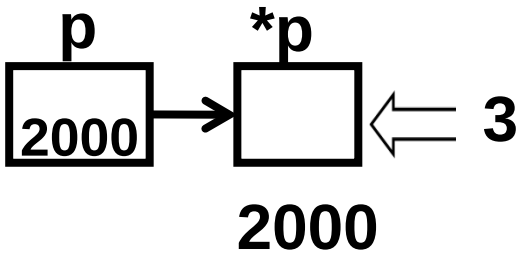
符号表					
变量名	i	j	k	p	q
地址	2000	2004	2008	3020	4016
类型	int	int	int	int *	int *

直接/间接访问类比：酒店找人

- **直接访问**：例如“i=3;”
  - 跟i很熟，知道它住在哪个房间，所以可以直接去访问它



- **间接访问**：例如已知p=&i; 执行“*p=3;”
  - 警察秘密调查i，只知道i住在酒店里，i的房间号放在前台（即p）那里，所以先询问前台（通过p获得i的地址），然后去访问它



指针的优点

- 能完成一些直接访问无法实现的功能
- 突破代码屏障去操作计算机内存，提供更大的编程灵活度
- 类比一下，在绕过某人（在某人不知情）的情况下获得某人的地址并进行操作
  - 权力机关才允许做的事情
  - 坏人容易利用来做些不可告人的勾当……

使用指针变量的例子

- 通过指针变量访问整型变量。
- **解题思路：**先定义2个整型变量，再定义2个指针变量，分别指向这两个整型变量，通过访问指针变量，可以找到它们所指向的变量，从而得到这些变量的值。

使用指针变量的例子

```
1. #include <stdio.h>
2. int main()
3. {
4.     int a=100,b=10;
5.     int *pointer_1, *pointer_2;
6.     pointer_1 = &a;
7.     pointer_2 = &b;

8.     printf("a=%d,b=%d\n",a,b);
9.     printf("*pointer_1=%d,*pointer_2=%d\n",
10.                          *pointer_1,*pointer_2);
11.    return 0;
12.}
```

定义两个指针变量

```
a=100,b=10
*pointer_1=100,*pointer_2=10
```

使pointer_1指向a

使pointer_2指向b

直接输出变量a和b的值

间接输出变量a和b的值

使用指针变量的例子

```
1. #include <stdio.h>
2. int main()
3. {
4.     int a=100,b=10;
5.     int *pointer_1, *pointer_2;
6.     pointer_1 = &a;
7.     pointer_2 = &b;

8.     printf("a=%d,b=%d\n",a,b);
9.     printf("*pointer_1=%d,*pointer_2=%d\n",
10.                          *pointer_1, *pointer_2);
11.    return 0;
12.}
```

此处*与类型名在一起。此时共同定义指针变量

此处*与指针变量一起使用。此时代表指针变量所指向的变量

怎样引用指针变量

- 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("%o",p);

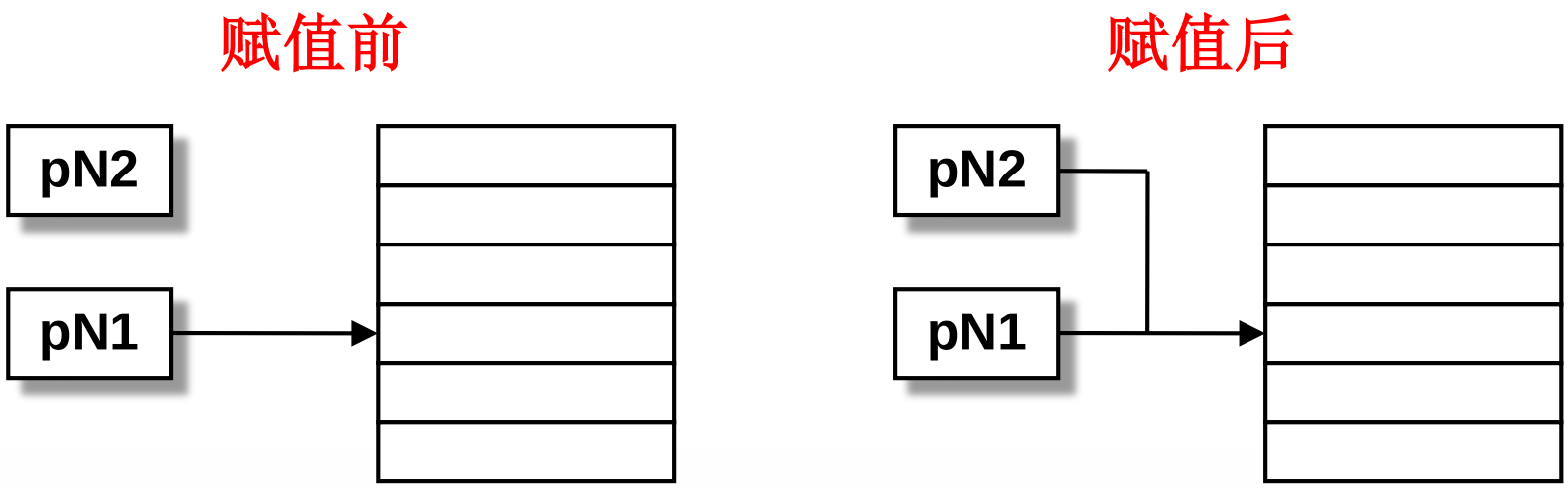
使p指向a

*p相当于a

以八进制输出a的地址

同类型指针的赋值

- 设pN1和pN2是两个相同类型的指针，
  - 执行“pN2=pN1;”这样一个操作后，pN1和pN2将指向同样的地址，即两个指针指向同一个内存单元
  - 对*pN1的任何改动都会影响*pN2的值，反之亦然



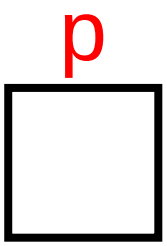
指针应用举例

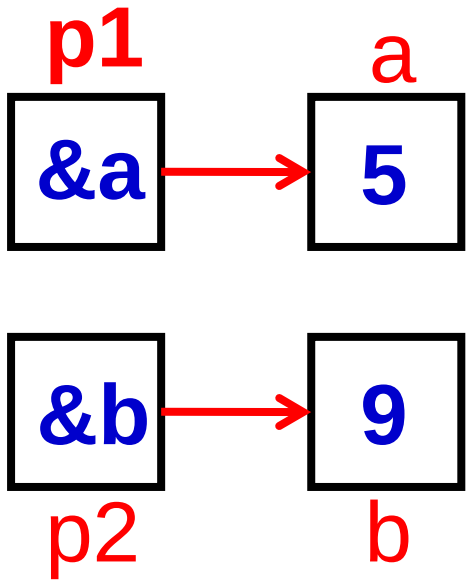
- 输入a和b两个整数，按先大后小的顺序输出a和b，要求使用指针。
- **解题思路：**用指针方法来处理这个问题。不交换整型变量的值，而是交换两个指针变量的值。

指针应用举例

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int *p1, *p2, *p, a, b;
5.      printf("integer numbers:");
6.      scanf("%d,%d", &a, &b);
7.      p1 = &a;
8.      p2 = &b;
9.      if (a<b)
10.     {
11.         p = p1;
12.         p1 = p2;
13.         p2 = p;
14.     }
15.     printf("a=%d,b=%d\n", a, b);
16.     printf("%d,%d\n", *p1, *p2);
17.     return 0;
18. }
```

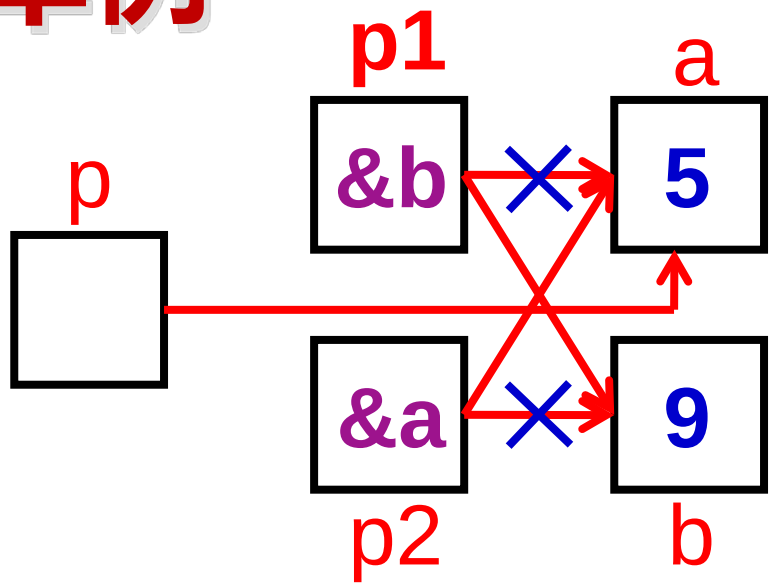
5,9





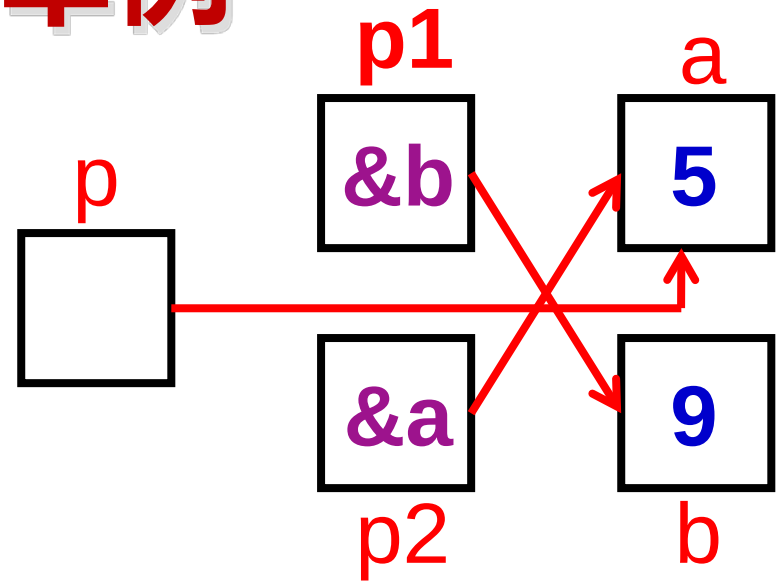
指针应用举例

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int *p1, *p2, *p, a, b;
5.      printf("integer numbers:");
6.      scanf("%d,%d", &a, &b);
7.      p1 = &a;
8.      p2 = &b;
9.      if (a<b)
10.     {
11.         p = p1;
12.         p1 = p2;
13.         p2 = p;
14.     }
15.     printf("a=%d,b=%d\n", a, b);
16.     printf("%d,%d\n", *p1, *p2);
17.     return 0;
18. }
```



指针应用举例

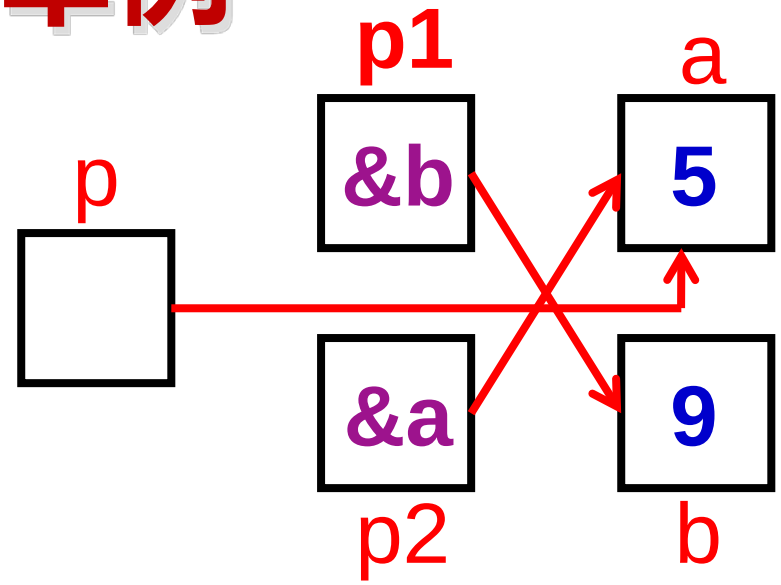
```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int *p1, *p2, *p, a, b;
5.      printf("integer numbers:");
6.      scanf("%d,%d", &a, &b);
7.      p1 = &a;
8.      p2 = &b;
9.      if (a<b)
10.     {
11.         p = p1;
12.         p1 = p2;
13.         p2 = p;
14.     }
15.     printf("a=%d,b=%d\n", a, b);
16.     printf("%d,%d\n", *p1, *p2);
17.     return 0;
18. }
```




```
a=5,b=9
9,5
```

指针应用举例

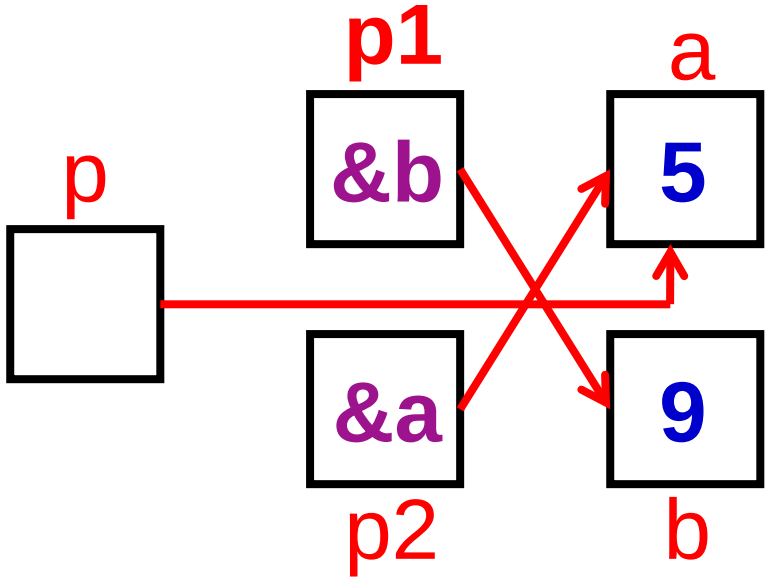
```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int *p1, *p2, *p, a, b;
5.      printf("integer numbers:");
6.      scanf("%d,%d", &a, &b);
7.      p1 = &a;
8.      p2 = &b;
9.      if (a<b)
10.     {
11.         p = p1;
12.         p1 = p2;
13.         p2 = p;
14.     }
15.     printf("a=%d,b=%d\n", a, b);
16.     printf("%d,%d\n", *p1, *p2);
17.     return 0;
18. }
```



可否改为p1=&b; p2=&a;?

指针应用举例

- 注意:
  - a和b的值并未交换，它们仍保持原值
  - 但p1和p2的值改变了。p1的值原为&a，后来变成&b，p2原值为&b，后来变成&a
  - 这样在输出*p1和*p2时，实际上是输出变量b和a的值，所以先输出9，然后输出5



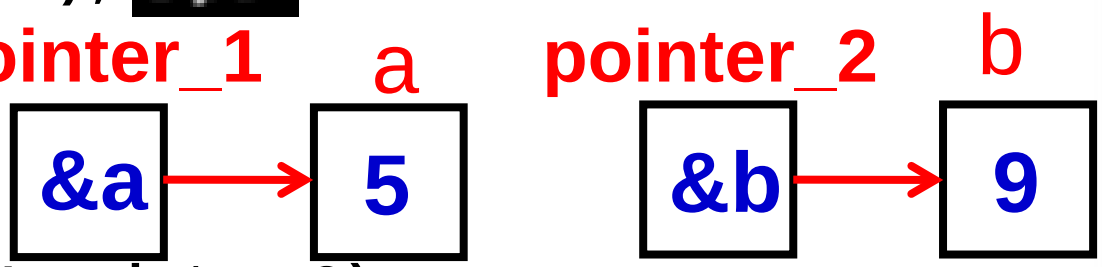
指针变量作函数参数的例子

- 对输入的两个整数按大小顺序输出。现用函数处理，而且用指针类型的数据作函数参数
- **解题思路：**定义一个函数swap，将指向两个整型变量的指针变量作为实参传递给swap函数的形参指针变量，在函数中通过指针实现交换两个变量的值。

指针变量作函数参数的例子

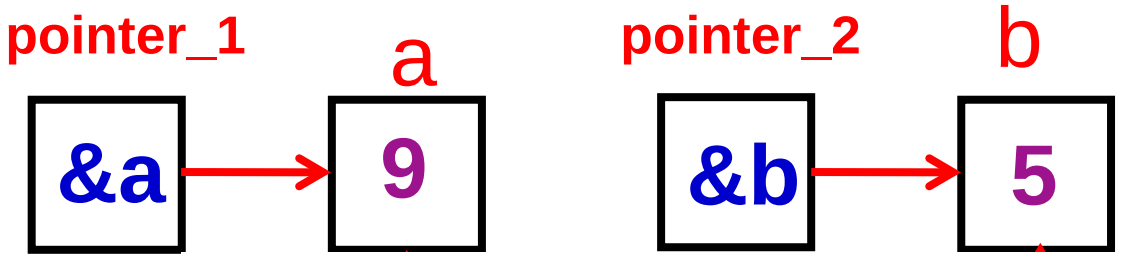
```
1. #include <stdio.h>
2. int main()
3. {
4.     void swap(int *p1,int *p2);
5.     int a,b;  int *pointer_1,*pointer_2;
6.     printf("please enter a and b:");
7.     scanf("%d,%d",&a,&b);
8.     pointer_1 = &a;
9.     pointer_2 = &b;
10.    if (a<b)
11.        swap(pointer_1,pointer_2);
12.    printf("max=%d,min=%d\n",a,b);
13.    return 0;
14.}
```

5,9



指针变量作函数参数的例子

```
void swap(int *p1,int *p2)
{
    int temp;
    temp = *p1;
    *p1 = *p2;
    *p2 = temp;
}
```




```
please enter a and b:5,9
max=9,min=5
```

swap的失败例子之一

```
void swap(int *p1,int *p2)
{
    int temp;
    temp = *p1;
    *p1 = *p2;
    *p2 = temp;
}
```

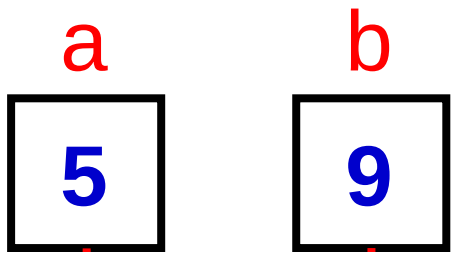
错！！！
无确定的指向

```
void swap(int *p1,int *p2)
{
    int *temp;
    *temp = *p1;
    *p1 = *p2;
    *p2 = *temp;
}
```

swap的失败例子之二

```
#include <stdio.h>
int main()
{   ……
    if (a<b)
        swap(a,b);
    printf("max=%d,min=%d\n",a,b);
    return 0;
}
void swap(int x,int y)
{
    int temp;

    temp = x;
    x = y;
    y = temp;
}
```



错！！！
无法交换a,b

swap的失败例子之三

```
#include <stdio.h>
int main()
{    ……
     scanf("%d,%d",&a,&b);
     pointer_1=&a;  pointer_2=&b;
     if (a<b)
          swap(pointer_1, pointer_2);
     printf("max=%d,min=%d\n",a,b);
     return 0;
}
void swap(int *p1,int *p2)
{
     int *p;
     p = p1;
     p1 = p2;
     p2 = p;
}
```

```
5,9
```

```
max=5,min=9
```

错！！！
只交换形参指向

swap例子启示录

- 函数的调用可以（而且只可以）得到一个返回值（即函数值），而**使用指针变量作参数，可以得到多个变化了的值**。如果不用指针变量是难以做到这一点的。
- 要从函数调用获得多个返回值，可以利用**指针参数传递**法。

指针参数传递法

- 如果想通过函数调用得到 n 个要改变的值：
  - ① 在主调函数中设 n 个变量，用 n 个指针变量指向它们
  - ② 设计一个函数，有n个指针形参。
  - ③ 在主调函数中调用这个函数，在调用时将这n个指针变量作实参，或直接将它们的地址传给该函数的形参
  - ④ 在执行该函数的过程中，通过形参指针变量，改变它们所指向的 n 个变量的值
  - ⑤主调函数中就可以使用这些改变了值的变量

指针参数传递实例

```
1. #include <stdio.h>
2. void fun(int *p, int *q)
3. {
4.     printf("*p=%d, q=%d\n", *p, *q);
5.     (*p)++;
6.     (*q)--;
7.     printf("*p=%d, q=%d\n", *p, *q);
8. }
9. int main()
10.{
11.    int x=0, y=0;
12.    printf("x=%d, y=%d\n", x, y);
13.    fun(&x, &y);
14.    printf("x=%d, y=%d\n", x, y);
15.    return 0;
16.}
```

指针作参数

改变指针指向的变量的值

```
x=0, y=0
*p=0, q=0
*p=1, q=-1
x=1, y=-1
```

作业 2017/11/30

- 按下列要求编写程序，提交手写源代码

1. 编写一个程序，输入一个整数，并设计一个函数pabs利用指针变量返回该数的绝对值，然后输出该返回值，其中pabs函数参数为整数指针，返回绝对值的指针。

2. *****尽快补齐上机课没有做完的上机题！*****