清华大学
Tsinghua University

程序设计基础

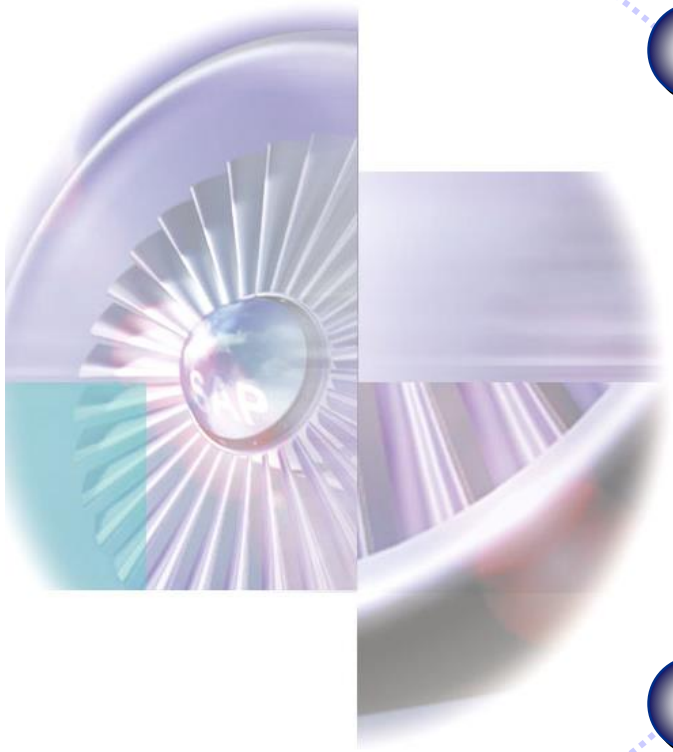
Fundamental of Programming

清华大学软件学院

刘玉身

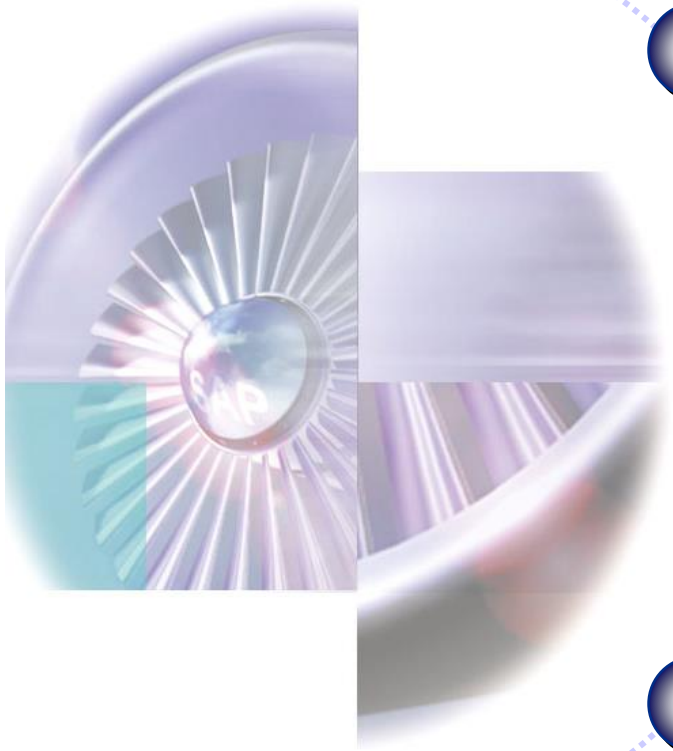
liuyushen@tsinghua.edu.cn

Lecture 7: 指针



1. 基本概念
2. 指针变量
3. 指针与数组
4. 指针与字符串

Lecture 7: 指针



1. 基本概念
2. 指针变量
3. 指针与数组
4. 指针与字符串

基本概念

指针是C语言中最重要的特性之一，要成为一个好的 C/C++ 程序员，就必须理解指针的概念，并善于使用它。

Pointers are often thought to be the most difficult aspect of C !!!

国家宝藏


CONGRESS
A DECLARATION
REPRESENTATIVES
UNITED STATES OF AMERICA
JOHN HANCOCK

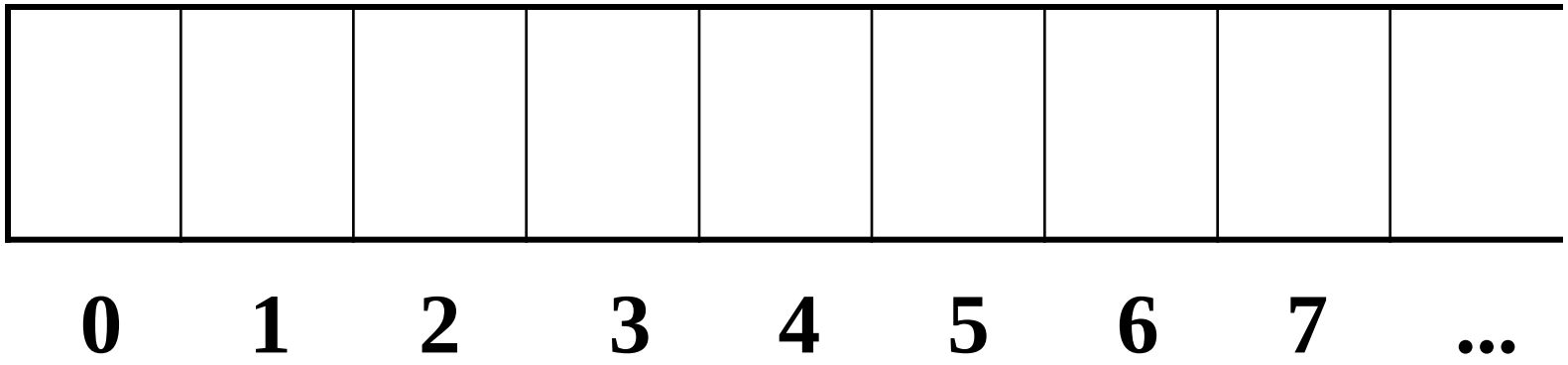

BROADWAY
WALL ST


TRINITY
CHURCH

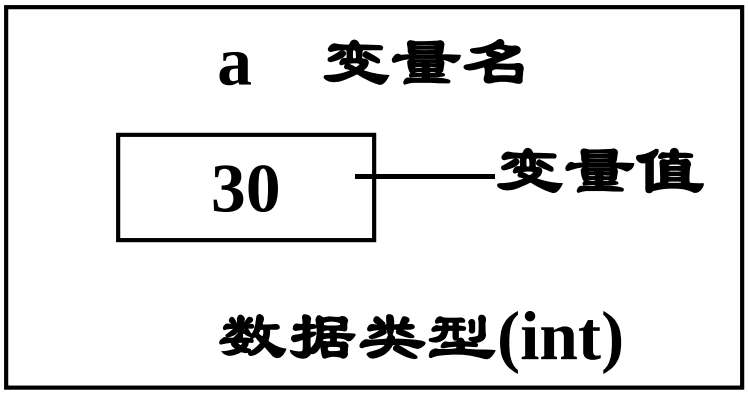

heere at the wall

Why 指针？

思考：如何访问内存中的数据？

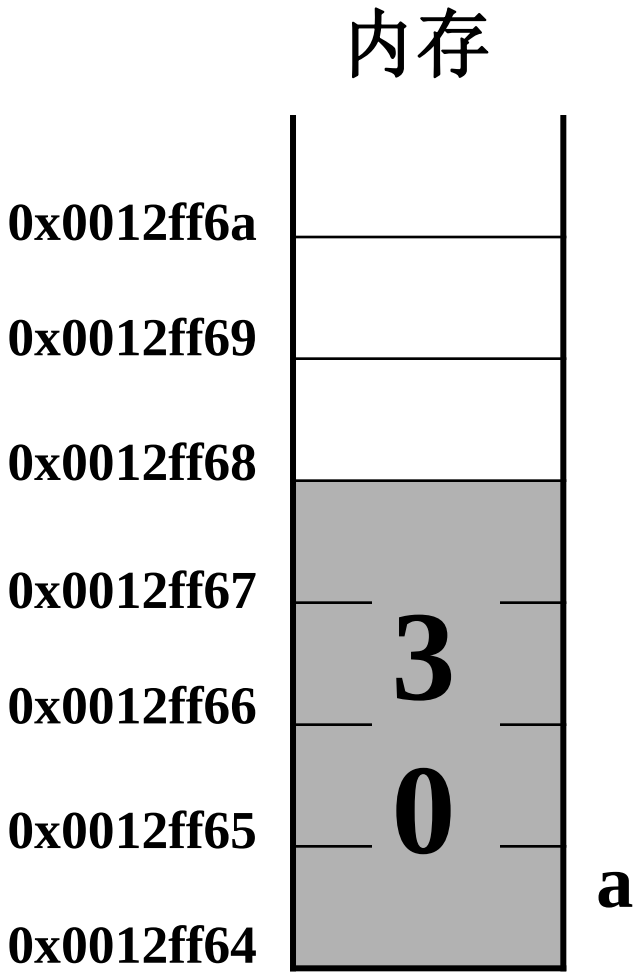


1. 字节为单位
2. 数据长度不同



高手的苦恼：

1. **内存有多大?**
2. **如何指哪打哪?**

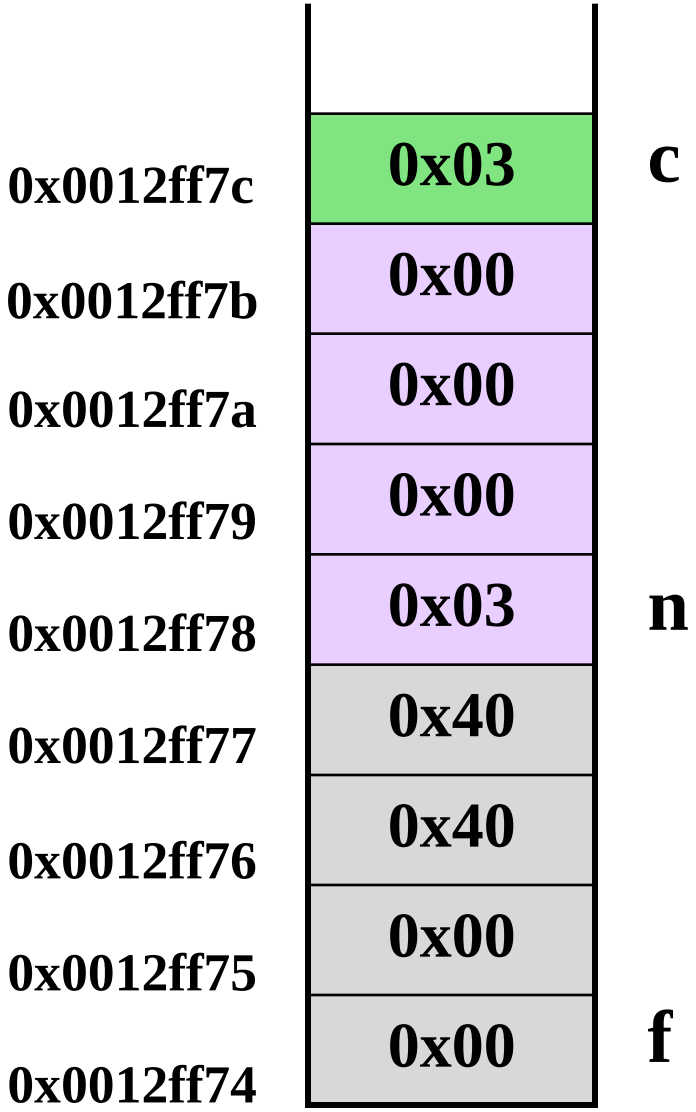


如何访问任意内存单元中的数据？

访问一个数据需要知道：

- 它的起始地址
- 它的数据类型

```
char c;
int n;
float f;

c = 3;      // 整数3
n = 3;      // 整数3
f = 3;      // 实数3
```



起始地址 + 数据类型 = 指哪打哪

什么是指针？

***A pointer* is a *variable* whose *value* is a *memory address*.**

The C Programming Language (K & R):

A pointer is a variable that contains the memory address of another variable.

指针是一种特殊的变量：

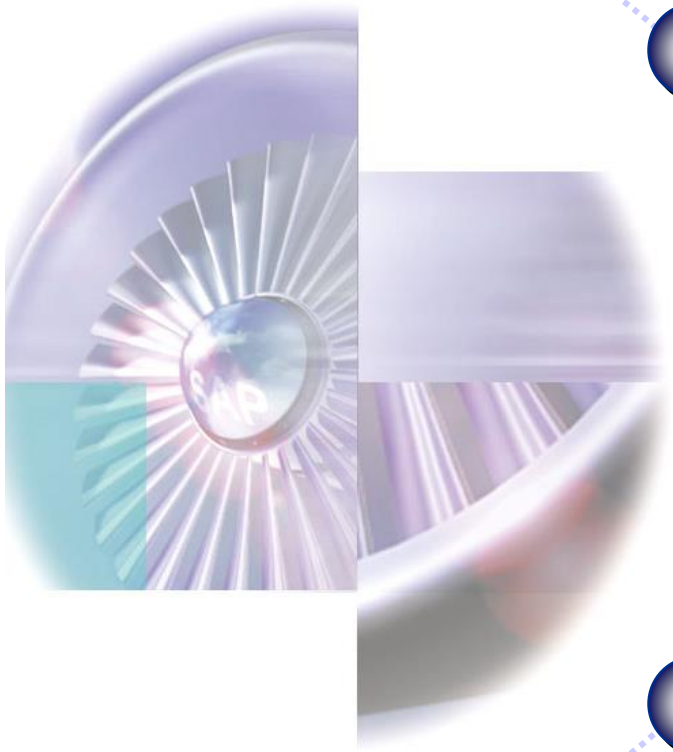
1. 变量名：与一般的变量命名规则相同；
2. 变量的值：是另一个变量的内存地址；
3. 变量的类型：包括两种类型，
   - 指针变量本身的类型，即**指针类型**，它的长度为4个字节；**(sizeof(p) == 4)**
   - 指针指向的变量（数据）的类型。

指针 ＝ 变量 ≠ 地址

买椟还珠. . .

“地址变量”

Lecture 7: 指针



1. 基本概念
2. **指针变量**
3. 指针与数组
4. 指针与字符串

指针的定义

指针定义的一般形式：

[存储类型] 基类型 *指针变量名;

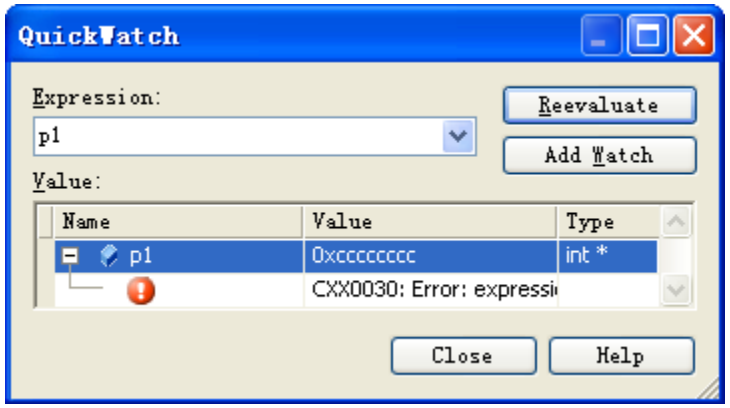
表示定义指针变量
不是‘*’运算符

例如：

```
int   x;       /* 定义了一个整型变量 */
int  *p;       /* 定义了一个指向整型变量的指针变量 */
static  char  *name;
```

注意

```
int   *p1, *p2;
int   *p1,   p2;
double *p3;
```



1. 指针变量名是 p1, p2 , 不是 *p1, *p2
2. 指针变量只能指向定义时所规定类型的变量 （p3 ≠ p1）
3. 指针变量定义后，变量值不确定，应用前必须先赋值

现在我们有了三种新的数据类型：

int * “pointer to int”

double * “pointer to double”

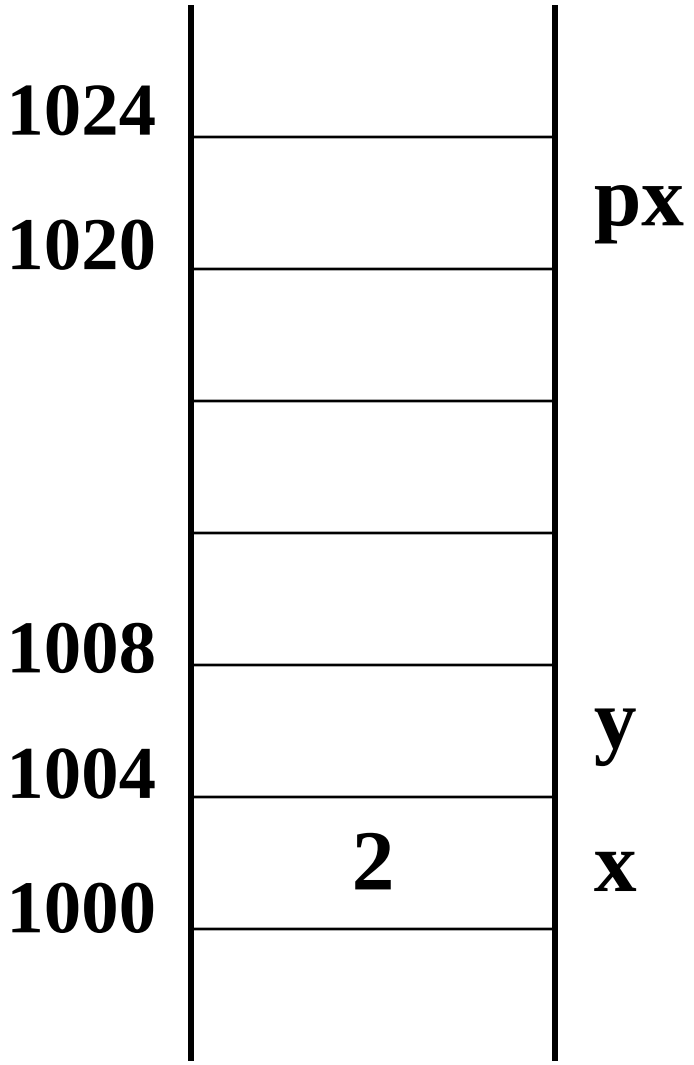
char * “pointer to char”

相同点？不同点？

```
int   x,   y,   *px;

x   =   2;
```

一些问题：

1. 如何把变量 x 的地址存放在指针 px 当中？
2. 如何使用指针 px 来访问或修改变量 x 的值？
3. 为何要如此自找麻烦？



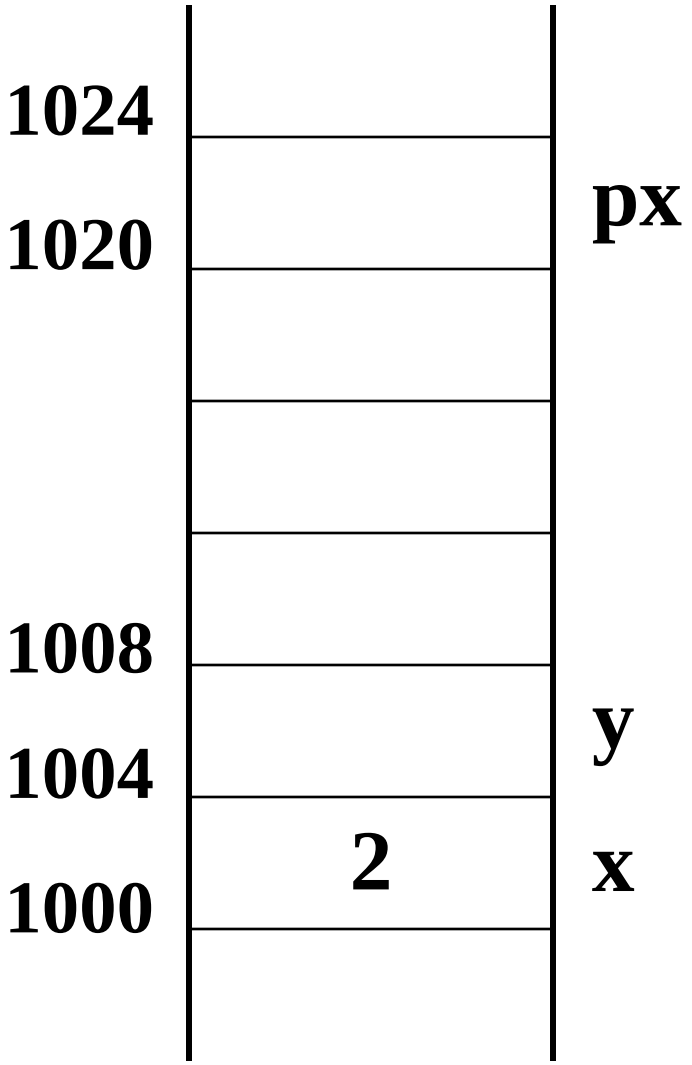
&运算符

```
int   x,   y,   *px;
x   =   2;
```

1. 如何把变量 x 的地址存放在指针 px 当中？

能否：px = 1000?

- 1000从何而来？
- 类型不匹配。



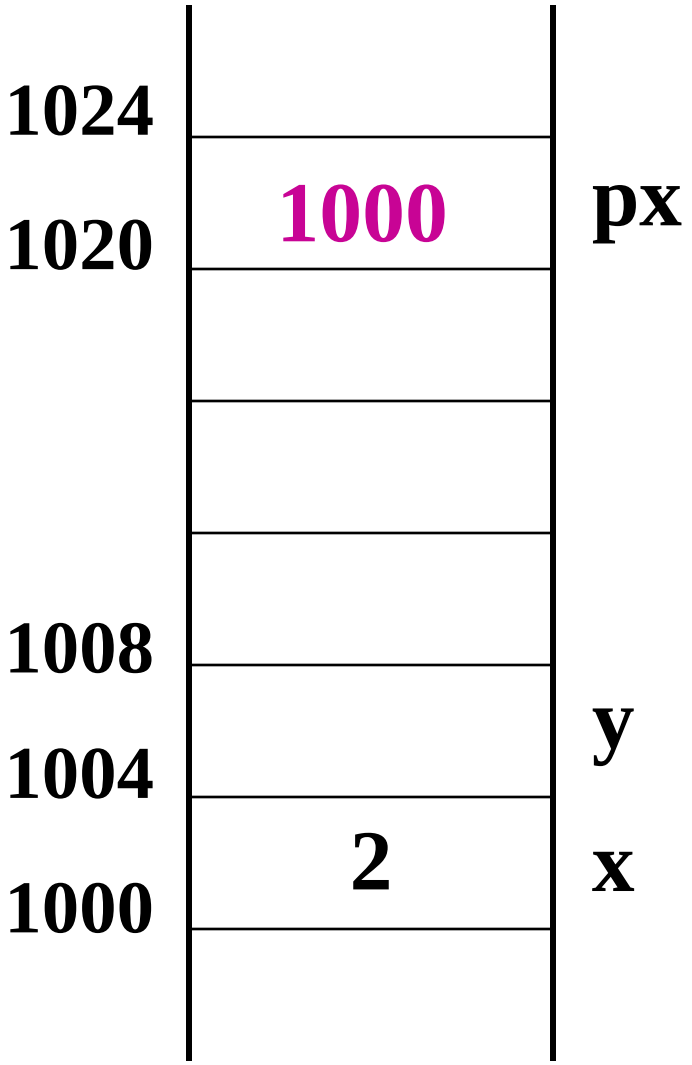
我们想要做的事情：

px = x 的内存地址；

我们写的语句：

px = &x;

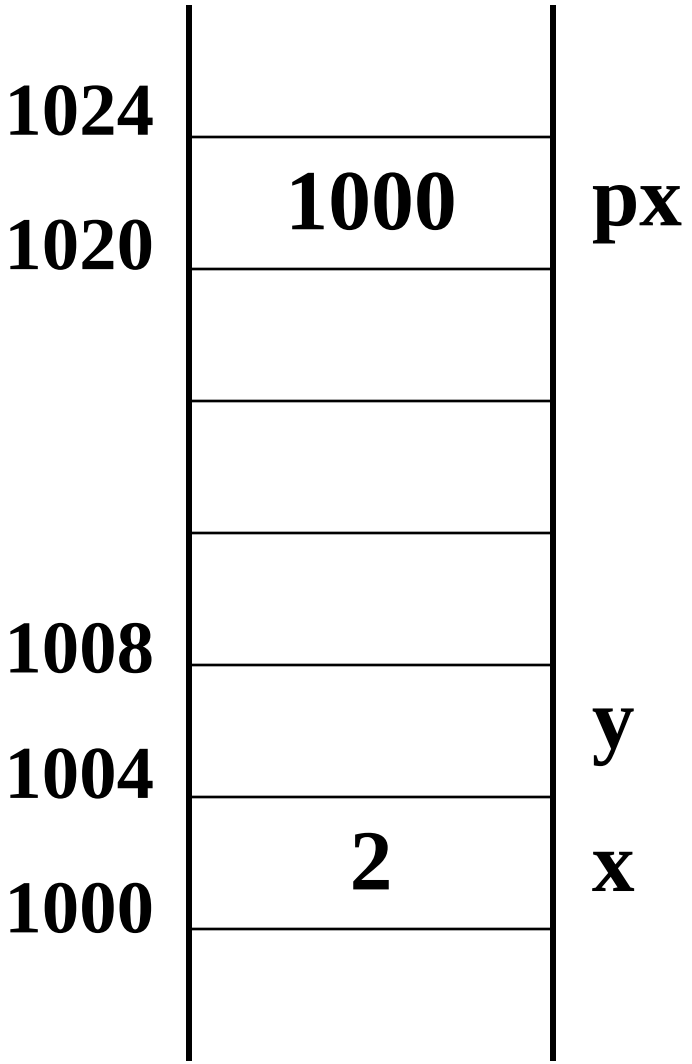
&：取地址运算符



*运算符

```
int   x,   y,   *px;
x   =   2;
px   =   &x;
```

2. 如何通过指针 px 来访问变量 x 的值？



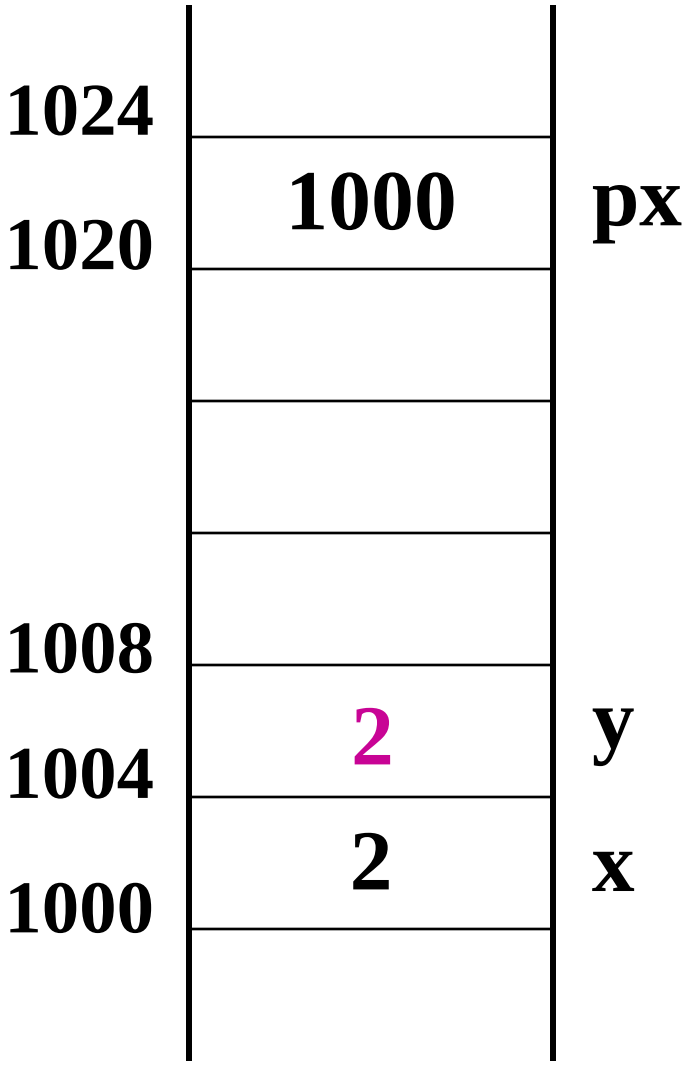
我们想要做的事情：

y = px所指向的变量的值；

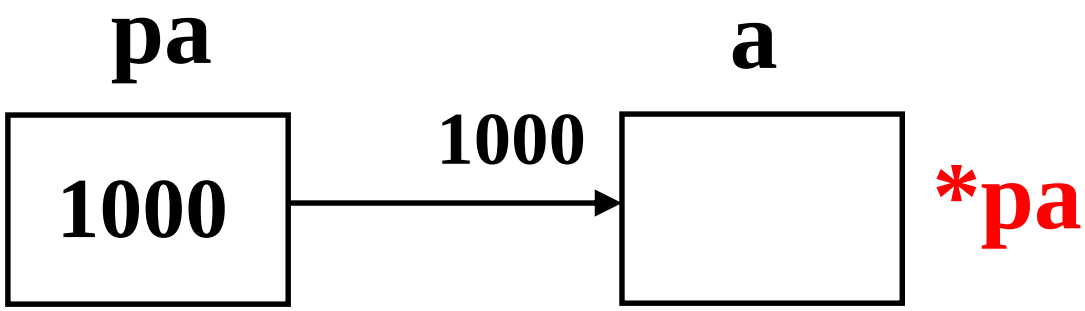
我们写的语句：

y = *px;

*：指针运算符（或称“间接访问”运算符）。实现取指针所代表地址的内容运算符

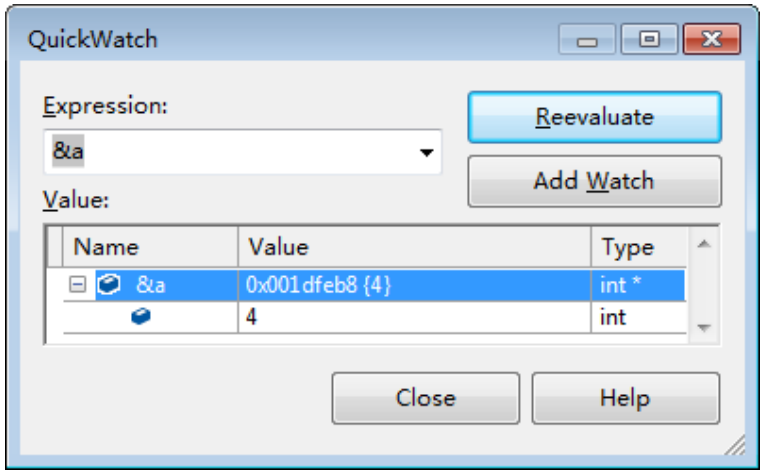


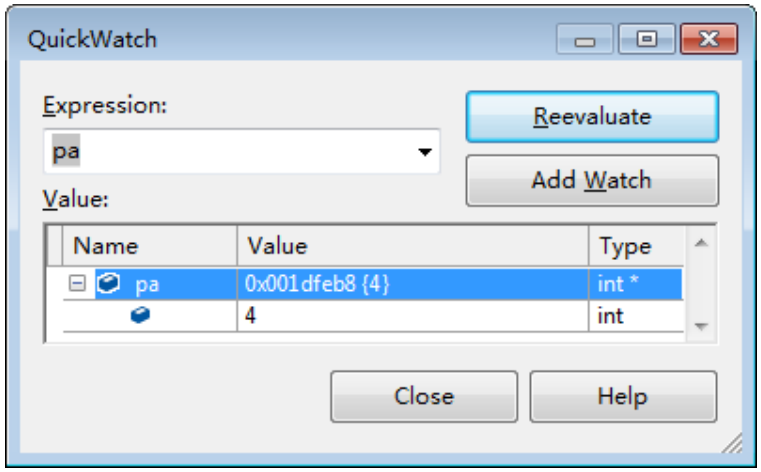
pa = &a;



示例：指针变量赋值比较

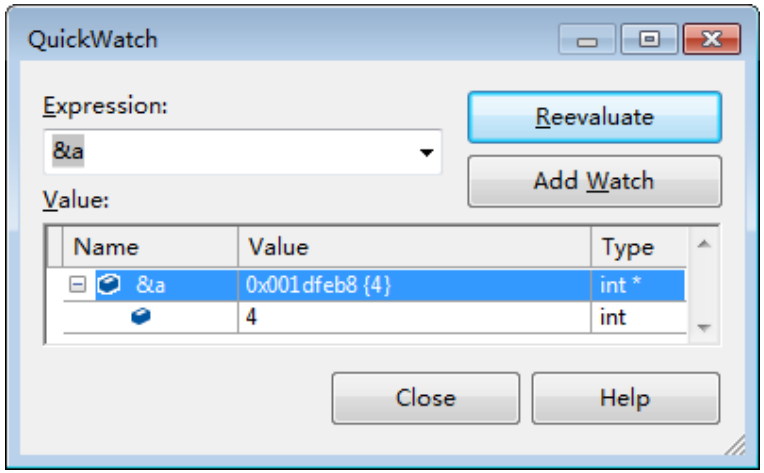
```
#include <stdio.h>
int main()
{
    int a = 4;
    int *pa;

    pa = &a;
    printf("%d", *pa);
    return 0;
}
```





示例：指针变量赋值比较

```
#include <stdio.h>
int main()
{
    int a = 4;
    int *pa;

    //pa = &a;
    pa = 0x001dfeb8;
    printf("%d", *pa);
    return 0;
}
```



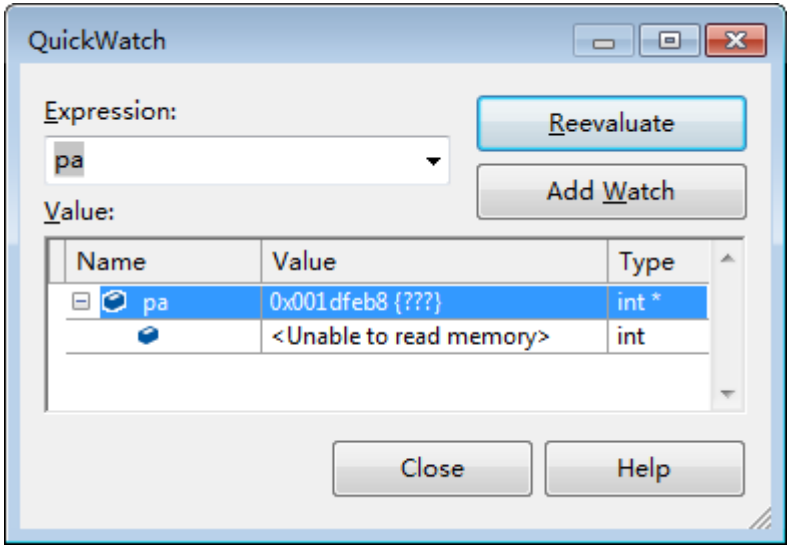
×

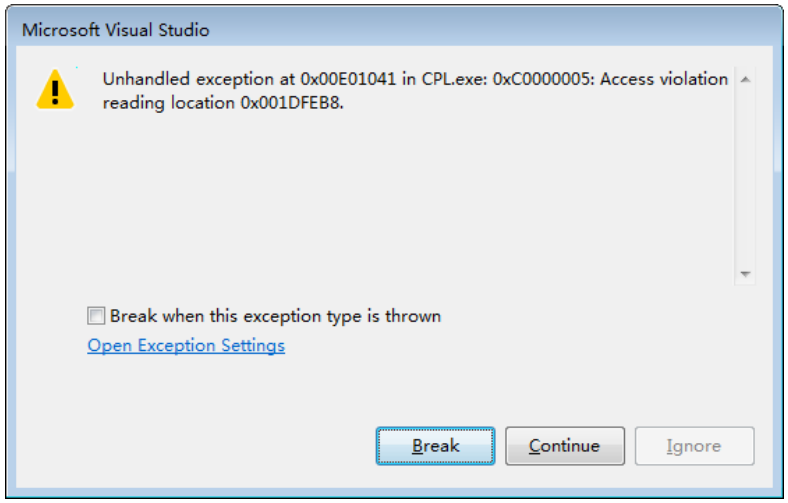
错误的原因？

error C2440: '=' : cannot convert from 'int' to 'int *'
类型不匹配

示例：指针变量赋值比较

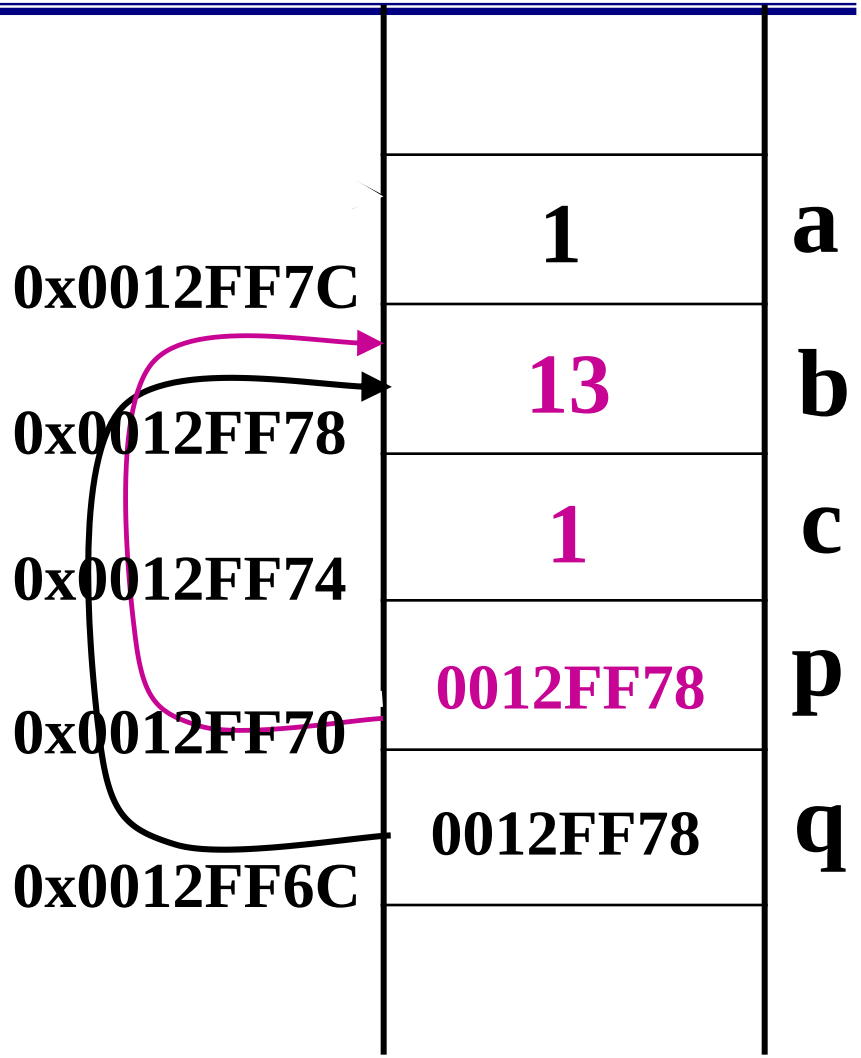
```
#include <stdio.h>
int main()
{
    int a = 4;
    int *pa;

    //pa = &a;
    pa = (int *)0x001dfeb8;
    printf("%d", *pa);
    return 0;
}
```





语法正确，语义错误！

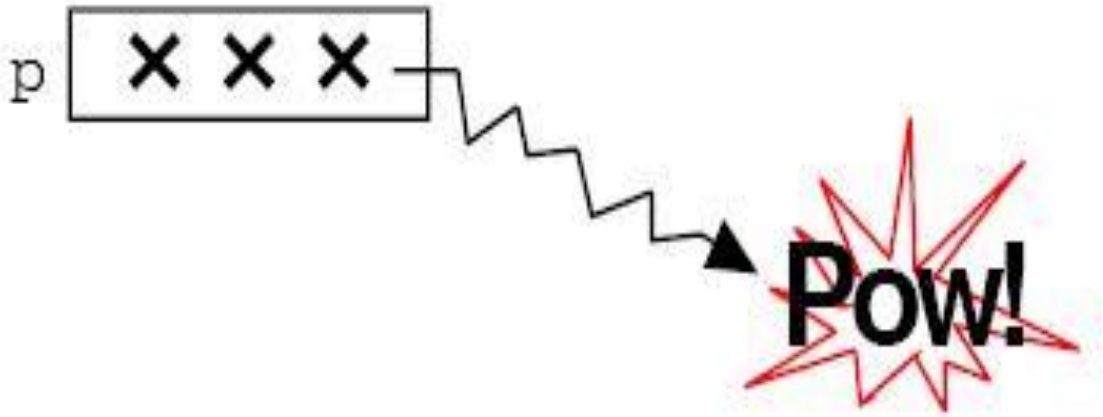
```
int   a,  b,  c, *p,  *q;

a   =   1;

b  =   2;

c   =   3;

p   =   &a;

q   =   &b;

c   =   *p;

p   =   q;

*p   =   13;
```



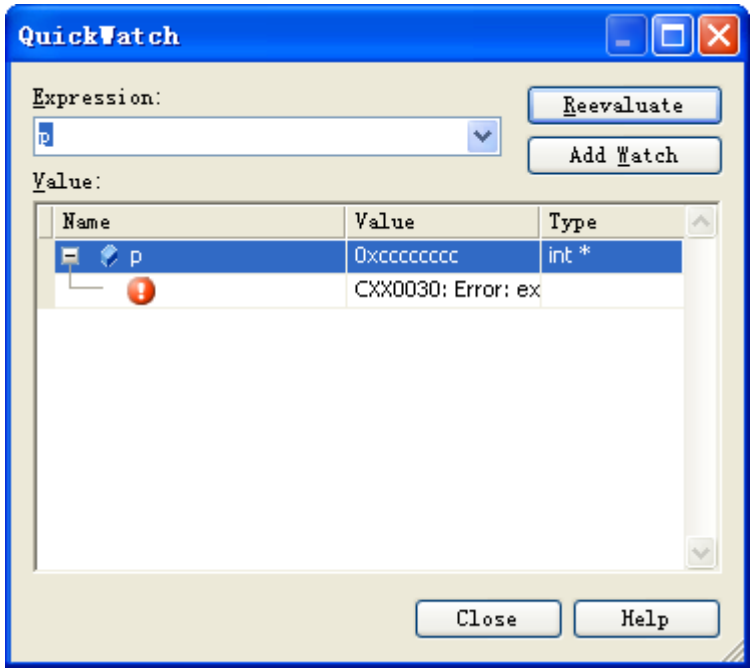
- 指针常量NULL表示一个指针不指向任何有效的数据。

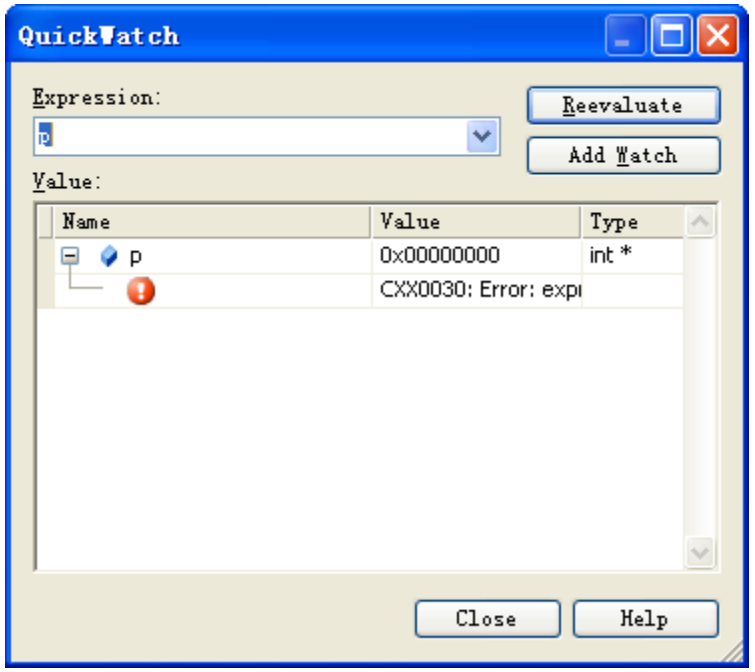
```
int  x, *p;
p = &x;
*p = 1;
p = NULL;
*p = 2;
```

一个其值为NULL的指针不同于一个未初始化的指针。一个指针在定义后，是未被初始化的，此时若对它进行访问，将会出错。



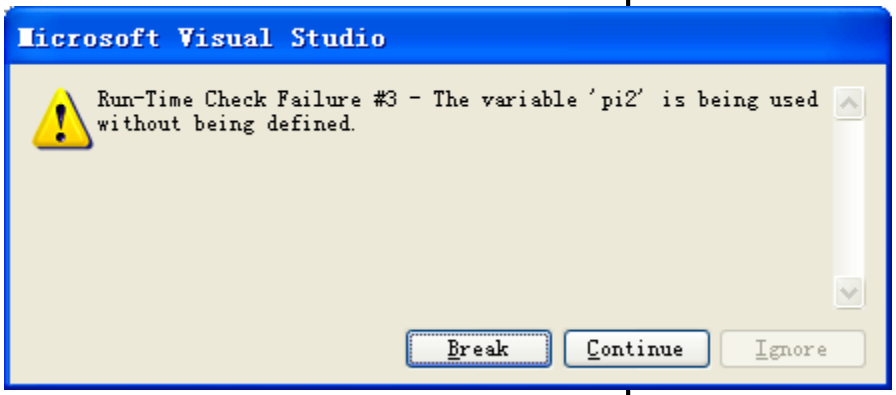
```
int   *p;                  // 未被初始化的指针
int   *p = NULL;           // 空指针
```





分析程序

```
int foobar(int *pi)
{
    *pi = 1024;
    return *pi;
}
int main()
{
    int *pi2;
    int ival = foobar(pi2);
    printf("%d", ival);
}
```



指针变量的初始化

一般形式：数据类型 *指针名 = 变量地址;

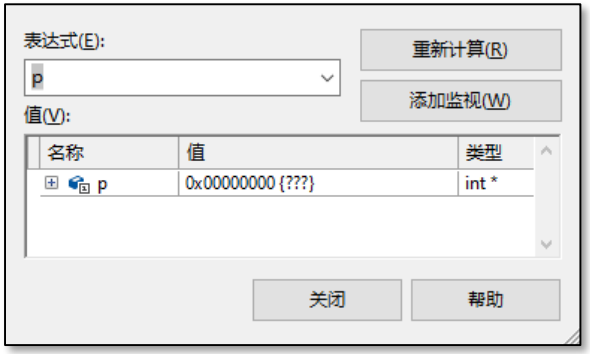
```
int i;
int  *p=&i;
```

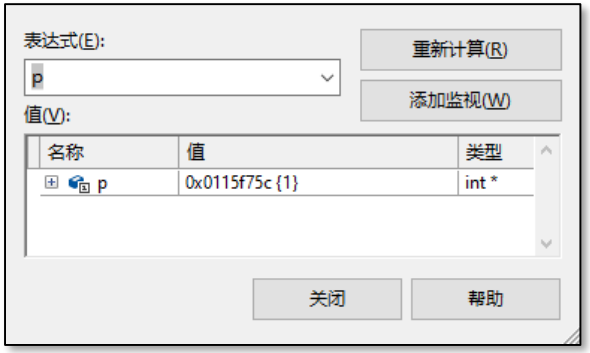
变量必须已说明过，类型应一致

```
int i;
int  *p=&i;
int  *q=p;
```

用已初始化指针变量作初值

```
main( )
{ int  i = 1;
  static  int  *p;
  p = &i;
  ......
}
```



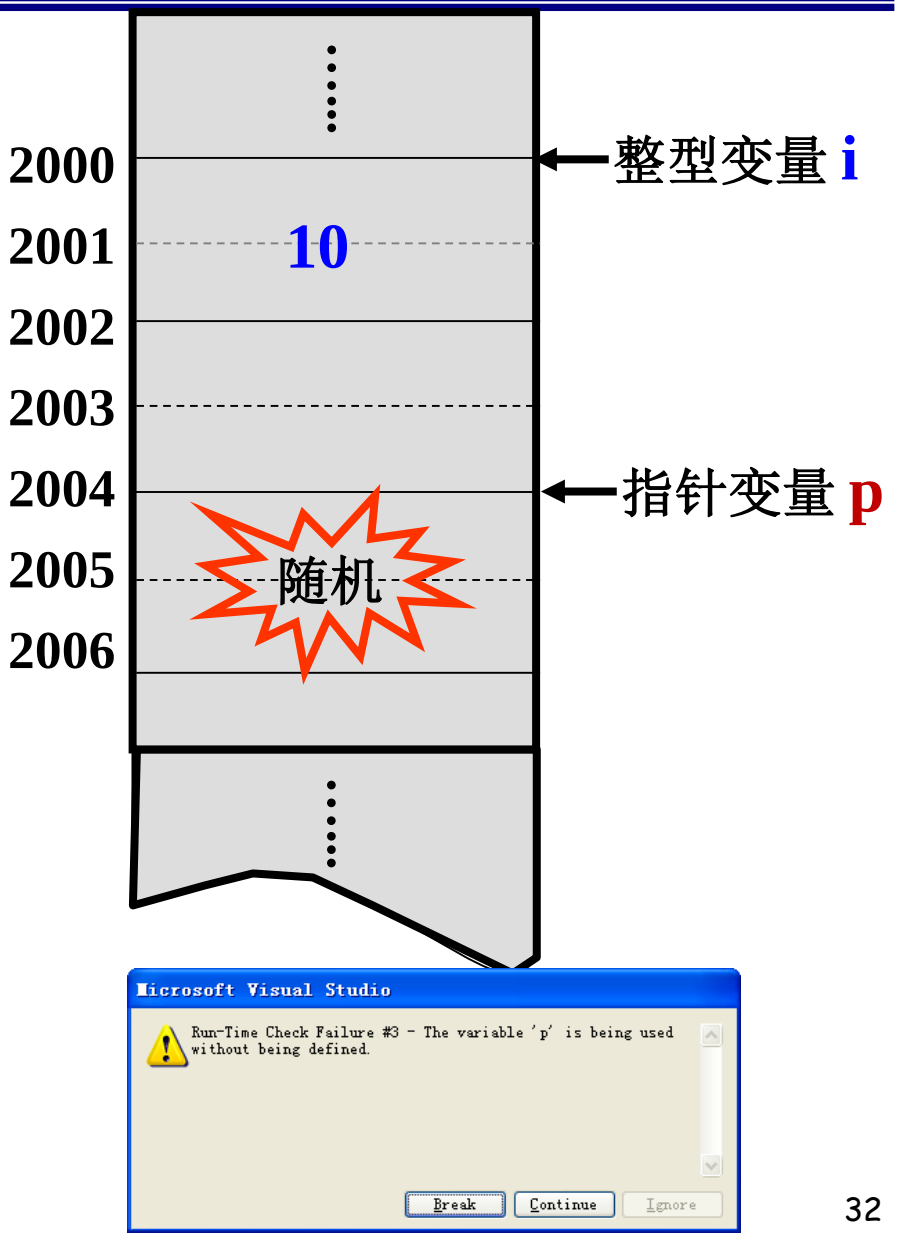


指针变量必须先赋值，再使用

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

危险！

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



零指针与空类型指针

零指针（NULL）：

- 定义:指针变量值为零
- 表示： int * p=0;

p指向地址为0的单元，系统保证该单元不作它用

表示指针变量值没有意义

```
#define   NULL   0
int   *p=NULL:
```

- p=NULL与未对p赋值不同
- 用途:
  - ◆避免指针变量的非法引用
  - ◆在程序中常作为状态比较

```
int   *p;
......
while(p != NULL)
{   ......
}
```

void *类型指针 （无类型指针）

- 表示：void *p;
- 使用时要进行强制类型转换

```
char  *p1;
void  *p2;
p1=(char  *)p2;
p2=(void *)p1;
```

表示不指定p是指向哪一种类型数据的指针变量

指针的算术运算

指针算术运算的结果依赖于指针的基类型

- p ± i : 指针p的值 ± i × d (i为整型数，d为p指向的变量所占字节数)
- p++, p--, p+i, p-i, p+=i, p-=i 等
- 若p1与p2指向同一数组，(p1 - p2) = 两指针间元素个数 ⇔ (p1 - p2) / d
- p1 + p2 无意义

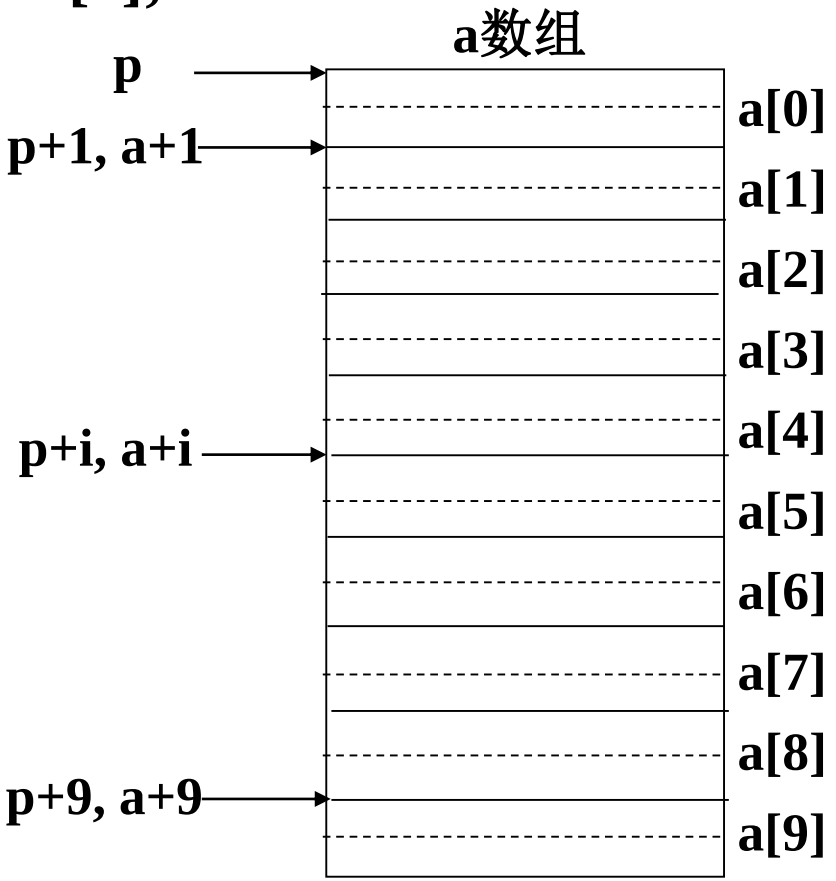
error C2110: '+' : cannot add two pointers

例 p指向float数，则 p + 1 ⇔ p + 1 × 4

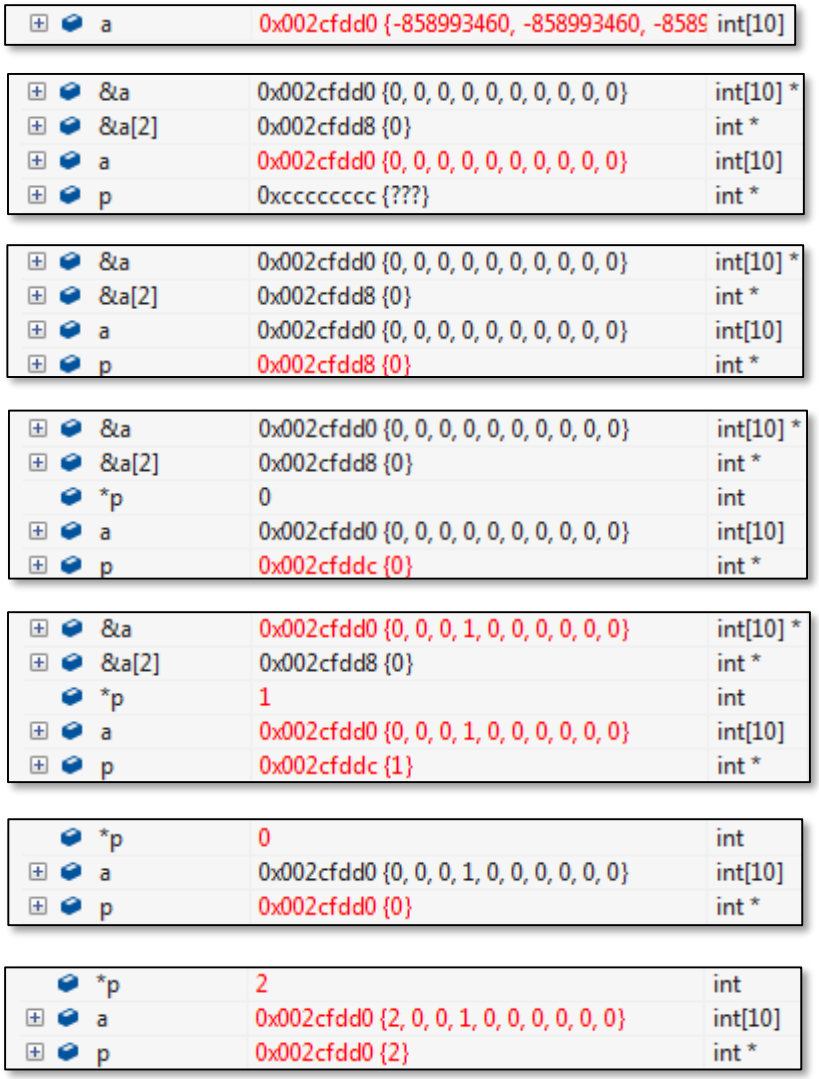
例 p指向int型数组，且p = &a[0];
则p + 1 指向a[1]

例 int a[10];
int *p = &a[2];
p++;
*p = 1;

例 int a[10];
int *p1 = &a[2];
int *p2 = &a[5];
则：p2 - p1 = 3;



```c
#include <stdio.h>
int main( )
{
    int a[10] = {0};
    int *p = &a[2];
    p++;
    *p = 1;
    p = a;
    *p = 2;
    return 0;
}
```



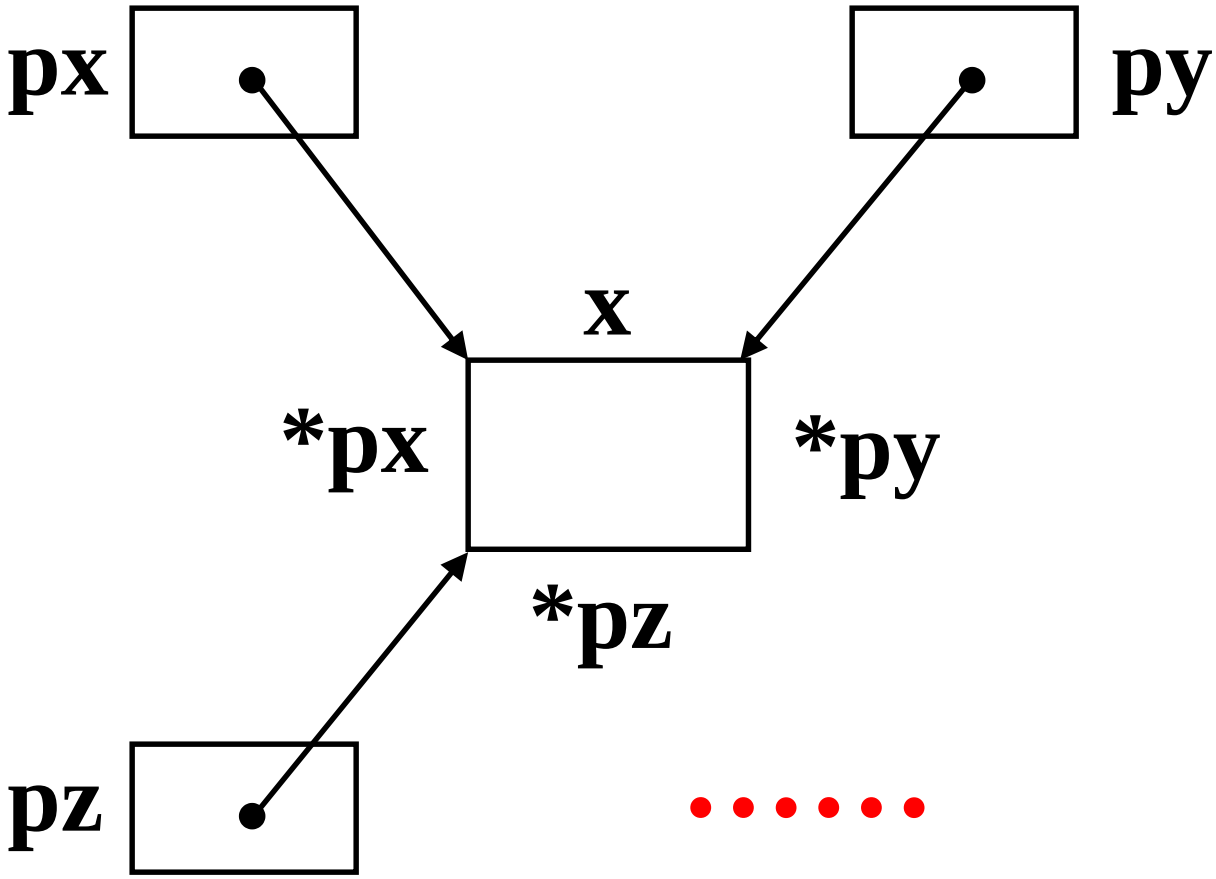
指针变量的关系运算

- 若p1和p2指向同一数组，则
  - p1 < p2　表示p1指的元素在前
  - p1 > p2　表示p1指的元素在后
  - p1 == p2　表示p1与p2指向同一元素
- 若p1与p2不指向同一数组，比较无意义
- p == NULL 或 p != NULL

指针的基本概念.....

指针能做什么？

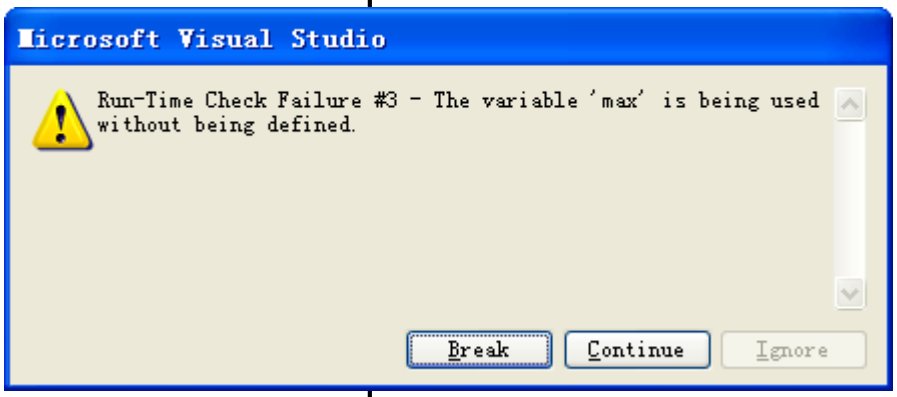
用处之一：提供了一种共享数据的方法，可以在程序的不同位置、使用不同的名字（指针）来访问相同的一段共享数据。如果在一个地方对该数据进行了修改，那么在其他的地方都能看到修改以后的结果。

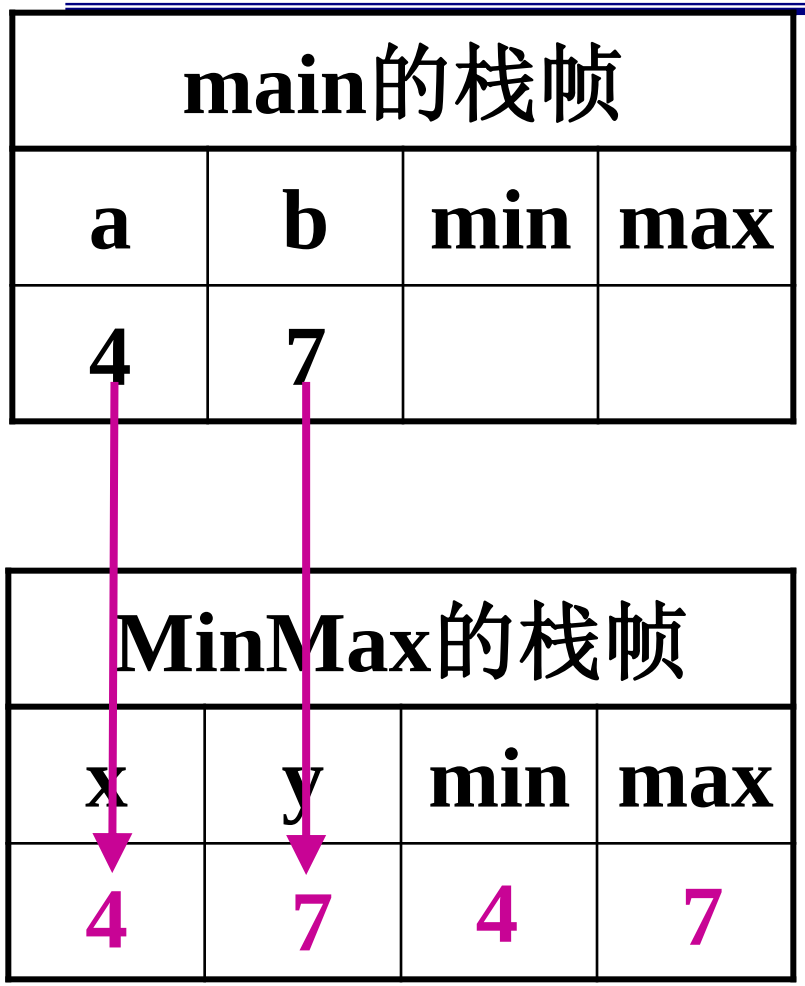

px
py
x
*px
*py
*pz
pz

```
void MinMax(int x, int y,
            int min, int max)
{
    min = x; max = y;
    if(x > y)
    {
        max = x;
        min = y;
    }
}
void main( )
{
   int  a, b, min, max;
   a = 4;  b = 7;
   MinMax(a, b, min, max);
   printf("%d, %d\n", min,max);
}
```

输出结果：

-858993460, -858993460

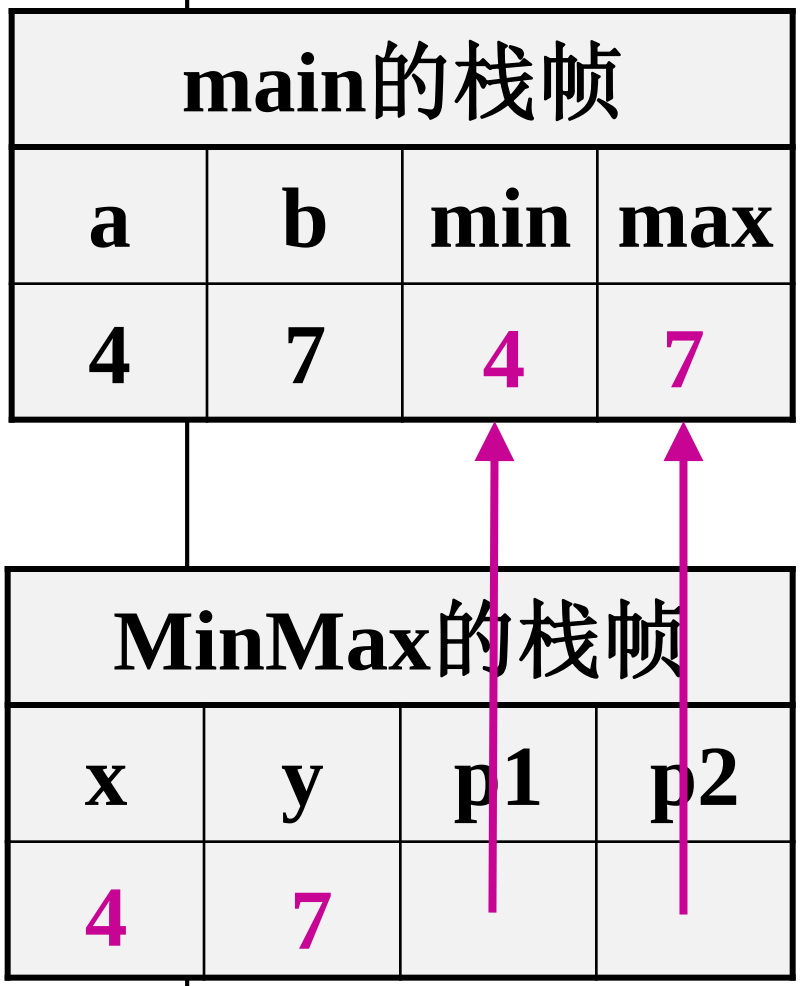





```
void MinMax(int x, int y,
            int min, int max)
{
     min = x; max = y;
     if(x > y)
     {
          max = x;
          min = y;
     }
}
void main( )
{
    int  a, b, min, max;
    a = 4;  b = 7;
    MinMax(a,b,min,max);
    printf("%d, %d\n", min,
           max);
}
```

```
void MinMax(int x, int y,
        int *p1, int *p2)
{
    *p1 = x; *p2 = y;
    if(x > y)
    {
        *p1 = x;
        *p2 = y;
    }
}
void main( )
{
   int  a, b, min, max;
   a = 4;  b = 7;
   MinMax(a, b, &min, &max);
   printf("%d, %d\n", min, max);
}
```

min、max的访问？



传值还是传地址？

- 传值：给予
- 传地址：索取

问题描述：计算一元二次方程的根。

$$a x^2 + b x + c = 0$$

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}, \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

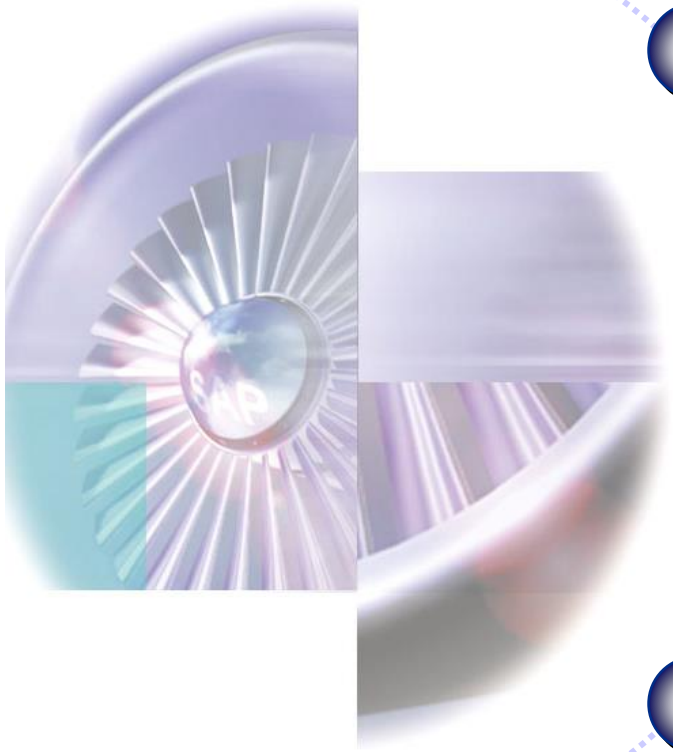
```
int  main( )
{
      double   a,  b,  c,  x1,  x2;

      /* 从键盘读入方程式的系数a, b, c  */
      GetCoefficients(?a,  ?b,  ?c);

      /* 求解方程式的两个根x1,  x2  */
      SolveQuadratic(?a,  ?b,  ?c,  ?x1,  ?x2);

      /*  显示方程式的两个根x1,  x2  */
      DisplayRoots(?x1,  ?x2);
}
```

```
int  main( )
{
      double   a,  b,  c,  x1,  x2;

      /* 从键盘读入方程式的系数a, b, c  */
      GetCoefficients(&a,  &b,  &c);

      /* 求解方程式的两个根x1,  x2  */
      SolveQuadratic(a,  b,  c,  &x1,  &x2);

      /*  显示方程式的两个根x1,  x2  */
      DisplayRoots(x1,  x2);
}
```

总结

- C语言程序设计中使用指针的好处
  - 使程序简洁、紧凑、高效
  - 有效地表示复杂数据结构（如: 链表、八叉树）
  - 动态分配内存
  - 得到函数的多个返回值

Lecture 7: 指针



1. 基本概念
2. 指针变量
3. 指针与数组
4. 指针与字符串