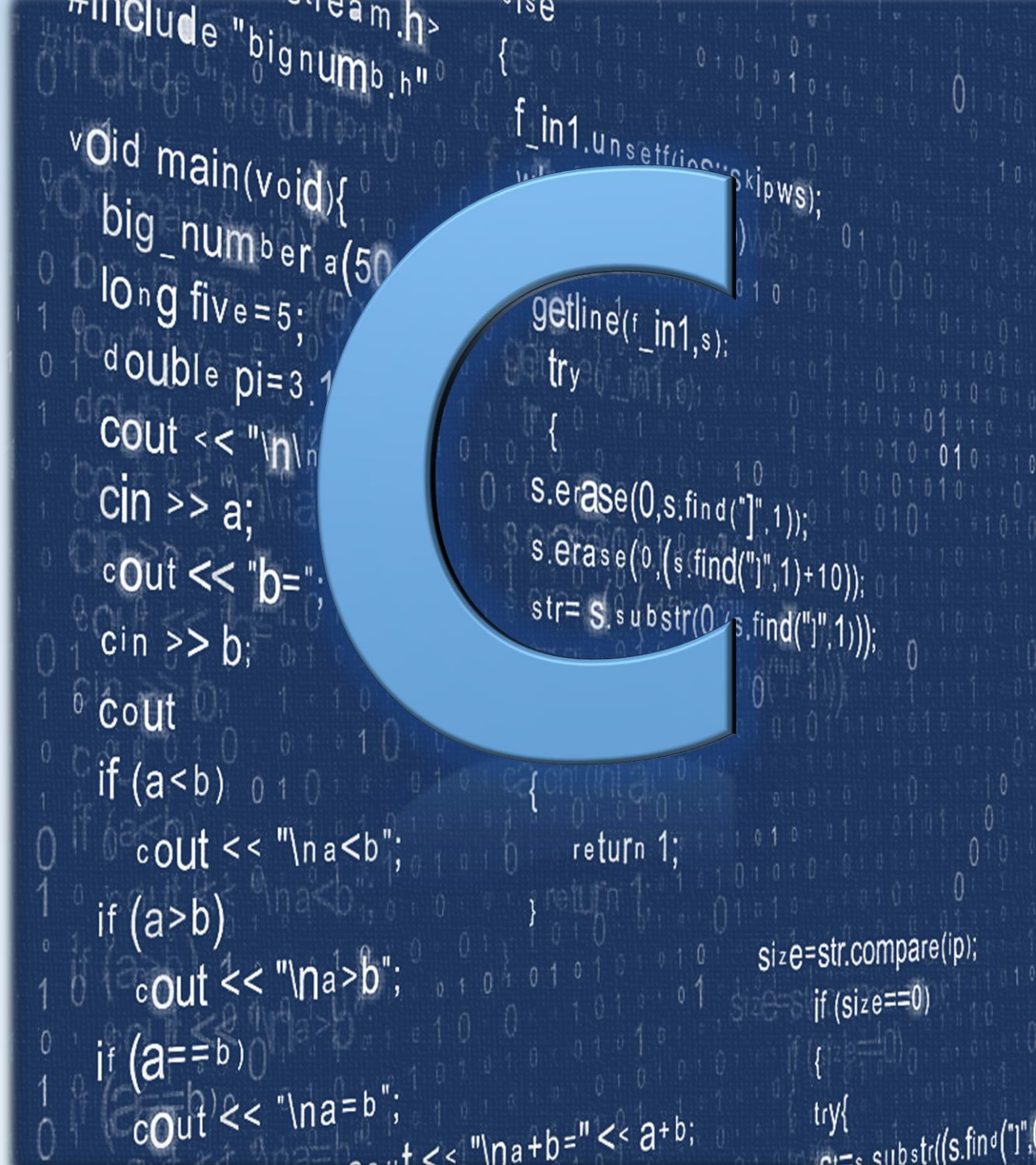
《 C语言程序设计》

丁盟
C语言课程组



上一讲知识复习

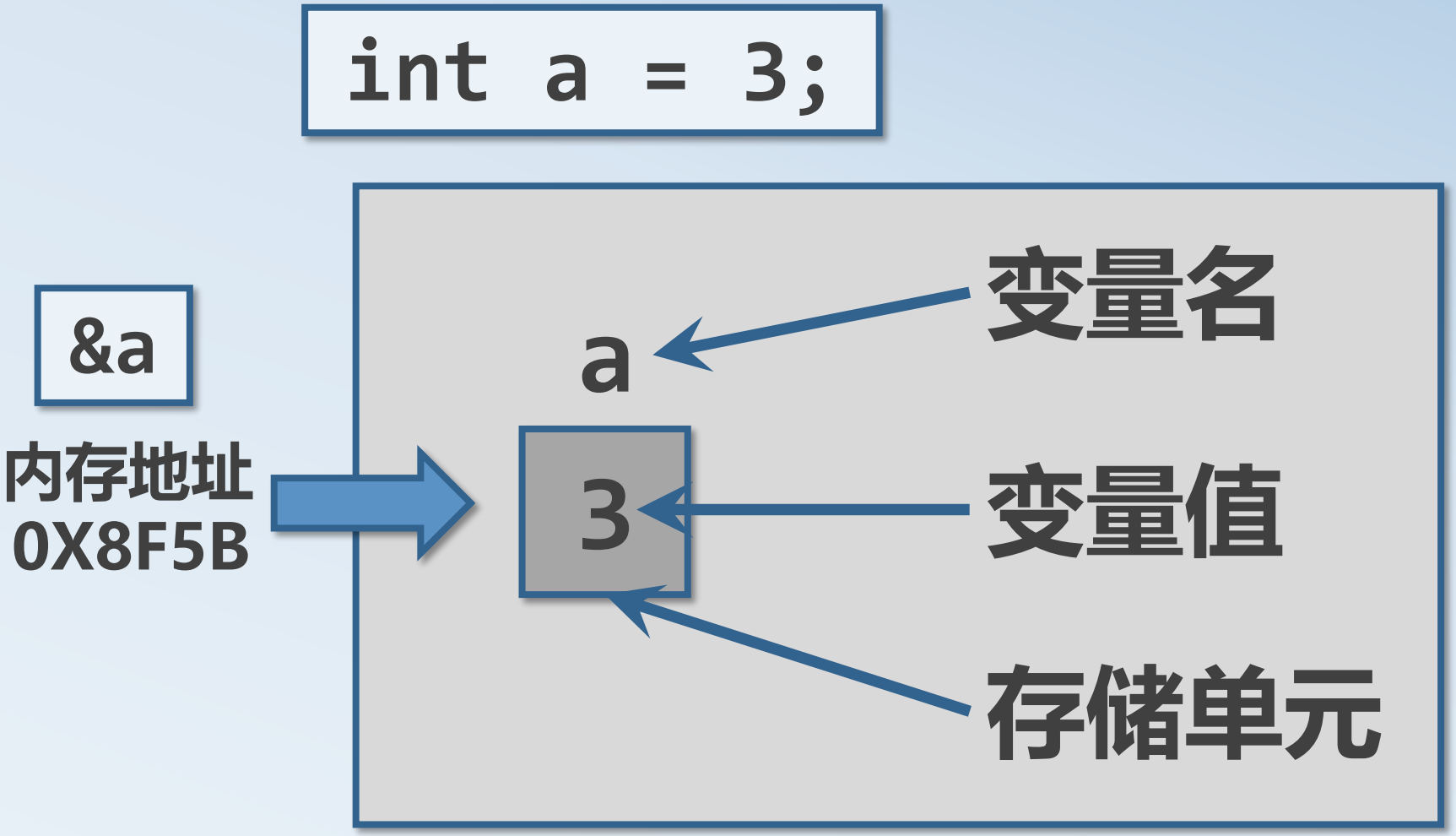
- 掌握数组声明的方法。
- 掌握一维数组、二维数组在内存中的存储。
- 掌握通过下标方式访问数组中各元素的方法。

本讲教学目标

- 掌握指针变量的声明方法。
- 理解指针变量的两个关键点：
  - 存放地址
  - “捆绑”一块内存空间(它是有类型的)
- 掌握多重指针的声明及初始化。
- 掌握通过指针访问所指内存空间中数据对象的方法。
- 理解const指针。
- 了解空指针及通用指针的作用。
- 了解指针变量的运算。

本章授课内容

指针变量的声明及使用



指针变量的声明及使用

```
int a = 3;
```

&a

8F5B	8F5C	8F5D	8F5E
0	0	0	3

什么是指针？ **地址就是指针！**

什么决定了数据的长度？ **类型！**

指针是否有类型？ **有！**

&a是指针吗？ **是！**

指针变量的声明及使用

```
#include <stdio.h>
int main(void)
{
    int a = 3;
    double b = 3.14;

    printf("int a 内存中存储的数据为：%d\n", a);
    printf("int a 内存中占用的字节数为：%d\n", sizeof(a));
    printf("int a 内存中存储的首地址为：%p\n", &a);

    printf("double b 内存中存储的数据为：%lf\n", b);
    printf("double b 内存中占用的字节数为：%d\n", sizeof(b));
    printf("double b 内存中存储的首地址为：%p\n", &b);

    return 0;
}
```

&a,&b是指针吗？

打印地址使用%p

指针变量的声明及使用

- **注：C语言中所有的变量都是要占据内存的，并且其占据内存大小是由变量类型所决定的。**

- **注：所有的指针变量都占据相同大小的内存，在32位电脑上，指针变量占4个字节。**

指针变量的声明及使用

- 指针变量的声明语法：

```
int * p = &x;
```

在变量前加 * 即可

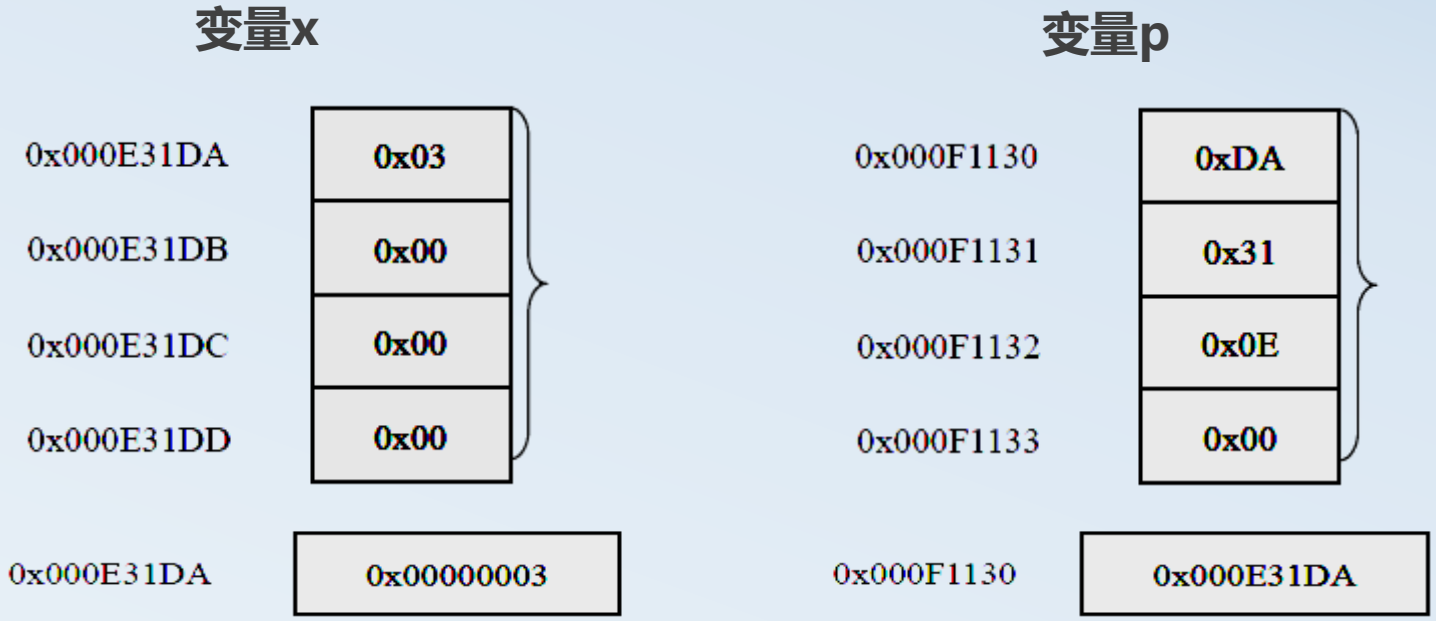
  - 指针变量是用来存地址的
  - 类型说明符决定了指针存储的地址的类型

- 关于指针的两种基本操作：
  - **&变量**：取出变量本身所在的内存首地址
  - ***地址**：*(变量地址) ⇔ 变量
    - 修改存储的变量地址中的数据值（出现在等号左边）
    - 取出存储的变量地址中的数据值

指针变量的声明及使用

- 例 设有int x=3; 请声明一个名为p的指针变量，p的值为数据对象3的存放地址

int * p=&x; 或 int * p; p=&x;



指针变量的声明及使用

- 例 设有double x=3.5;请声明一个名为p的指针变量，p值为数据对象3.5的存放地址。请画出p、x的存储关系示意图。

- 例 设有int x = 3; int * p = &x;，则表达式*p 的结果为3，请对此进行分析。

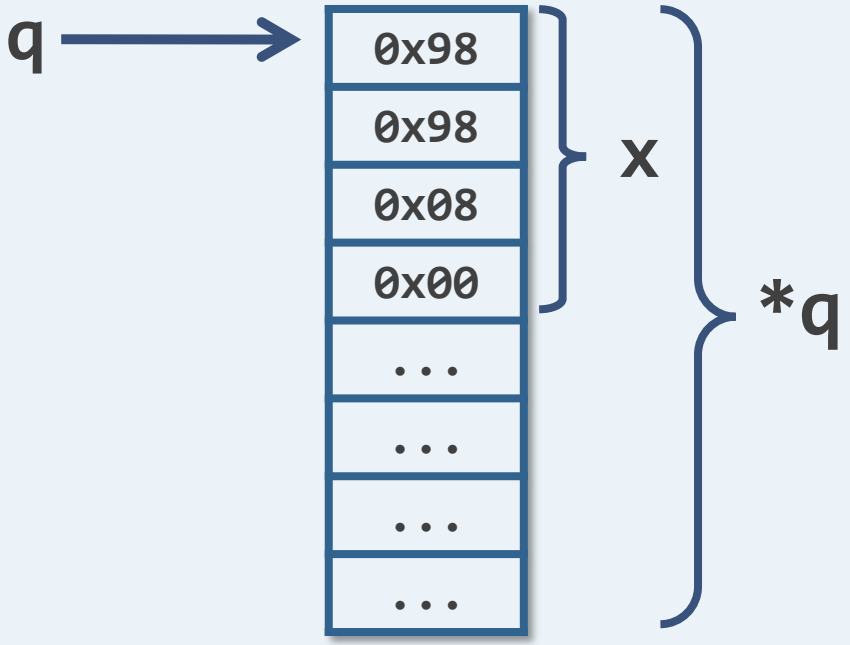
指针变量的声明及使用

- 例 设int x = 3; int * p = &x;，请分析下述表达式的值：x、&x、p、&p、*p。

```
#include <stdio.h>
int main(void)
{
    int x = 3;
    int * p = &x;
    printf("表达式x 的值为:  %d\n", x);
    printf("表达式&x 的值为: %p\n", &x);
    printf("表达式p 的值为:  %p\n", p);
    printf("表达式&p 的值为: %p\n", &p);
    printf("表达式*p 的值为: %d\n", *p);
    return 0;
}
```

指针变量的声明及使用

```
#include <stdio.h>
int main(void)
{
    int x = 0x89898;
    double * q = (double*)&x;

    printf("%lf\n", *q);

    printf("%d\n", *q);
    return 0;
}
```



注意：当指针类型与赋值变量的类型不同时会发生内存溢出或者内存截断的问题.

指针变量的声明及使用

- 指针定义小结：
  - 指针变量也是变量，& + 变量名 表示该变量的地址
  - 如果指针变量p的值为变量a的地址，则称p指向a
  - 指针变量p也是有类型的

（它的类型含义：该变量只能存放它所对应的类型去掉一个*后的那个类型的变量或常量的地址，例如：int * p，则p只能存放int的变量或常量的地址。）

  - p是指针变量，&p 为p的地址，*p 表示p指向单元的值

本章授课内容

指针变量的运算

- 指针的运算关系：
  - 表示该指针（操作数）所指向的内存空间（*）
  - 取该“变量”（操作数）地址（&）
  - 指针的赋值（=）
  - 指针的加减运算（+、-）
  - 指针的关系运算（<、>、==）

指针变量的运算

- 指针的赋值分为两种情况：
  - 将变量地址赋给指针----指向变量
  - 将指针赋给指针----与被赋值的指针指向同一个变量
- 例 设有int x=3; int * p = 0; int * q = 0; 请观察下面的语句

```
p = &x;
q = p;
*q = 5;  // 通过指针来修改它所指向的内存区域中的内容
```

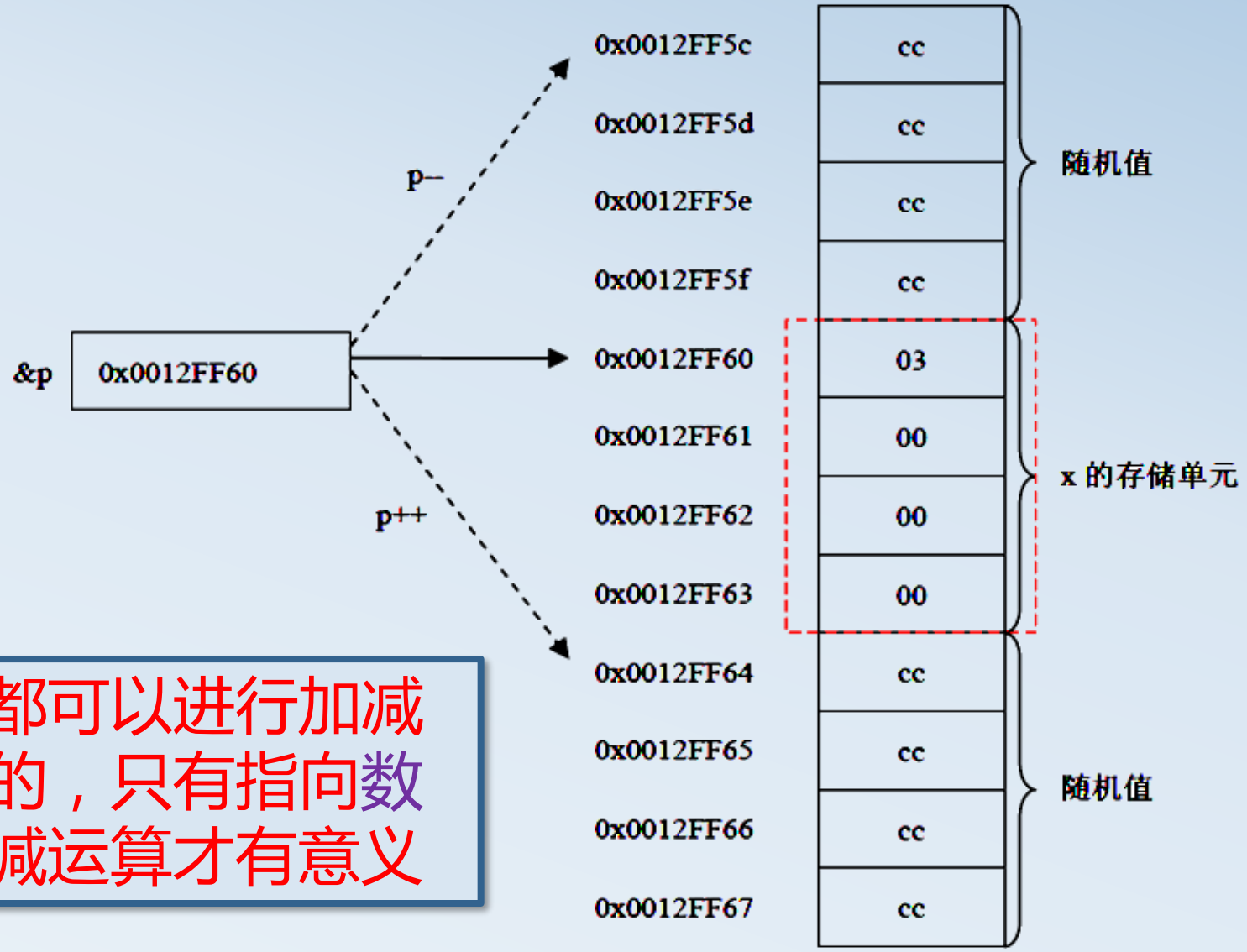
指针变量的运算

```
#include <stdio.h>
int main(void)
{
    int x = 3;
    int *p = &x;
    printf("p = %p, *p = %d\n", p, *p);
    p++; printf("p = %p, *p = %d\n", p, *p);
    p--; printf("p = %p, *p = %d\n", p, *p);
    p--; printf("p = %p, *p = %d\n", p, *p);
    return 0;
}
```

```
p = 0093F848, *p = 3
p = 0093F84C, *p = -858993460
p = 0093F848, *p = 3
p = 0093F844, *p = -858993460
```

指针的加减运算中是以指针的类型的长度为单位的。

指针变量的运算



指针并不是都可以进行加减和比较运算的，只有指向数组的指针加减运算才有意义

指针变量的运算

- 设有int x = 3, y = 4; int * p = &x, * q = &y; 请依次观察下面的语句。

```
if(p!=q)
{
    printf("p,q指向了两块不同的内存区域\n");
}
p=q;
if(p==q)
{
    printf("p,q指向了两块相同的内存区域\n");
}
```

指针变量的运算

- 例 设有int x = 3, y = 4, z; int * p = &x, * q=&y;
  请依次观察下面的语句。

```
y = *p + 5;
y = *p * 6;
z = *p + *q;
```

指针：

1.指针也是变量，也可以构成表达式。
2.指针可以将不同内存区域的数据联系起来。
3.灵活掌握指针的用法是一个程序员必备的。

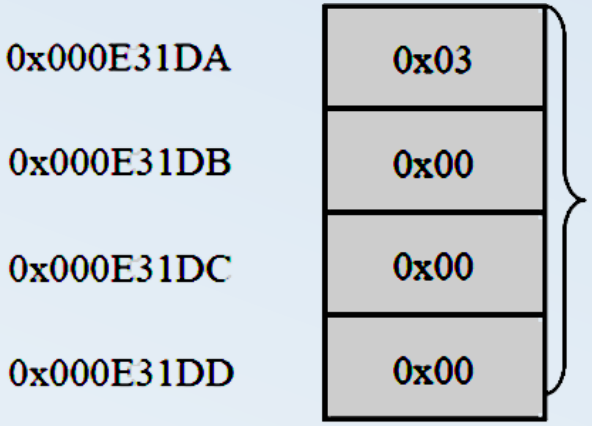
本章授课内容

多重指针的声明及使用

- 若指针的存储内容为另一个指针的地址。则成称这个指针为多重指针。

```
int x = 3;
```

变量x

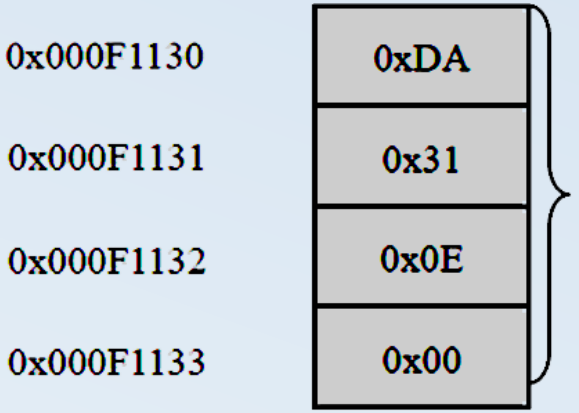


0x000E31DA

0x00000003

```
int * p = &x;
```

变量p

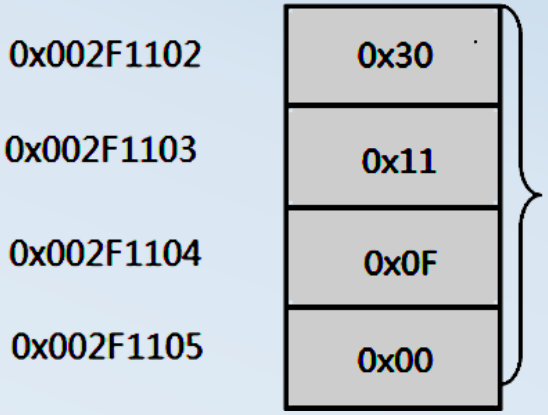


0x000F1130

0x000E31DA

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

变量q



0x002F1102

0x000F1130

多重指针的声明及使用

- 多重指针：
  - 指向指针的指针
  - 二重指针变量：

    声明说明符 * 类型限定符列表 * 类型限定符列表
  - 三重指针变量：

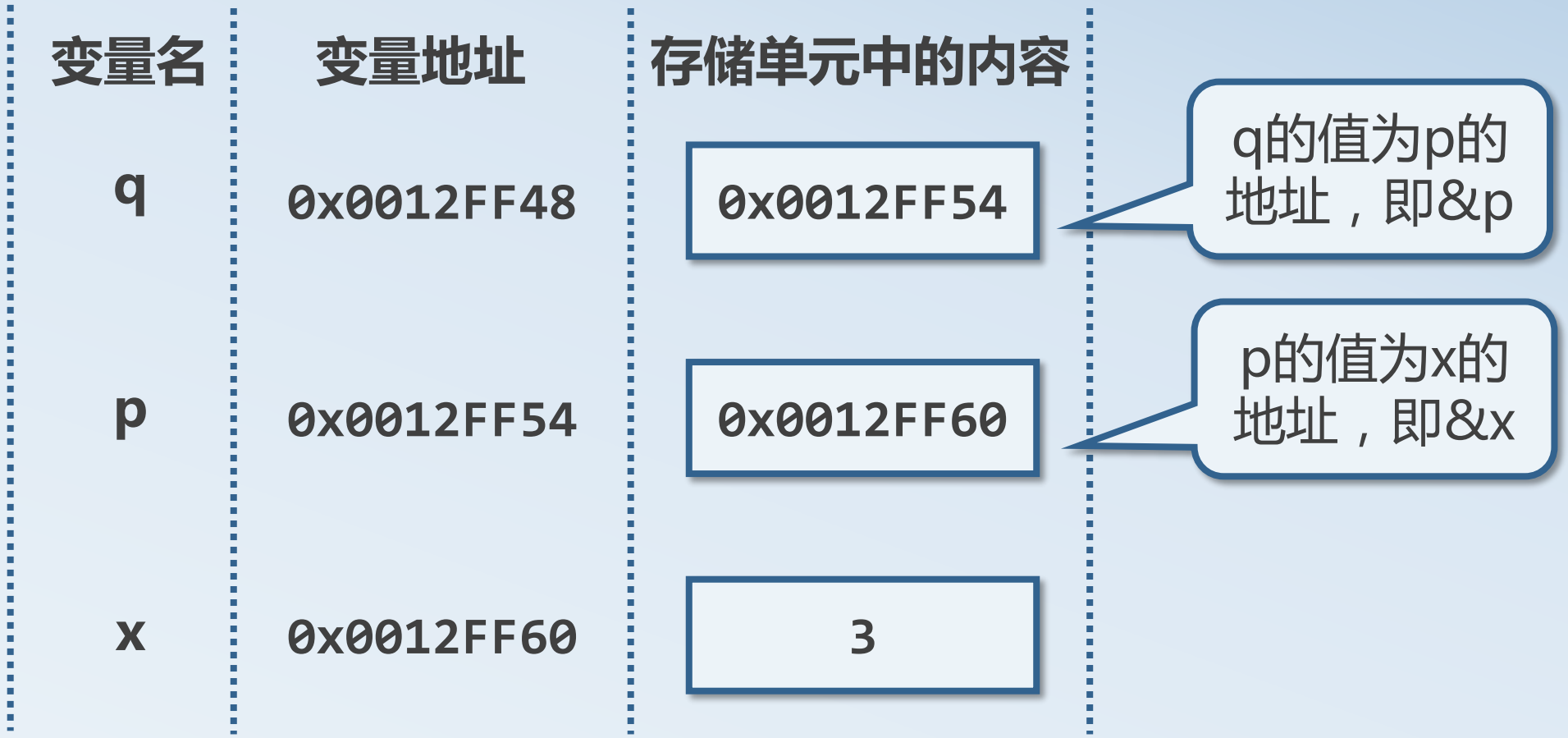
    声明说明符 * 类型限定符列表 * 类型限定符列表 * 类型限定符列表
  - 多重指针

多重指针的声明及使用

```
#include <stdio.h>
int main(void)
{
    int x = 3;  int * p = &x;  int ** q = &p;
    printf("表达式 x  的值为: %d\n", x  );
    printf("表达式 *p 的值为: %d\n", *p );
    printf("表达式 **q的值为: %d\n", **q);
    printf("表达式 &x 的值为: %p\n", &x );
    printf("表达式 p  的值为: %p\n", p  );
    printf("表达式 *q 的值为: %p\n", *q );
    printf("表达式 &p 的值为: %p\n", &p );
    printf("表达式 q  的值为: %p\n", q  );
    printf("表达式 &q 的值为: %p\n", &q );
    return 0;
}
```

多重指针的声明及使用

```
int x = 3;  int * p = &x;  int ** q = &p;
```



多重指针的声明及使用

- 例 根据下列语句画出存储示意图。

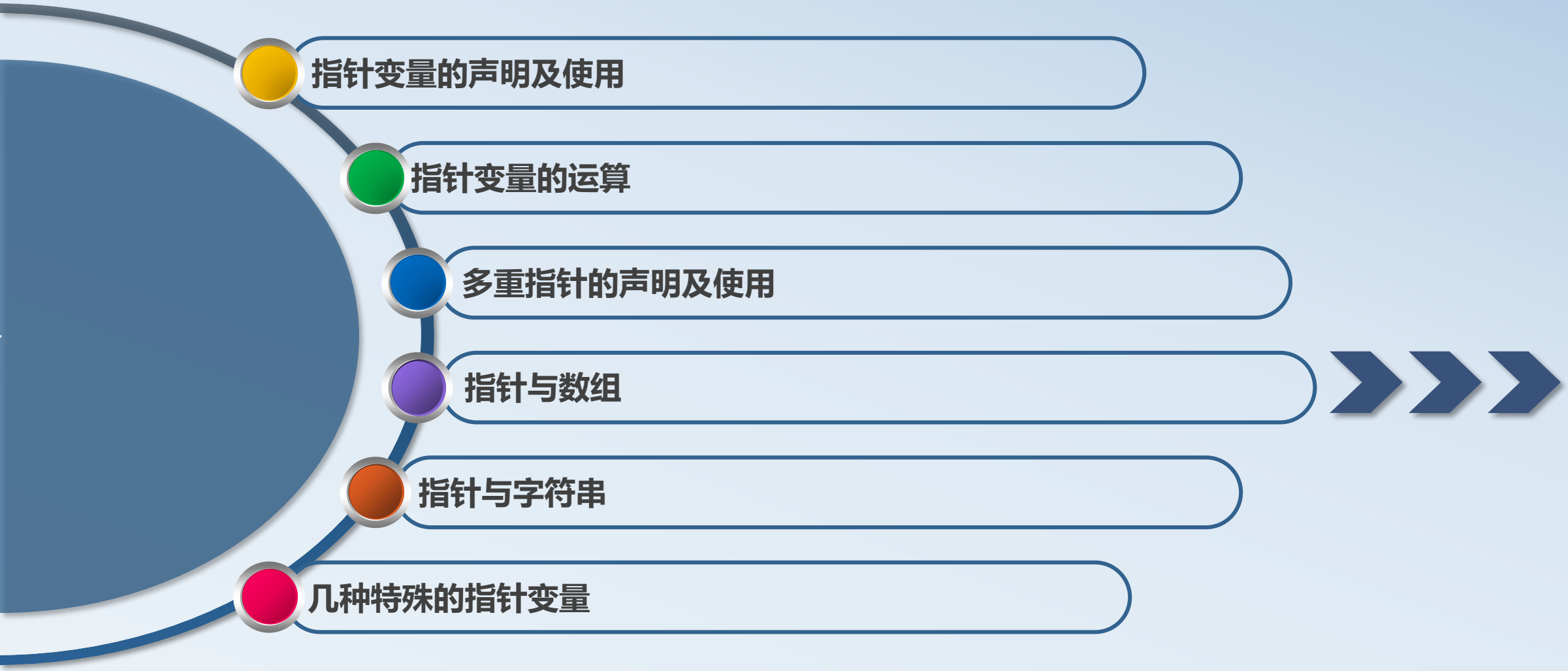
```
double f = 3.24;
double *p1 = &f;
double ** p2 = &p1;
double *** p3 = &p2;
```

多重指针的声明及使用

- **注意：**

  1. **多重指针变量也是变量.**
  2. **多重指针变量赋值过程中一定要注意类型.**
  3. **多重指针中，变量的等价形式.**

本章授课内容



指针与数组

- 在数组声明符中，当仅出现一个中括号时，所声明的数组为一维数组

```
int arr[5] = {11, 12, 13, 14, 15};
```

数组名	数组各元素地址表示	数组各元素表示	数组各元素内容
arr	&arr[0]	arr[0]	11
	&arr[1]	arr[1]	12
	&arr[2]	arr[2]	13
	&arr[3]	arr[3]	14
	&arr[4]	arr[4]	15

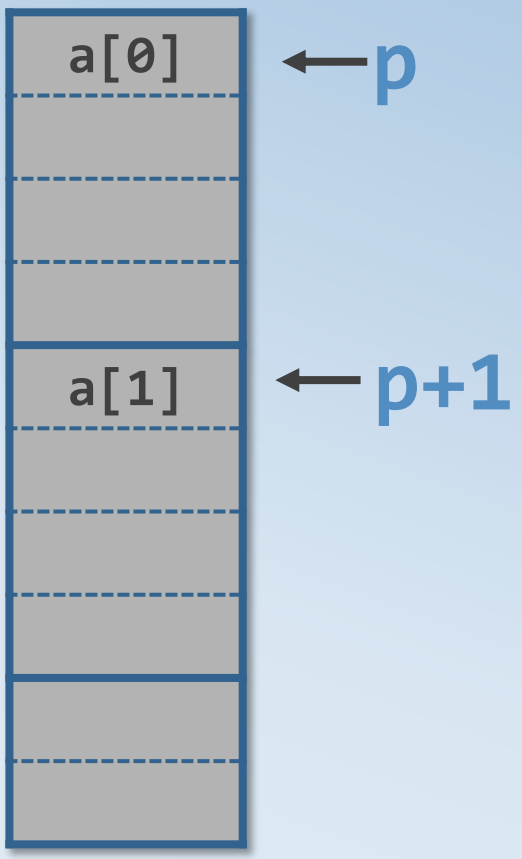
指针与数组

```
#include <stdio.h>
int main(void)
{
    int a[5] = {11, 12, 13, 14, 15};

    printf("&a[0]   = %p\n", &a[0]);
    printf("&a[0]+1 = %p\n", &a[0] + 1);

    printf("a   = %p\n", a);
    printf("a+1 = %p\n", a + 1);

    printf("&a   = %p\n", &a);
    printf("&a+1 = %p\n", &a + 1);
    return 0;
}
```

```
&a[0]   = 00BEFDD4
&a[0]+1 = 00BEFDD8
a   = 00BEFDD4
a+1 = 00BEFDD8
&a   = 00BEFDD4
&a+1 = 00BEFDE8
```

指针与数组

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



- p值增加多少？
- 如果int改成double，p值增加多少呢？
- 指针的加减运算是以其所指类型的字节长度为单位的

p+1 <==> p 移动 sizoef(所指类型）字节

指针与数组

- 条件：p,q是指向同一数组。
  - 可以进行**指针和整数的加减运算**
  - **同类型指针之间的减法运算**

运算方式	说　　明
p+n	p之后第n个元素的地址
p-n	p之前第n个元素的地址
p++	p作为当前操作数，然后后移一个元素
++p	p后移一个元素，然后作为当前操作数
p--	p作为当前操作数，然后前移一个元素
--p	p前移一个元素，然后作为当前操作数
p-q	表示p和q两者之间的元素个数

指针与数组

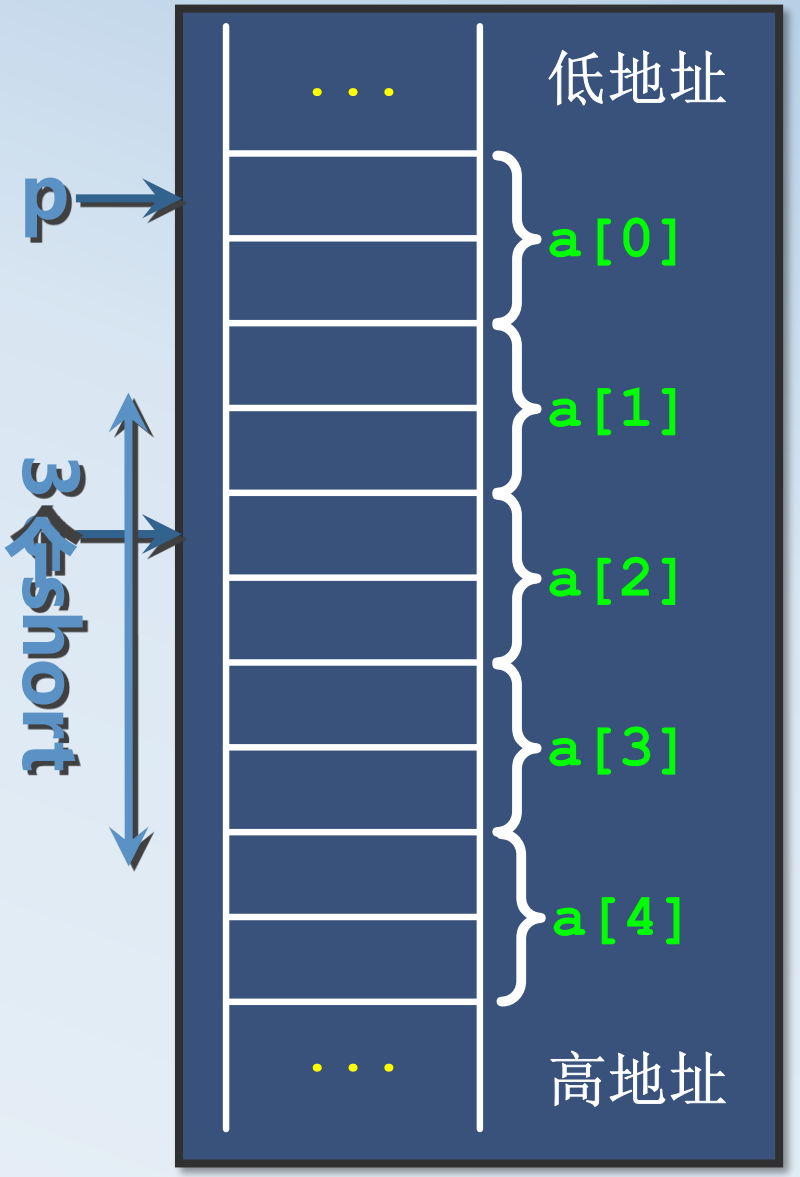
- 条件：指向同一数组的两个指针才能进行关系运算

  p<q、p<=q、p==q、p!=q、p>=q、p>q

  - p<q:判断p所指元素是否在q所指元素之前
  - 其他运算的含义与上述类似
  - 若p，q不是指向同一数组的指针，则运算无意义
- 不能与非指针类型变量进行比较,但可与NULL(即0值)进行等或不等的关系运算
  - 判断p是否为空指针
  - P == NULL
  - p != NULL

指针与数组

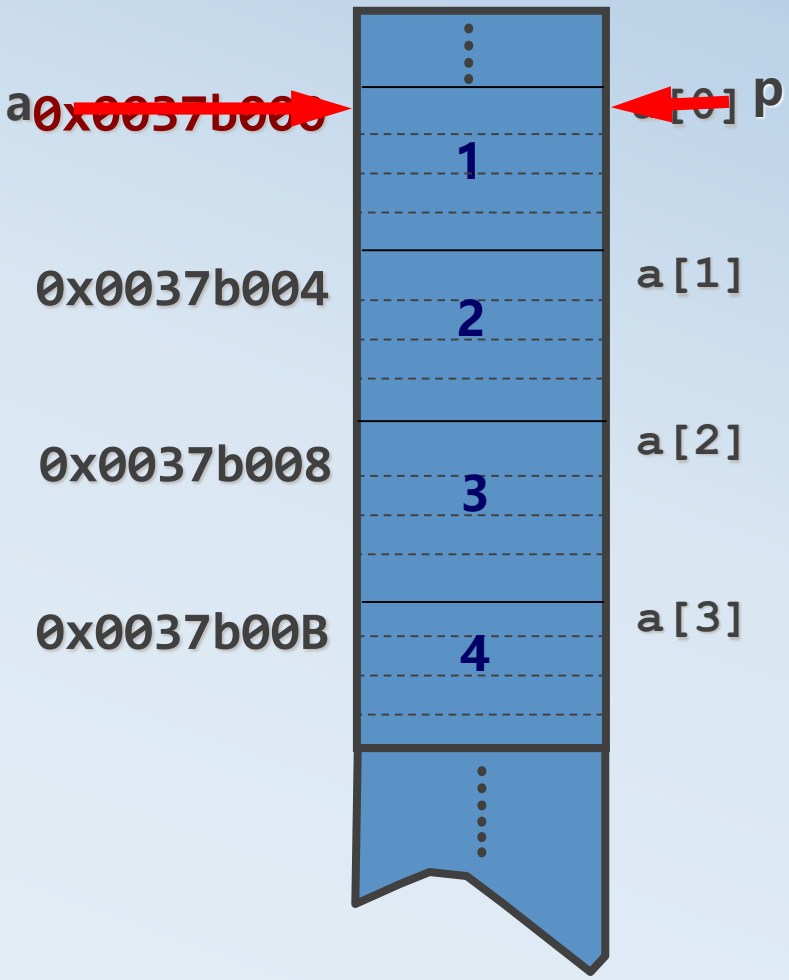
```
short a[5] = {1,2,3,4,5};
short *p, *q;
p = &a[0];
q = p+2;
p += 3;
printf("%d", *p++);
scanf("%d", --q);
if (p>q)
  printf("%d", p-q);
else
  printf("%d", q-p);
```



指针与数组

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

- 数组名就是一个常量指针
- 当一个指针p指向数组首元素，p也可以做数组名用
- 数组元素的几种等价形式
  - a[i]
  - *(a+i)
  - p[i]
  - *(p+i)



指针与数组

- 方法1:下标法

```
#include <stdio.h>
int main(void)
{
  int a[4] = {1,2,3,4};
  int * pa = a, i;
  for(i=0;i<4;i++)
  {
    printf("a[%d]:%d\n",i,a[i]);
  }
  for(pa = a,i = 0; i < 4;i++)
  {
    printf("pa[%d]:%d\n",i,pa[i]);
  }
  return 0;
}
```

- 方法2:指针法

```
#include <stdio.h>
int main(void)
{
  int a[4] = {1,2,3,4};
  int *pa, i;
  for(i=0;i<4;i++)
  {
    printf("*(a+%d):%d\n",i,*(a+i));
  }
  for(pa = a;pa < a + 4;pa++)
  {
    printf("*pa:%d\n",*pa);
  }
  return 0;
}
```

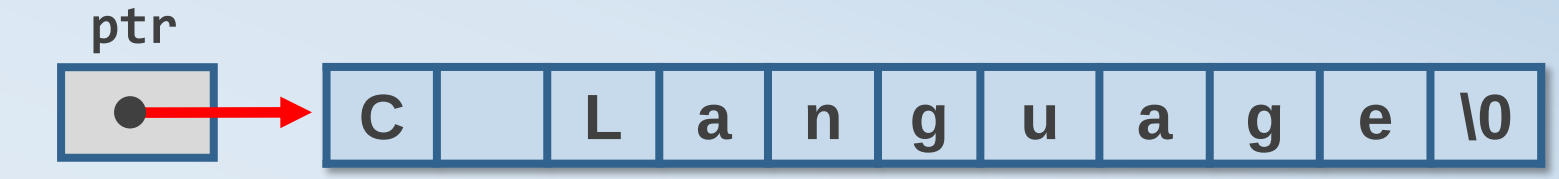
本章授课内容

指针与字符串

- 在C语言中，可以将字符指针指向字符串
- 被指向的字符串可以是一个**常量字符串**，也可以是一个存储着字符串的**字符数组**

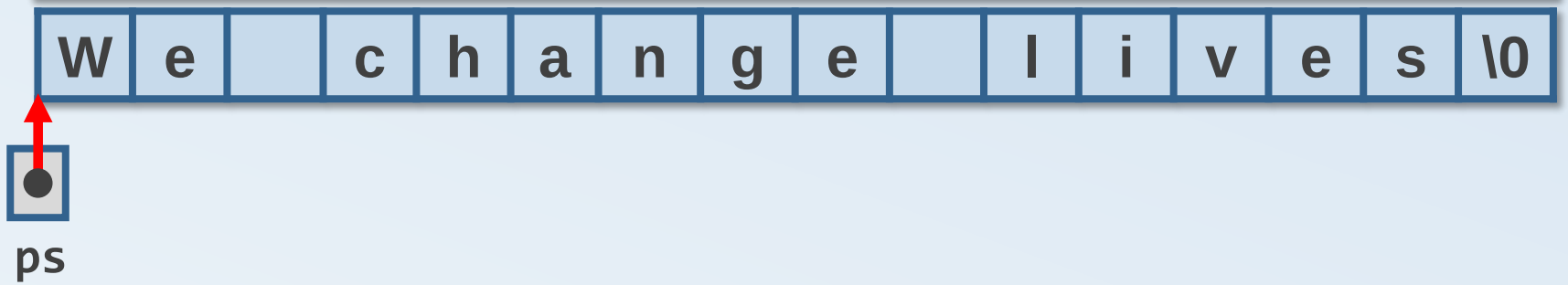
指针与字符串

```
char * ptr = "C Language";
```




```
char * ps="We change lives";
int n=10;
ps=ps+n;
printf("%s\n",ps);
```

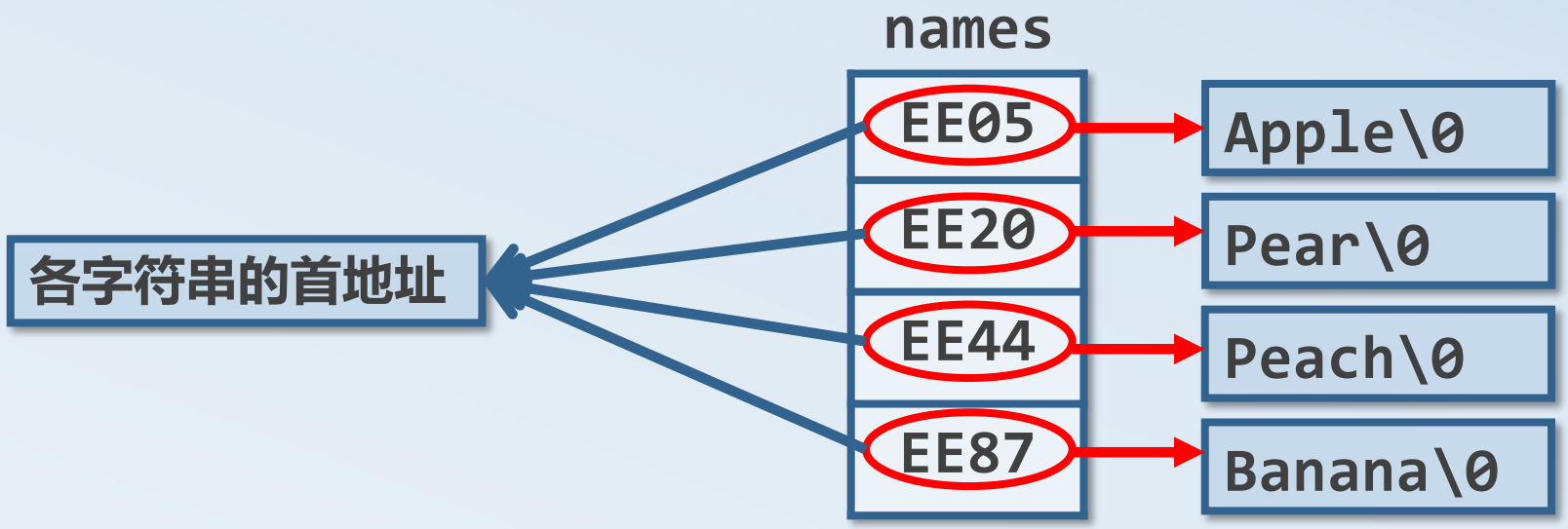
lives



指针与字符串

- 字符指针数组：一个数组中的各个元素都是字符指针

```
char *names [] = {"Apple", "Pear", "Peach", "Banana"};
```



指针与字符串

```
int main(void)
{
    char * names[6]
      = {"Guanyu", "Zhangfei",
         "Zhaoyun", "Machao",
         "Huangzhong", "Liubei"};
    char * temp;
    printf("%s %s\n",names[2],names[3]);
    temp = names[2];
    names[2] = names[3];
    names[3] = temp;
    printf("%s %s\n",names[2],names[3]);
    printf("\n");
    return 0;
}
```


```
Zhaoyun Machao
Machao Zhaoyun
```

本章授课内容

几种特殊的指针变量

- const指针
- 空指针
- 通用指针

几种特殊的指针变量

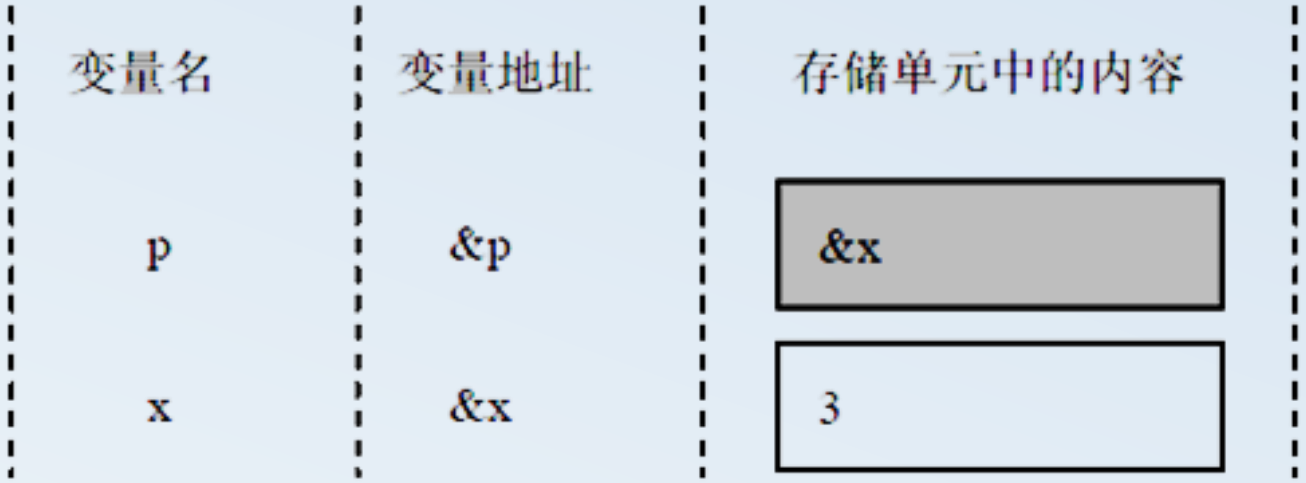
- 在指针声明符中，若使用了const关键字，该指针称为const指针。
  - 一重变量的const指针
  - 二重变量的const指针
  - 多重变量的const指针
- 作用：
  - 限定变量的可使用权限

几种特殊的指针变量

- const紧跟一重指针变量，表明该指针变量的值初始化后不可改变。
- 例 设有int x = 3; int y = 4; 请分析下面语句。

  int * const q; ❌

  int * const p = &x; p = &y; ❌

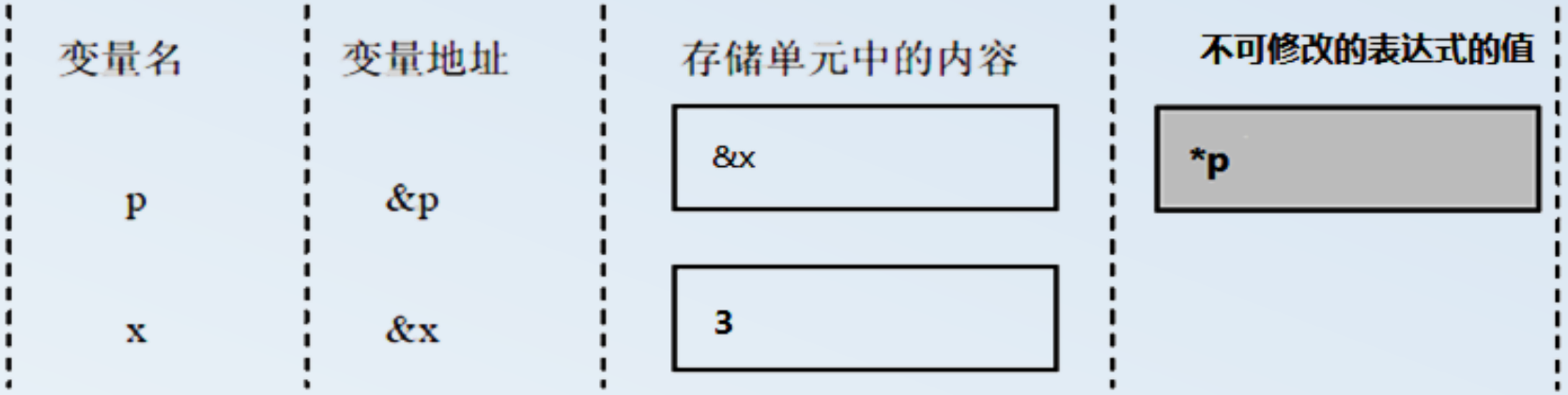


几种特殊的指针变量

- const后面为 * **一重指针变量**，表明限制指针p的权限，技巧是紧跟const后面的式子不可变。
- 例 设有int x = 3; int y = 4; 请分析下面语句。

```
const int * p = &x;
p = &y;
*p = 10;
```

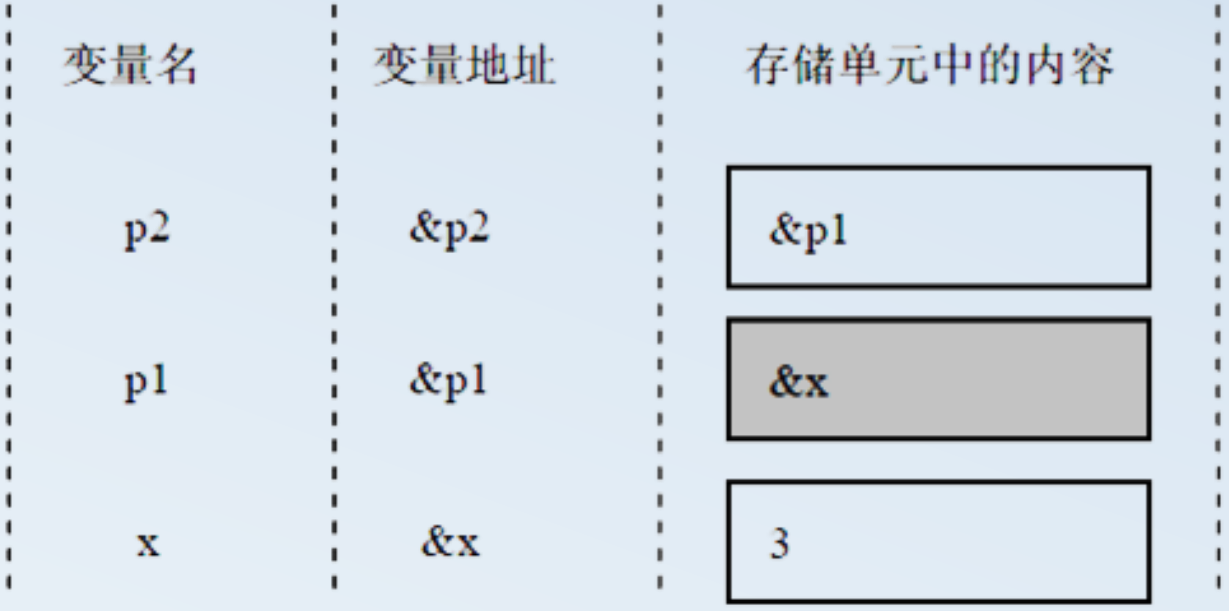
❌



几种特殊的指针变量

❖例 请画出内存存储示意图，并分析const 对p 的作用。

```
int x = 3;
int * p1 = &x;
int * const * p2 = &p1;
```



几种特殊的指针变量

注意：

1. const的位置不同意义不同
2. 紧跟const后面的部分代表的值不可变

几种特殊的指针变量

- **空指针**字面值的形式：
  - **0**
  - **NULL**
- 一个值为空指针字面值的指针变量被称为空指针
- 例：

```
int * p = 0;
double * p = NULL;
```

空指针注意：

若声明一个指针变量时，如果不指定它所指向的变量，则将空指针字面值赋给该指针.

几种特殊的指针变量

```
#include <stdio.h>
int main(void)
{
    int * p = NULL;
    if(p) // if(NULL != p)
    {
        printf("%d\n", *p);
    }
    return 0;
}
```

空指针注意：

建议使用if(p)的形式

建议使用if(NULL != p)的形式

几种特殊的指针变量

- 声明指针变量时，如果声明说明符为 **void**，则该指针为**通用指针**。
- 注意：
  - 对通用指针，不能使用*或下标运算符，也不能作为加减运算的操作数。
  - 任何指针都可以转换为void *类型，然后再转换为原类型，值保持不变。
- 例 设有int x = 3; int * p = &x; void * q;，请分别观察下面的语句。

```
q = p;
p = (int *)q;
```

Thank You !