《计算概论》课程 程序设计部分

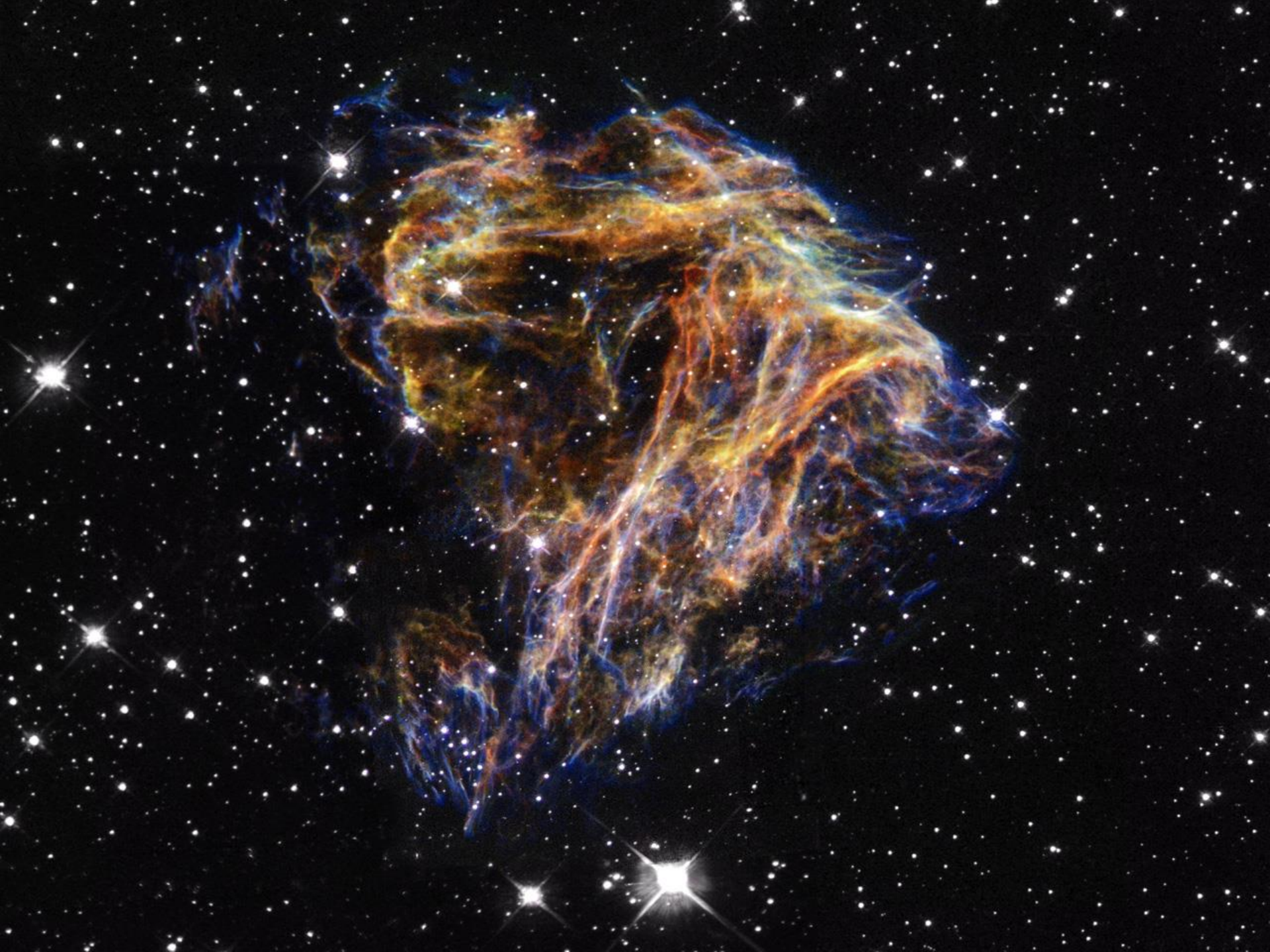
指针（1）

李　戈

北京大学 信息科学技术学院 软件研究所

lige@sei.pku.edu.cn

什么是“指针”？



互联网上的资源——地址

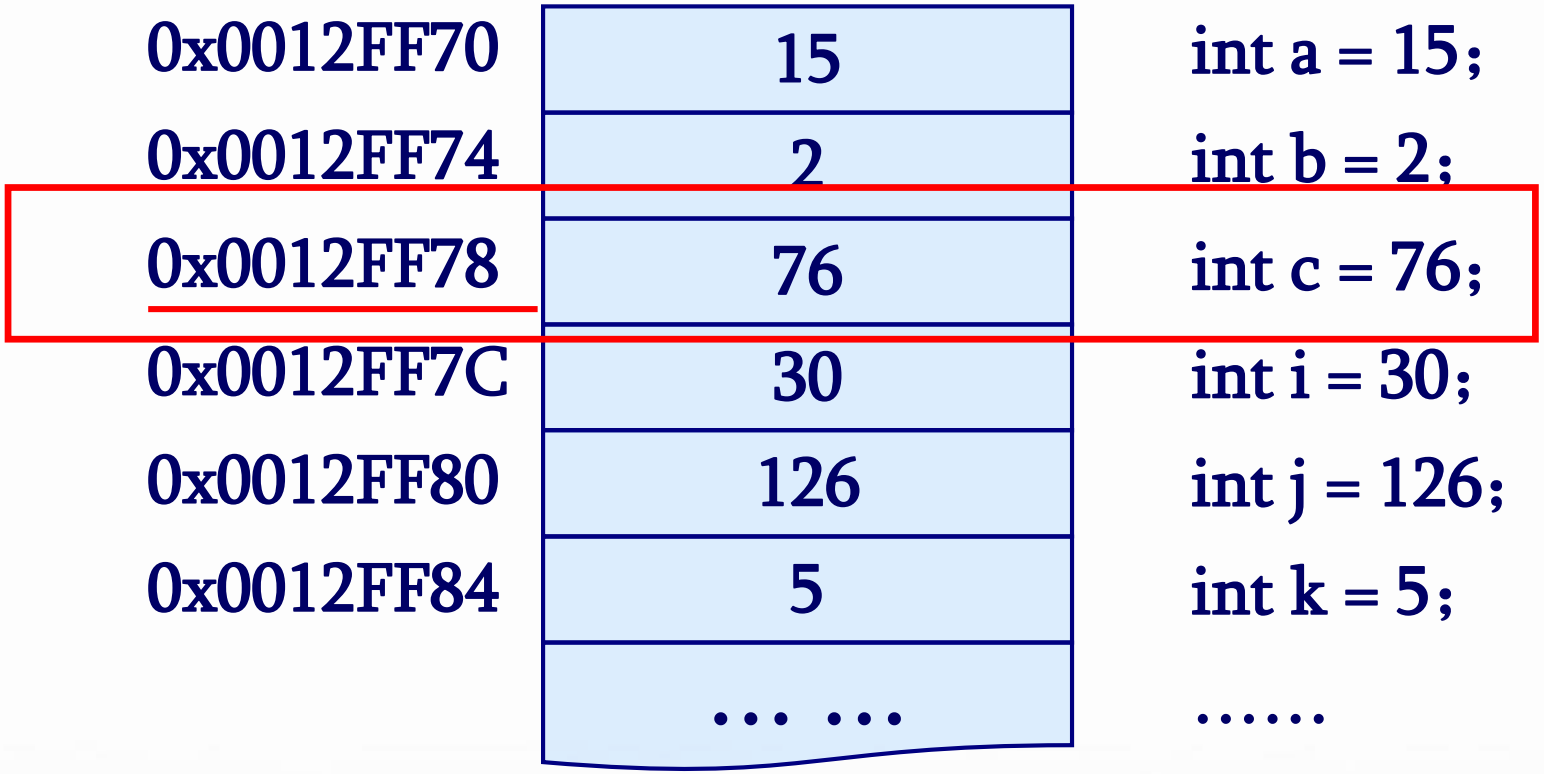
http://www.nasa.gov/images/content/166502.jpg

N49 Nebula

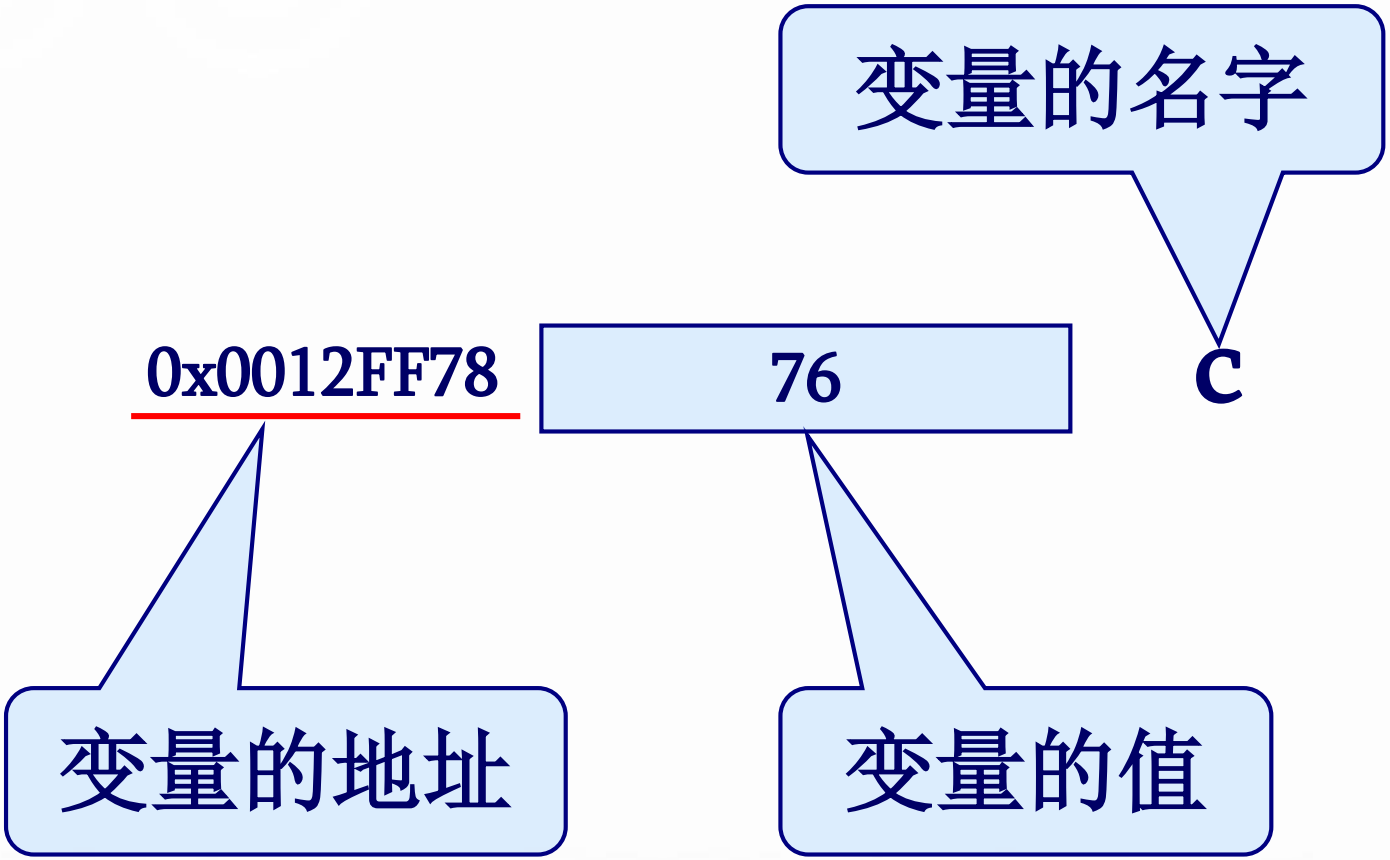
可以把“网址”称为 指向资源的“指针”

内存中的位置——地址

```
void main( )
{
    int a = 15;
    int b = 2;
    int c = 76;
    int i = 30;
    int j = 126;
    int k = 5;
    ……
}
```



变量的三要素



内存中的资源——地址

把某个变量的地址称为“指向该变量的指针”

0x0012FF74 | 76 | C

http://www.nasa.gov/images/content/166502.jpg

N49 Nebula

能不能拿到、看到一个变量的地址？

- 可以利用 **取地址运算符“&”** 实现

- cout<<&c<<endl;
  - 结果： 0041FE24 ；（VC++2013环境）
- cout<<sizeof(&c)<<endl;
  - 结果：4；（VC++2013环境）

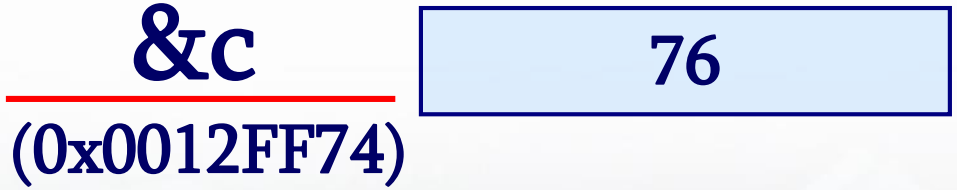
变量地址（指针）的作用

- 我们可以通过资源地址（指针）访问网络资源

http://www.nasa.gov/images/content/166502.jpg

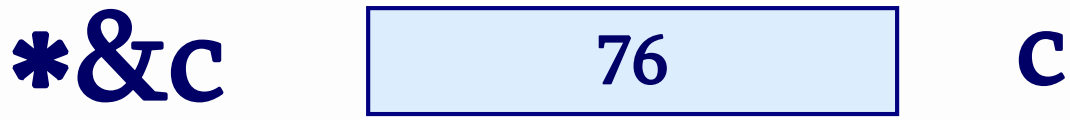
N49 Nebula

- 计算机通过变量的地址（指针）操作变量



通过变量的地址（指针）操作变量

- 可以利用 **指针运算符*** 实现



- cout<<*&c<<endl;
- cout<< c <<endl;

通过变量的地址（指针）操作变量

- 可以利用 **指针运算符*** 实现

***&c　　等价于　　c**

编译时，编译器建立 变量名 到 地址 的映射

- cout<< a <<endl;　等价于 cout<<*&a<<endl;
  - 找到变量a的地址；
  - 从地址 0x0012FF74 开始的四个字节中取出数据；
  - 将取出的数据送到显示器；

什么叫“指针变量”？

存放地址（指针）的变量

- 我们可以设置一个变量，来存放网络资源的地址

http://www.nasa.gov/images/content/166502.jpg

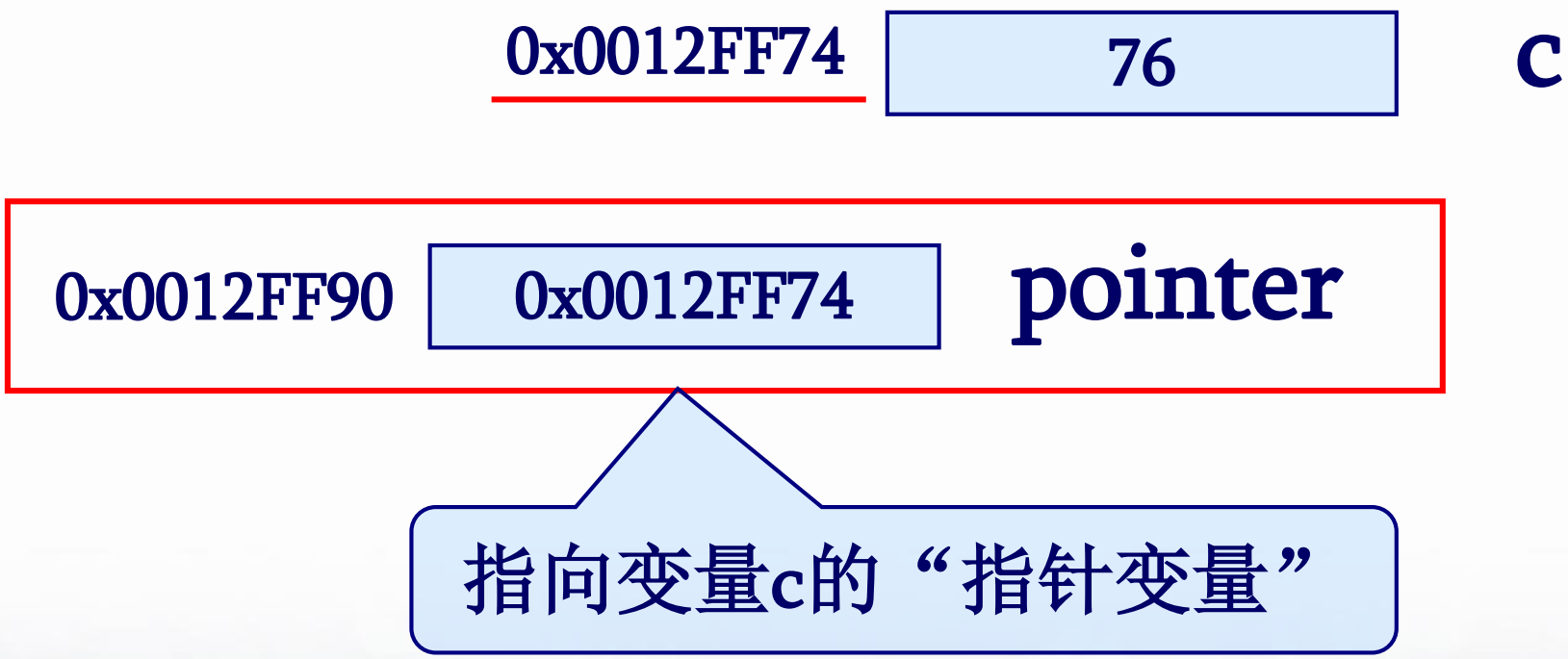
N49 Nebula

- 当然，我们也可以设置一个变量，来存放变量的地址（变量的指针）

0x0012FF74 | 76 | C

指针变量

- 指针变量
  - 专门用于存放指针（某个变量的地址）的变量



定义一个指针变量

int * pointer ;

指针变量的基类型

指针运算符 pointer的类型

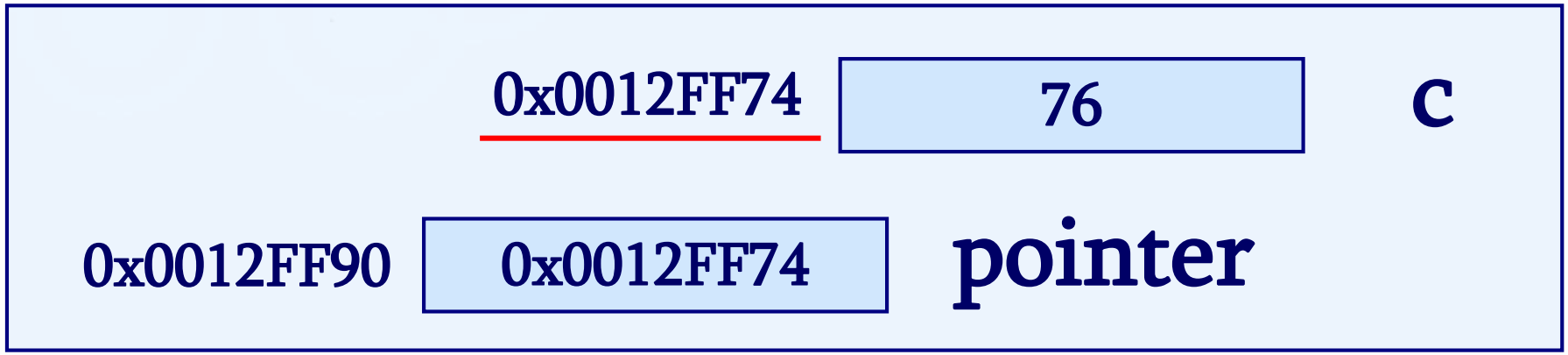
指针变量的名字

基类型：指针变量指向的变量的类型

0x0012FF74 | 76（int型） | c

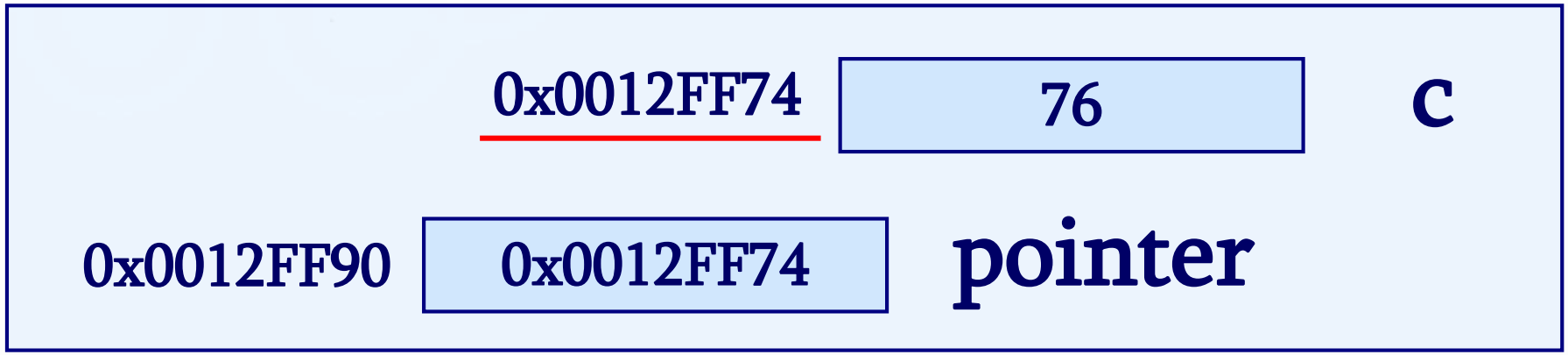
0x0012FF90 | 0x0012FF74 | pointer

指针变量的定义




```
int c = 76;
int *pointer;    //定义名字为pointer的指针变量;
pointer = &c;
        //将变量c的地址赋值给指针变量pointer；
        //赋值后，称指针变量pointer指向了变量c
```

指针变量的定义




```
int c = 76;
int *pointer;    //定义名字为pointer的指针变量;
pointer = &c;
```

能不能写成

```
pointer = c ;
```

绝对不行！

- 因为pointer是存放地址的变量，所以只能存放地址！

指针变量的使用

- 问题：
  - 既然指针变量中存放的是“某个变量的地址”；
  - 可否通过“指针变量”访问“它所指向的变量”呢？
- 例如：

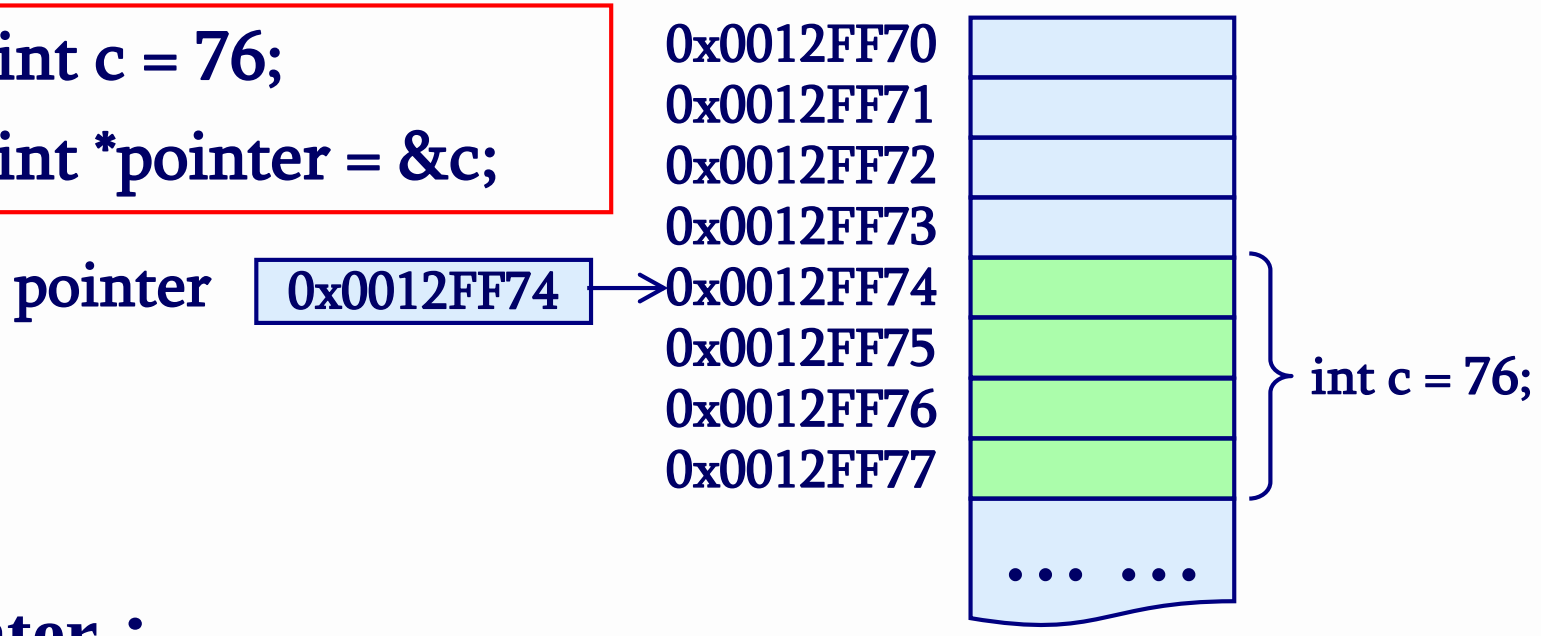
  - 可否利用pointer访问到变量c的值“76”呢？

指针变量的使用

- 也利用 指针运算符* 实现

若有：

```
int c = 76;
int *pointer = &c;
```



则*pointer：

- 为“pointer所指向的存储单元的内容”；
- “pointer所指向的存储单元的内容” 是 变量c

指针变量的地址

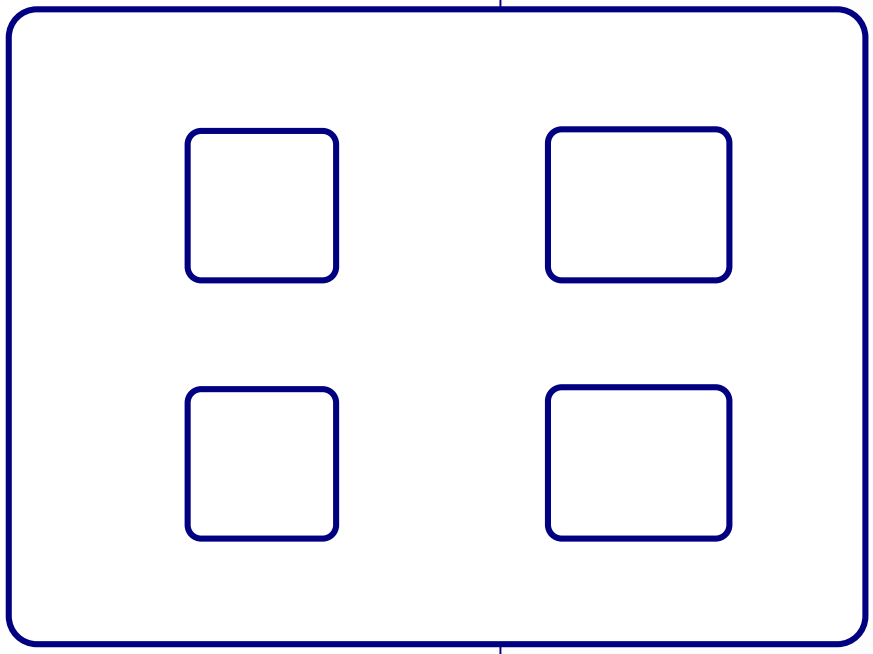
- 指针变量也是变量，是变量就有地址

```
#include<iostream>
using namespace std;
int main()
{
    int iCount = 18;
    int * iPtr = &iCount;
    *iPtr = 58;
    cout << iCount << endl;
    cout << iPtr << endl;
    cout << &iCount << endl;
    cout << *iPtr << endl;
    cout << &iPtr << endl;
    return 0;
}
```

```
58
0028FB10
0028FB10
58
0028FB04
```

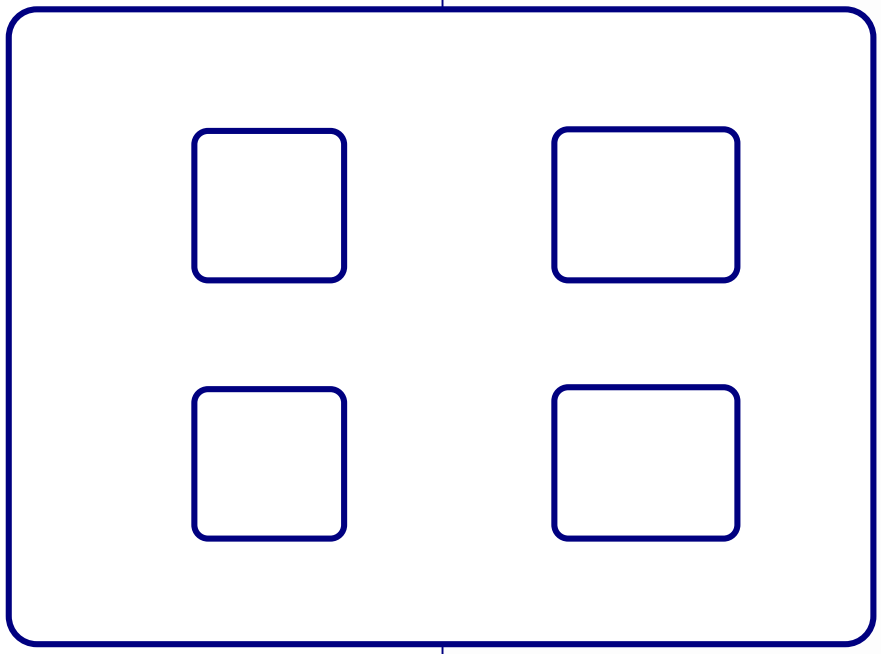
指针变量示例

```
#include<iostream>
using namespace std;
int main()
{
    int a = 0, b = 0, temp;
    int *p1 = NULL, *p2 = NULL;
    cin >> a >> b;
    p1 = &a;
    p2 = &b;
    if (*p1 < *p2)
    {
        temp = *p1; *p1 = *p2; *p2 = temp;
    }
    cout << "max = " << *p1 << ", min = " << *p2 << endl;
    return 0;
}
```

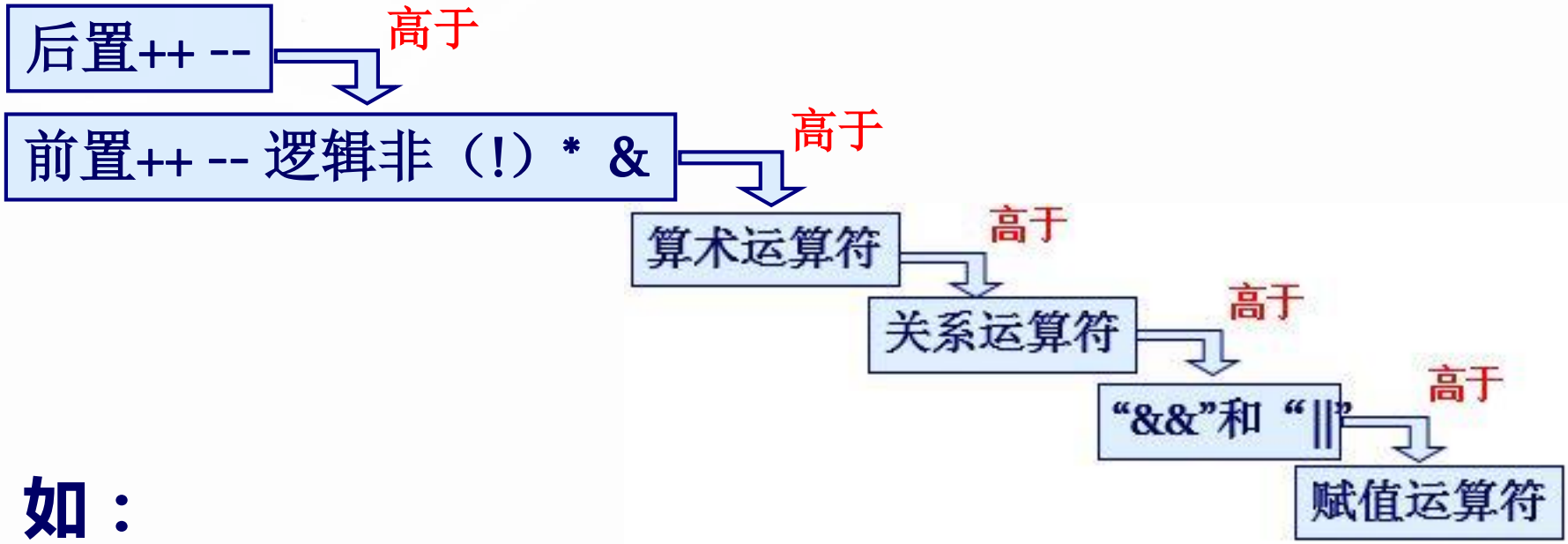


指针变量示例

```
#include<iostream>
using namespace std;
int main()
{
    int a = 0, b = 0;
    int *p1 = NULL, *p2 = NULL;
    int *temp = NULL;
    cin >> a >> b;
    p1 = &a;
    p2 = &b;
    if (*p1 < *p2)
    {
        temp = p1; p1 = p2; p2 = temp;
    }
    cout << "max = " << *p1 << ", min = " << *p2 << endl;
    return 0;
}
```

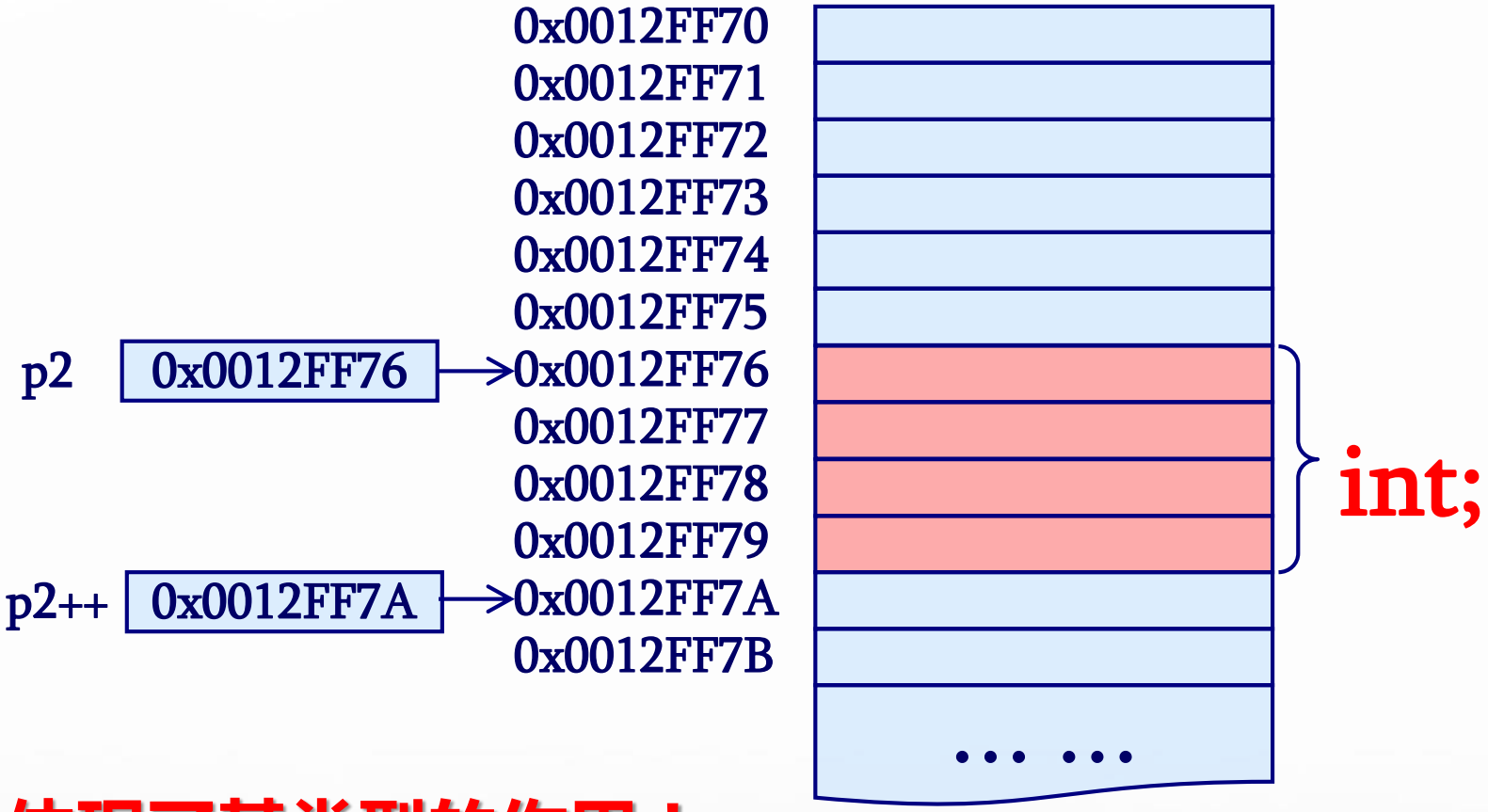


& 与 * 的运算优先级



- 如：
  - &*pointer = &(*pointer)
  - *&a = * (&a)
  - （ *pointer ）++ **不等于** *pointer++

pointer++ 的含义



体现了基类型的作用！

程序示例

```
#include<iostream>
using namespace std;
int main() {
    int n = 0;
    int * p = &n;
    cout << p << endl;
    p++;
    cout << p << endl;
    return 0;
}
```

```
00C6FED8
00C6FEDC
```

讨论：iPtr++的含义

- 假设：iPtr 当前所存地址是**0x00000100**
  - 若 iPtr 指向一个整型元素（占四个字节），
    则 **iPtr++** 等于 iPtr+1*4 = **0x00000104**
  - 若 iPtr 指向一个实型元素（占四个字节），
    则 **iPtr++** 等于 iPtr+1*4 = **0x00000104**
  - 若 iPtr 指向一个字符元素（占一个字节），
    则 **iPtr++** 等于 iPtr+1*1 = **0x00000101**

小结

- 讲过的概念
  - 什么是“**指针**”
  - 什么是“**指针变量**”
    - *pointer 的含义
    - pointer++的含义

数组与指针

指向数组元素的指针

```
#include<iostream>
using namespace std;
int main()
{
    int a[5] = { 1, 2, 3, 4, 5 };
    int *p = &a[3];
    cout << *p << endl;
    *p = 100;
    cout << a[3] << endl;
    return 0;
}
```

指针与数组

```
#include<iostream>
using namespace std;
int main() {
    int a[5] = {10, 11,12, 13, 14};
    int * p = NULL;
    cout << a << endl;
    p = a;
    cout << p << endl;
    return 0;
}
```

```
0109FAD8
0109FAD8
```

指针与数组

```
#include<iostream>
using namespace std;
int main()
{
    int a[5] = { 10, 11, 12, 13, 14 };
    cout << a << endl;
    cout << *a << endl;
    cout << &a[0] << endl;
    cout << a[0] << endl;
    return 0;
}
```

10	11	12	13	14
a[0]	a[1]	a[2]	a[3]	a[4]

```
0017F754
10
0017F754
10
```

数组的地址（数组的指针）

- 数组名代表数组首元素的地址

【数组名是指向数组第一个元素的指针】

- 对于数组 a[10]，数组名a代表数组a[10]中第一个元素a[0]的地址；
  - 即 a 与 &a[0] 等价
- 注意：
  - a是地址常量，不是变量，不能给a赋值。

指针与数组

```
#include<iostream>
using namespace std;
int main() {
    int a[5] = {10, 11,12, 13, 14};
    int * p = NULL;
    cout << a << endl;
    p = a;
    cout << p << endl;
    cout << *p << endl;
    cout << *p++ << endl;
    cout << *p++ << endl;
    return 0;
}
```

10	11	12	13	14
a[0]	a[1]	a[2]	a[3]	a[4]

```
00C5FCB4
00C5FCB4
10
10
11
```

利用指针变量引用数组元素

- 若定义
  - **数组 int a[10]；指针int *pointer;**
- 则：
  - **pointer = a； 等价于 pointer = &a[0];**
- 数组访问
  - **pointer + i ；等价于 a + i；等价于 &a[i];**
  - ***(pointer + i) 等价于 *(a + i) 等价于 a[i]**
- 表示形式
  - **pointer[i] 等价于 *(pointer + i)**

需要注意的问题

- **int *p = &a[0];**
  - a++是没有意义的，但p++会引起p变化。
  - p可以指向数组最后一个元素以后的元素。
- **指针做加减运算时一定注意有效的范围**

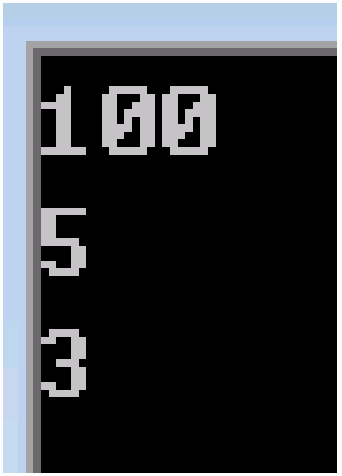
```
int a[5];
int *iPtr = &a[1];
iPtr --;          (指向&a[0])
*iPtr = 3;        (ok, a[0]=3)
iPtr --;          (指向&a[-1], dangerous)
*iPtr = 6;        (damage)
```

需要注意的问题

- 若定义 int a[5] = {1, 2, 3, 4, 5}; int *p;
  - 设当前：i=3, a[i]=4;
  - 则：t=*p--相当于t=a[i--]
- 特别注意：
  - *++p 相当于a[++i]，先将p自加，再作*运算。
  - *--p 相当于a[--i]，先使p自减，再作*运算。
  - *p++ 相当于a[i++]，先做*运算，再将p自加。
  - *p-- 相当于a[i--]，先做*运算，再将p自加。

程序举例

```
#include<iostream>
using namespace std;
int main()
{
    int a[5] = { 1, 2, 3, 4, 5 };
    int *p = &a[3];
    *p = 100;
    cout << *p++ << endl;
    cout << *p-- << endl;
    cout << *--p << endl;
    return 0;
}
```



使用指针代替数组下标

```
int main()
{
    int a[10],i;
    for (i=0;i<10;i++)
          cin >> a[i];
    for (i=9;i>=0;i--)
         cout<<setw(3)<<a[i];
    return 0;
}
```

```
int main( )
{
    int a[10], i, *p=a;
    for (i= 0;i<10;i++)
          cin >> *p++;
    for (p--;p>=a; )
          cout<<setw(3)<<*p--;
    return 0;
}
```

倒置数组元素

```
#include<iostream>
#include <iomanip>
using namespace std;
int main()
{
    int a[10], *p = NULL, *q = NULL, temp;
    for (p = a; p < a + 10; p++)
        cin >> *p;
    for (p = a, q = a + 9; p < q; p++, q--)
    {
        temp = *p; *p = *q; *q = temp;
    }
    for (p = a; p < a + 10; p++)
        cout << setw(3) << *p;
    return 0;
}
```

```
1 2 3 4 5 6 7 8 9 0
  0  9  8  7  6  5  4  3  2  1
```