《计算概论》课程 程序设计部分

指针（1）

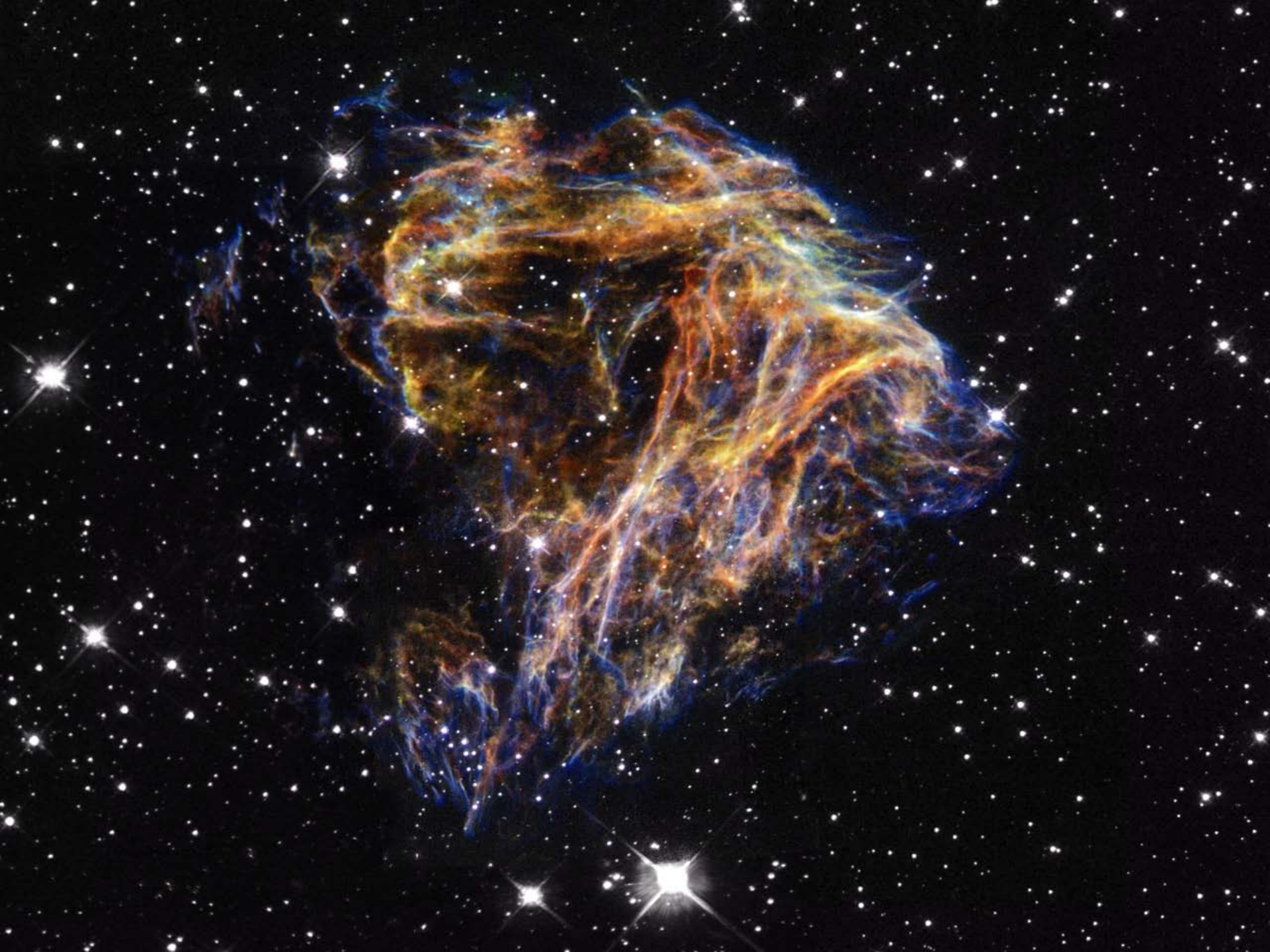
李　戈

北京大学 信息科学技术学院 软件研究所

2010年12月10日

北京大学

什么是“指针”？



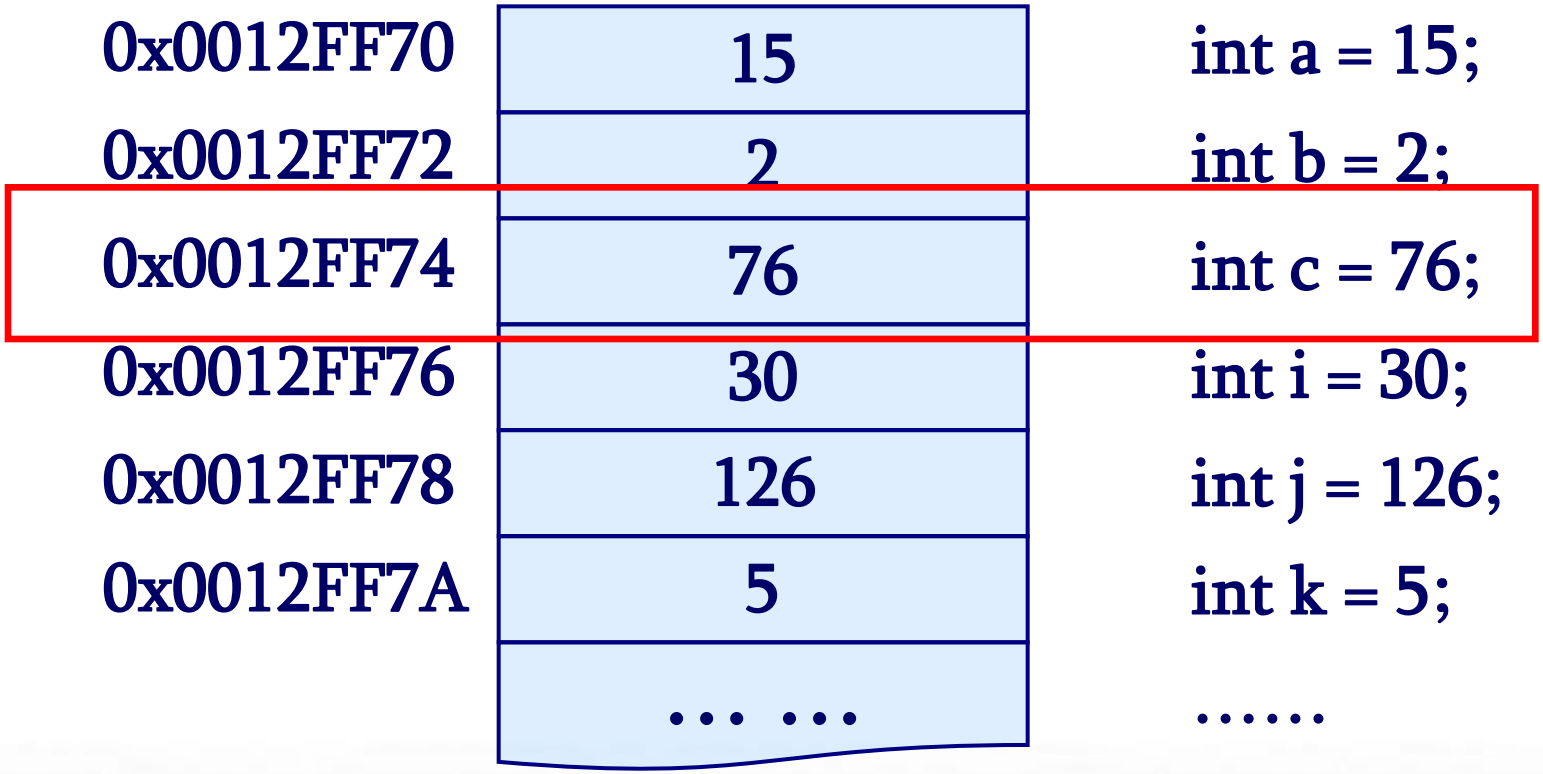
互联网上的资源——地址

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

N49 Nebula

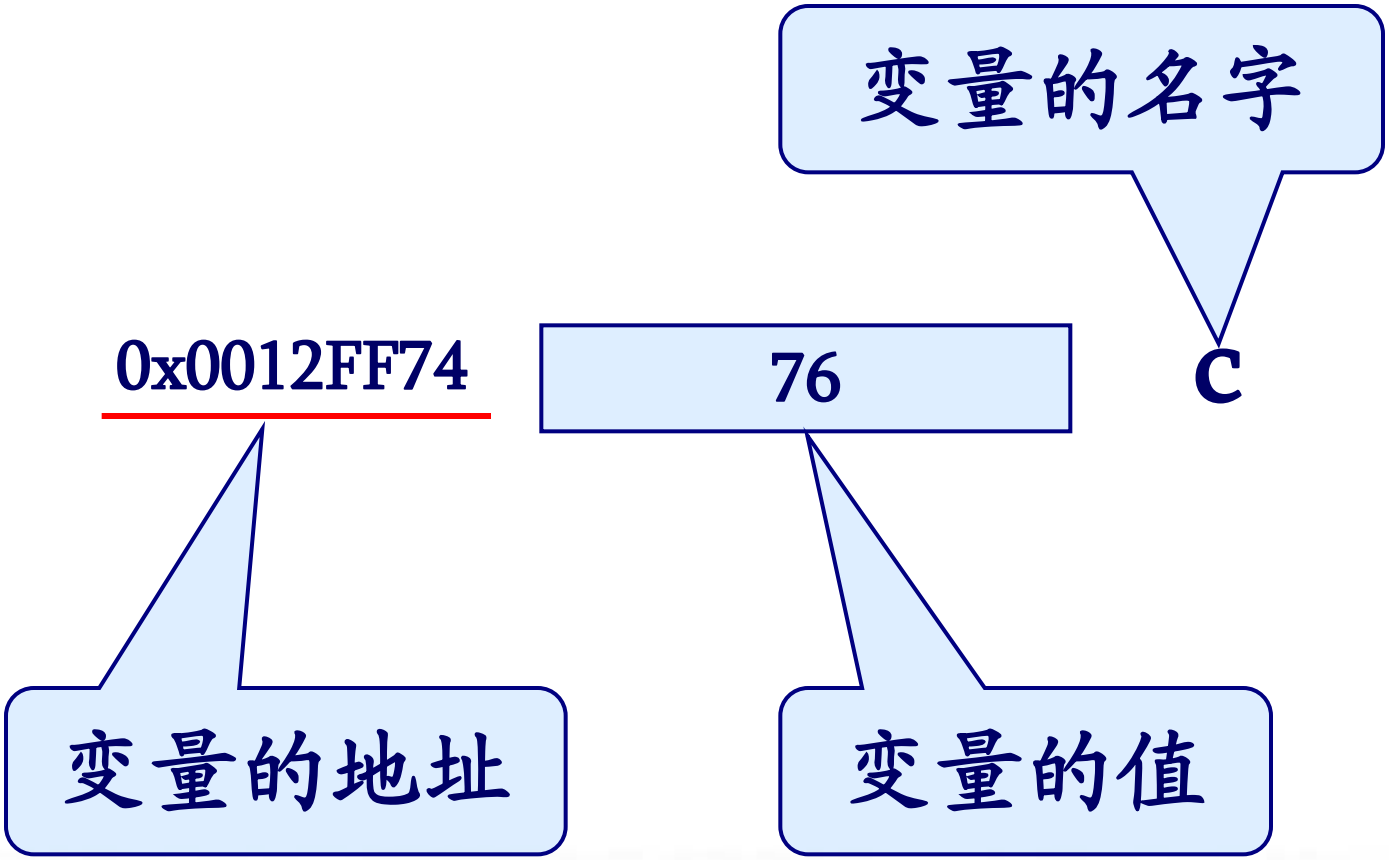
可以把“网址”称为 指向资源的“指针”

内存中的资源——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”

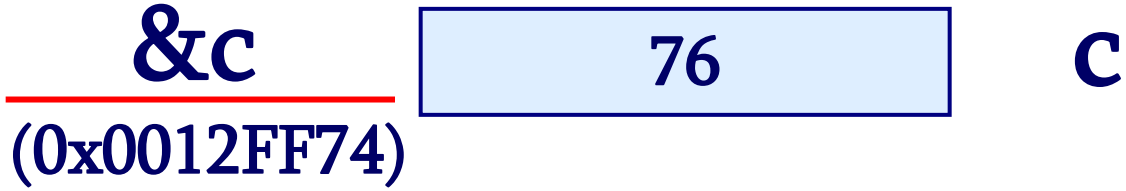
0x0012FF74 | 76 | C

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： 12FF74; （VC++6.0环境）
- cout<<sizeof(&c)<<endl;
  - 结果： 4; （VC++6.0环境）

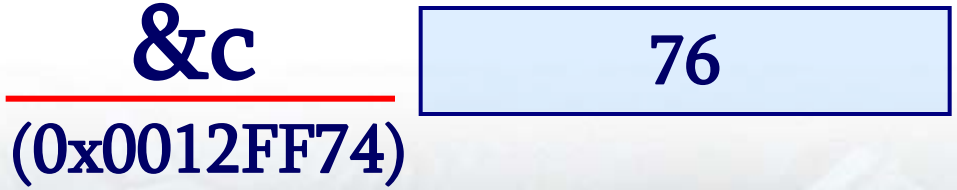
变量地址（指针）的作用

- 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

*&c　　76　　c

- cout<< a <<endl;
- cout<<*&a<<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”？

北京大学

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

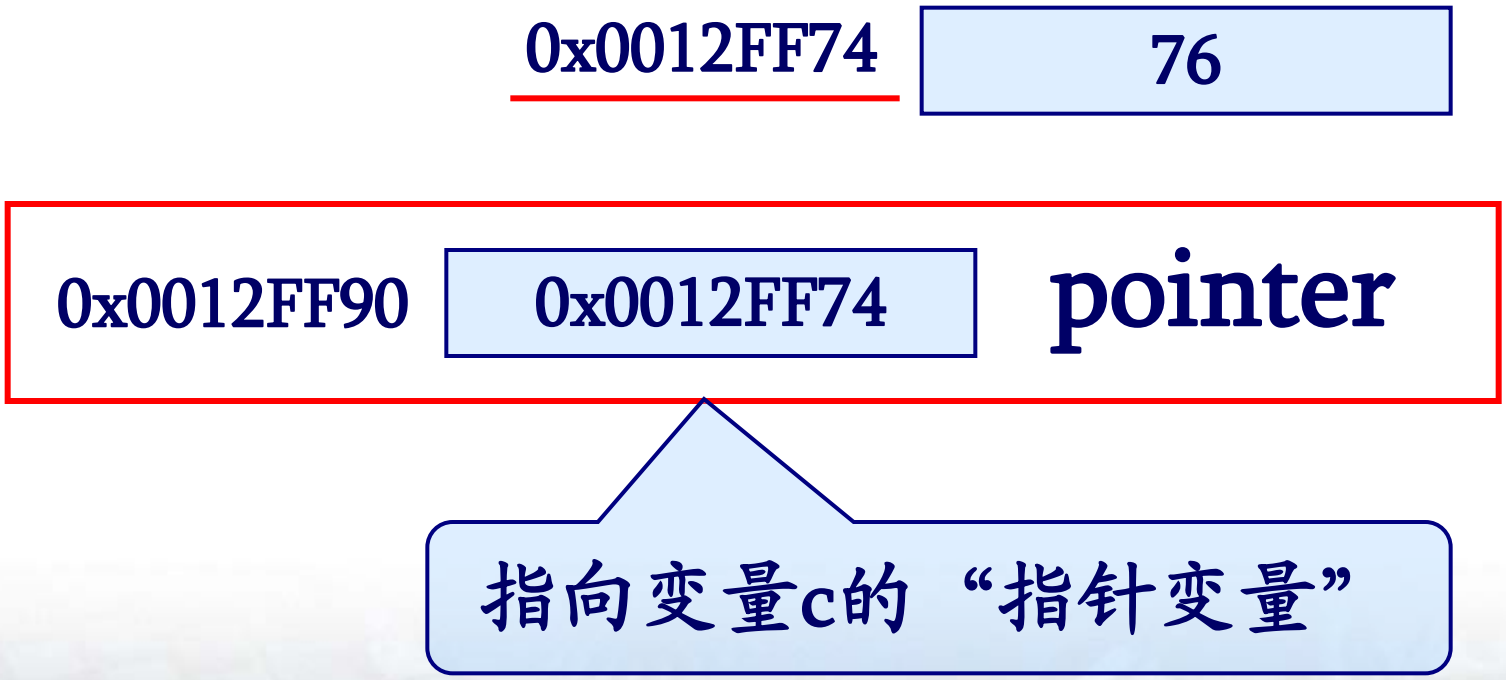
N49 Nebula

- 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）

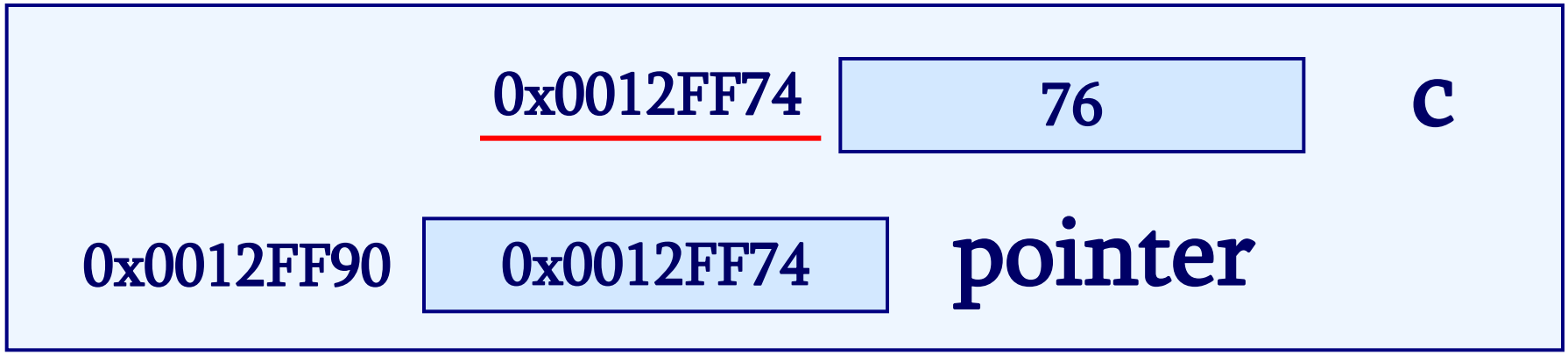
0x0012FF74 | 76 | C

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



指针变量的定义




```
int c = 76;          //定义int型变量c，并赋值76;
int *pointer;        //定义名字为pointer的指针变量;
        //“*”表示变量pointer的类型为指针类型;
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“基类型”

int * pointer ;

- 指针变量的基类型
- 指针运算符
- 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“基类型”

- 问题：
  - 指针变量是用来存放“变量的地址”的；
  - 既然“变量的地址”的格式都一样（VC6中4字节）；
  - 为什么还要指定指针变量的“基类型”？

```
int c = 76;              //定义int型变量c，并赋值76;
int *pointer;            //定义名字为pointer的指针变量;
pointer = &c;            //为指针变量赋值;
```

回顾：指针变量的定义、赋值

- 定义一个指向int型变量c的指针变量
  - int *pointer ;
  - pointer = &c ;

  定义时也可以进行初始化，写成:
  - int *pointer = &c ;
- 能不能写成:
  - int *pointer ;
  - 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

指针变量也有自己的地址吗?

- 指针也是变量，是变量就有地址

```
int main()
{  int iCount= 18;
   int * iPtr = &iCount;
   *iPtr = 58;
   cout<<iCount<<endl;          58
   cout<<iPtr<<endl;            0x0067fe00
   cout<<&iCount<<endl;         0x0067fe00
   cout<<*iPtr<<endl;           58
   cout<<&iPtr<<endl;           0x0067fdfc
   return 0;
}
```

讨论：&*pointer的含义

- 设:
  - 定义整型变量 int a = 3;
  - 定义一个指向变量a的指针变量pointer
    - int *pointer = &a;
- &*pointer的含义
  - *pointer等价于整型变量a
  - &*pointer等价于&a;
  - （*pointer）++ 等价于 a++;

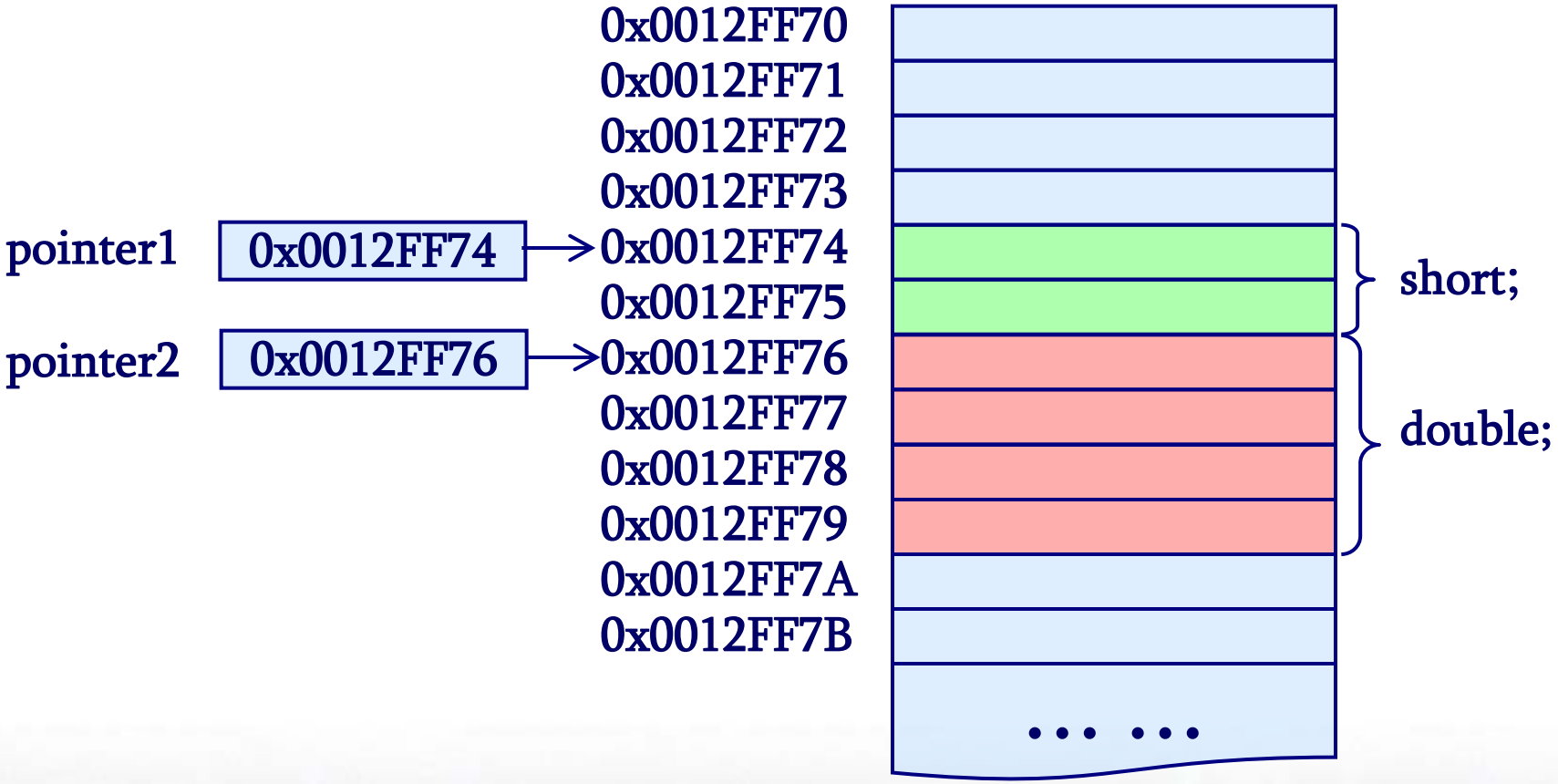
& 与 * 的运算优先级

- 与其他运算符相比
  - 高于算术运算符
- 几个同级的运算符
  - *, &, ++, --
  - 按照自右而左的结合方向
- 如:
  - &*pointer = &(*pointer)
  - *&a = * (&a)
  - （*pointer）++ 不等于 *pointer++

讨论：iPtr++的含义

- 假设iPtr所代表的地址是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

为何要指定指针变量的“基类型”？



指针使用举例

北京大学

指针的使用示例

```
void main( )
{
    int c;
    int *pointer;
    c = 76;
    pointer = &c;
    cout<<*pointer<<endl;
}
```

- cout<<*pointer<<endl;
  - 相当于 cout<<c<<endl;
  - 结果：76

指针的使用示例

```
void main( )
{
    int c;
    int *pointer;
    c = 76;
    pointer = &c;
    cout<<*pointer<<endl;
}
```

可否写成：*pointer = &c;

指针的使用示例

```
void main( )
{
    int c;
    int *pointer;
    c = 76;
    pointer = &c;
    cout<<*pointer<<endl;
}
```

可否写成：*pointer = 72;

指针变量的初始化

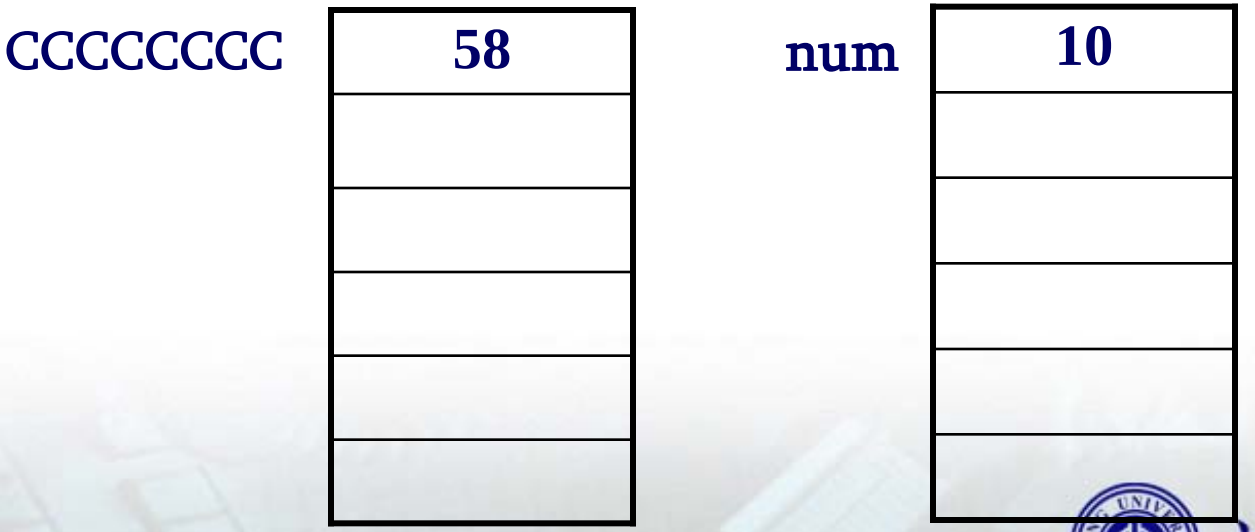
- 指针定义却不赋初值

```
int num;
int *iPtr;
num = 10;
*iPtr = 58;
```

(num = -858993460)

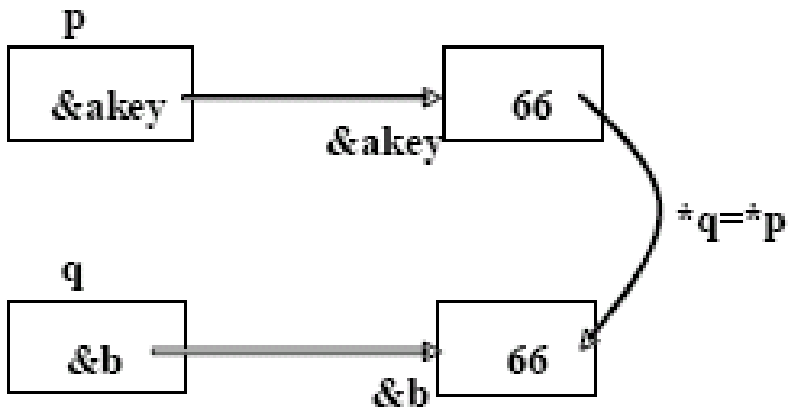
(iPtr = CCCCCCCC)

危险！地址CCCCCCCC可能不在用户区



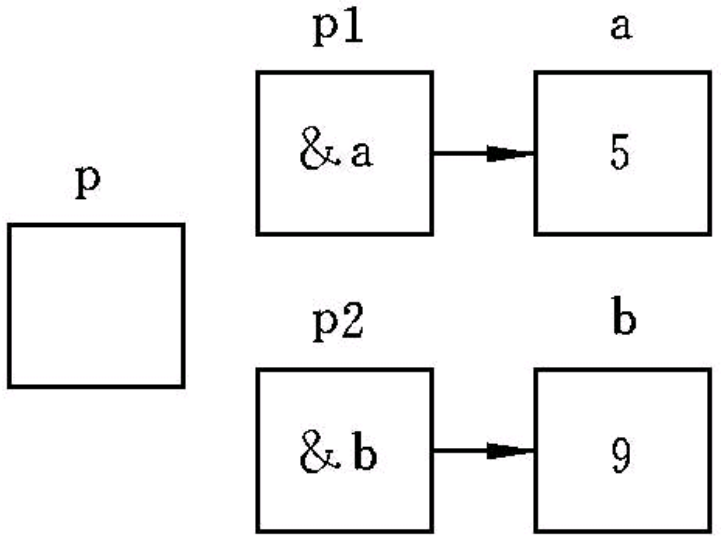
程序举例（1）

```
void main()
{
    int akey = 0, b = 0;
    int *p = NULL, *q = NULL;
    akey = 66;
    p = &akey;
    q = &b;
    *q = *p;
    cout<<“b = ”<<b<<endl;
    cout<<“*q = ”<<*q<<endl;
}
```



程序举例（2）

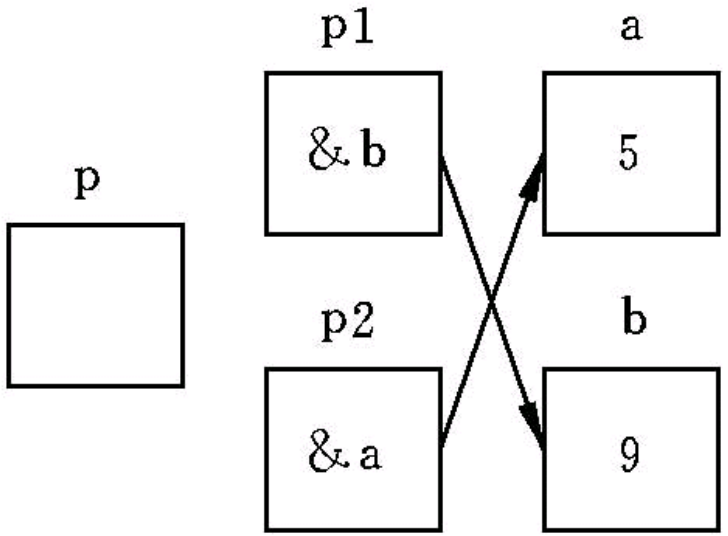
```
void main()
{
    int *p1, *p2, *p;
    int a, b;
    cin>>a>>b;
    p1 = &a;   p2 = &b;
    if (a < b)
          { p = p1; p1 = p2; p2 = p; }
    cout<<“a = ”<<a<<“, b = ”<<b<<endl;
    cout<<“max=”<<*p1<<“, min=”<<*p2<<endl;
}
```



北京大学

程序举例（2）

```
void main()
{
    int *p1, *p2, *p;
    int a, b;
    cin>>a>>b;
    p1 = &a;   p2 = &b;
    if (a < b)
          { p = p1; p1 = p2; p2 = p; }
    cout<<“a = ”<<a<<“, b = ”<<b<<endl;
    cout<<“max=”<<*p1<<“, min=”<<*p2<<endl;
}
```



数组与指针

北京大学

指向数组元素的指针

```
#include<iostream.h>
void main()
{
  int a[5]={1,2,3,4,5};
  int *p = &a[3];
  cout<<*p<<endl;
  *p = 100;
  cout<<a[3]<<endl;
}
```

指向数组元素的指针

```
#include<iostream.h>
void main()
{
  int a[5]={10, 11, 12, 13, 14};
  cout<<a<<endl;
  int *p = a;
  cout<<*p<<endl;
}
```

指向数组元素的指针

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

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。

数组的元素

- int a[10]
  - 定义了10个存放int型数据的连续空间;
- a + n
  - “a+n”代表数组a中第n+1个元素的地址。
    - 则a+1是数组a[10]的第2个元素a[1]的地址
- 指向数组元素的指针可以用做下标，
  - [ ]与*的作用相同
  - 如：p[i]与*(p+i)等价

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--]，先做* 运算
- 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

- 利用指针实现数组a的输入输出

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

程序举例

```
#include<iostream.h>
void main()
{
    int a[5]={1, 2, 3, 4, 5};
    int *p = &a[3];
    *p = 100;
    cout<<*p++<<endl;
    cout<<*p--<<endl;
    cout<<*--p<<endl;
}
```

练习—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;
}
```

练习—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;
}
```

好好想想，有没有问题?

谢　谢　！

北京大学