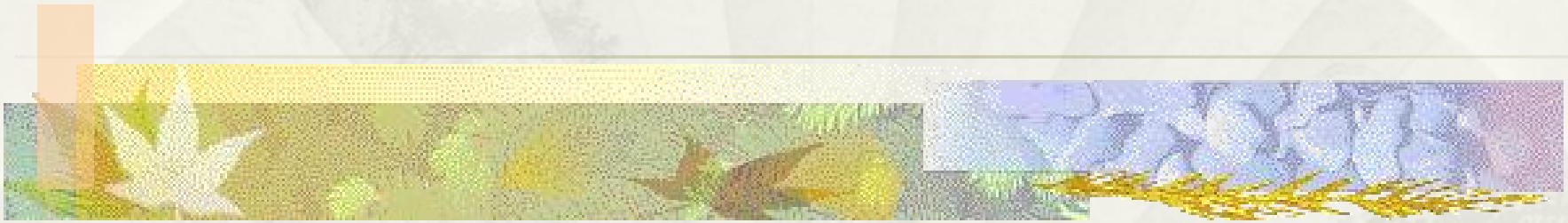
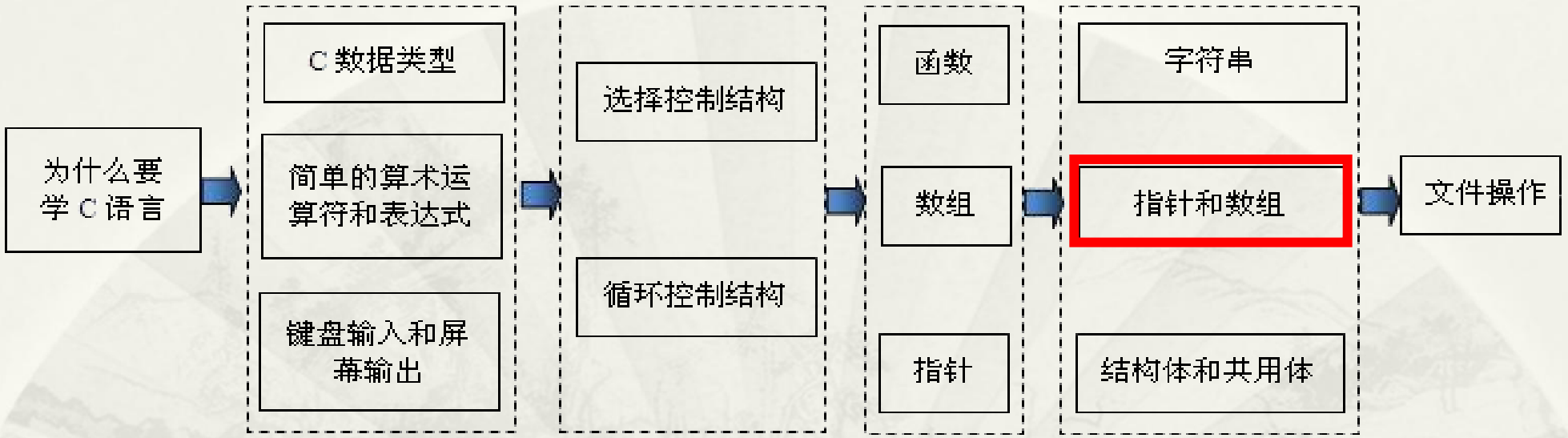
哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

规格严格 功夫到家

第11章 指针和数组



哈尔滨工业大学（深圳）
计算机科学与技术学院
刘洋
Liu.yang@hit.edu.cn



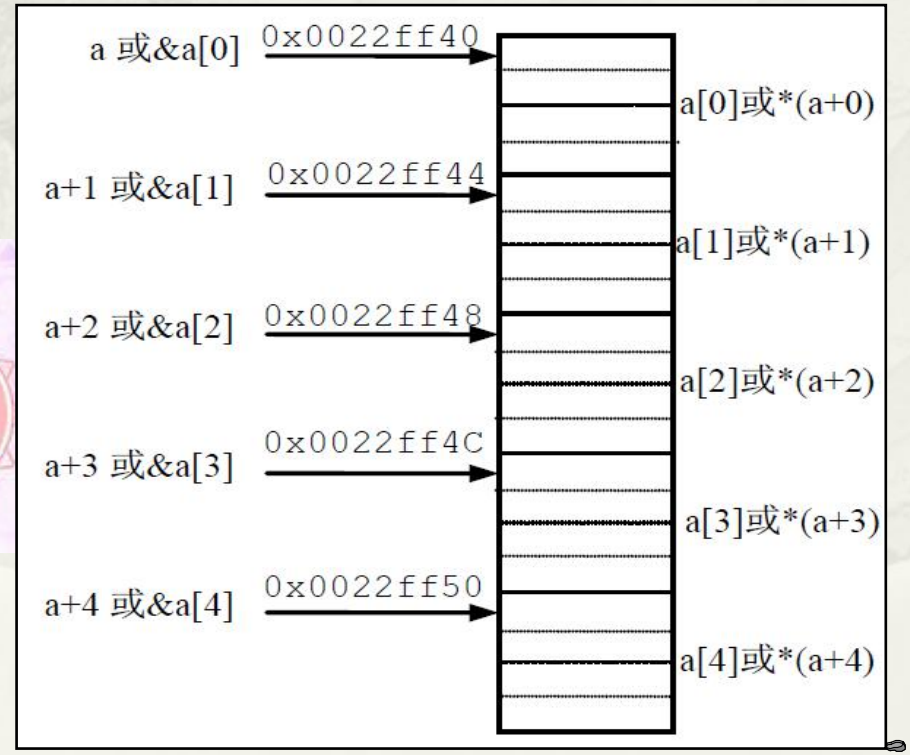
简单的数据结构 → 复杂的数据结构

第11章　学习内容

- 指针与一维数组间的关系
- 指针与二维数组间的关系
- 指针数组及其应用
- 动态数组，动态内存分配

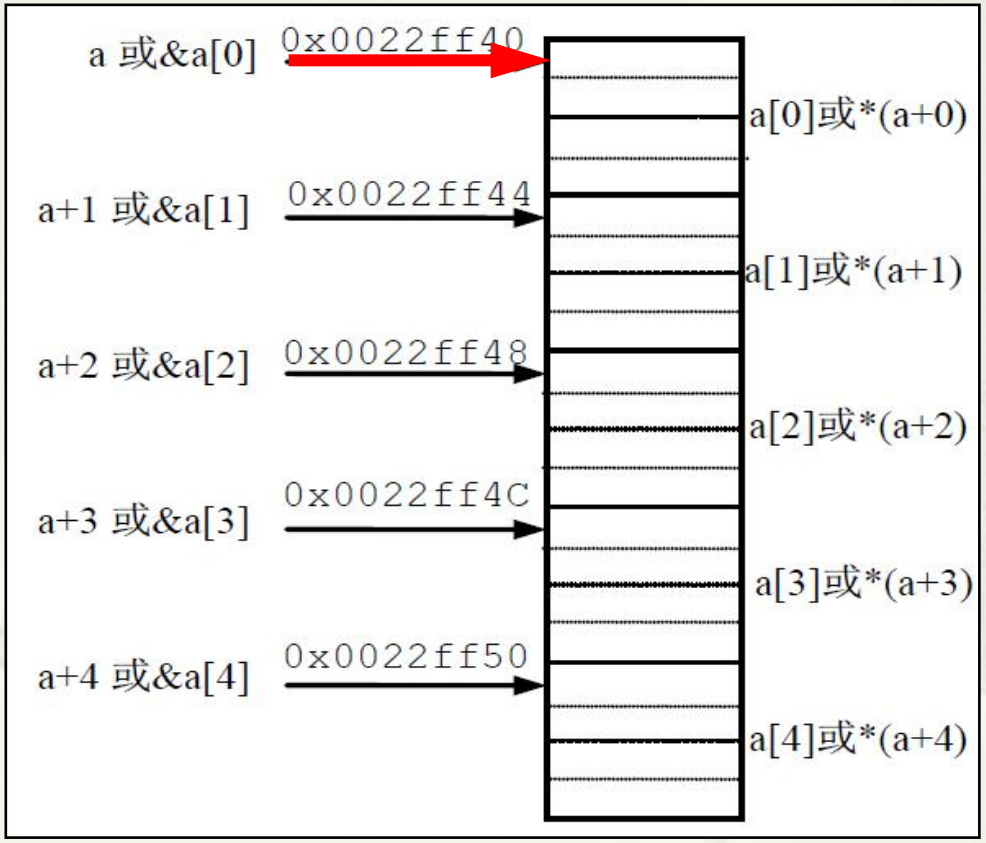
11.1 指针和一维数组间的关系

- 指针和数组的关系极为密切
  - 数组元素的等价引用形式：a[i] ↔ *(a+i)
  - 用下标形式访问数组元素，本质是计算该元素在内存中的地址



11.1 指针和一维数组间的关系

- 为什么一个int型指针能够指向一个整型数组呢？




```
int a[5];
int *p = a;
```

数组名是数组的首地址

```
int *p = &a[0];
```

&a[0]是整型元素的地址

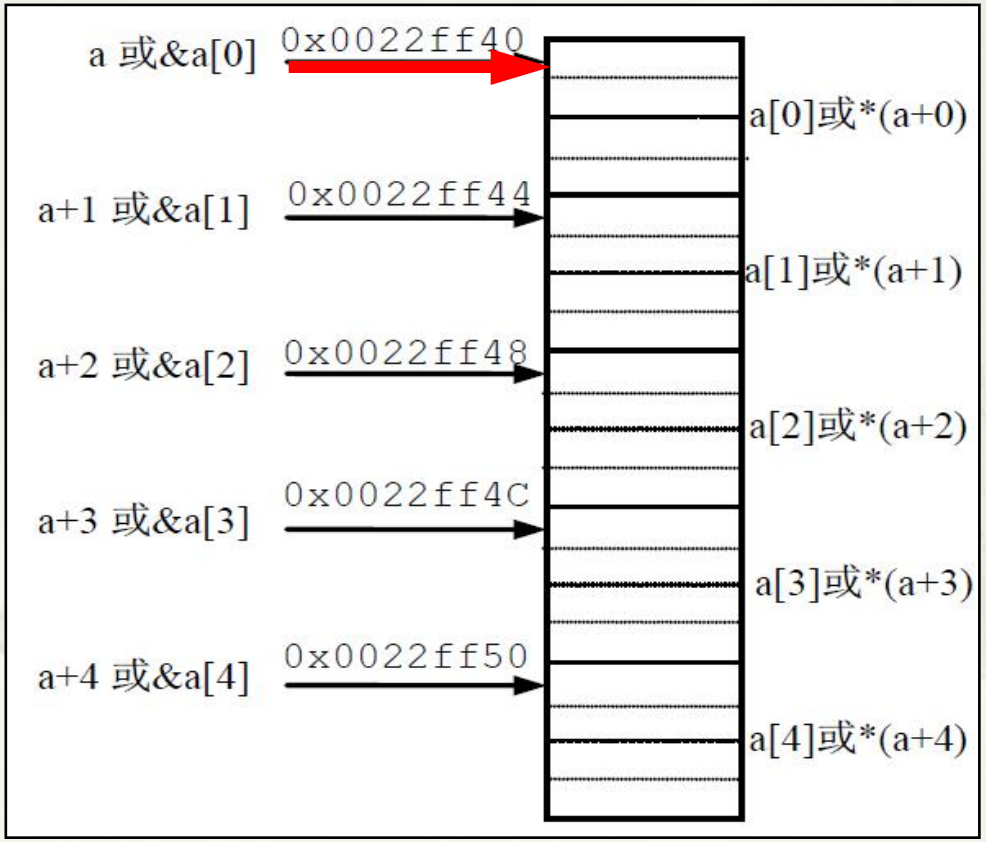
p是整型指针

a[0]的类型和p的基类型相同

数组元素的等价引用形式：a[i] ↔ *(a+i)

11.1 指针和一维数组间的关系

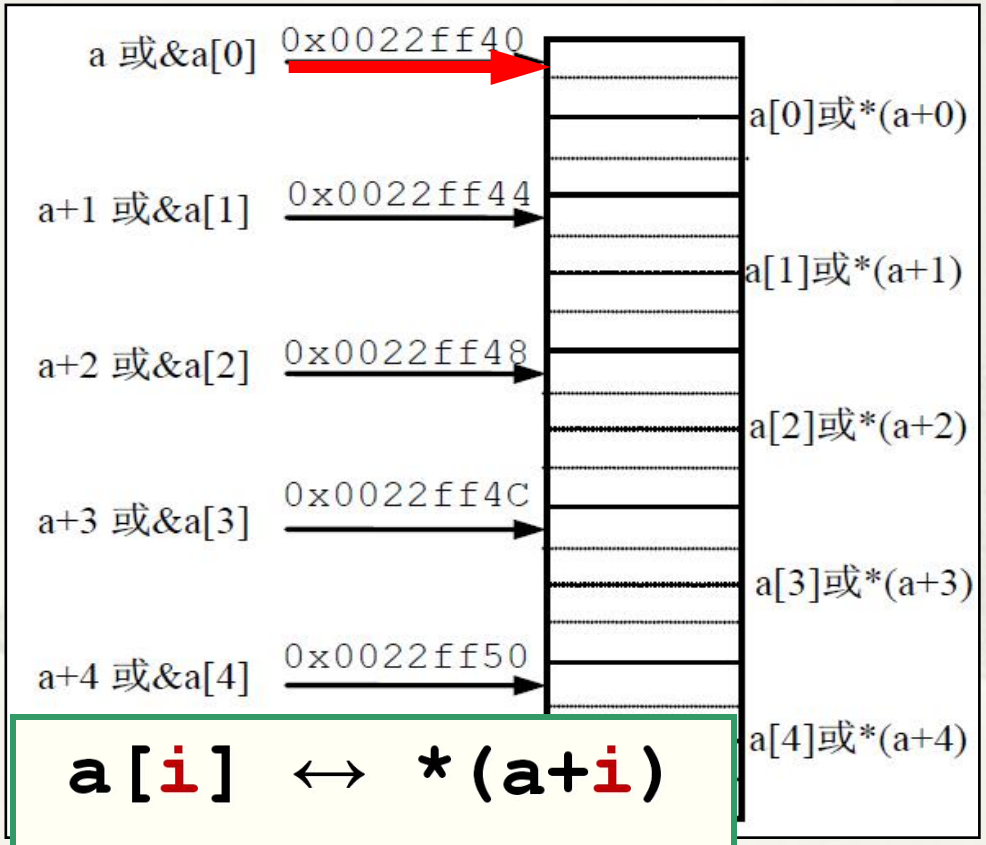
【例11.1】演示数组元素的引用方法




```
1    #include <stdio.h>
2    int  main()
3    {
4        int  a[5], i;
5        printf("Input five numbers:");
6        for (i=0; i<5; i++)
7        {
8            scanf("%d", &a[i]);
9        }
10       for (i=0; i<5; i++)
11       {
12           printf("%4d", a[i]);
13       }
14       printf("\n");
15       return 0;
16   }
```

11.1 指针和一维数组间的关系

【例11.1】演示数组元素的引用方法



a[i] ↔ *(a+i)

&a[i] ↔ (a+i)

```
1   #include <stdio.h>
2   int  main()
3   {
4       int  a[5], i;
5       printf("Input five numbers:");
6       for (i=0; i<5; i++)
7       {
8           scanf("%d", a+i);
9       }
10      for (i=0; i<5; i++)
11      {
12          printf("%4d", *(a+i));
13      }
14      printf("\n");
15      return 0;
16  }
```

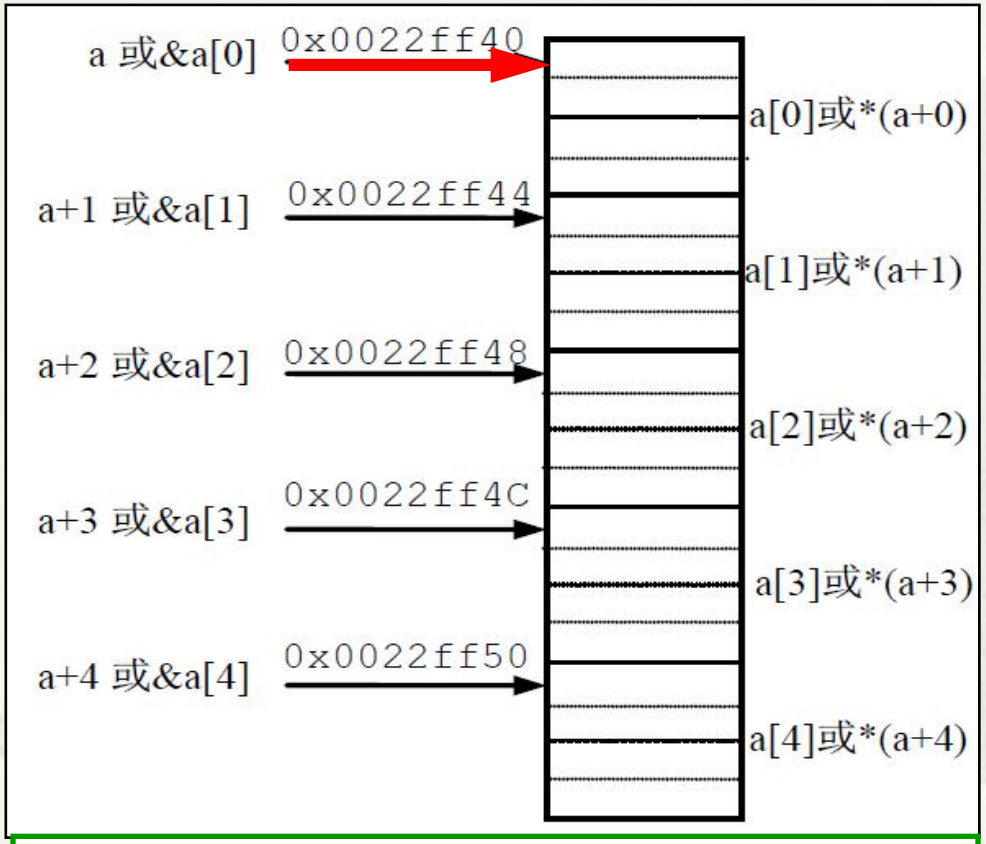
11.1 指针和一维数组间的关系

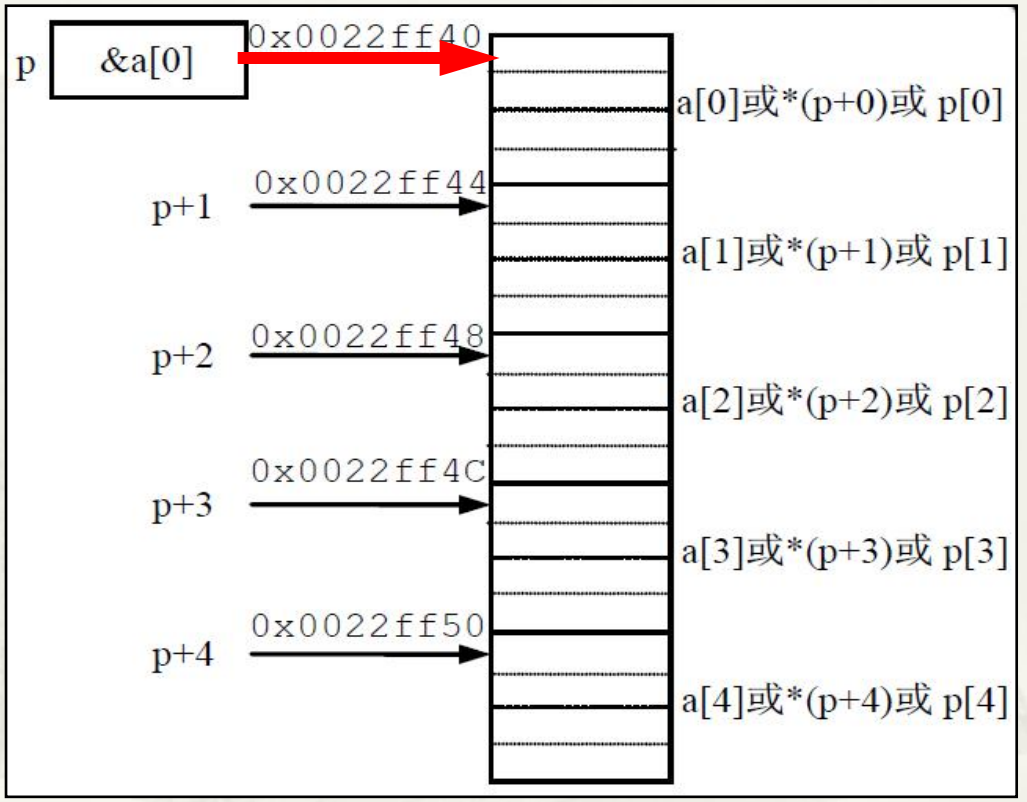
【例11.1】演示数组元素的引用方法


```
1   #include <stdio.h>
2   int  main()
3   {
4       int  a[5], *p;
5       printf("Input five numbers:");
6       for (p = a; p<a+5; p++)
7       {
8           scanf("%d", p);
9       }
10      for (p = a; p<a+5; p++)
11      {
12          printf("%4d", *p);
13      }
14      printf("\n");
        return 0;
```

p++不是增加1字节，取决于p的基类型

11.1 指针和一维数组间的关系





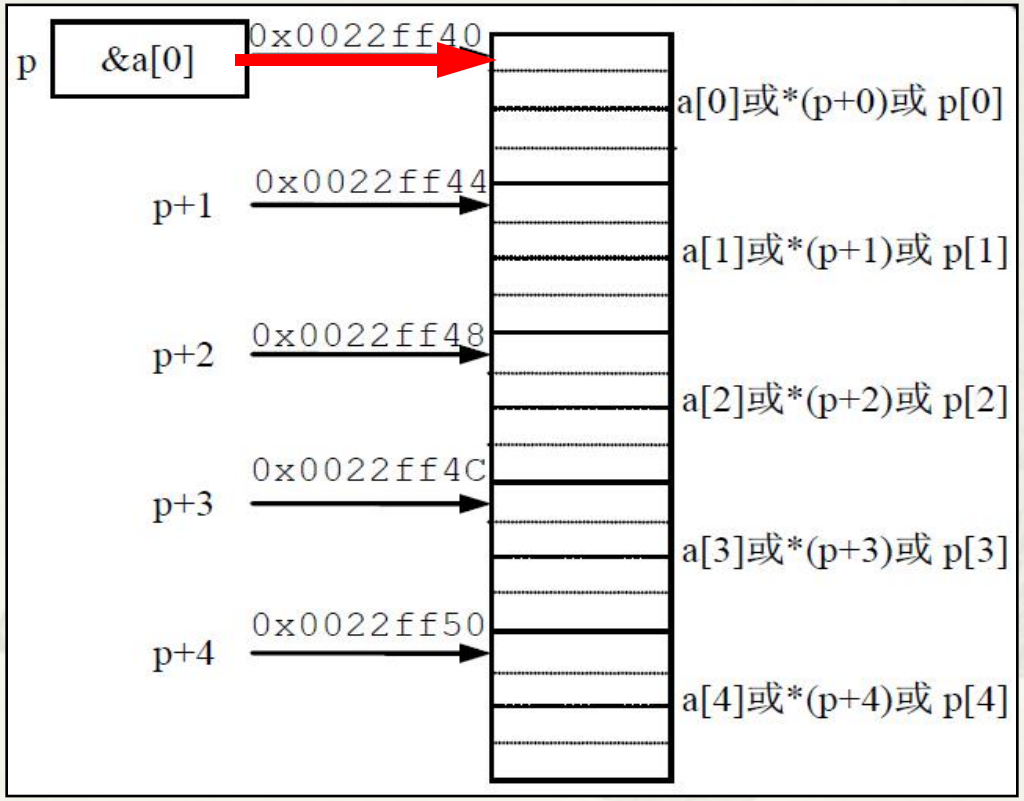
因p保存的是数组的首地址
所以p也可看成是数组名

另两种等价的引用形式：

p[i] ↔ *(p+i)

11.1 指针和一维数组间的关系

【例11.1】演示数组元素的引用方法



p[i] ↔ *(p+i)

```
1   #include <stdio.h>
2   int  main()
3   {
4       int  a[5], *p = NULL, i;
5       printf("Input five numbers:");
6       p = a;
7       for (i=0; i<5; i++)
8       {
9           scanf("%d", &p[i]);
10      }
11      p = a;
12      for (i=0; i<5; i++)
13      {
14          printf("%4d", p[i]);
15      }
16      printf("\n");
17      return 0;
18  }
```

11.1 指针和一维数组间的关系

【例11.2】演示数组和指针变量作函数参数

```
3    void InputArray(int a[], int n)
4    {
5       int  i;
6       for (i=0; i<n; i++)
7       {
8           scanf("%d", &a[i]);
9       }
10   }
```

被调函数的形参声明为数组类型，用下标法访问数组元素

```
11   void OutputArray(int a[], int n)
12   {
13      int  i;
14      for (i=0; i<n; i++)
15      {
16          printf("%4d", a[i]);
17      }
18      printf("\n");
19   }
```

11.1 指针和一维数组间的关系

【例11.2】演示数组和指针变量作函数参数

```
3    void InputArray(int *pa, int n)
4    {
5        int i;
6        for (i=0; i<n; i++, pa++)
7        {
8            scanf("%d", pa);
9        }
10   }
```

被调函数的形参声明为指针类型，用指针法访问数组元素

```
11   void OutputArray(int *pa, int n)
12   {
13       int i;
14       for (i=0; i<n; i++, pa++)
15       {
16           printf("%4d", *pa);
17       }
18       printf("\n");
19   }
```

11.1 指针和一维数组间的关系

【例11.2】演示数组和指针变量作函数参数

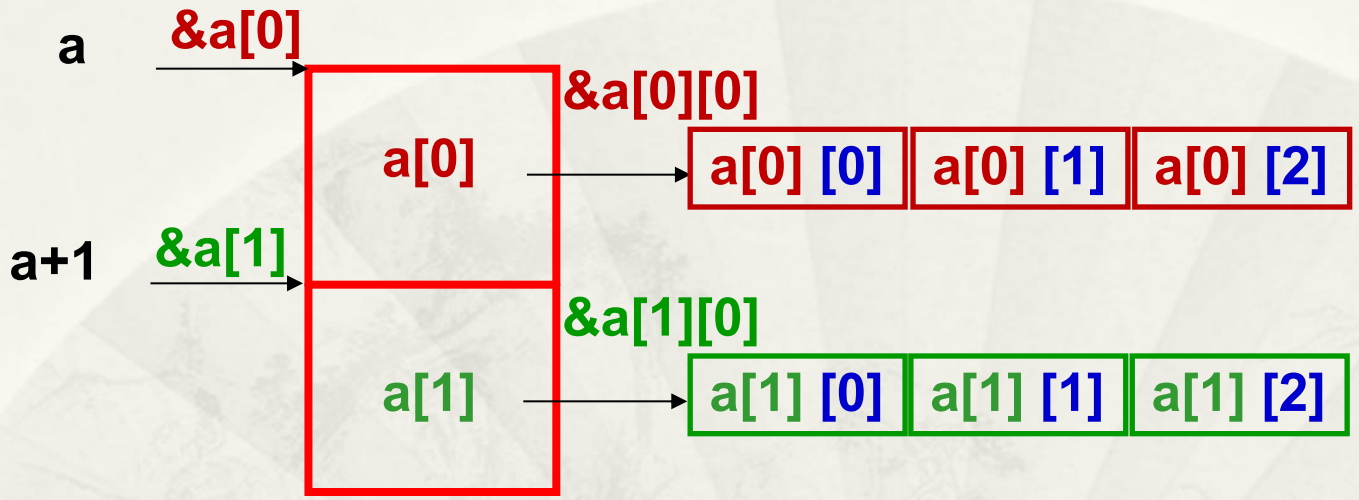
```
1    #include <stdio.h>
2    int  main()
3    {
4       int  a[5];
5       printf("Input five numbers:");
6       InputArray(a, 5);
7       OutputArray(a, 5);
8       return 0;
9    }
```

在主函数中这样做
没有实际意义

```
1    #include <stdio.h>
2    int  main()
3    {
4       int  a[5];
5       int *p = a;
6       printf("Input five numbers:");
7       InputArray(p, 5);
8       OutputArray(p, 5);
9       return 0;
10   }
```

11.2 指针和二维数组间的关系

- 将二维数组a看成一维数组，有2个“int[3]型”元素




```
int a[2][3];
```

a[i][j]

((a+i)+j)

a代表二维数组的首地址，第0行的地址，行地址

a+i 代表第i行的地址

但并非增加i个字节！

a[i] ↔ *(a+i)

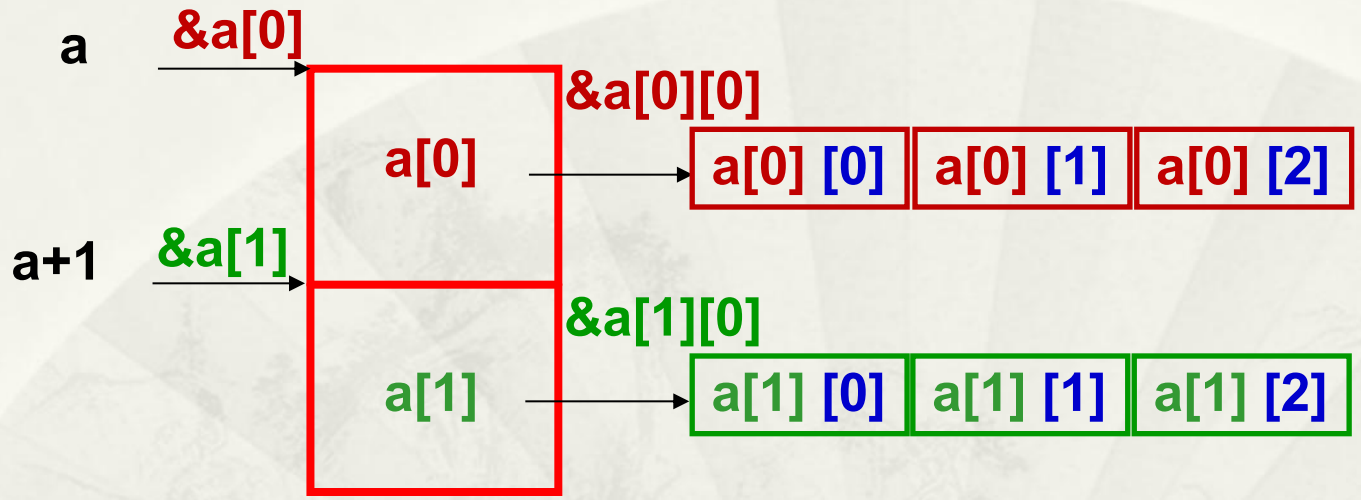
&a[i] ↔ (a+i)

1）a包含2个元素a[0],a[1]

2）a[0],a[1]又分别是一个一维数组，包含3个元素

11.2 指针和二维数组间的关系

- 将二维数组a看成一维数组，有2个“int[3]型”元素




```
int a[2][3];
```

```
a[i][j]
*(*(p+i)+j)
```

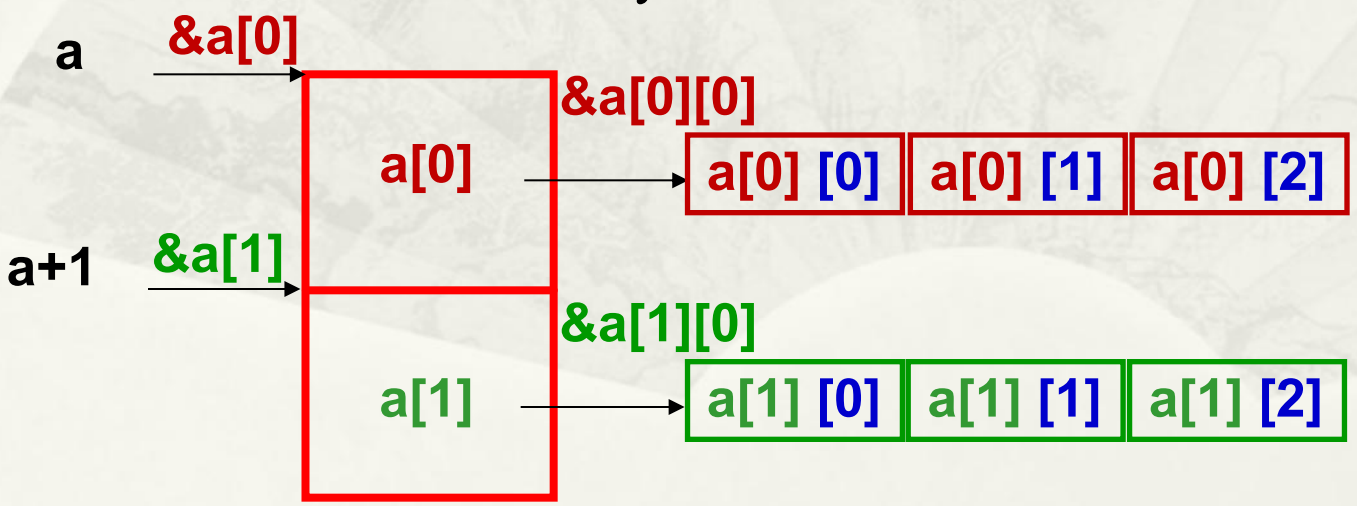
- 若要让一个指针指向它，则应定义为
  - `int (*p)[3];` //行指针，基类型是int[3]
  - `p = a;`
  - `p = &a[0];` //指向第0行的“int[3]型”元素

11.2 指针和二维数组间的关系

- 逐行查找→
- 逐列查找

```
int (*p)[3];
p = a;
```

```
for (i=0; i<m; i++)
{
   for (j=0; j<n; j++)
   {
      printf("%d", *(*(p+i)+j));
   }
}
```

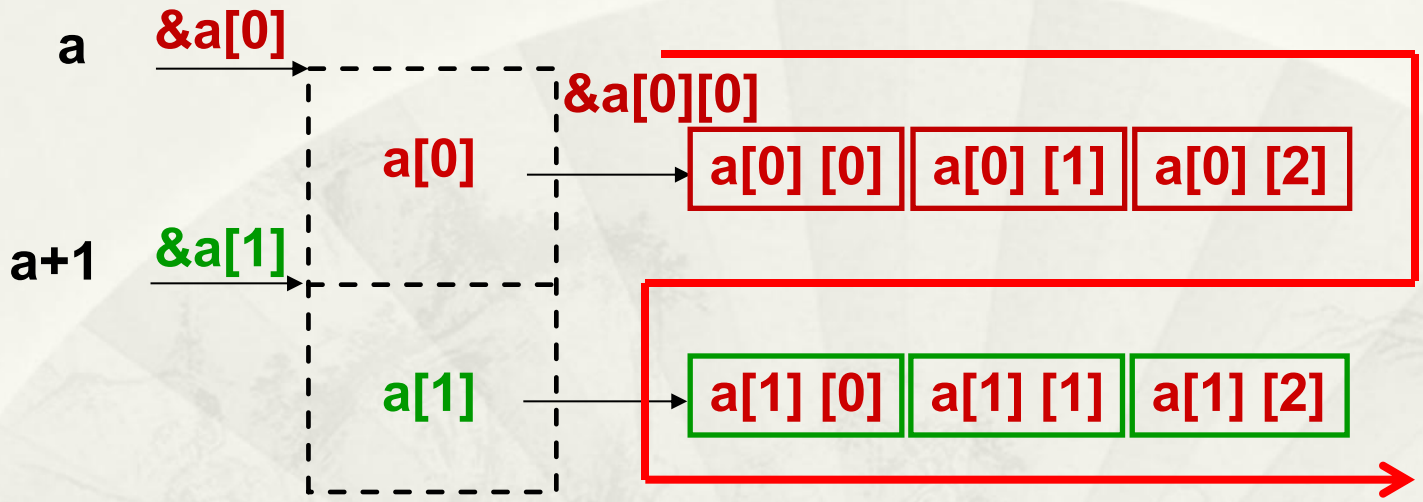



```
int a[2][3];
```

```
a[i][j]
*(*(p+i)+j)
```

11.2 指针和二维数组间的关系

- 将二维数组a看成一维数组，有6个int型元素



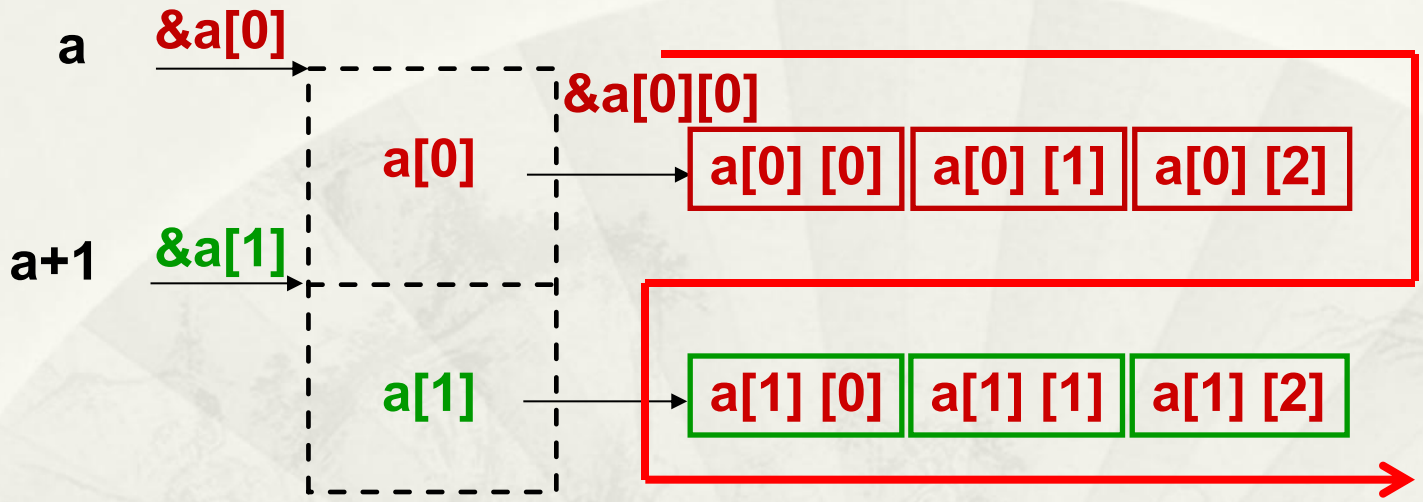
*(a + i)即 a[i]代表第i行第0列的地址，列地址

*(a+i)+j 即 a[i]+j
代表第i行第j列的地址&a[i][j]

((a+i)+j) 即 a[i][j]
代表第i行第j列的内容

11.2 指针和二维数组间的关系

- 将二维数组a看成一维数组，有6个int型元素



a[i][j]

*(p+i*n+j)

p[i*n+j]

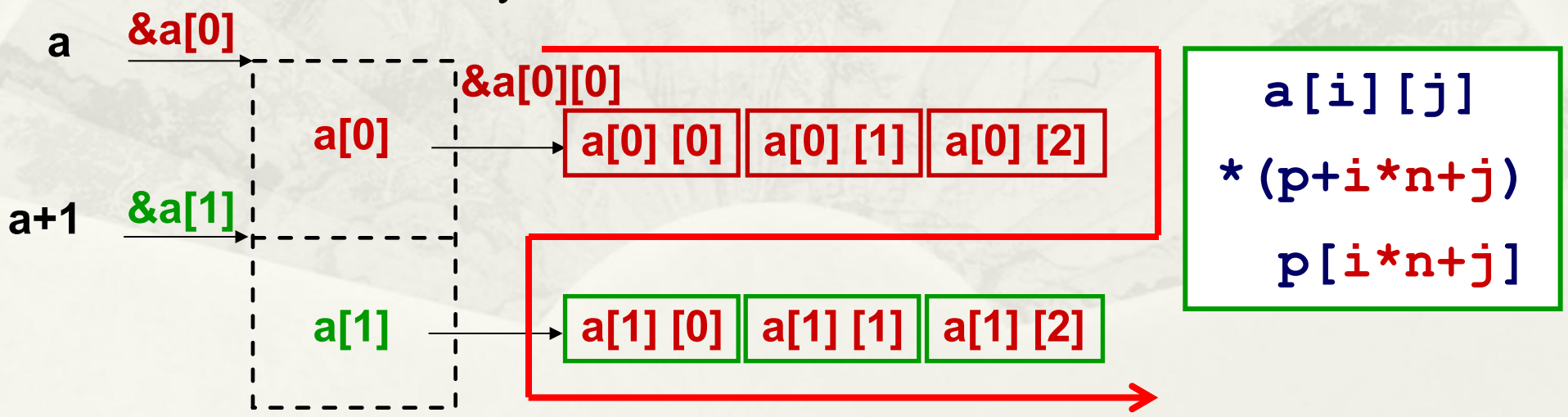
- 若要让一个指针指向它，则应定义为
  - int *p; //列指针，基类型是int
  - p = a[0];
  - p = &a[0][0]; //指向第0行第0列的int型元素

11.2 指针和二维数组间的关系

- 逐列查找
- 根据相对偏移量

```
int *p;
p = &a[0][0];
```

```
for (i=0; i<m; i++)
{
   for (j=0; j<n; j++)
   {
      printf("%d", *(p+i*n+j));
   }
}
```



11.2 指针和二维数组间的关系

【例11.3】输入一个3行4列的二维数组，然后输出这个二维数组的元素值

```
void InputArray(int p[][N], int m, int n)
{
    int i, j;
    for(i = 0; i<m; i++)
    {
        for(j = 0; j<n; j++)
        {
            scanf("%d", &p[i][j]);
        }
    }
}
```

形参声明为二维数组，列数须为常量

```
void OutputArray(int p[][N], int m, int n)
{
    int i, j;
    for(i = 0; i<m; i++)
    {
        for(j = 0; j<n; j++)
        {
            printf("%4d", p[i][j]);
        }
        printf("\n");
    }
}
```

```
InputArray(a, 3, 4);
OutputArray(a, 3, 4);
```

11.2 指针和二维数组间的关系

【例11.3】输入一个3行4列的二维数组，然后输出这个二维数组的元素值

```
void InputArray(int (*p)[N], int m, int n)
{
    int i, j;
    for(i = 0; i<m; i++)
    {
        for(j = 0; j<n; j++)
        {
            scanf("%d", *(p+i)+j);
        }
    }
}
```

形参声明为二维数组的**行指针**，列数须为常量

```
void OutputArray(int (*p)[N], int m, int n)
{
    int i, j;
    for(i = 0; i<m; i++)
    {
        for(j = 0; j<n; j++)
        {
            printf("%4d", *(*(p+i)+j));
        }
        printf("\n");
    }
}
```

```
InputArray(a, 3, 4);
OutputArray(a, 3, 4);
```

11.2 指针和二维数组间的关系

【例11.3】输入一个3行4列的二维数组，然后输出这个二维数组的元素值

```
void InputArray(int *p, int m, int n)
{
    int i, j;
    for(i = 0; i<m; i++)
    {
        for(j = 0; j<n; j++)
        {
            scanf("%d", &p[i*n+j]);
        }
    }
}
```

形参声明为二维数组的**列指针**，列数可为变量

```
InputArray(*a, 3, 4);
OutputArray(*a, 3, 4);
```

```
void OutputArray(int *p, int m, int n)
{
    int i, j;
    for(i = 0; i<m; i++)
    {
        for(j = 0; j<n; j++)
        {
            printf("%4d", p[i*n+j]);
        }
        printf("\n");
    }
}
```


Thank you for
your attention !
SuXiaoHong


Q&A