

程序设计语言C

一维数组和二维数组的定义和使用



河海大学

计算机与信息学院

王继民

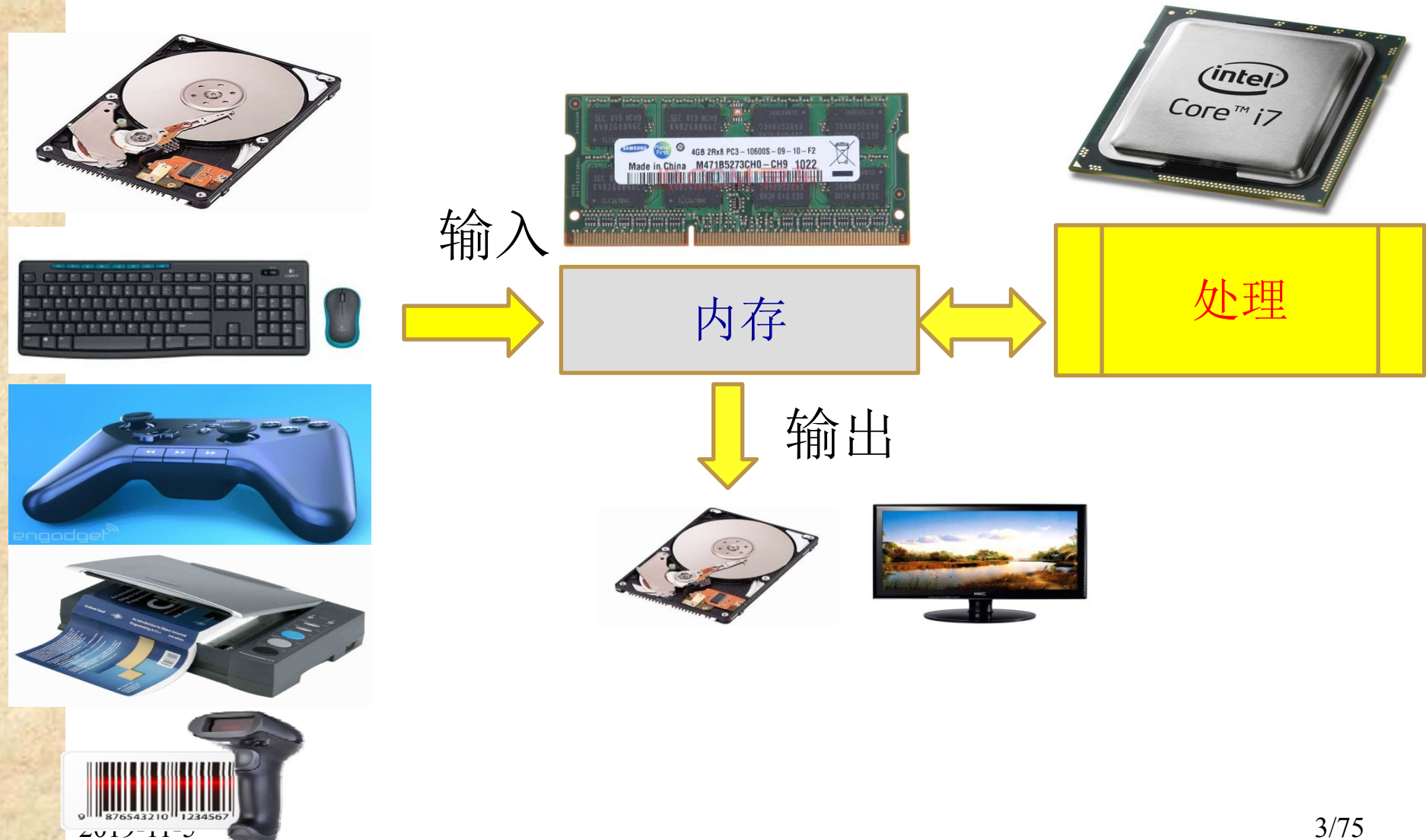
wangjimin@hhu.edu.cn



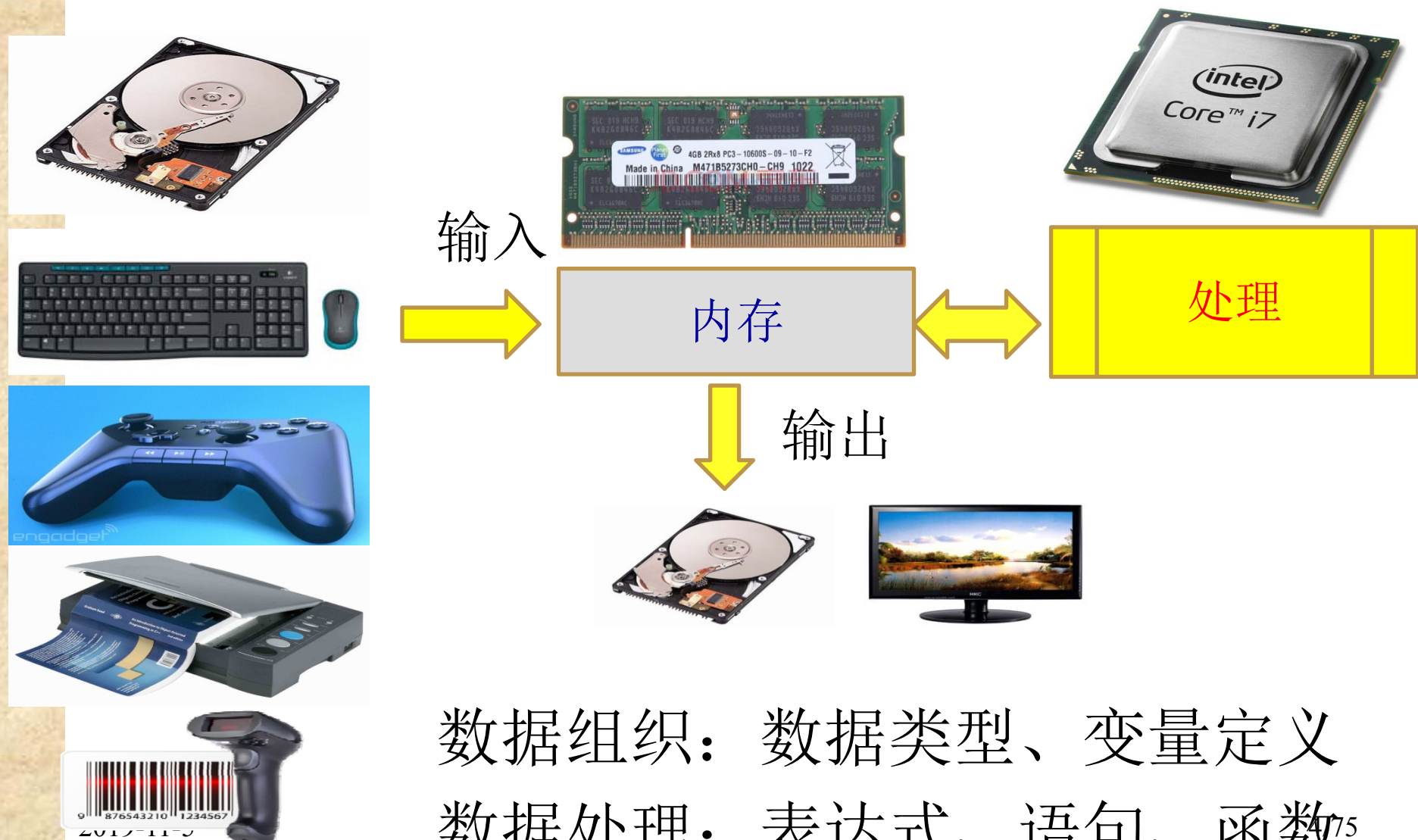
内容提要

- **6.1 一维数组的定义和使用**
- **6.2 二维数据的定义和使用**
- **补充：常用的算法**

IPO: Input Process Output 输入处理输出



IPO: Input Process Output 输入处理输出



数据组织: 数据类型、变量定义
数据处理: 表达式、语句、函数⁷⁵


为什么使用数组

- 例子：要读入某班全体**50**位同学某科学习成绩，然后进行简单处理（求平均成绩、最高分、最低分.....）
- 若用简单变量，需**50**个不同变量名，要用很多个scanf命令

```
int score1,score2,...score50;  
scanf("%d,%d,%d",&score1,&score2,&score3);  
scanf("%d,%d,%d",&score4,&score5,&score6);
```

- 而用数组，可共用一个scanf命令并利用循环结构读取

```
int score[50],i;  
for (i=0;i<50;i++)  
{  
    scanf("%d",&score[i]);  
}
```



保存大量同类型的相关数据



6.1 一维数组的定义和使用

1. 一维数组(Array)的定义

定义

数据类型 数组名 [常量表达式];

```
int a[10];
```

- 数组大小必须是常量表达式(常量或符号常量)，其值必须为正，不能为变量；
- 数组一旦定义，不能改变大小；
- 数组大小最好用宏来定义，以适应未来可能的变化

```
#define SIZE 10
```

```
int a[SIZE];
```

2. 一维数组元素的存储

- 数组是同一类型数据（数组元素）的集合。
- 一个数组在内存中占一片连续的存储单元。

如有 `int a[4];`

假设数组的起始地址为**2000**，则该数组在内存中的存储情况如图所示。

| 数组a | |
|------|------|
| 2000 | a[0] |
| 2001 | |
| 2002 | |
| 2003 | |
| 2004 | a[1] |
| 2005 | |
| 2006 | |
| 2007 | |
| 2008 | a[2] |
| 2009 | |
| 2010 | |
| 2011 | |
| 2012 | a[3] |
| 2013 | |
| 2014 | |
| 2015 | |

3. 一维数组的初始化

- 数组定义后的初值仍然是随机数，一般需要我们来初始化

```
int a[5] = { 12, 34, 56 ,78 ,9 };
```

```
int a[5] = { 0 };
```

```
int a[] = { 11, 22, 33, 44, 55 };
```

```
char digit[] = {'0','1','2','3','4','5','6','7','8','9'};
```

4. 一维数组元素的引用

数据元素引用方式:

数组名[下标]

- 下标可以是整型常量，变量或整型表达式。范围从0到 数组元素个数-1.

如，`int a[10];` 下标是从0开始，使用`a[0]`、`a[1]`、`a[2]`、……、`a[9]`这样的形式访问每个元素。数组下标(index)既可以是常量，也可以是整型表达式，如`a[i]`，通过改变下标变量访问不同元素。

4. 一维数组元素的引用

//将字符数组元素顺序输出

```
#include <stdio.h>
```

```
int main(){
```

```
    char digit[] = {'0','1','2','3','4','5','6','7','8','9'};
```

```
    for(int index=0; index<10; index++){
```

```
        printf("%c",digit[index]);
```

```
    }
```

```
    return 0;
```

```
}
```

4. 一维数组元素的引用

```
int a[10];
```

...

a[0] = a[5] + a[7] - a[2*3];//a[0],a[5]等都分别表示一个整型变量。

程序执行时，先计算方括号内表达式的值，然后将改值作为数组元素的序号，与数组名组合表示某个元素。

例 数组元素的引用。

```
#include <stdio.h>
int main( )
{
    int a[10]; //定义具有10个元素的整型数组。
    int i = 0;
    //循环将a[0],...,a[9]分别赋值
    for (i=0;i<=9;i++)
        a[i]=i;

    //分别将a[0],...,a[9]分别输出
    for (i=9;i>=0;i--)
        printf("%d ",a[i]);

    printf("\n");
    return 0;
}
```

程序使a[0]~a[9]的值为0~9，然后按逆序输出。通过循环动态修改循环控制变量i，让a[i]在不同时刻代表不同的数组元素。

运行结果如下：

9 8 7 6 5 4 3 2 1 0


```
#include <stdio.h>
```

```
int main( )
```

```
{
```

```
    int a[10];
```

```
    int index = 0;
```

```
    for(index=0; index<10; index++){  
        printf("the %d :%d",index,a[index]);  
    }
```

```
    int sum = 0;
```

```
    for(index=0; index<10; index = index+1){  
        sum = sum + a[index];  
    }
```

```
    printf("this sum is %d",sum);
```

```
    return 0;
```

```
}
```

例. 数组元素的引用,求数组元素和。



5. 一维数组使用举例

例1. 找出一个整型一维数组中最大和最小的值以及其下标。

例2. 有两个一维数组a,b，其内的元素值已经按照从小到大排序，请将a,b中的值按照从小到大排序，存储到数组c中。



例3. 用数组来处理求Fibonacci数列问题。

斐波那契数列，又称黄金分割数列，指的是这样一个数列：0、1、1、2、3、5、8、13、21、34、.....
在数学上，斐波纳契数列以如下被以递递归的方法定义：

$$F(0) = 0,$$

$$F(1) = 1,$$

$$F(n) = F(n-1) + F(n-2) \quad (n \geq 2, n \in \mathbb{N}^*)$$

求Fibonacci数列的前20个项。

可以用20个元素代表数列中的20个数，从第3个数开始，⁴⁹⁻¹¹⁻⁵可以直接用表达式 $f[i] = f[i-2] + f[i-1]$ 求出各数。^{6/75}

```
#include <stdio.h>
```

```
int main( )
```

```
{
```

```
    int i;
```

```
    int f[20] = {1,1}; //f [0]
```

```
    for(i=2; i<20; i++)
```

```
        f[i]= f[i-2] + f[i-1]; //在i的值为2时， f [2] =f [0] +f [1] ， 依此类推
```

```
    for(i=0; i<20;i++) //此循环的作用是输出20个数
```

```
    {
```

```
        printf("%d,",f[i]);
```

```
        if(i%5==0)
```

```
            printf("\n"); //控制换行， 每行输出5个数据
```

```
    }
```

```
    return 0;
```

```
}
```

| | | | | |
|-----|------|------|------|------|
| 1 | 1 | 2 | 3 | 5 |
| 8 | 13 | 21 | 34 | 55 |
| 89 | 144 | 233 | 377 | 610 |
| 987 | 1597 | 2584 | 4181 | 6765 |

如果下标值小于0或超过数组长度时会出现什么情况?

```
#include <stdio.h>
```

```
int main(){  
    int a=1, c=2, b[10], i;
```

```
    int index = 0;
```

```
    printf("a=%0x c=%0x b=%0x  b[9]=%0x \n", &a,&c, &b, &b[9]);
```

```
    for(index=0;index<12;index++){  
        b[index] = index;  
    }
```

```
    printf("%d %d\n", a, c);
```

```
    return 0;
```

```
}
```

| | |
|------|----|
| b[0] | 0 |
| b[1] | 1 |
| b[2] | 2 |
| b[3] | 3 |
| b[4] | 4 |
| b[5] | 5 |
| b[6] | 6 |
| b[7] | 7 |
| b[8] | 8 |
| b[9] | 9 |
| c | 10 |
| a | 11 |

```
a=28ff18  
c=28ff14  
b=28feec  
b[9]=28ff10  
11 10
```

运行程序或单步执行观察变量变化情况可以看到，变量c和a的值因数组越界而被悄悄破坏了



计算数组元素个数

`sizeof(数组名)`:返回数组的存储空间大小。

`sizeof(元素类型)`: 返回一个元素占用的存储空间。

`sizeof(数组名)/sizeof(元素类型)`:数组元素个数。



6.2 二维数组的定义和使用



存储所有同学的C语言的成绩，可以采用一维数组。

存储所有同学的C语言、高数、英语的成绩，采用什么样的方式？

| | | | | | |
|------|--|--|--|--|--|
| C语言 | | | | | |
| 高数 | | | | | |
| 英语 | | | | | |
| 操作系统 | | | | | |

存储所有同学的家庭住址？

一个同学的家庭住址：jiangsusheng nanjingshi jiangningqu，
需要一个字符数组。

全班同学呢？

1. 二维数组的定义

定义二维数组:

类型标识符 数组名[常量表达式][常量表达式];

例如

float a[3][4], b[5][10];

定义a为 3×4 (3行4列)的单精度数组, b为 5×10 (5行10列)的单精度数组。注意不能写成“float a[3,4], b[5,10];”。


| | | 列 | | | |
|---|------|---------|---------|---------|---------|
| | | 0 | 1 | 2 | 3 |
| 行 | a[0] | a[0][0] | a[0][1] | a[0][2] | a[0][3] |
| | a[1] | a[1][0] | a[1][1] | a[1][2] | a[1][3] |
| | a[2] | a[2][0] | a[2][1] | a[2][2] | a[2][3] |

2. 二维数组元素的引用

- 二维数组元素的引用

数据名[行下标][列下标]

- 行下标也称为第一维下标，列下标也称为第二维下标；
- 行下标和列下标可以是常量、变量以及表达公式；
- 行下标的范围是0-行数-1，列下标的范围是0-列数-1。



例. 定义四个矩阵 $A_{n \times m}$, $B_{n \times m}$, $C_{n \times m}$, $D_{n \times m}$, 编程输入矩阵元素值, 并求A,B的和放到C中, 求A,B的差放到D, 打印输出C,D。

例. 有个3 X 4二维数组, 采用随机数的方式产生每个元素的值, 输出二维数组元素中最大值和最小值。

C语言中产生随机数

```
#include <stdio.h>
```

```
#include <time.h>
```

```
#include <stdlib.h>
```

//rand() 会随机生成一个位于 0 ~ RAND_MAX 之间的整数

//srand(unsigned int) 设置随机数种子，如果每次种子设置相同，则产生的随机数也相同

```
int main(){
```

```
    srand((unsigned)time(NULL));//time返回当前时间，每次运行  
    //获取的时间都不同，因此种子也不同。
```

```
    int n = 0;
```

```
    for(n=0; n<10; n++){
```

```
        printf("%d\n", rand()%10);
```

```
    }
```

```
    return 0;
```

```
}  
2019-11-5
```

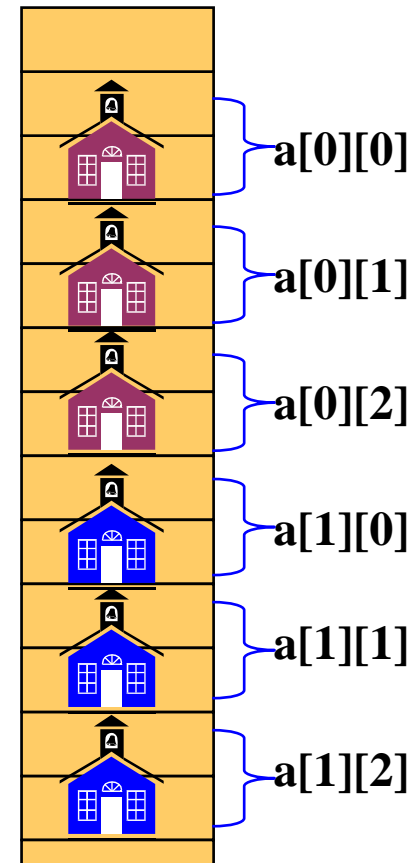
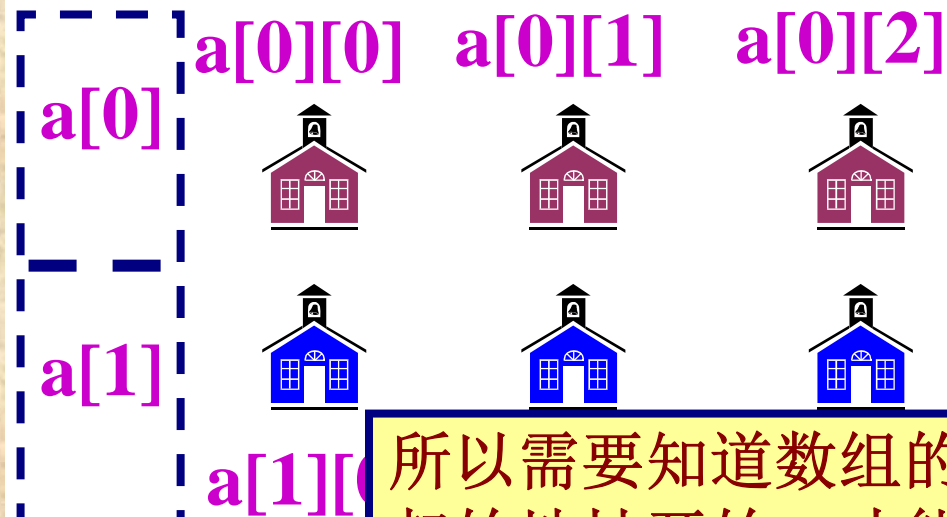
3. 二维数组的存储结构

存放顺序：按行存放

先顺序存放第0行的元素

再存放第1行的元素

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



所以需要知道数组的每一行 有多少列，这样从起始地址开始，才能正确的读出数组的元素



3. 二维数组的存储结构

练习：编程验证二维数组元素的顺序存储特性。

4. 二维数组其实还是一维数组

可以把二维数组看作是一种特殊的一维数组：

它的元素是一个一维数组。例如，可以把a看作是一个一维数组，它有3个元素：a[0], a[1], a[2]，每个元素又是一个包含4个元素的一维数组，见下图。a[0], a[1], a[2] 是3个一维数组的名字。

| | | 列 | | | |
|---|------|---------|---------|---------|---------|
| | | 0 | 1 | 2 | 3 |
| 行 | a[0] | a[0][0] | a[0][1] | a[0][2] | a[0][3] |
| | a[1] | a[1][0] | a[1][1] | a[1][2] | a[1][3] |
| | a[2] | a[2][0] | a[2][1] | a[2][2] | a[2][3] |

5. 二维数组的初始化

(1) 分行给二维数组赋初值。如

```
int a[3][4] = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}};
```

把第1个花括号内的数据赋给第1行的元素，第2个花括号内的数据赋给第2行的元素.....即按行赋初值。

(2) 可以将所有数据写在一个花括号内，按数组排列的顺序对各元素赋初值。如

```
int a[3][4] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
```

效果与前相同。但以第1种方法为好，一行对一行，界限清楚。用第2种方法如果数据多，写成一大片，容易遗漏，也不易检查。

(3) 可以对部分元素赋初值。如

```
int a[3][4]= {{1}, {5}, {9}};
```

它的作用是只对各行第1列的元素赋初值，其余元素值自动置为0。赋初值后数组各元素为

1 0 0 0

5 0 0 0

9 0 0 0

也可以对各行中的某一元素赋初值：

```
int a[3][4] = {{1}, {0, 6}, {0,0,11}};
```

初始化后的数组元素如下：

1 0 0 0

0 6 0 0

0 0 11 0

(4) 在定义时也可以通过分行赋初始值，省略第一维的长度。如

```
int a[ ][4]={ {0, 0, 3}, {}, {0, 10}};
```

这样的写法，能通知编译系统：数组共有3行。数组各元素为

0 0 3 0

0 0 0 0

0 10 0 0



补充：排序算法

从键盘输入一批数据，并对数据按照从小到大排序，并输出。

- 交换法排序
- 选择法排序

补充：排序算法

随机数产生算法：

- #include <stdio.h>

- #include <stdlib.h>

- #include <time.h>

- int main() {

 - int a;

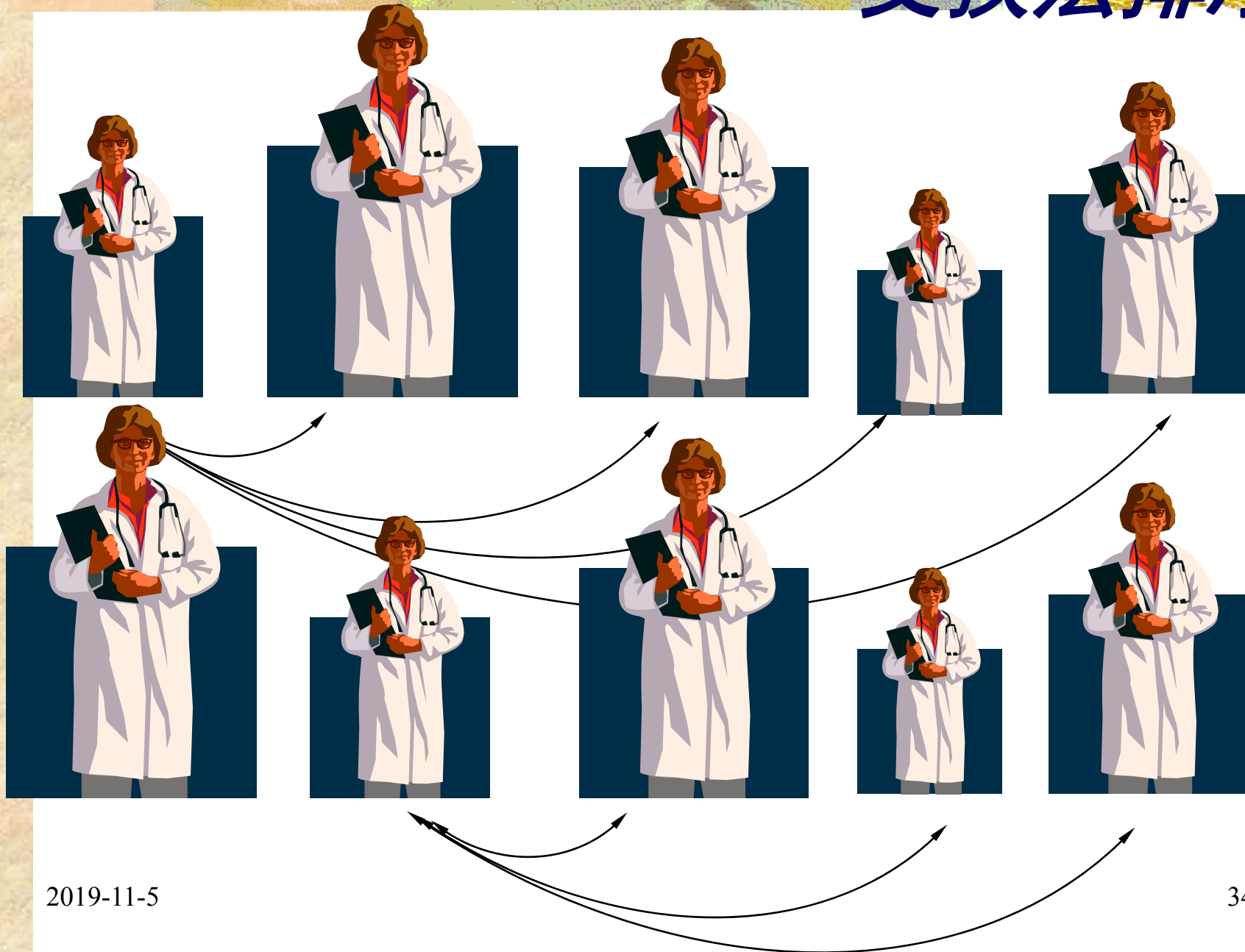
 - srand((unsigned)time(NULL)); // 设置随机数种子

 - a = rand(); // 产生[0, 1)之间的随机数

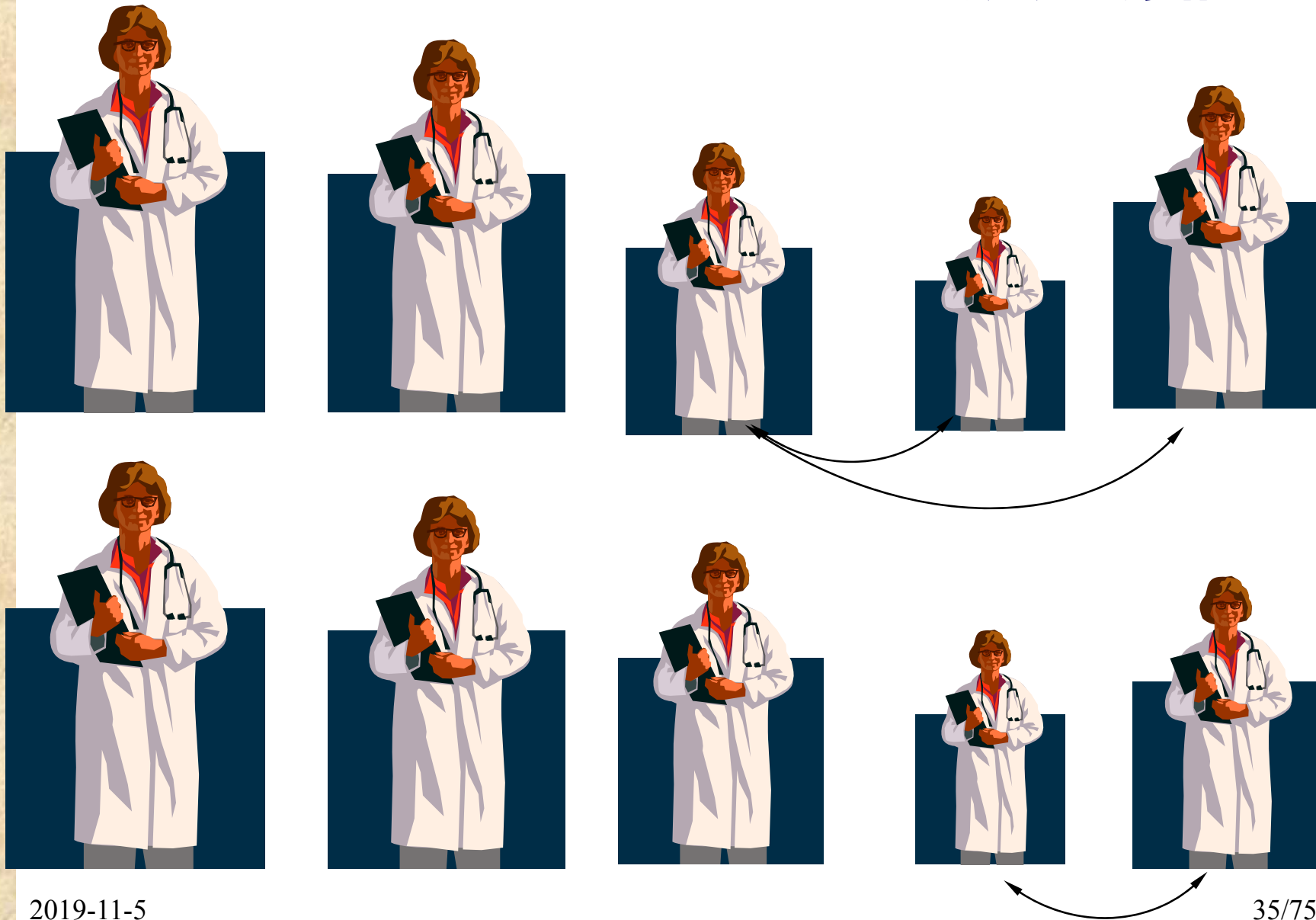
 - printf("%d\n", a);

 - return 0;

交换法排序



交换法排序



交换法排序

交换法排序

```
for (i=0; i<n-1; i++)  
{  
    for (j=i+1; j<n; j++)  
    {  
        if (score[j] > score[i])  
            "交换成绩score[j]和score[i]",  
            "交换学号num[j]和num[i]";  
    }  
}
```

选择法排序

$k=0$

$k=1$



$k=1$

$k=2$



选择法排序



从小到大排序

例 i=1 初始: [13 38 65 97 76 49 27]

Diagram showing initial array with pointers k and j. k points to 13, j points to 38. Arrows indicate comparisons between adjacent elements.

i=2 一趟: 13 [27 65 97 76 49 38]

Diagram showing the first pass of the bubble sort. The element 27 is now at index 1. k points to 27, j points to 38. Arrows indicate comparisons between adjacent elements.

二趟: 13 27 [65 97 76 49 38]

Diagram showing the second pass of the bubble sort. The elements 13 and 27 are now at their final sorted positions. k points to 65, j points to 38. Arrows indicate comparisons between adjacent elements.

三趟: 13 27 38 [97 76 49 65]

Diagram showing the third pass of the bubble sort. The elements 13, 27, and 38 are now at their final sorted positions. k points to 97, j points to 65. Arrows indicate comparisons between adjacent elements.

四趟: 13 27 38 49 [76 97 65]

Diagram showing the fourth pass of the bubble sort. The elements 13, 27, 38, and 49 are now at their final sorted positions. k points to 76, j points to 65. Arrows indicate comparisons between adjacent elements.

五趟: 13 27 38 49 65 [97 76]

Diagram showing the fifth pass of the bubble sort. The elements 13, 27, 38, 49, and 65 are now at their final sorted positions. k points to 97, j points to 76. Arrows indicate comparisons between adjacent elements.

六趟: 13 27 38 49 65 76 [97]

Diagram showing the sixth pass of the bubble sort. The array is now fully sorted: 13, 27, 38, 49, 65, 76, 97. k points to 97, j points to 97. Arrows indicate comparisons between adjacent elements.

选择法排序

选择法排序

```
for (i=0; i<n-1; i++)  
{  
    k = i;  
    for (j=i+1; j<n; j++)  
    {  
        if (score[j] > score[k])  
            记录此轮比较中最高分的元素下标 k = j;  
    }  
    若k中记录的最大数不在位置i, 则  
        "交换成绩score[k]和score[i]",  
        "交换学号num[k]和num[i]";  
}
```


作业

- 1.从键盘输入5X5阶的矩阵，编程计算（1）两条对角线上各元素之和；（2）两条对角线上行、列下标均为偶数的各元素之积。
- 2.编程打印杨辉三角。要求：先定义二维数组存储杨辉三角的元素值，然后打印。

| | | | | | |
|---|---|----|----|---|---|
| 1 | | | | | |
| 1 | 1 | | | | |
| 1 | 2 | 1 | | | |
| 1 | 3 | 3 | 1 | | |
| 1 | 4 | 6 | 4 | 1 | |
| 1 | 5 | 10 | 10 | 5 | 1 |

作业

- 3. 定义矩阵 $A_{n,m}$, 编程将该数组右上角变成0, 并输出该矩阵的值。
- 4. 输入一行字符, 统计其中的英文字符、数字字符、空格及其他字符的个数。

```
char a[10];  
for(int index=0; index<10; index++){  
    a[index] = getchar();  
}
```

- 定义矩阵 $A_{n,m}$, $B_{m,p}$, $C_{n,p}$, 计算 $A*B$ 的结果, 并存放至 C 中。



6. (选做) 实现CNN中的卷积操作和池化操作。

定义 28×28 的二维数组data, 每个元素采用随机数方式产生, 范围为0-1。定义 3×3 的卷积核, 对该二维数组进行卷积操作, 步长为1, 输出卷积结果cdata; 对cdata进行池化操作, 分别输出采用均值池化/最大值池化的操作结果pdata, 假设池化操作的核大小为 2×2 , 步长为2。

卷积核权重为:

0.1 0 0.3

0.3 0 0.1

0.5 0.4 0

卷积操作:

输入数据

| | | | | | |
|---|----|----|----|----|----|
| 1 | 2 | 3 | 4 | 5 | 6 |
| 2 | 3 | 4 | 5 | 6 | 7 |
| 3 | 4 | 5 | 6 | 7 | 8 |
| 4 | 5 | 6 | 7 | 8 | 9 |
| 5 | 6 | 7 | 8 | 9 | 10 |
| 6 | 7 | 8 | 9 | 10 | 11 |
| 7 | 8 | 9 | 10 | 11 | 12 |
| 8 | 9 | 10 | 11 | 12 | 13 |
| 9 | 10 | 11 | 12 | 13 | 14 |

| | | |
|-----|-----|-----|
| 0.1 | 0.1 | 0.1 |
| 0.1 | 0.1 | 0.1 |
| 0.1 | 0.1 | 0.1 |

卷积核

| | | | |
|-----|--|--|--|
| 2.4 | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

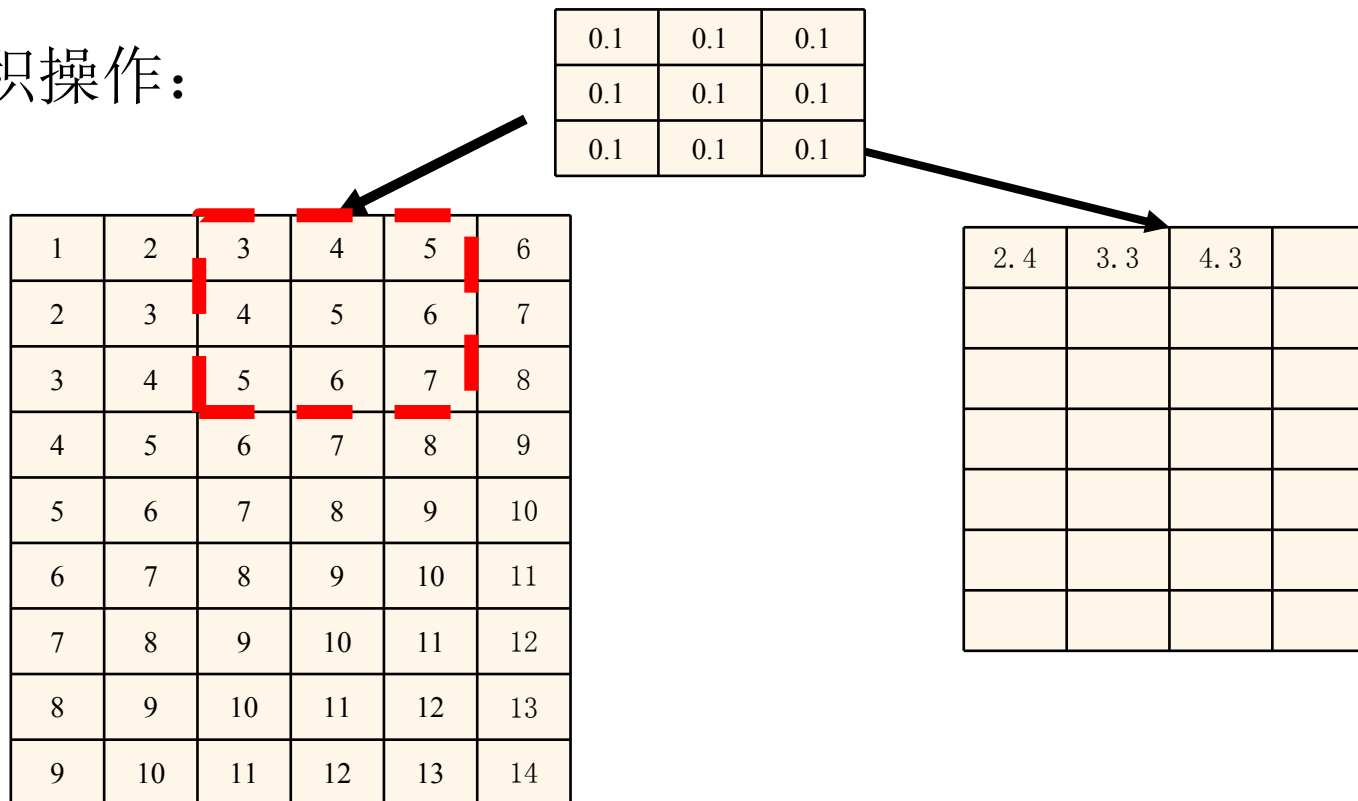
卷积操作:

| | | |
|-----|-----|-----|
| 0.1 | 0.1 | 0.1 |
| 0.1 | 0.1 | 0.1 |
| 0.1 | 0.1 | 0.1 |

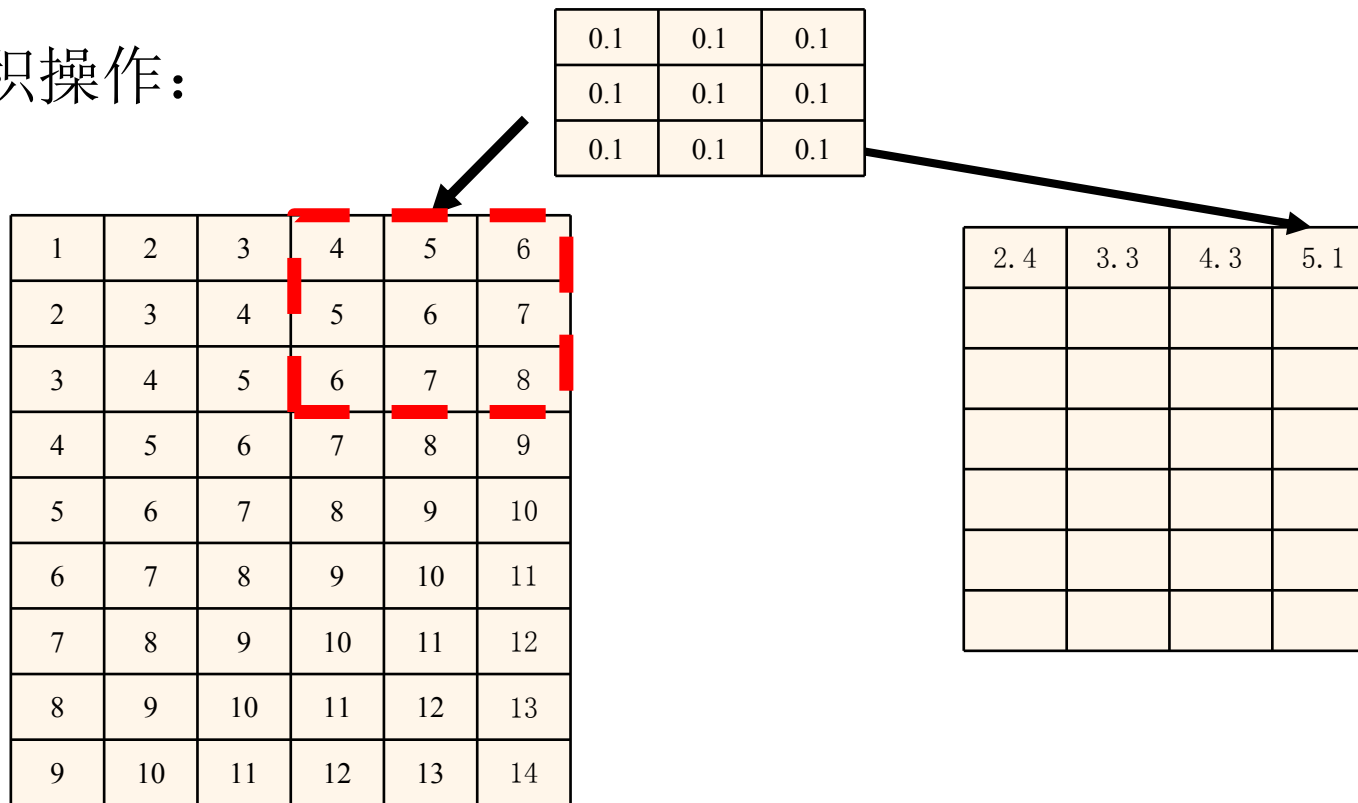
| | | | | | |
|---|----|----|----|----|----|
| 1 | 2 | 3 | 4 | 5 | 6 |
| 2 | 3 | 4 | 5 | 6 | 7 |
| 3 | 4 | 5 | 6 | 7 | 8 |
| 4 | 5 | 6 | 7 | 8 | 9 |
| 5 | 6 | 7 | 8 | 9 | 10 |
| 6 | 7 | 8 | 9 | 10 | 11 |
| 7 | 8 | 9 | 10 | 11 | 12 |
| 8 | 9 | 10 | 11 | 12 | 13 |
| 9 | 10 | 11 | 12 | 13 | 14 |

| | | | |
|-----|-----|--|--|
| 2.4 | 3.3 | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

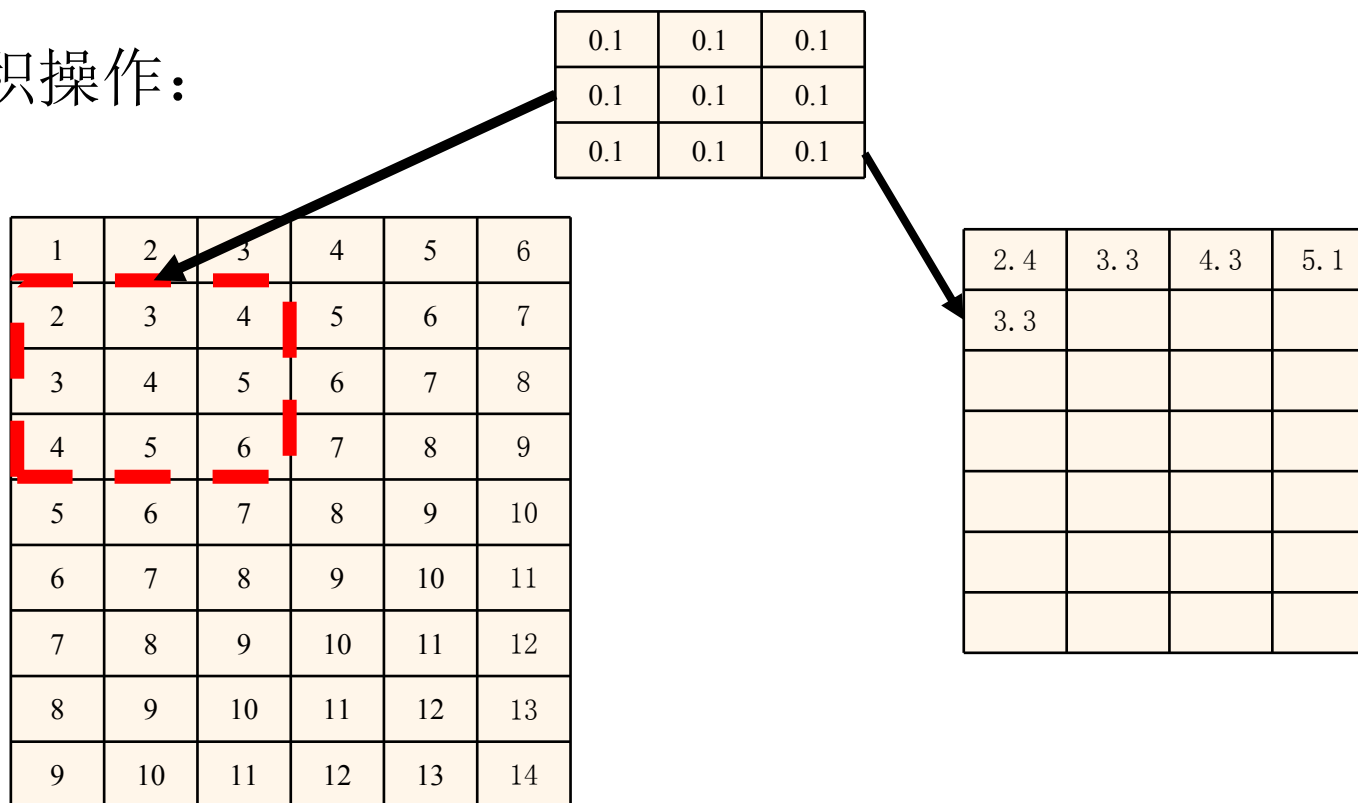
卷积操作:



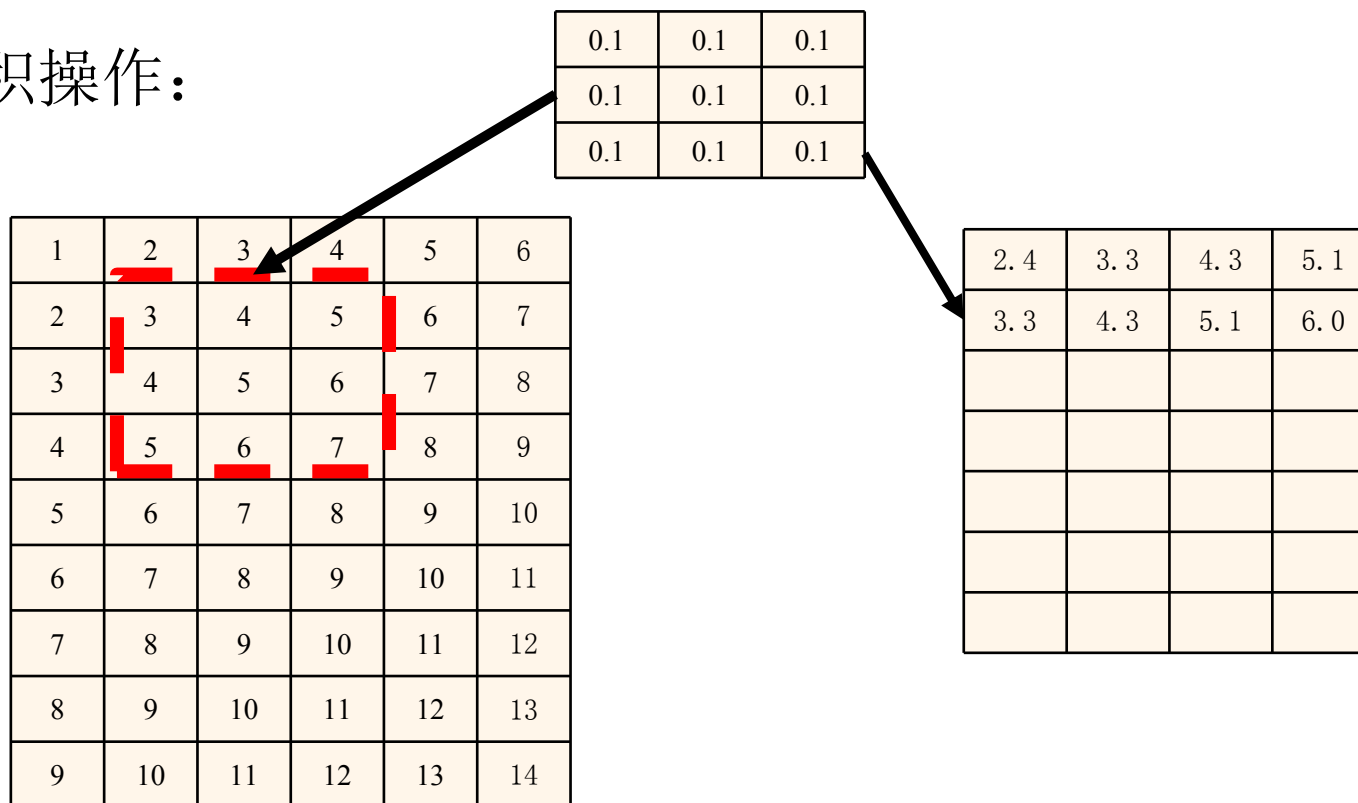
卷积操作:



卷积操作:



卷积操作:



池化操作:

池化核大小 2×2

输入数据

| | | | |
|-----|-----|-----|-----|
| 2.4 | 3.3 | 4.3 | 5.1 |
| 3.3 | 4.3 | 5.1 | 6.0 |
| | | | |
| | | | |
| | | | |
| | | | |

均值

| | |
|------|--|
| 3.33 | |
| | |
| | |

最大值

| | |
|-----|--|
| 4.3 | |
| | |
| | |

最小值

| | |
|-----|--|
| 2.4 | |
| | |
| | |

池化操作:

池化核大小 2×2

输入数据

| | | | |
|-----|-----|-----|-----|
| 2.4 | 3.3 | 4.3 | 5.1 |
| 3.3 | 4.3 | 5.1 | 6.0 |
| | | | |
| | | | |
| | | | |
| | | | |

均值

| | |
|-----|-----|
| 3.3 | 5.1 |
| | |
| | |

最大值

| | |
|-----|-----|
| 4.3 | 6.0 |
| | |
| | |

最小值

| | |
|-----|-----|
| 2.4 | 4.3 |
| | |
| | |

池化操作:

池化核大小 2×2

输入数据

| | | | |
|-----|-----|-----|-----|
| 2.4 | 3.3 | 4.3 | 5.1 |
| 3.3 | 4.3 | 5.1 | 6.0 |
| | | | |
| | | | |
| | | | |
| | | | |

均值


| | |
|-----|-----|
| 3.3 | 5.1 |
| | |
| | |

最大值

| | |
|-----|-----|
| 4.3 | 6.0 |
| | |
| | |

最小值

| | |
|-----|-----|
| 2.4 | 4.3 |
| | |
| | |



7.员工管理系统v2，实现一个员工信息管理系统，员工信息包括：姓名，工号，工资。假设员工数量不超过100人。

功能要求：

- (1) 员工信息增加；
- (2) 员工信息删除；
- (3) 查询员工信息：工号。

8. 熟悉C语言的math.h头文件中的函数.

欢迎交流





学号_姓名

群名称:程序设计语言C

群 号:683689303