

1 函数

1.1 数的概述

发现不断进行加法运算，为了提高代码的复用性，就把该功能独立封装成一段独立的小程序，当下次需要执行加法运算的时候，就可以直接调用这个段小程序即可，那么这种封装形形式的具体表现形式则称作函数。

练习：把两个整数相加：

```
public class FunctionDemo1 {
    public static void main(String[] args) {
        /*
        int a = 4+5;
        System.out.println("a="+a);
        int b = 3+9;
        System.out.println("b="+b);

        */
        add(4,5)
    }
    public static int add(int a, int b) {
        return a+b;
    }
}
```

举例：地铁自动售票机，接收纸币或硬币，返回车票，该售票机具有独立功能，可以理解为函数。有未知内容参与运算（要投币，纸币硬币，多少钱）。有返回值（返回车票）

举例 2：手机，手机具备打电话功能，有未知内容（电话号码），键盘是（形参），输入的号码是实际参数。

1.2 函数的格式

修饰符 返回值类型 函数名 (参数类型 形式参数1, 参数类型 形式参数2, ...)

```
{
    执行语句;
    return 返回值;
}
```

返回值类型：运行这段程序得出的一个运算结果，结果类型，如果函数没有返回值则用void来表示该函数没有返回值。

函数名：仅仅是一个标识符，可以随意起名字。

形式参数：是一个变量，用于存储调用函数传递进来的实际参数。

实际参数：传递给形式参数的具体数值。

返回值：返回给调用者。

定义函数：

1: 是否有未知内容参与运算

2: 是否有运算结果（返回值）

案例：获取2个整数中的较大的数。

```
public static int getMax(int x, int y) {  
    int result;  
    if (x > y) {  
        result = x;  
    } else {  
        result = y;  
    }  
    return result;  
}
```

解析：getMax方法

该方法方法名为：getMax，方法的作用是获取找出两个整数中较大的值。该方法有两个int型参数，：x和y，方法返回两个数中较大的一个。

public static 是方法的修饰符

int 是方法的返回值类型

getMax 是方法的方法名

(int x,int y) 是参数列表，x和y是形式参数。

{ } 花括号内的代码是方法体

return result; result 是返回值。

方法定义完成之后，如何调用一个方法？

函数调用：

想要使用方法，必须调用它。

```
public static void main(String[] args) {  
    int max = getMax(5, 7);  
    System.out.println(max);  
}
```

一：在main方法中调用getMax()方法，5和7就是给该方法传递的实际参数。如果方法有返回值，可以定义一个变量接收返回值，变量类型和方法返回值类型一致。本例中通过int类型变量max接收了getMax方法的返回值。

完整程序

这里的getMax(i, j); i和j就是实际参数。

```
public class Demo6 {  
    public static void main(String[] args) {
```

```

    int i=5;
    int j=7;
    int max = getMax(i, j);
    System.out.println(i+"和"+j+"的最大值是: "+max);
}

public static int getMax(int x, int y) {
    int result;
    if (x > y) {
        result= x;
    } else {
        result= y;
    }
    return result;
}
}

```

二：上述案例中调用getMax方法，并将结果赋值给了max变量。也可以直接打印getMax()方法的结果。

```

public class Demo6 {
    public static void main(String[] args) {
        int i=5;
        int j=7;
        //打印方法的结果
        System.out.println(getMax(i, j));
    }

    public static int getMax(int x, int y) {
        int result;
        if (x > y) {
            result= x;
        } else {
            result= y;
        }
        return result;
    }
}

```

注意：main方法是程序的入口由虚拟机调用，方法和方法之间不能嵌套，方法之间通过调用来使用。

方法什么时候执行完毕：

当执行完return语句，或者执行到方法末尾的花括号时方法结束。

该类中包含了两个方法，main方法和getMax方法。main方法由java虚拟机调用，并且main方法的写法是固定的。Main方法可以调用其他方法。

当调用 getMax 方法时，变量 i 的值传递给方法中的 x，j 的值传递给方法中的 y，并开始

执行 getMax 方法中的语句，执行 return，并返回运算结果。getMax 方法运行完毕。

1.3 函数的特点

- 1、定义函数可以将功能代码进行封装
- 2、便于对该功能进行复用
- 3、函数只有被调用才会被执行
- 4、函数的出现提高了代码的复用性
- 5、对于函数没有具体返回值的情况，返回值类型用关键字void表示，那么该函数中的return语句如果在最后一行可以省略不写。

注意：

函数中只能调用函数，不可以在函数内部定义函数。

定义函数时，函数的结果应该返回给调用者，交由调用者处理。

函数的返回值void

需求：根据学生考试成绩划分 ABCD A90-100 B80-89 C70-79 D60-69 E0-59，建议成绩使用 double。将该程序使用函数定义。

```
public static void main(String[] args) {  
    printGrade(90);  
    printGrade(59.5);  
}  
  
public static void printGrade(double score) {  
    char grade;  
    if (score >= 90.0)  
        System.out.println("A");  
    else if (score >= 80.0)  
        System.out.println("B");  
    else if (score >= 70.0)  
        System.out.println("C");  
    else if (score >= 60.0)  
        System.out.println("D");  
    else  
        System.out.println("E");  
}
```

```
public static void main(String[] args) {
```

```

    getResult(5);
}

public static int getResult(int x) {
    System.out.println(return x*8);
    //调用该函数会报错.缺少返回值类型.
}

/*
 * 该方法没有具体的返回值,那么返回值的类型 是不可以写int 了
 * 但是又和函数的格式不符合了,怎么解决?
 * 当函数运算后, 没有具体的返回值时,这时返回值类型用一个特殊的关键字做标志.
 * 该关键字就是void void 表示没有具体的返回值类型.
 * 当函数的返回值类型是void 时,函数中的return语句可以省略不写.
 */

```

注意：函数中只能调用函数，不能在函数内部定义函数。函数之间是平级的，相互之间是调用的关系。

错误写法

```

public static void main(String[] args) {
    public static void getResult(int x) {
        System.out.println(x * 8);
        // 主函数，里边嵌套函数。错误！
    }
}

```

1.4 函数的应用

案例一：画矩形。

```

/*
    为了提高代码的复用性
    定义一个画矩形的函数
    1、确定函数的运算结果的数据类型，void
    2、确定没有未知参数。
 */

public static void draw(int width , int height){
    for(int i = 0 ; i< height ; i++){
        for(int j = 0 ; j < width ; j++){
            System.out.print("*");
        }
        System.out.println();
    }
}

```

案例二：两个数字对比是否相等

```
/*
    1、确定函数的运算结果的数据类型，boolean
    2、确定没有未知参数。a,b
*/
public static boolean equlas(int a , int b){
    /**
    if(a==b){
        return true;
    }else{
        return false;
    }
    **/
    return a==b?true:false;

}
```

案例三：比较两个数的大小

```
/*
    比较两个整数谁大.
*/
public static int max(int a,int b)
{
    /**
    if(a>b)
        return a;
    else
        return b;
    */

    return a>b?a:b;
}
```

函数的使用注意事项：

1. 函数中只能调用函数，不能定义函数，没有定义函数之前，不能调用函数。
2. 输出语句只能对有具体返回结果的函数的进行打印。
3. 返回值类型是 void 的函数，是不可以被输出语句打印的。
4. 函数需要实现功能，那么函数只实现所需功能即可，不要实现不需要的功能。

1.5 函数的重载

1、**函数重载的定义**：在同一个类中，有一个以上的同名函数，只要函数的参数列表或参数类型不一样即可，与返回值无关， 这些统称为方法的重载。

2、**函数的重载存在的原因**：为了增强方法的阅读性，优化了程序设计。

案例 1：九九乘法表

```
private static void print99() {
    for(int i = 1 ; i<= 9 ; i ++){
        for(int j = 1 ; j<=i ; j++){
            System.out.print(i+"*"+j+"="+i*j)+" ";
        }
        System.out.println();
    }
}

private static void print99(int num) {
    for(int i = 1 ; i<= num ; i ++){
        for(int j = 1 ; j<=i ; j++){
            System.out.print(i+"*"+j+"="+i*j)+" ";
        }
        System.out.println();
    }
}
```

练习：判断那个方法是重载

```
void show(int w, double c, char b){}
```

```
void show(int x, char y, double z){}      true
void show(int a, double c, char b){}      false
void show(int a, char b){}                true
void show(double c){}                     true
double show(int x, char y, double z){}    true
```

2 数组

概念
同一种类型数据的集合。其实数组就是一个容器。
数组的好处
可以自动给数组中的元素从 0 开始编号，方便操作这些元素。
格式 1:
元素类型[] 数组名 = new 元素类型[元素个数或数组长度]; 示例: <code>int[] arr = new int[5];</code>
格式 2:
元素类型[] 数组名 = new 元素类型[]{元素, 元素,}; <code>int[] arr = new int[]{3,5,1,7};</code> <code>int[] arr = {3,5,1,7};</code>

如果需要存储大量的数据，例如如果需要读取 100 个数，那么就需要定义 100 个变量，显然重复写 100 次代码，是没有太大意义的。如何解决这个问题，Java 语言提供了数组（array）的数据结构，是一个容器可以存储相同数据类型的元素，可以将 100 个数存储到数组中。

1 数组的概念

同一种类型数据的集合。其实数组就是一个容器。运算的时候有很多数据参与运算,那么首先需要做什么.不是如何运算而是如何保存这些数据以便于后期的运算,那么数组就是一种用于存储数据的方式,能存数据的地方我们称之为容器,容器里装的东西就是数组的元素,数组可以装任意类型的数据,虽然可以装任意类型的数据,但是定义好的数组只能装一种元素,也就是数组一旦定义,那么里边存储的数据类型也就确定了。

2 数组的好处

存数据和不存数据有什么区别吗?数组的最大好处就是能都给存储进来的元素自动进行编号.注意编号是从 0 开始。方便操作这些数据。

例如 学生的编号,使用学号就可以找到对应的学生。

3 数组的格式

元素类型[] 数组名 = new 元素类型[元素个数或数组长度];

示例: `int[] arr = new int[5];`

案例:

需求:想定义一个可以存储 3 个整数的容器

实现:

1 声明数组变量

为了使用数组必须在程序中声明数组,并指定数组的元素类型

=左半部分:

先写左边明确了元素类型 是 `int` , 容器使用数组,那么如何来标识数组?.那么用一个特殊的符号[]中括号来表示。想要使用数组是需要给数组起一个名字的,那么我们在这里给这个数组起名字为 `x` .接着跟上等号。

代码体现:

```
int [] x
```

注意: `int x[]` 也是一种创建数组的格式。推荐使用 `int [] x` 的形式声明数组。

2 创建数组

=右半部分:

要使用一个新的关键字,叫做 `new`。`new` 用来在内存中产生一个容器实体,数据要存储是需要有空间的,存储很多数据的空间用 `new` 操作符来开辟, `new int[3]`; 这个 3 是元素的个数。右边这部分就是在内存中定义了一个真实存在的数组,能存储 3 个元素。

`new int[3]` 做了两件事情,首先使用 `new int[3]` 创建了一个数组,然后把这个数组的引用赋值给数组变量 `x`。

```
int [] x=new int[3];
```

x 是什么类型?

任何一个变量都得有自己的数据类型。注意这个 `x` 不是 `int` 类型的。`int` 代表的是容器里边元素的类型。那么 `x` 是数组类型的。

数组是一种单独的数据类型。数据类型分为 2 大派,分为基本数据类型和引用数据类型。第二大派是引用数据类型。那么大家现在已经接触到了引用数据类型三种当中的一种。就是数组类型 `[]` 中括号就代表数组。

4、`int[] arr = new int[5]`;在内存中发生了什么?

内存任何一个程序,运行的时候都需要在内存中开辟空间。`int[] arr = new int[5]`; 这个程序在内存中是什么样?这就涉及到了 `java` 虚拟机在执行程序时所开辟的空间,那么 `java` 开辟启动了多少空间呢? 继续学习 `java` 的内存结构。

1. 数组的定义

格式 1:
元素类型[] 数组名 = new 元素类型[元素个数或数组长度]; 示例: <code>int[] arr = new int[5];</code>
格式 2:
元素类型[] 数组名 = new 元素类型[] {元素, 元素,}; <code>int[] arr = new int[] {3,5,1,7};</code> <code>int[] arr = {3,5,1,7};</code>

注意: 给数组分配空间时,必须指定数组能够存储的元素个数来确定数组大小。创建数组之后不能修改数组的大小。可以使用 `length` 属性获取数组的大小。

遍历数组

2. 数组初始化

数组的格式

```
int[] x = new int[3];  
x[0] = 1;
```

```
x[1] = 2;
```

另一种定义：该形式可以直接明确数组的长度,以及数组中元素的内容

```
int[] x = { 1, 2, 3 };
```

```
int[] x=new int[]{1,2,3};
```

初始化方式 1: 不使用运算符 new

```
int[] arr = { 1, 2, 3, 4, 5 };
```

```
int[] arr2 = new int[] { 1, 2, 3, 4, 5 };
```

初始化方式2:

```
int[] arr3=new int[3];
```

```
arr3[0]=1;
```

```
arr3[1]=5;
```

```
arr3[2]=6;
```

如果数组初始化中不使用运算符 new。需要注意：下列写法是错误的。

```
int[] arr;
```

```
arr={1,2,3,4,5};
```

此时初始化数组，必须将声明，创建，初始化都放在一条语句中个，分开会产生语法错误。

所以只能如下写：

```
int[] arr={1,2,3,4,5};
```

3. 数组遍历

```
public static void main(String[] args) {  
    int[] x = { 1, 2, 3 };  
    for (int y = 0; y < 3; y++) {  
        System.out.println(x[y]);  
        // System.out.println("x["+y+"]="+x[y]); 打印效果 x[0]=1;  
    } // 那么这就是数组的第一个常见操作.遍历  
}
```

数组中有一个属性可以获取到数组中元素的个数,也就是数组的长度. 数组名.length

```
public static void main(String[] args) {  
    int[] x = { 1, 2, 3 };  
    for (int y = 0; y < x.length; y++) {  
        System.out.println(x[y]);  
        // System.out.println("x["+y+"]="+x[y]); 打印效果 x[0]=1;  
    } // 那么这就是数组的第一个常见操作.遍历  
}
```

4. 数组的常见异常

一数组角标越界异常:，注意：数组的角标从 0 开始。

```
public static void main(String[] args) {  
    int[] x = { 1, 2, 3 };  
    System.out.println(x[3]);  
    //java.lang.ArrayIndexOutOfBoundsException  
}
```

二 空指针异常:

```
public static void main(String[] args) {  
    int[] x = { 1, 2, 3 };  
    x = null;  
    System.out.println(x[1]);  
    // java.lang.NullPointerException  
}
```

数组:

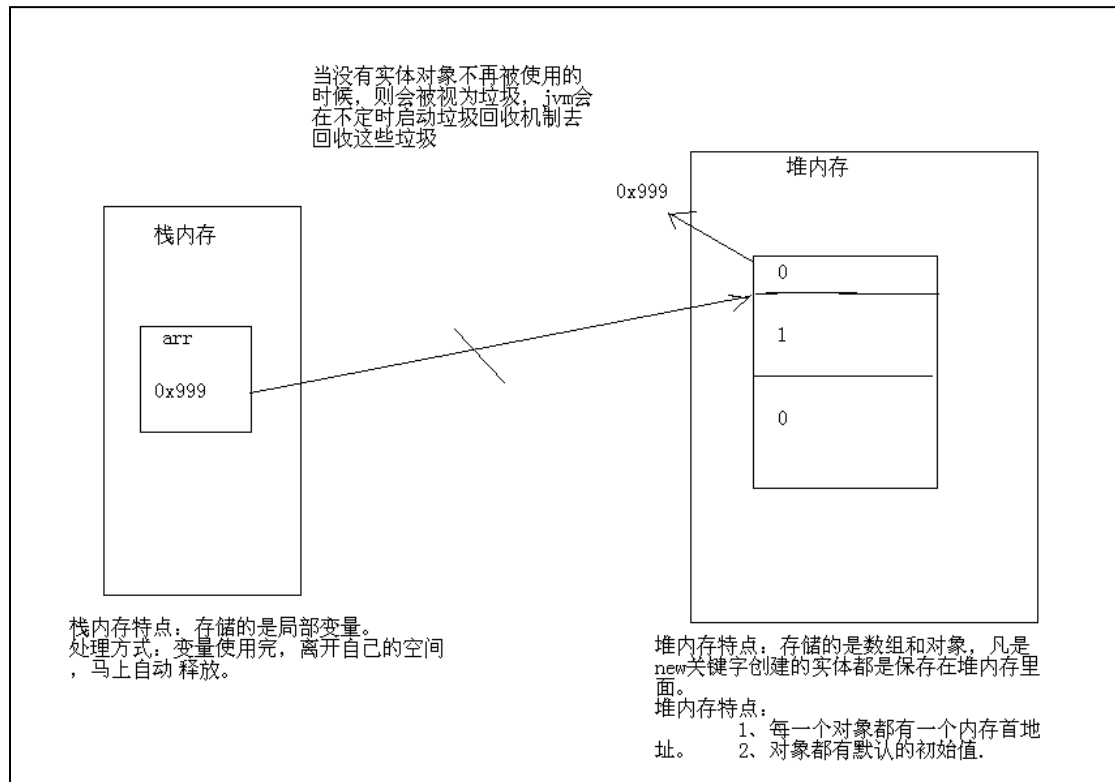
什么时候使用数组:当元素较多时为了方便操作这些数组,会先进行来临时存储，所使用的容器就是数组。

特点:

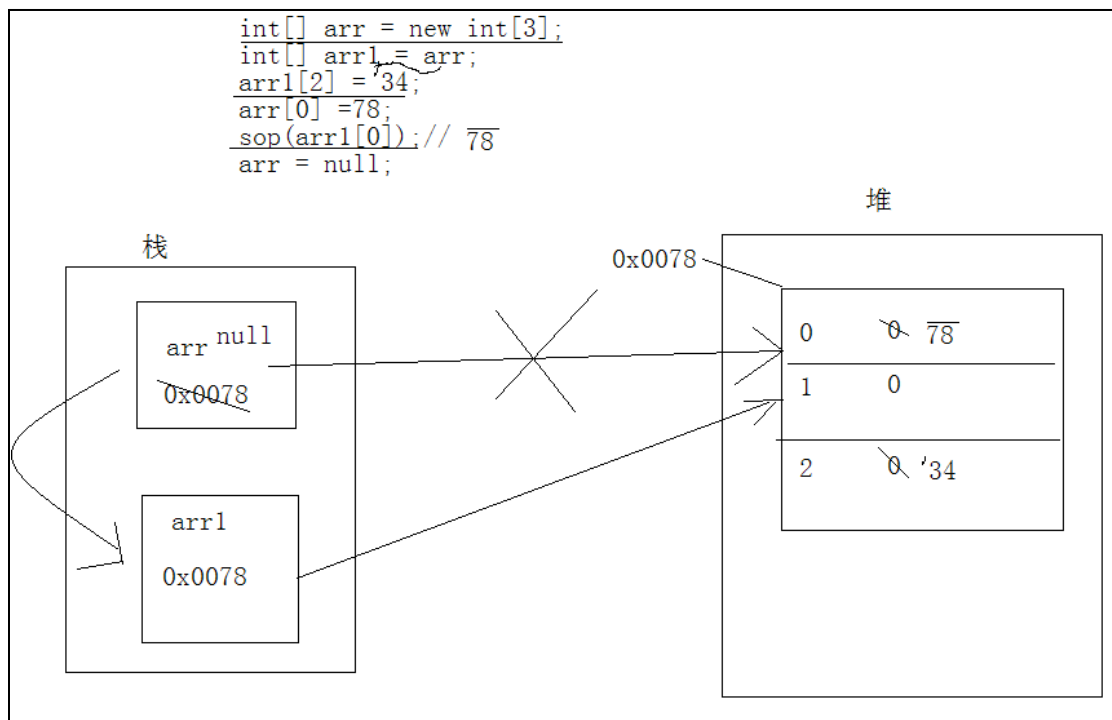
数组长度是固定的。

5. 数组的内存分析

案例分析一:



案例分析二：



2. 数组的常见操作

6.1 案例一个数组取出最大值

/*定义一个获取最大值的功能：

1、确定结果：返回值类型 int

2、未知内容：要获取哪个数组的最大值没有确定，则是数组没有确定

思路：

1、定义一个变量，记录住数组的比较大的元素。

2、遍历整个数组，让数组的每一个元素都和该变量进行对比即可。

3、当变量遇到比它大的元素，则让该变量记录该元素的值，当循环结束时，最大值产生了

*/

```
public static int getMax(int[] arr)
{
    //定义变量记录较大的值，初始化为数组中的任意一个元素。
    int max = arr[0];

    for(int x=1; x<arr.length; x++)
    {
        if(arr[x]>max)
            max = arr[x];
    }
    return max;
}
```

6.2 直接排序

案例二：使用直接排序对数组进行排序：

/*

选择排序。

以一个角标的元素和其他元素进行比较。

在内循环第一次结束，最值出现的头角标位置上。

*/

```
public static void selectSort(int[] arr)
{
    for(int x=0; x<arr.length-1; x++)
```

```

{
    for(int y=x+1; y<arr.length; y++) //为什么y的初始化值是 x+1?
    因为每一次比较,
                                     //都用x角标上的元素和下一个元素进
    行比较。
    {
        if(arr[x]>arr[y])
        {
            int temp = arr[x];
            arr[x] = arr[y];
            arr[y] = temp;
        }
    }
}

```

6.3 冒泡排序

案例三：冒泡排序

```

/*
冒泡排序。
比较方式：相邻两个元素进行比较。如果满足条件就进行位置置换。
原理：内循环结束一次，最值出现在尾角标位置。
*/
public static void bubbleSort(int[] arr)
{
    for(int x=0; x<arr.length-1; x++)
    {
        for(int y=0; y<arr.length-x-1; y++) //-x:让每次参与比较的元减。
                                     //-1:避免角标越界。
        {
            if(arr[y]>arr[y+1])
            {
                int temp = arr[y];
                arr[y] = arr[y+1];
                arr[y+1] = temp;
            }
        }
    }
}

```

6.4 折半查找(二分法)

案例四:

```
/*
    为了提高查找效率，可使用折半查找的方式，注意：这种查找只对有序的数组有效。
    这种方式也成为二分查找法。
*/
public static int halfSeach(int[] arr,int key)
{
    int min,mid,max;
    min = 0;
    max = arr.length-1;
    mid = (max+min)/2;

    while(arr[mid]!=key)
    {
        if(key>arr[mid])
            min = mid + 1;
        else if(key<arr[mid])
            max = mid - 1;

        if(min>max)
            return -1;

        mid = (max+min)/2;
    }
    return mid;
}
```

案例五：数组翻转

```
/*
    反转其实就是头角标和尾角标的元素进行位置的置换，
    然后在让头角标自增。尾角标自减。
    当头角标<尾角标时，可以进行置换的动作。
*/
public static void reverseArray(int[] arr)
{
    for(int start=0,end=arr.length-1; start<end; start++,end--)
    {
        swap(arr,start,end);
    }
}
```

```
//对数组的元素进行位置的置换。
public static void swap(int[] arr,int a,int b)
{
    int temp = arr[a];
    arr[a] = arr[b];
    arr[b] = temp;
}
```

3 二维数组

Arrays 的使用

```
int [] a = new int[]{80,23,45,3,56};
System.out.println( Arrays.toString( a ) );    // [80,23,45,3,56]
Arrays.sort( a );
System.out.println( Arrays.toString( a ) );    // [3, 23, 45, 56, 80]
System.out.println( "最小值 : " + a[0] + " , 最大值 : " + a[a.length-1] );    // 排序求最值
System.out.println( Arrays.binarySearch( a , 450 ) );
```

遍历: toString() 将数组的元素以字符串的形式返回

排序: sort() 将数组按照升序排列

查找: binarySearch()在指定数组中查找指定元素, 返回元素的索引, 如果没有找到返回 (-插入点-1) 注意: 使用查找的功能的时候, 数组一定要先排序。

二维数组:

吸烟:

没钱	零买	1 根	一个变量
稍微有钱	一包	一维数组	20 根变量
很有钱	一条	10 包(二维数组)	二维数组

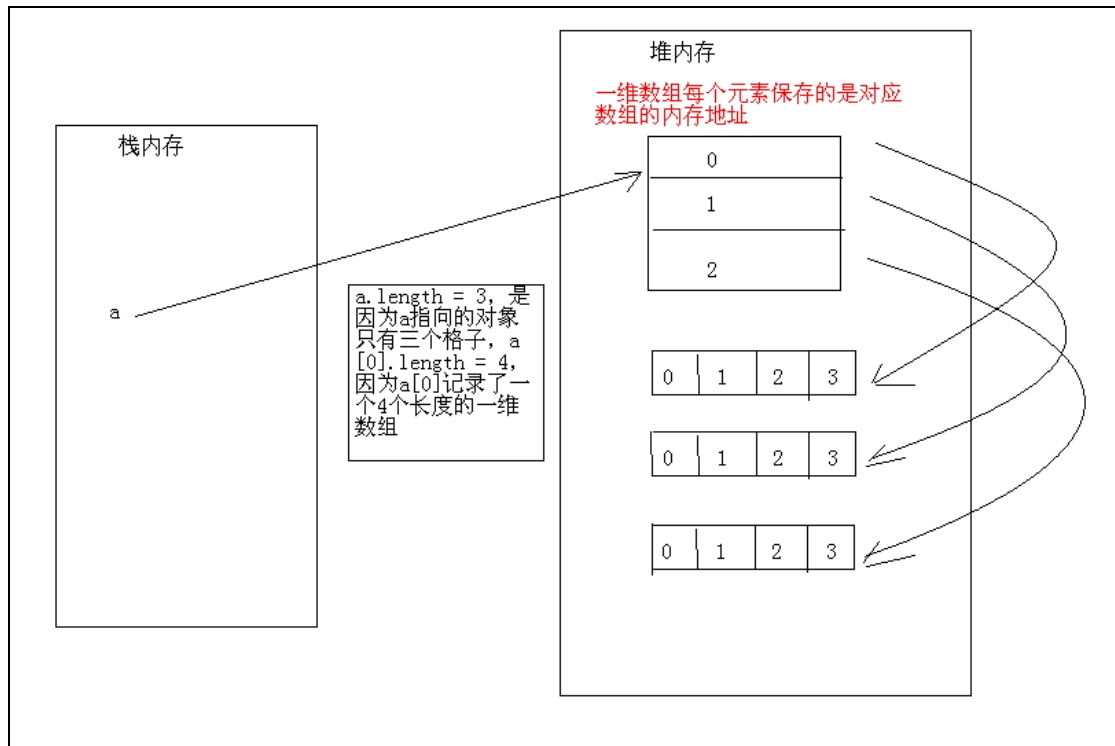
二维数组: 实质就是存储是一维数组。

数组定义:

数组类型[][] 数组名 = new 数组类型[一维数组的个数][每一个一维数组中元素的个数];

```
二维数组的定义:
int [][] a = new int[3][4];
System.out.println( a );           // [[I@de6ced
System.out.println( a.length );    // 获取二维数组中存储的一维数组的个数 3
System.out.println( a[0] );        // 获取的是二维数组中第一个一维数组 4
System.out.println( a[0].length ); // 4
System.out.println( Arrays.toString( a[0] ) );
```

疑问: 为什么 a.length = 3, a[0].length = 4?



数组的初始化:

静态初始化:

```
int [][] a = new int[][] { { 12,34,45,89},{34,56,78,10},{1,3,6,4} };
```

动态初始化:

```
int [][] a = new int[3][4];          // 3*4=12

for ( int i = 0; i < a.length ; i++ ) // 1
{
    for ( int j = 0; j < a[i].length ; j++ )
    {
        a[i][j] = ++value;          // a[1][0] = 4
    }
}
```

二维数组常见的操作:

1. 遍历二维数组
2. 对二维数组求和

```
class Demo3
{
    // 定义一个遍历二维数组的功能函数
    public static void printArr2( int [][] a ){
        // 1. 拆开二维数组
        for ( int i = 0 ; i < a.length ; i++ )
        {
            // 2. 拆开一维数组获取数据
            for ( int j = 0 ; j < a[i].length ; j++ )
            {
                System.out.print( a[i][j]+" , " );
            }
        }

    }

    // 定义一个函数计算二维数组中的元素的累加和
    public static long getSum( int [][] a ){
        // 0. 定义一个结果变量
        long sum = 0L;
        // 1. 拆开二维数组
        for ( int i = 0 ; i < a.length ; i++ )
        {
            // 2. 拆开一维数组获取数据
            for ( int j = 0 ; j < a[i].length ; j++ )
            {
                sum+=a[i][j];
            }
        }
        return sum;
    }

    // 统计二维数组中元素的个数
    public static int getDataCount( int [][] a ){
        // 0. 记录元素个数
        int count = 0;
        // 1. 拆开二维数组
        for ( int i = 0 ; i < a.length ; i++ )
        {
            // 2. 拆开一维数组获取数据
            for ( int j = 0 ; j < a[i].length ; j++ )
            {
                count++;
            }
        }
    }
}
```

```
        return count;
    }

    public static void main(String[] args)
    {

        int [][] a = new int[][]{ {23,4,5},{2},{4,5,78,56,90} };
        printArr2( a );
        System.out.println();
        System.out.println("累加和是: "+getSum( a ) );
        System.out.println("统计元素个数: "+getDataCount( a ) );
        System.out.println("Hello World!");

    }
}
```

3. 作业

1. 如何理解函数（方法）？
2. 怎么定义一个函数？
3. 函数重载什么时候使用？
4. 数组是什么和数组的特点是什么？
5. java 的内存(栈和堆)的特点？
6. 定义一个长度为 10 的 int 数组,统计数组的最大值、最小值、奇数和偶数的个数。