

非贷款,0元入学,不1万就业不给1分钱学费,我们已干四年了!

一、Java语言基础组成-Part 4 1.9 数组

笔记总链接:http://bbs.itheima.com/thread-200600-1-1.html

1.9.4 数组操作常见操作

对数组操作最基本的动作就是存和取。 核心思想:就是对角标的操作。

示例:遍历并打印数组元素

class ArrayDemo{ 01.

public static void main(String[] args) { 02. $int[] arr = {89,34,270,17};$ 03. 04. 05. for(int x = 0; x < arr.length; x += 1){ System.out.println("arr[" + x + "] = " + arr[x] + ";"); 06. } 07. } 08. } 09.

复制代码 运行结果: - - X 📷 管理员: C:\Windows\system32\cmd.exe D:∖code∖day05>javac ArrayDemo.java

Ξ D:∖code∖day05>java ArrayDemo arr[0] = 89; arr[1] = 34;arr[2] = 270;arr[3] = 17;

常见操作一:获取最值(最大值,最小值) 思路:

1、需要进行比较,并定义变量记录住每次比较后较大的值。 2、对数组中的元素进行遍历取出,和变量中记录的元素进行比较。

如果遍历到的元素大于变量中记录的元素,就用变量该记录住大的值。 3、遍历结果,该变量记录就是最大值。

两个明确: 明确一:结果。是数组中的元素:int类型。 明确二:未知内容。数组。

示例1:通过定义变量记录较大的值的方式实现。 class ArrayDemo{ 01. public static void main(String[] args) { 02. int[] arr= {89,34,-270,17,3,100}; 03. int max = getMax(arr); 04.

System.out.println("max = " + max); 05. } 06. 07. public static int getMax(int[] arr){ 08. int maxElement = arr[0]; 09. for(int x = 1; x < arr.length; x++){ 10. if(arr[x] > maxElement) 11. maxElement = arr[x]; 12. 13. }

return maxElement;

public static int getMax(int[] arr){

for(int x = 1; x < arr.length; x++){

int maxIndex = 0;

常见操作二:排序(选择排序,冒泡排序)

3、依次类推,直到最后一个元素。

选择排序

思路:

素,就互换内容。

第二小的元素。

代码:

18. 19.

20. 21.

22. 23.

24.

25.

26.

27.

28.

29.

30. 31.

32.

提高。

代码:

01. 02.

03.

04.

05.

06.

运行结果:

冒泡排序

思路:

示例:

import java.util.Arrays;

public static void main(String[] args) {

printArray(arr); Arrays.sort(arr);

printArray(arr);

int[] arr= {89,34,-270,17,3,100};

System.out.print("排序前数组:");

System.out.print("排序后数组:");

public static void printArray(int[] arr){

public static void main(String[] args) {

int[] arr= {4,1,5,7,8,4,2}; int index = getIndex(arr,2);

System.out.println("index = " + index);

public static int getIndex(int[] arr, int key){

for(int x = 0; x < arr.length; x++){

if(arr[x] == key)

return -1;

return x;

for(int x = 0; x < arr.length; x++){

if(x != arr.length - 1)

System.out.print(arr[x] + ",");

System.out.println(arr[x] + "]");

- - X

System.out.print("[");

else

}

画 管理员: C:\Windows\system32\cmd.exe

常见操作三:折半查找(二分查找)

class ArrayDemo{

}

示例:简单遍历查找方式

01.

02. 03.

04.

05. 06.

07.

08.

09.

10. 11.

12.

13. 14. 15.

}

代码1:

代码2:

01.

02.

03.

04.

05.

06. 07.

08.

09.

10.

11. 12.

13.

14. 15.

16. 17.

18.

19. 20.

21. 22.

23.

24.

25.

}

复制代码

}

}

}

D:\code\day05>javac ArrayDemo.java

前数组:[89,34,-270,17,3,100]

class ArrayDemo{

}

}

01.

02.

03.

04.

05.

06. 07.

08.

09. 10.

11.

12.

13.

14.

15.

16. 17.

18.

19.

20. 21.

22.

}

运行结果:

复制代码

■ 管理员: C:\Windows\system32\cmd.exe

D:\code\day05>java ArrayDemo

素大于右边的元素,那么两个元素就互换。

的元素,那么两个元素就互换。

2、经过第一轮比较,最大的元素就已经存储到数组最右边的结点中了。

4、依照此方式,一直到只有第一和第二个元素互相比较而结束。

D:∖code∖day05>javac ArrayDemo.java

排序前数组: [89,34,-270,17,3,100] 排序后数组: [-270,3,17,34,89,100]

class ArrayDemo{

}

14. 15.

16.

07.

08.

09.

10.

}

复制代码

运行结果: 画 管理员: C:\Windows\system32\cmd.exe D:\code\day05>javac ArrayDemo.java Ξ D:∖code∖day05>java ArrayDemo max = 100示例2:通过定义变量记录较大的值的索引方式实现。 class ArrayDemo{ 01. public static void main(String[] args) { 02. int[] arr= {89,34,-270,17,3,100}; 03. int max = getMax(arr); 04. System.out.println("max = " + max); 05. 06.

if(arr[x] > arr[maxIndex]) 11. 12. maxIndex = x;13. return arr[maxIndex]; 14. 15. 16. 复制代码 运行结果: 画 管理员: C:\Windows\system32\cmd.exe D:\code\day05>javac ArrayDemo.java Ε D:∖code∖day05>java ArrayDemo max = 100

1、首先拿数组第一个元素依次与除其自身外的其他每个元素顺序比较,如果第一个元素大于剩下的某个元

2、经过第一轮比较之后,此时,第一个元素就是数组中最小的元素。然后再拿第二个元素与除第一个元素和

其自身的元素进行比较,如果第二个元素大于剩下的某个元素,就互换内容。此时,第二个元素就是数组中倒数

01. class ArrayDemo{ public static void main(String[] args) { 02. int[] arr= {89,34,-270,17,3,100}; 03. System.out.print("排序前数组:"); 04. printArray(arr); 05. selectSort(arr); 06. System.out.print("排序后数组:"); 07. 08. printArray(arr); } 09. 10. public static void selectSort(int[] arr){ 11. for(int x = 0; x < arr.length - 1; x++){ 12. for(int y = x + 1; y < arr.length; y++){ 13. $if(arr[x] > arr[y]){$ 14. 15. int temp = arr[x]; 16. arr[x] = arr[y];arr[y] = temp; 17.

}

public static void printArray(int[] arr){

for(int x = 0; x < arr.length; x++){

if(x != arr.length - 1)

System.out.print(arr[x] + ",");

System.out.println(arr[x] + "]");

System.out.print("[");

else

}

}

}

}

}

复制代码 运行结果: - 0 x 📷 管理员: C:\Windows\system32\cmd.exe D:\code\day05>javac ArrayDemo.java D:\code\day05>java ArrayDemo Ε 序前数组:[89,34,-270,17,3,100] 序后数组: [−270,3,17,34,89,100] P.S. 1、上面的selectSort方法之所以不用返回int数组的原因是因为:arr引用变量是对传入selectSort方法中作为 参数的数组的引用,selectSort方法执行完毕之后,我们依然可以通过arr引用变量操作传入的数组。所以,没有 必要再通过返回值获取。 2、上面的选择排序算法效率比较低,因为数组每一个元素与剩下的元素比较就是为了获得最小的元素并且与

之互换。例如:{89,34,-270,17,3,100}这个数组,第一轮就要互换4次才能使第一个元素存储的是这个数组中最

小的元素。如果是这样,那么更高效率的方式则是只需要通过两个变量,一个记录最小值,一个记录最小值所在

的角标即可。等当前元素与余下的所有元素比较完,直接互换,这样只需互换一次就能达到目标,效率自然就会

public static void main(String[] args) {

printArray(arr);

selectSort(arr);

int[] arr= {89,34,-270,17,3,100}; System.out.print("排序前数组:");

System.out.print("排序后数组:"); 07. printArray(arr); 08. 09. } 10. public static void selectSort(int[] arr){ 11. 12. for(int x = 0; x < arr.length - 1; x++){ int num = arr[x]; 13. int index = x; 14. for(int y = x + 1; y < arr.length; y++){ 15. if(num > arr[y]){ 16. num = arr[y];17. index = y;18. 19. } 20. //如果最小的就是自己,就没有必要执行swap操作 21. if(index != x)22. swap(arr,x,index); 23. } 24. 25. } 26. public static void swap(int[] arr, int a,int b){ 27. 28. int temp = arr[a]; 29. arr[a] = arr[b]; arr[b] = temp; 30. } 31. 32. public static void printArray(int[] arr){ 33. System.out.print("["); 34. 35. for(int x = 0; x < arr.length; x++){ if(x != arr.length - 1) 36. System.out.print(arr[x] + ","); 37. else 38. 39. System.out.println(arr[x] + "]"); } 40. } 41. 42. } 复制代码

_ 0 X

Ξ

代码: 01. class ArrayDemo{ public static void main(String[] args) { 02. int[] arr= {89,34,-270,17,3,100}; 03. System.out.print("排序前数组:"); 04. printArray(arr); 05. bubbleSort(arr); 06. System.out.print("排序后数组:"); 07. printArray(arr); 08. 09. } 10. public static void bubbleSort(int[] arr){ 11. for(int x = 0; x < arr.length - 1; x++){ 12. for(int y = 0; y < arr.length - 1 - x; y++){ 13. 14. $if(arr[y] > arr[y+1]){$ 15. int temp = arr[y]; arr[y] = arr[y+1];16. 17. arr[y+1] = temp;18. } } 19. } 20. } 21. 22. public static void printArray(int[] arr){ 23. 24. System.out.print("["); for(int x = 0; x < arr.length; x++){ 25. if(x != arr.length - 1) 26. System.out.print(arr[x] + ","); 27. 28. else System.out.println(arr[x] + "]"); 29. 30. 31. } 32. 复制代码 运行结果: - - X om 管理员: C:\Windows\system32\cmd.exe D:\code\day05>javac ArrayDemo.java D:\code\day05>java ArrayDemo 排序前数组: [89,34,—270,17,3,100] 后数组: [-270,3,17,34,89,100] 在真实开发中,是不可能让我们自己去写这些排序算法的,因为JDK中已经提供好了API可以直接供我们调 用。

1、首先在第一轮排序中,数组从第一个元素到倒数第二个元素依次与其右边的元素进行比较,如果左边的元

3、第二轮排序则是从第一个元素到倒数第三个元素依次与其右边的元素进行比较,如果左边的元素大于右边

复制代码 运行结果: 📷 管理员: C:\Windows\system32\cmd.exe D:∖code∖day05>javac ArrayDemo.java D:\code\day05>java ArrayDemo index = 6 P.S. 如果一个数组是无序的,那么可以通过简单遍历查找的方式查找到某个元素所在的角标。但是如果一个数组 是有序的,那么就可以通过一种更高效的方式达到相同的目的,也就是二分查找。 思路: 1、设置三个变量记录角标:min、max、mid。min初始值为0,max为数组最大角标,mid为 $(\max + \min)/2$ 2、查看mid角标的元素是否与待查找的值相等,如果相等,则直接返回角标值,程序终止执行。 3、如果待查找的值小于角标为mid的元素值,那么说明待查找的元素的位置可能在min与mid角标之间。设 置max = mid - 1 , mid = (max + min)/2 , 重复第1、2步的操作。

4、如果待查找的值大于角标为mid的元素值,那么说明待查找的元素的位置可能在mid与max角标之间。设

5、如果数组中不存在待查找的元素,那么按照如上流程,最终min角标值会大于max角标值,此时返回-1。

置min = mid + 1, mid = (max + min)/2, 重复第1、2步的操作。

class ArrayDemo{ 01. 02. public static void main(String[] args) { 03. int[] arr= {13,15,19,28,33,45,78,106}; int index = binarySearch(arr,78); 04. System.out.println("index = " + index); 05. 06. } 07. public static int binarySearch(int[] arr, int key){ 08. int max,min,mid; 09. 10. min = 0;max =arr. length - 1; 11. mid = (max + min)/2;12. 13. while(arr[mid] !=key){ 14. if(key > arr[mid]) 15. min = mid + 1;16. 17. else if (key < arr[mid])</pre> max = mid - 1;18. if(max < min)</pre> 19. return -1; 20. mid = (max + min)/2;21. 22. } 23. return mid; 24. } 25. } 复制代码 运行结果: om 管理员: C:\Windows\system32\cmd.exe D:\code\day05>javac ArrayDemo.java D:\code\day05>java ArrayDemo index = 6

class ArrayDemo{ public static void main(String[] args) { int[] arr= {13,15,19,28,33,45,78,106}; int index = binarySearch(arr,78); System.out.println("index = " + index); public static int binarySearch(int[] arr, int key){ int max,min,mid; min = 0;max = arr. length - 1; while(min <= max){</pre> mid = (max + min) >> 1;if(key > arr[mid]) min = mid + 1;else if (key < arr[mid])</pre> max = mid - 1;else return mid; return -1;

| ### ### ### ### ### ### ### ### ### ## | 05.06. | <pre>int[] arr= {13,15,19,28,33,45,78,106}; int index = binarySearch(arr,44); System.out.println("index = " + index); }</pre> |
|--|---|--|
| | 07.08.09.10.11.12. | <pre>public static int binarySearch(int[] arr, int key){ int max,min,mid; min = 0;</pre> |
| ### 1000 ### | 13. 14. 15. 16. 17. | <pre>mid = (max + min) >> 1; if(key > arr[mid]) min = mid + 1;</pre> |
| ### 1995 Contraction processes and the contract of the contrac | 19. 20. 21. 22. | <pre>max = mid - 1; else return mid; } return min;</pre> |
| ### 1990 Proceedings Procedure Proce | 25. } | |
| ### 1000 | D: \code \da | ay05>javac ArrayDemo.java |
| Control Contro | 说明:由上面 | 的结果可以看到,如果要向数组{13,15,19,28,33,45,78,106}中插入值为44的元素,那么应该 |
| Compared the compared Compared the compared to the compare | 在实际开发中 | ¹ ,二分查找也不需要我们自己去写,JDK中已经提供了相关的API供调用。 |
| ### ### ### ### ### ### ### ### ### ## | 01. impor | |
| The Continue of the Continue o | 05. 06. 07. 08. | <pre>int[] arr= {13,15,19,28,33,45,78,106}; //如果存在返回的具体的角标位置,不存在返回的是-插入点-1 int index = Arrays.binarySearch(arr,44); System.out.println("index = " + index);</pre> |
| ### 1 | 10. } 复制代 | |
| ### Comparison of the Comparis | D: \code \da D: \code \da | ay05>javac ArrayDemo.java |
| ### 1 | | |
| ### 1 | public static int b 使用二分搜索 进行排序,则 参数: a - 要排 | int key) (法来搜索指定的 int 型数组,以获得指定的值。必须在进行此调用之前对数组进行排序(通过 gost(int[]) 方法)。如果没有对数组则结果是不确定的。如果数组包含多个带有指定值的元素,则无法保证找到的是哪一个。 (搜索的数组 |
| 1. (日本の大学の場合を含まって、 | 返回 , 如果它包 索引,5 | 包含在数组中,则返回搜索键的索引,否则返回(-(<i>嫌人点</i>) - 1)。 <i>插入点</i> 被定义为将键插入数组的那一点,即第一个大于此键的元素如果数组中的所有元素都小于指定的键,则为 a. length。注意,这保证了当且仅当此键被找到时,返回的值将 >= 0。 |
| | 1、什么时候的 | |
| 1. 特別的の場所の名前(日本教授の、1. 10世紀の名前) 2. 他 下午の時間の表面である。 1. 10世紀の名前 1. 他 下午の時間の表面の表面である。 1. 10世紀の表面の表面の表面の表面の表面の表面の表面の表面の表面の表面の表面の表面の表面の | 3、将这些数 | 据存储到数组中,根据运算的结果作为角标直接去查数组中对应的元素即可,这种方式称为查 |
| (中のなりますの場合は、第一に関手を対象が1分割である。 | 1、首先判断的 。 | |
| ### ### ### ### ### ### ### ### ### ## | 再将60无 3、由上面的f | 符号右移4位,再与15进行与操作,其值就是60的十六进制的倒数第二位。 例子可以总结出,将一个十进制数转换为十六进制的步骤就是: |
| ### 1 | 再右移4位 4、进而可以持 | 立,与15相与…直到相与结果为0为止。 |
| | 10进制转 | |
| ### 1 | 01. impor 02. 03. class 04. | <pre>ArrayDemo{ public static void main(String[] args) {</pre> |
| public static void coint(not may) | 05. 06. 07. 08. | toHex(60); toBin(-6); } //十进制>二进制 |
| ### trans(rum,15,4); ### public static void tootal(int rum)(| 10. 11. 12. 13. | <pre>public static void toBin(int num){ trans(num,1,1); } //十进制>十六进制</pre> |
| ### Common Control of the Control o | 16. 17. 18. 19. | trans(num,15,4); } //十进制>八进制 |
| 25. | 21. 22. 23. 24. | trans(num,7,3); } //进制转换的通用方法 public static void trans(int num, int base,int offset){ |
| ### Comparison of the compari | 26. 27. 28. 29. | <pre>if(num == 0){ System.out.println("0"); return; } char[] chs = {'0','1','2','3','4','5','6','7','8','9','A','B</pre> |
| ### A continue of the contin | 31. 32. 33. 34. | <pre>,'D' ,'E' ,'F' }; char[] arr = new char[32]; int pos = arr.length ; while(num != 0){</pre> |
| 41. for(int x = pos; x < arc.length; x++)(42. System.out.print(arr(x)); 43. } 44. System.out.println()); 44. } 46. } 46. } 46. } Erista: | 36. 37. 38. 39. | <pre>arr[pos] = chs[temp]; num = num >>> offset; } System.out.println("pos = " + pos);</pre> |
| を行動果: ② 世界 CAWindowskystem32cmdese Discode Valy(5) jova firesystems jova Discode Valy(5) jova firesystems jova Discode Valy(5) jova firesystems Discode Valy(5) jova fir | 41. 42. 43. 44. | <pre>for(int x = pos; x < arr.length; x++){ System.out.print(arr[x]); } System.out.println();</pre> |
| Divodo NagoSjava 和resplano Divodo NagoSjava Mresplano Divodo NagoSjava 和resplano Divodo NagoSjava Aresplano | 46. } 复制代 | |
| 在真实开发中,进制转换也不需要转机写,JDK已经通过API的方式提供给我们,直接调用即可, 示例: 1. class ArrayDemo { 2. | D: \code \da D: \code \da pos = 30 3C pos = 0 | ay05>javac ArrayDemo.java ay05>java ArrayDemo |
| ### String weeks = { "", "是期 " , "是 | 在真实开发中 | |
| 84.) | 01. class 02. | <pre>public static void main(String[] args) { System.out.println(Integer.toHexString(60));</pre> |
| ではないとは、 | 04. 05. } 复制代 | } |
| ### To a class ArrayDemo { ### To a class Array | 丽 管理员: C D: \code \d D: \code \d | ay05>javac ArrayDemo.java |
| public static void main(String[] args) { String week = getWeek(4); System.out.println(week); public static String getWeek(int num){ if (num > 7 num < 1){ return "错误的星期"; } String[] weeks = { "","星期一","星期二","星期三","星期四","星期五","星期日"}; return weeks[num]; return weeks[num]; } **Effshh **ETTT** | 3c 示例(查表法 | ξ): |
| public static String getWeek(int num) { 188. | 02. 03. 04. 05. | <pre>public static void main(String[] args) { String week = getWeek(4); System.out.println(week); }</pre> |
| ### ### ### ### ### ### ### ### #### #### | 07. 08. 09. 10. | if(num > 7 num < 1){ return "错误的星期"; } String[] weeks = { "","星期一","星期二","星期三","星期四","星期五" |
| 运行结果: Table C:\Windows\system32\cmd.exe D:\code\day05>javac ArrayDemo.java D:\code\day05>javac ArrayDemo END~ | 12. 13. 14. | return weeks[num]; |
| | D: \code \d | ay05>javac ArrayDemo.java |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | 配管理员: Ca D: \code \da D: \code \da 星期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | 配管理员: Ca D: \code \da D: \code \da 星期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | 配管理员: Ca D: \code \da D: \code \da 星期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | m 管理员: Ca D: \code \da D: \code \da 星期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | m 管理员: Ca D: \code \da D: \code \da 星期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | m 管理员: Ca D: \code \da D: \code \da 星期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | m 管理员: Ca D: \code \da D: \code \da 星期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | m 管理员: Ca D: \code \da D: \code \da 星期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | m 管理员: Ca D: \code \da D: \code \da 星期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | m 管理员: Ca D: \code \da D: \code \da 星期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | m 管理员: Ca D: \code \da D: \code \da 星期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | m 管理员: Ca D: \code \da D: \code \da 星期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | m 管理员: Ca D: \code \da D: \code \da 星期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | m 管理员: Ca D: \code \da D: \code \da 星期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | 配管理员: Ca D: \code \da D: \code \da 星期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dagger D: \code \dagger E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \dag D: \code \dag E 期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |
| | D: \code \da D: \code \da 星期四 | ay05>javac ArrayDemo.java ay05>java ArrayDemo ~END~ |

运行结果:

P.S.

角标如何获取?

代码:

画 管理员: C:∖Windows\system32\cmd.exe

D:\code\day05>java ArrayDemo index = 6

D:\code\day05>javac ArrayDemo.java

比min大的角标所在的数组元素全部往后顺延一个位置。

给定一个有序的数组,如果往该数组中存储一个元素,并保证这个数组还是有序的,那么这个元素的存储的

可以先通过二分查找,返回min的值,然后将待插入元素存在角标为min的数组位置,数组中角标为min以及

- - X