### JAVA第一阶段—DAY05-JAVA 答案

1. 简答题：简述一维数组的定义语法。

**答案：**

数据类型[] 数组名字 = new 数据类型[长度];

数据类型[] 数组名 = new 数据类型[]{元素1,元素2,元素3...};

数据类型[] 数组名 = {元素1,元素2,元素3...};

1. 简答题：简述二维数组的定义语法。

**答案：**

数据类型[][] 数组名字 = new 数据类型[长度][长度];

数据类型[][] 数组名字 = new 数据类型[长度][];

数据类型[][] 数组名字 = new 数据类型[][]{{元素1},{元素2}...};

数据类型[][] 数组名字 = {{元素1},{元素2}...};

1. 简答题：简述不同类型数组默认初始化值。

**答案：byte、short、int、long类型数组初始化值为0；float、double类型数组初始化值为0.0；boolean类型数组初始化值为false；char类型数组默认初始化值为初始值为'\u0000'，表现为空字符''；引用数据类型数组默认初始化值为null。**

1. 简答题：简述Java数组内存分配管理情况。

**答案：Java 程序在运行时，需要在内存中分配空间。为了提高运算效率，就对空间进行了不同区域的划分，因为每一片区域都有特定的处理数据和内存管理方式。Java的new出来的数组对象存在堆内存中，而引用变量存在本地方法栈中。**

1. 简答题：定义一个数组并使用多种方式进行遍历。

**答案：**

public static void main(String[] args) {

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

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

System.out.println(arr[i]);

}

}

public static void main(String[] args) {

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

for (int e : arr) {

System.out.println(e);

}

}

1. 编程题：定义一个int类型数组，获取数组的最小值。

**答案：**

public static void main(String[] args) {

int[] arr = { 5, 15, 2000, 10000, 100, 4000 };

//定义变量，保存数组中0索引的元素

int min = arr[0];

//遍历数组，取出每个元素

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

//遍历到的元素和变量max比较

//如果数组元素大于max

if (arr[i] < min) {

//max记录住大值

min = arr[i];

}

}

System.out.println("数组最小值是： " + min);

}

1. 编程题：完成数组的冒泡排序和选择排序。

**答案：**

// 冒泡排序

public static void main(String[] args) {

// 定义一个元素顺序杂乱的数组

int[] arr = { 4, 1, 3, 6, 2, 5};

// 外层循环表示比较轮数

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

// 内层循环对比相邻两个数

for (int j = 0; j < arr.length-i; j++) {

// 定义一个临时变量

int tempMax=0;

// 进行元素位置交换

if(arr[j]>arr[j+1]) {

tempMax=arr[j];

arr[j]=arr[j+1];

arr[j+1]=tempMax;

}

}

}

// 打印

System.out.println(Arrays.toString(arr));

}

// 选择排序

public static void main(String[] args) {

// 定义一个元素顺序杂乱的数组

int[] arr = { 4, 1, 3, 6, 2, 5};

// 外层循环循环每一个元素，拿着与内存循环的每个元素对比

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

// 内层循环对比相邻两个数

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

// 定义一个临时变量

int temp=0;

// 进行元素位置交换

if(arr[i] > arr[j]) {

temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

}

// 打印

System.out.println(Arrays.toString(arr));

}

1. 编程题：实现数组元素的二分查找。

**答案：**

public static void main(String[] args) {

// 定义一个元素顺序有序的数组

int[] arr = { 1, 2, 3, 4, 5, 6};

// 定义一个要查找的元素

int a=61;

// 初始化查找的起始位置、终止位置和中间位置

int begin=0;

int end=arr.length;

int mid=(begin+end)/2;

// 确保不会出现重复查找，越界

while(begin <=end) {

if(a> arr[mid]) {

begin=mid+1;

mid=(begin+end)/2;

}else if(a< arr[mid]) {

end=mid-1;

mid=(begin+end)/2;

}else {

System.out.println("元素位置为："+mid);

break;

}

// 如果起始角标已经超过终止角标还没找到，那就是数组没有这个原生

if(end<=begin) {

System.out.println("没有该元素");

break;

}

}

}

1. 编程题：实现数组元素的反转。

**答案：**

public static void main(String[] args) {

// 定义一个元素顺序杂乱的数组

int[] arr = { 4, 1, 3, 6, 2, 5};

// 对半遍历数组即可

for (int i = 0; i < arr.length/2; i++) {

// 首尾元素进行对换

int temp=arr[i];

arr[i]=arr[arr.length-1-i];

arr[arr.length-1-i]=temp;

}

// 打印

System.out.println(Arrays.toString(arr));

}

1. 编程题：实现2个同类型的数组合并。

**答案：**

public static void main(String[] args) {

// 定义2个同样类型的数组

int[] arr= {1,2};

int[] arr2= {3,4,5};

// 定义一个新数组，并指定长度

int[] newArr=new int[arr.length+arr2.length];

for (int i = 0; i < newArr.length; i++) {

if(i<arr.length) {

newArr[i]=arr[i];

}else {

newArr[i]=arr2[i-arr.length];

}

}

// 打印

System.out.println(Arrays.toString(newArr));

}