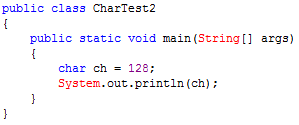
1. char类型占几字节？为什么下面的程序128及之后的数运行之后都是“？”



**解释：**可能是因为java中的英文字符与中文字符是不连续的，也可能是因为java字库的

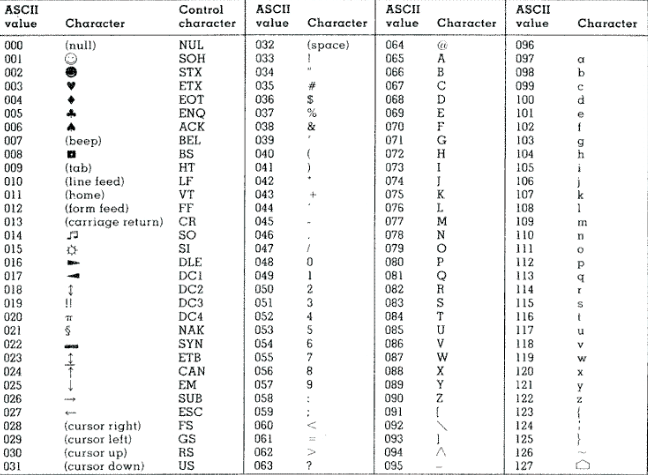
原因，128到19968之间的数所代表的字符，不是英文字符也不是中文字符，所以在

中国境内不显示。

如：Unicode码从0~127表示英文字符；而中文字符是从19968到40869。共收录

22901个中文字符。19968代表中文 ‘一’ ，40869代表中文字符 ‘龥’ 。





1. 这种交换两个数的方法，不用定义第三个变量来存储数据，那么将a+b取出来的时候，

存放在哪里？

int a=10;

int b=20;

a=a+b; //这种方法不用定义第三个变量就能实现两个

b=a-b; //的交换

a=a-b;

1. 为什么从键盘输入的空格不能计算，而且空格之后的字符也不能计算，而直接定义好的空格就可以？

举例：Scanner input = new Scanner(System.in);

String str = input.next();

Char[] arr = str.toCharArray();//将输入的字符串转换成字符数组

System.out.println(Arrays.toSring())//将数组中的内容打印出来

//如果从键盘输入“1234567890qwe ” 那么打印出来的是什么呢？

//打印结果：1234567890qwe 后面的空格不会打印出来。

//如果从键盘输入“1234567890 qwe” 那么打印出来的结果是什么呢？

//打印结果：1234567890 后面的空格以及qwe都不打印。

举例：String str1 = “1234567890qwe ”

String str2 = “1234567890 qwe”

System.out.println(Arrays.toString(str1));

//打印结果为：“1234567890qwe ” 注意：最后的空格也打印出来了

System.out.println(Arrays.toString(str2));

//打印结果为：“1234567890 qwe” 中间的空格也打印出来了。

4、在下满的例子中，DD继承了CC的fun（）方法与print（）方法，且DD中的print（）方法不算重写，那么DD中是不是有两个print（）方法呢？这样不是两个一样的方法？为什么不会报错？

|  |
| --- |
| **public** **class** ChongXieTest2  {  **public** **static** **void** main(String[] args)  { DD dd = **new** DD();  dd.fun(); //输出结果： Oracle java  dd.print(); //输出结果： PHP  }  }  **class** CC  {  **public** **void** fun()  { System.***out***.println("Oracle");  print();  }  **private** **void** print()  { System.***out***.println("java"); }  }  **class** DD **extends** CC  {  **public** **void** print()  {  System.***out***.println("PHP");  }  } |

1. 子类继承父类，是复制一份父类对象内容？还是引用父类对象？

如果直接复制一份，为什么不能直接使用私有属性和方法？

如果是引用父类对象，为什么能够直接使用非私有属性和方法，而不需要引用？

6、

|  |
| --- |
| import java.util.ArrayList;  import java.util.Iterator;  class ArrayListDemo5  {  public static void main(String[] args)  {  ArrayList<String> al = new ArrayList<String>();  al.add("hello");  al.add("world");  al.add("java");  al.add("C");  al.add("C++");  Iterator<String> it = al.iterator();  while(it.hasNext())  {  String str = it.next();  System.out.println(str);  }  /\*  问题：Iterator<String> it = al.iterator();  Iterator是接口，iterator()方法返回的是“返回按适当顺序在列表的元素上进行迭代的迭代器”。  因为Iterator是接口不能返回，那么难道是说：al.iterator()方法返回的是实现了Iterator接口的迭代器？  \*/  }  } |

问题：Iterator<String> it = al.iterator();

Iterator是接口，iterator()方法返回的是“返回按适当顺序在列表的元素上进行

迭代的迭代器”。

因为Iterator是接口不能返回，那么难道是说：al.iterator()方法返回的是实现了

Iterator接口的迭代器？

解答：“返回按适当顺序在列表的元素上进行迭代的迭代器”返回的是实现Iterator接口的具体实现类。

7.Scanner问题

|  |
| --- |
| 问题描述： |
| **public** **static** **void** main(String[] args)  {  Scanner sc = **new** Scanner(System.*in*);  **int** sum = 0;  **int** count = 0;  **while**(sc.hasNextInt())  {  count++;  // int ob= sc.nextInt(); 如果nextInt()方法不写在这里，而是写在if条件句中，那么程序就会进入死循环，因为当输入的数据大于5之后，不满足if条件句，他就不会执行里面的代码，那么你再次输入的数据就会没有变量接收，程序就会死循环了。  **if**(count<=5)  {  // sum += ob;  sum += sc.nextInt();  }  **else**  {  sum += 0;  }    }  System.*out*.println(sum\*1.0/5);  } |

1. Scanner问题：

|  |
| --- |
| 问题描述：next()与nextLine()的区别 |
| **public** **static** **void** main(String[] args)  {  Scanner in = **new** Scanner(System.*in*);  String str1 = in.next();  System.*out*.println(str1);  } 解析：next()一定要读取到有效字符后才可以结束输入，对输入有效字符之前遇到的空格键、Tab键或Enter键等结束符，next()方法会自动将其去掉，只有在输入有效字符之后，next()方法才将其后输入的空格键、Tab键或Enter键等视为分隔符或结束符。   也就是说在遇到hello这个有效字符之前，若是遇到空格、tab键、enter键等结束键的话，next()方法会自动将其去掉，只留下输入的有效字符；而在hello这个有效字符之后的遇到的空格、Tab键、enter键，next()方法都会将其视作结束键，从而结束本次输入。所以最后的输入结果hello前后都是不带空格、Tab键、enter键的。 |
| **public** **static** **void** main(String[] args)  {  Scanner in = **new** Scanner(System.*in*);  String str1 = in.nextLine();  System.*out*.println(str1);  } 解析：nextLine()方法得到的是一行的数据，包括空格、Tab键，nextLine()反方的结束符只是Enter键，即nextLine()方法返回的是Enter键之前的所有字符，它是可以得到带空格的字符串的。   如果在heppl之前输入enter键的话，整个程序就会结束，而如果在heppl之前遇到空格、Tab键，以及在aa之后遇到的空格、Tab键，都会完整的输出。 |

1. IO流问题：

|  |
| --- |
| 问题描述：在将文件复制到另一个目录下的时候，注意产生死循环（以下操作为死循环，不要尝试） |
| **public** **static** **void** main(String[] args)  {  File file = **new** File("d:\\KuGou\\Groove Coverage - God is a Girl.mp3");  File target = **new** File("e:\\codeTest\\God is a girl.mp3");  **try**  {  FileInputStream fis = **new** FileInputStream(file);  FileOutputStream fos = **new** FileOutputStream(target);  **long** time1 = System.*currentTimeMillis*();  **byte**[] buf = **new** **byte**[1024\*1024]; //声明一个1M大小的字节数组 ,用于存放输入流中的字节，这就相当于是一个缓存区了  //public int read(byte[] b) throws IOException 从输入流中读取一定数量的字节，并将其存储在缓冲区数组 b 中。以整数形式返回实际读取的字节数    **int** len = fis.read(buf); //内存读取缓存区中的数据，返回字节的长度  **while**(len!=-1)  {  fos.write(buf,0,len);  }  /\*  \* 总结：上面会形成一个死循环，从而不断的复制数据。  \* 解释：FileInputStream会将源文件中的数据读取出来，存放在缓存区byte[1024\*1024]数组中，此时缓存区中能够缓存的数据最多只有1024\*1024byte，即1M。  \* 如果源文件file中数据的大小大于1M，那么输入流就无法达到源文件file的末尾，所以fis.red(buf)!=-1恒成立，  \* 那么此时执行while中的判断“len!=-1”也是恒成立的，那么这就是死循环了。  \* 由此造成的后果就是会将源文件1M大小的数据不断复制追加进目标文件target中，这样target文件的大小就会一直变大，直到系统死机。  \* 【注意：因为缓存区中的缓存只有1M，所以只能读取源文件file中1M大小的数据，而后死循环，将这1M大小的数据不断追加进target文件中。】  \*/  fis.close();  fos.close();  **long** time2 = System.*currentTimeMillis*();  System.*out*.println("使用缓存需要的时间是："+(time2-time1)+"毫秒");  System.*out*.println("文件复制成功！");  } **catch** (Exception e)  {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  } |
| 以下是正确的代码：（有两种方法） |
| 方法一：将缓存区的大小设置为大于等于file文件的大小，让得fis.read = 1 能够成立 |
| **public** **static** **void** main(String[] args)  {  File file = **new** File("d:\\KuGou\\Groove Coverage - God is a Girl.mp3");  File target = **new** File("e:\\codeTest\\God is a girl.mp3");  **try**  {  FileInputStream fis = **new** FileInputStream(file);  FileOutputStream fos = **new** FileOutputStream(target);  **long** time1 = System.*currentTimeMillis*();  **byte**[] buf = **new** **byte**[(**int**)file.length()];  **int** len = fis.read(buf); //内存读取缓存区中的数据，返回字节的长度  fos.write(buf,0,len);  fis.close();  fos.close();  **long** time2 = System.*currentTimeMillis*();  System.*out*.println("使用缓存需要的时间是："+(time2-time1)+"毫秒");  System.*out*.println("文件复制成功！");  } **catch** (Exception e)  {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  } |
| 方法二： |
| **public** **static** **void** main(String[] args)  {  File file = **new** File("d:\\KuGou\\Groove Coverage - God is a Girl.mp3");  File target = **new** File("e:\\codeTest\\God is a girl.mp3");  **try**  {  FileInputStream fis = **new** FileInputStream(file);  FileOutputStream fos = **new** FileOutputStream(target);  **long** time1 = System.*currentTimeMillis*();  **byte**[] buf = **new** **byte**[1024\*1024];  **int** len = -1; //首先判断  **while**((len=fis.read(buf))!=-1)  {  fos.write(buf,0,len);  }  fis.close();  fos.close();  **long** time2 = System.*currentTimeMillis*();  System.*out*.println("使用缓存需要的时间是："+(time2-time1)+"毫秒");  System.*out*.println("文件复制成功！");  } **catch** (Exception e)  {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  } |

1. Struts2框架问题一：

|  |
| --- |
| 若一个Action类可以应答多个action请求，每个action请求又是使用不同的 验证规则，怎么办？  》为每一个不同的action请求定义其对应的验证文件：  ActionClassName**-**别名**-**validation**.**xml  》不带别名的配置文件：ActionClassName-validation**.**xml中的验证规则依然会起作 用。可以把各个action共有的验证规则配置在其中，但需要注意的是，只适用于 某一个action请求的验证规则就不要配置在这里了。  【参考struts2-7/ validation.jsp、validation2.jsp、TestValidationAction.java、success.jsp、TestValidationAction-testValidation-validation.xml、TestValidationAction-testValidation2-validation.xml、】 |
| 问题：validation.jsp和validation2.jsp都对应TestValidationAction.java类,但是这两个JSP文件的验证规则又是不一样的，那么为何validation.jsp就能准确的找到TestValidationAction-testValidation-validation.xml文件，而validation2.jsp能准确的找到TestValidationAction-testValidation2-validation.xml文件？ |