Android笔试总结

1.请谈一下Android系统的架构。

答：Android系统采用了分层架构，从高层到低层分别是应用程序层、应用程序框架层、系统运行库层和linux核心层。

2.谈谈android大众常用的五种布局。

答：在Android中，共有五种布局方式，分别是：FrameLayout(框架布局)，LinearLayout (线性布局)，AbsoluteLayout(绝对布局)，RelativeLayout(相对布局)，TableLayout(表格布局)。

（1）FrameLayout 框架布局,放入其中的所有元素都被放置在最左上的区域，而且无法为这些元素指定一个确切的位置,下一个子元素会重叠覆盖上一个子元素，适合浏览单张图片。

（2）LinearLayout 线性布局,是应用程序中最常用的布局方式，主要提供控件水平或者垂直排列的模型，每个子组件都是以垂直或水平的方式来定位.(默认是垂直)

（3）AbsoluteLayout 绝对定位布局,采用坐标轴的方式定位组件，左上角是（0，0）点，往右x轴递增，往下Y轴递增,组件定位属性为android:layout\_x 和 android:layout\_y来确定坐标。

（4）RelativeLayout 相对布局,根据另外一个组件或是顶层父组件来确定下一个组件的位置。和CSS里面的类似。

（5）TableLayout 表格布局,类似Html里的Table.使用TableRow来布局，其中TableRow代表一行，TableRow的每一个视图组件代表一个单元格。

3.谈谈android数据存储方式。

答：Android提供了5种方式存储数据：

（1）使用SharedPreferences存储数据；它是Android提供的用来存储一些简单配置信息的一种机制，采用了XML格式将数据存储到设备中。只能在同一个包内使用，不能在不同的包之间使用。

（2）文件存储数据；文件存储方式是一种较常用的方法，在Android中读取/写入文件的方法，与Java中实现I/O的程序是完全一样的，提供了openFileInput()和openFileOutput()方法来读取设备上的文件。

（3）SQLite数据库存储数据；SQLite是Android所带的一个标准的数据库，它支持SQL语句，它是一个轻量级的嵌入式数据库。

（4）使用ContentProvider存储数据；主要用于应用程序之间进行数据交换，从而能够让其他的应用保存或读取此Content Provider的各种数据类型。

（5）网络存储数据；通过网络上提供给我们的存储空间来上传(存储)和下载(获取)我们存储在网络空间中的数据信息。

4.Android中Activity, Intent, Content Provider, Service各有什么区别。

答：Activity： 活动，是最基本的android应用程序组件。一个活动就是一个单独的屏幕，每一个活动都被实现为一个独立的类，并且从活动基类继承而来。

Intent： 意图，描述应用想干什么。最重要的部分是动作和动作对应的数据。

Content Provider：内容提供器，android应用程序能够将它们的数据保存到文件、SQLite数据库中，甚至是任何有效的设备中。当你想将你的应用数据和其他应用共享时，内容提供器就可以发挥作用了。

Service：服务，具有一段较长生命周期且没有用户界面的程序。

5.View, surfaceView, GLSurfaceView有什么区别。

答：view是最基础的，必须在UI主线程内更新画面，速度较慢。

SurfaceView 是view的子类，类似使用双缓机制，在新的线程中更新画面所以刷新界面速度比view快

GLSurfaceView 是SurfaceView的子类，opengl 专用的

6.Adapter有什么作用？常见的Adapter有哪些？

答：Adapter是连接后端数据和前端显示的适配器接口。常见的Adapter有ArrayAdapter, BaseAdapter, CursorAdapter, HeaderViewListAdapter, ListAdapter, ResourceCursorAdapter, SimpleAdapter, SimpleCursorAdapter, SpinnerAdapter, WrapperListAdapter等

7.Manifest.xml文件中主要包括哪些信息？

答：manifest：根节点，描述了package中所有的内容。

uses-permission：请求你的package正常运作所需赋予的安全许可。

permission： 声明了安全许可来限制哪些程序能你package中的组件和功能。

instrumentation：声明了用来测试此package或其他package指令组件的代码。

application：包含package中application级别组件声明的根节点。

activity：Activity是用来与用户交互的主要工具。

receiver：IntentReceiver能使的application获得数据的改变或者发生的操作，即使它当前不在运行。

service：Service是能在后台运行任意时间的组件。

provider：ContentProvider是用来管理持久化数据并发布给其他应用程序使用的组件。

8.请写一段代码（SAX, DOM, 或者pull ）来解析XML文档。

答：下面是要解析的XML文件：

<?xml version="1.0" encoding="UTF-8"?>

<persons>

<person id="1">

<name>张三</name>

<age>22</age>

</person>

<person id="2">

<name>李四</name>

<age>23</age>

</person>

</persons>

复制代码

定义一个名为Person的javaBean用于存放上面解析出来的xml内容

publicclassPerson {

privateInteger id;

privateString name;

privateShort age;

publicInteger getId() {

returnid;

}

publicvoidsetId(Integer id) {

this.id =id;

}

publicString getName() {

returnname;

}

publicvoidsetName(String name) {

this.name =name;

}

publicShort getAge() {

returnage;

}

publicvoidsetAge(Short age) {

this.age =age;

}

}

（1）使用SAX读取XML文件；它采用的是事件驱动，并不需要解析完整个文档，速度快并且占用内存少。需要为SAX提供实现ContentHandler接口的类。

PersonDefaultHandler.java

Android有四大组件：Activity、Service、Broadcast Receiver、Content Provider

在这些组件之间的通讯中，主要是由Intent协助完成的。

Intent负责对应用中一次操作的动作、动作涉及数据、附加数据进行描述，Android则根据此Intent的描述，负责找到对应的组件，将 Intent传递给调用的组件，并完成组件的调用。

因此，Intent在这里起着一个媒体中介的作用，专门提供组件互相调用的相关信息，实现调用者与被调用者之间的解耦。

例如，在一个联系人维护的应用中，当我们在一个联系人列表屏幕(假设对应的Activity为listActivity)上，点击某个联系人后，希望能够跳出此联系人的详细信息屏幕(假设对应的Activity为detailActivity)

为了实现这个目的，listActivity需要构造一个 Intent，这个Intent用于告诉系统，我们要做“查看”动作，此动作对应的查看对象是“某联系人”，然后调用startActivity (Intent intent)，

将构造的Intent传入，系统会根据此Intent中的描述，到ManiFest中找到满足此Intent要求的Activity，系统会调用找到的 Activity，即为detailActivity，最终传入Intent，detailActivity则会根据此Intent中的描述，执行相应的操作。

android应用开发框架是 Application Framework. 其系统架构由5部分组成，分别是：Linux Kernel、Android Runtime、Libraries、Application Framework、Applications。第二部分将详细介绍这5个部分。下面自底向上分析各层。

Android架构

1、Linux Kernel

Android基于Linux 2.6提供核心系统服务，例如：安全、内存管理、进程管理、网络堆栈、驱动模型。Linux Kernel也作为硬件和软件之间的抽象层，它隐藏具体硬件细节而为上层提供统一的服务。 如果你学过计算机网络知道OSI/RM，就会知道分层的好处就是使用下层提供的服务而为上层提供统一的服务，屏蔽本层及以下层的差异，当本层及以下层发生了变化不会影响到上层。也就是说各层各尽其职，各层提供固定的SAP（Service Access Point），专业点可以说是高内聚、低耦合。 如果你只是做应用开发，就不需要深入了解Linux Kernel层。

2、Android Runtime

Android包含一个核心库的集合，提供大部分在Java编程语言核心类库中可用的功能。每一个Android应用程序是Dalvik虚拟机中的实例，运行在他们自己的进程中。Dalvik虚拟机设计成，在一个设备可以高效地运行多个虚拟机。Dalvik虚拟机可执行文件格式是.dex，dex格式是专为Dalvik设计的一种压缩格式，适合内存和处理器速度有限的系统。 大多数虚拟机包括JVM都是基于栈的，而Dalvik虚拟机则是基于寄存器的。两种架构各有优劣，一般而言，基于栈的机器需要更多指令，而基于寄存器的机器指令更大。dx 是一套工具，可以将 Java .class 转换成 .dex 格式。一个dex文件通常会有多个.class。由于dex有时必须进行最佳化，会使文件大小增加1-4倍，以ODEX结尾。 Dalvik虚拟机依赖于Linux 内核提供基本功能，如线程和底层内存管理。

3、Libraries

Android包含一个C/C++库的集合，供Android系统的各个组件使用。这些功能通过Android的应用程序框架（application framework）暴露给开发者。下面列出一些核心库： 系统C库--标准C系统库（libc）的BSD衍生，调整为基于嵌入式Linux设备 媒体库--基于PacketVideo的OpenCORE。这些库支持播放和录制许多流行的音频和视频格式，以及静态图像文件，包括MPEG4、 H.264、 MP3、 AAC、 AMR、JPG、 PNG 界面管理--管理访问显示子系统和无缝组合多个应用程序的二维和三维图形层 LibWebCore--新式的Web浏览器引擎,驱动Android 浏览器和内嵌的web视图 SGL--基本的2D图形引擎 3D库--基于OpenGL ES 1.0 APIs的实现。库使用硬件3D加速或包含高度优化的3D软件光栅 FreeType --位图和矢量字体渲染 SQLite --所有应用程序都可以使用的强大而轻量级的关系数据库引擎

4、Application Framework

通过提供开放的开发平台，Android使开发者能够编制极其丰富和新颖的应用程序。开发者可以自由地利用设备硬件优势、访问位置信息、运行后台服务、设置闹钟、向状态栏添加通知等等，很多很多。 开发者可以完全使用核心应用程序所使用的框架APIs。应用程序的体系结构旨在简化组件的重用，任何应用程序都能发布他的功能且任何其他应用程序可以使用这些功能（需要服从框架执行的安全限制）。这一机制允许用户替换组件。 所有的应用程序其实是一组服务和系统，包括： 视图（View）--丰富的、可扩展的视图集合，可用于构建一个应用程序。包括包括列表、网格、文本框、按钮，甚至是内嵌的网页浏览器 内容提供者（Content Providers）--使应用程序能访问其他应用程序（如通讯录）的数据，或共享自己的数据 资源管理器（Resource Manager）--提供访问非代码资源，如本地化字符串、图形和布局文件 通知管理器（Notification Manager）--使所有的应用程序能够在状态栏显示自定义警告 活动管理器（Activity Manager）--管理应用程序生命周期,提供通用的导航回退功能

5、Applications

Android装配一个核心应用程序集合，包括电子邮件客户端、SMS程序、日历、地图、浏览器、联系人和其他设置。所有应用程序都是用Java编程语言写的。更加丰富的应用程序有待我们去开发！ 从上面我们知道Android的架构是分层的，非常清晰，分工很明确。Android本身是一套软件堆迭(Software Stack)，或称为「软件迭层架构」，迭层主要分成三层：操作系统、中间件、应用程序。从上面我们也看到了开源的力量，一个个熟悉的开源软件在这里贡献了自己的一份力量。

1. 如果代码中出现了乱码，这是什么问题造成的，在你常用的开发工具中，你是怎么去解决这个问题的。

Java编程语言默认的编码方式是UNICODE，但Java编译器所使用的字符集则是操作系统默认的字符集，中文的Windows是GBK，英文系统则是ISO－8859-1。出现乱码的根本原因是由于被操作的中文字符（变量）的编码格式与目标的编码格式不同，所有这些问题都是发生在字符的读入和输出过程中的，在Java程序中最重要的是保证程序入口和出口的汉字信息不失真。

解决方法：选择ＪＤＫ的中文本地化版本；选择合适的编译参数；通过编程的方式实现字符编码的转换代码；定义字符输出集。

1. Java的内存回收机制

首先要明确几点：  
        第一:在垃圾回收器回收垃圾之前，我们先来了解一下Java分配对象的方式，Java的堆更像一个传送带，每分配一个新对象，它就往前移动一格。这意味着对象存储空间的分配速度相当快。Java的“堆指针”只是简单地移动到尚未分配的领域。也就是说，分配空间的时候，“堆指针”只管依次往前移动而不管后面的对象是否还要被释放掉。如果可用内存耗尽之前程序就退出就再好不过了，这样的话垃圾回收器压根就不会被激活。但是由于“堆指针”只管依次往前移动，那么你肯定会想，总有一  天内存会被耗尽，垃圾回收器就开始释放内存。这里有人肯定会问：怎么判断某个对象该被回收呢？答案就是当堆栈或静态存储区没有对这个对象的引用时，就表示程序（员）对这个对象没有兴趣了，它就应该被回收了。有两种方法来知道这个对象有没有被引用：第一种是遍历堆上的对象找引用；第二种是遍历堆栈或静态存储区的引用找对象。前者的实现叫做“引用计数法”，意思就是当有引用连接至对象时，引用计数加1，当引用离开作用域或被置为null时，引用计数减1，这种方法有个缺陷，如果对象之间存在循环引用，可能会出现“对象应该被回收，但引用计数却不为零”的情况。  
  
        第二:Java采用的是后者，在这种方式下，Java虚拟机采用一种“自适应”的垃圾回收技术，如何处理找到的存活对象（也就是说不是垃圾），  
        Java有两种方式：  
                一种是“停止-复制”：理论上是先暂停程序的运行（所以它不属于后台回收模式），然后将所有存活的对象从当前堆复制到另一个堆，没有被复制的全是垃圾。当对象被复制到新堆上时，它们是一个挨着一个的，所以新堆保持紧凑排列（这也是为什么分配对象的时候“堆指针”只管依次往前移动）。然后就可以按前述方法简单、直接地分配内存了。这将导致大量内存复制行为，内存分配是以较大的“块”为单位的。有了块之后，垃圾回收器就可以不往堆里拷贝对象了，直接就可以往废弃的块里拷贝对象了。  
  
         另一种是“标记-清扫”：它的思路同样是从堆栈和静态存储区出发，遍历所有的引用，进而找出所有存活的对象。每当它找到一个存活对象，就会给对象一个标 记。这个过程中不会回收任何对象。只有全部标记完成时，没有标记的对象将被释放，不会发生任何复制工作，所以剩下的堆空间是不连续的，然后垃圾回收器 重新整理剩余的对象，使它们是连续排列的。  
  
         当垃圾回收器第一次启动时，它执行的是“停止-复制”，因为这个时刻内存有太多的垃圾。然后Java虚拟机会进行监视，如果所有对象都很稳定，垃圾回收器的效率降低的话，就切换到“标记-清扫”方式；同样，Java虚拟机会跟踪“标记-清扫”效果，要是堆空间出现很多碎片，就会切换到“停止-复制”方式。这就是  所谓的“自适应”技术。其实仔细想一下，“停止-复制”和“标记-清扫”无非就是：“在大量的垃圾中找干净的东西和在大量干净的东西里找垃圾”。不同的 环境用不同的方式，这样做完全是为了提高效率，要知道，无论哪种方式，Java都会先暂停程序的运行，所以，垃圾回收器的效率其实是很低的。

1. 计算机组成原理的东西，高位和低位计算的东西
2. 宏内核和微内核的区别

老一点的操作系统都是宏内核的，也就是说，整个操作系统是一个运行在核心态的单独的a.out文件，这个二进制文件包含进程管理、内存管理、文件系统以及其他。具体实例包括UNIX、MS-DOS、 VMS、MVS、OS/360、MULTICS等。

另一种便是微内核，在这种系统中操作系统的大部分都运行在单独的进程，而且多数在内核之外。它们之间通过消息传递来通信。内核的任务是处理消息传递、中断处理、底层的进程管理，以及可能的I/O。这种设计的实例有RC4000、Amoeba、Chorus、Mach，以及还没有发布的Windows/NT。

宏内核看上去试图包办一切，而微内核恰恰相反，它的任务只是“处理消息传递、中断处理、底层的进程管理，以及可能的I/O”，而其他事情都交给内核之外单独的进程来完成。

宏内核和微内核最大的差别在于，微内核仅在内核中完成如今成调度之类操作系统的最基本的功能，而宏内核则是倾向于将尽可能多的功能放在内核中实现。大家可以看看大牛Tanenbaum和Linus关于微内核和宏内核孰优孰劣的两次争论，很有价值！

1. 结合进制间的转换，以及byte的取值范围进行判断代码的输出结果

java byte 类型的取值范围是-128～127

int n1 = 14;

//十进制转成十六进制：

Integer.toHexString(n1);

//十进制转成八进制

Integer.toOctalString(n1);

//十进制转成二进制

Integer.toBinaryString(12);

//十六进制转成十进制

Integer.valueOf("FFFF",16).toString();

//十六进制转成二进制

Integer.toBinaryString(Integer.valueOf("FFFF",16));

//十六进制转成八进制

Integer.toOctalString(Integer.valueOf("FFFF",16));

//八进制转成十进制

Integer.valueOf("576",8).toString();

//八进制转成二进制

Integer.toBinaryString(Integer.valueOf("23",8));

//八进制转成十六进制

Integer.toHexString(Integer.valueOf("23",8));

//二进制转十进制

Integer.valueOf("0101",2).toString();

//二进制转八进制

Integer.toOctalString(Integer.parseInt("0101", 2));

//二进制转十六进制

Integer.toHexString(Integer.parseInt("0101", 2));

1. 至少说出三个Android常用的开发框架

ThinkAndroid是一个免费的开源的、简易的、遵循Apache2开源协议发布的Android开发框架，其开发宗旨是简单、快速的进行 Android应用程序的开发，包含Android mvc、简易sqlite orm、ioc模块、封装Android httpclitent的http模块, 具有快速构建文件缓存功能，无需考虑缓存文件的格式，都可以非常轻松的实现缓存，它还基于文件缓存模块实现了图片缓存功能， 在android中加载的图片的时候，对oom的问题，和对加载图片错位的问题都轻易解决。

afinal是一个开源的android的orm和ioc应用开发框架，其特点是小巧灵活，代码入侵量少。

libGdx是一个跨平台的2D/3D的游戏开发框架，它由Java/C/C++语言编写而成。

1. 进程和线程间的区别（好像有这题）

从一定意义上讲，进程就是一个应用程序在处理机上的一次执行过程，它是一个动态的概念，而线程是进程中的一部分，进程包含多个线程在运行。

通常在一个进程中可以包含若干个线程，它们可以利用进程所拥有的资源。在引入线程的操作系统中，通常都是把进程作为分配资源的基本单位，而把线程作为独立运行和独立调度的基本单位。由于线程比进程更小，基本上不拥有系统资源，故对它的调度所付出的开销就会小得多，能更高效的提高系统内多个程序间并发执行的程度。

线程和进程的区别在于，子进程和父进程有不同的代码和数据空间，而多个线程则共享数据空间，每个线程有自己的执行堆栈和程序计数器为其执行上下文。多线程主要是为了节约CPU时间，发挥利用，根据具体情况而定。线程的运行中需要使用计算机的内存资源和CPU。

**线程与进程的区别归纳：**

**a.地址空间和其它资源**：进程间相互独立，同一进程的各线程间共享。某进程内的线程在其它进程不可见。

**b.通信：**进程间通信IPC，线程间可以直接读写进程数据段（如全局变量）来进行通信——需要进程同步和互斥手段的辅助，以保证数据的一致性。

**c.调度和切换**：线程上下文切换比进程上下文切换要快得多。

d.在多线程OS中，进程不是一个可执行的实体。

8）写一个计算器，语言不限

#include "iostream"

#include "string"

using namespace std;

//---------------------------------- Stack.h --------------------------

//定义Stack类

const maxsize=20;

enum Error\_code { success, overflow, underflow };

template <class T>

class Stack {

public:

Stack();

bool empty() const;

bool full() const;

int size() const;

void clear();

Error\_code top(T &item) const;

Error\_code pop();

Error\_code push(const T &item);

private:

int count;

T entry[maxsize];

};

template <class T>

Stack<T>::Stack() {

count=0;

}

template <class T>

bool Stack<T>::empty () const {

return count==0;

}

template <class T>

bool Stack<T>::full () const {

return count==maxsize;

}

template <class T>

int Stack<T>::size() const {

return count;

}

template <class T>

void Stack<T>::clear() {

count=0;

}

template <class T>

Error\_code Stack<T>::top (T &item) const {

if (empty()) return underflow;

item= entry[count-1];

return success;

}

template <class T>

Error\_code Stack<T>::pop () {

if (empty()) return underflow;

count--;

return success;

}

template <class T>

Error\_code Stack<T>::push (const T &item) {

if (full()) return overflow;

entry[count++]=item;

return success;

}

//------------------------------------------额外函数------------------

bool user\_says\_yes()

{

int c;

bool initial\_response = true;

do { // Loop until an appropriate input is received.

if (initial\_response)

cout << " (y,n)? " << flush;

else

cout << "Respond with either y or n: " << flush;

do { // Ignore white space.

c = cin.get();

} while (c == '\n' || c ==' ' || c == '\t');

initial\_response = false;

} while (c != 'y' && c != 'Y' && c != 'n' && c != 'N');

return (c == 'y' || c == 'Y');

}

//---------------------------------- Main Program ----------------------

Stack<char> sign;

Stack<double> num;

int set; // 判断程序中的异常，以便适时退出

void process(char c) { //计算两个数的 + - \* / 运算

int k=0;

double a,b;

sign.pop();

if (num.top(b)==success){ //判断例外

num.pop();

if (num.top(a)==success) {

num.pop();

k=1;

}

}

if (k) {

switch (c) {

case '+': num.push(a+b); break;

case '-': num.push(a-b); break;

case '\*': num.push(a\*b); break;

case '/':

if (b==0) { //分母不能为0

set=4;

num.push(-1);

}

else

num.push(a/b);

break;

}

}

else {set=1;num.push(-1);}

}

//输入表达式

void get\_command(string &str) {

cout<<"\n请输入要进行运算的表达式,包括\" +,-,\*,/,=,(,)\"和数字,"<<endl

<<"例如:\" 3+2.5\*(6-25/4)-8.32= \"."<<endl

<<"注意: 以数字开头,等号结尾,中间括号要匹配."<<endl;

cin>>str;

}

//求值 表达式

double do\_command(const string &str) {

string s="";

double outcome=-1;

char c;

for (int i=0;str[i]!='\0';i++)

{

if (set!=0) break; //例外 则停止运行

while (1) { //分离数据与运算符

if (str[i]<='9' && str[i]>='0' || str[i]=='.') {

s+=str[i];

i++;

}

else {

if(s!="") {

if (num.push(atof(s.c\_str ()))==overflow)

set=3;

s="";

}

break;

}

}

char ch= str[i];

switch (ch) { //处理运算的优先级，并注意例外抛出

case '\*':

case '/':

if (sign.top(c)==success)

if(c=='\*'||c=='/') process(c);

if (sign.push(ch)==overflow)

set=3;

break;

case '+':

case '-':

while (sign.top(c)==success) {

if (c!='(') process(c);

else break;

}

if (sign.push(ch)==overflow)

set=3;

break;

case '(':

if (sign.push(ch)==overflow)

set=3;

break;

case ')':

while (sign.top(c)==success) {

if (c!='(') process(c);

else break;

}

sign.pop();

break;

case '=':

while (sign.top(c)==success) {

if (c!='(') process(c);

else break;

}

break;

default: set=2;break;

}

}

if (num.size()==1 && sign.size()==0)

num.top(outcome);

else set=1;

if (set==0) cout<<"运算结果是：\n"<<endl; //出错时的错误信息

else {

outcome=-1;

if (set==1) cout<<"\n您输入的不匹配，有错误发生。Result lost!!"<<endl;

if (set==2) cout<<"\n您输入了非法字符 , 请重新输入,谢谢合作!"<<endl;

if (set==3) cout<<"\nStack is full, Lost result!!"<<endl;

if (set==4) cout<<"\n 分母为0，不能进行除法运算，出现溢出， Lost result!!"<<endl;

}

return outcome;

}

// 主程序main()

int main() {

do {

string str,s;

set=0;

get\_command(str);

s=str;

if( str[0]=='-') str='0'+str; //处理表达式中的负号

for (int i=1;str[i]!='\0';i++) {

if (str[i]=='-' && str[i-1]=='(') {

str.insert (i,"0");

i++;

}

}

double out= do\_command(str);

cout<<s<<out<<endl; //输出结果

num.clear(); //清空栈

sign.clear();

cout<<"\n还要计算其它的吗"<<flush;

}while (user\_says\_yes()); //允许多次执行运算

return 1;

}

1. 给你一个数组，进行排序，要求所有的奇数在前面，所有的偶数在后面，并且都是从大到小的（大概是这样子的题目）
2. 让你写一个程序，能够实现10进制的数字自动转换成16进制的
3. 关于排序的题目，好像是写出你会的排序方法，然后写出其中一种你会的代码
4. 写两个线程，交替输出数字

在Java下实现两个线程交替执行，轮流打印数字和字母

在CSDN上看到的一个问题，多线程轮流打印数字和字母，如：

1 2 A 3 4 B……51 52 Z

通过管程Monitor实现，在Java中通过wait()和notify()来实现，但无法实现2个以上的线程按照特定的顺序执行，居然是有特定顺序的，还不如直接使用串行的代码。

下面是代码：

/\*\*

\* 管程，线程在这里面排队

\* @author admin

\*

\*/

class Monitor {

private static Monitor monitor = null;

private Monitor() {

}

public static Monitor getMonitor() {

if (monitor == null) {

monitor = new Monitor();

}

return monitor;

}

}

/\*\*

\* 连续打印两个数字，然后等待

\* @author admin

\*

\*/

class NumberPrinter implements Runnable {

private int num = 1;

public void run() {

synchronized (Monitor.getMonitor()) {

while (num < 52) {

System.out.print(num + " " + (num + 1) + " ");

num += 2;

Monitor.getMonitor().notify();

if (num < 52) {

try {

Monitor.getMonitor().wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

}

}

/\*\*

\* 打印一个字母，然后等待

\* @author admin

\*

\*/

class CharPrinter implements Runnable {

private char ch = 'A';

@Override

public void run() {

synchronized (Monitor.getMonitor()) {

while (ch <= 'Z') {

System.out.print(ch + " ");

ch += 1;

Monitor.getMonitor().notify();

if (ch <= 'Z') {

try {

Monitor.getMonitor().wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

}

}

public class Main {

public static void main(String[] args) {

Thread thread1 = new Thread(new NumberPrinter());

Thread thread2 = new Thread(new CharPrinter());

thread1.start();

thread2.start();

try {

thread1.join();

thread2.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

肯定离不开数据结构，操作系统，计算机网络，设计模式，算法，编程语言的了

1、有关系R和S，R∩S等价于（B）

     A、S-(R-S)    B、R-(R-S)   C、(R-S)US  D、RU(R-S)

2、叉树的先序遍历序列和后序遍历序列正好相反,则该二叉树满足的条件是(B)

      A、空或只有一个结点   B、高度等于其结点数   C、该二叉树是完全二叉树  D、所有结点无右孩子

3、下类函数中不是类的成员函数的是（D）

   A、构造   B、 析构   C、友元   D、拷贝构造

4、假设X是这样的自然数,X能被3整除或者X的个位数是3,那么在1000以内(小于1000)，求这样的X的累加之和。（199701）

1 #include<stdio.h>

2 int main(){

3 int i,sum=0;

4 for(i=0;i<=1000;i++){

5 if(i%3==0||i%10==3){

6 sum=sum+i;

7 }

8 }

9 printf("%d\n",sum);

10 scanf("%d",&sum);

11 }

5、栈的插入和删除操作在（B）进行。

        A、栈底     B、栈顶       C、任意位置     D、指定位置

定义：栈是规定只能在栈顶插入或删除的线性表，也就是先进后出的规则。

6、某公司数据库密码规定为5位组成的字符串，存储之前，需要将其加密，加密算法为：依次将每个字符依次将每个字符的ASC码值乘以2，再加上10，若计算到的新字符的值等于128，则继续将其除以3，否则不进行除法运算。最后将该得到的新字符串中所有字符前后互换（第一位和最后一位，第二位和倒数第二位交换，依次交换），编程求字符串“abcde”加密后的字符串。（FFEDD）

1 #include<stdio.h>

2 #include <string.h>

3 int main(){

4 char pwd[]="abcde",temp;

5 int len=strlen(pwd);

6 /\*赋值\*/

7 for(int i=0;i<len;i++){

8 printf("%c\n",pwd[i]);

9 if((pwd[i]\*2+10)>=128){

10 pwd[i]=(pwd[i]\*2+10)/3;

11 printf("大于或等于%c\n",pwd[i]);

12 }

13 else{

14 pwd[i]=(pwd[i]\*2+10);

15 printf("小于%c\n",pwd[i]);

16 }

17 }

18 /\*交换\*/

19 for(int j=0;j<len/2;j++){

20 temp=pwd[j];

21 pwd[j]=pwd[len-j-1];

22 pwd[len-j-1]=temp;

23 }

24 /\*打印出来\*/

25 for(int j=0;j<len;j++){

26 printf("最终%c\n",pwd[j]);

27 }

28 /\*预留输入\*/

29 scanf("%d",&len);

30 }

7、为了有效的捕获系统需求，应采用（C）

   A、瀑布模型    B、V 模型     C、原型模型       D、螺旋模型

 8、写出下面程序的运行结果（3667788）

1 #include<stdio.h>

2

3 void main(){

4 int c;

5 char aa[]="12345";

6 char \*p=aa;

7

8 while(\*p){

9 c=\*p;

10 switch(c-'2'){

11 case 0:

12 case 1:putchar(c+4);

13 case 2:putchar(c+4);break;

14 case 3:putchar(c+3);break;

15 default:putchar(c+2);break;

16 }

17 p++;

18 }

19 printf("\n");

20 }

9、C#中关于构造函数描述正确的是（C）

    A、构造函数可以声明返回类型  B、构造函数不可以用private修饰  C、构造函数必须与类名相同  D、构造函数不能带参数

 10、下面哪个是无线网络协议（C）

A、ADSL   B、100BaseT  C、WiMax  D、1000BaseT

 11、有关析构函数的说法不正确的是( C)。

A、析构函数有且只有一个   B、析构函数无任何函数类型    C、析构函数和构造函数一样可以有形参   D、析构函数的作用是在对象生命周期结束时收回先前分配的内存空间

 12、2到38642之间个位数为1的素数(只能被自己和1整除的数)的和是（）

1 #include<stdio.h>

2 int main(){

3 int i=0,sum=0;

4 for(i=2;i<=38642;i++){

5 int a,b,m=i/2;

6 /\*获取个位数\*/

7 a=i%10;

8 /\*判断素数\*/

9 while (i%m!=0 && m>1)

10 m--;

11 /\*判断是否个位为1+素数\*/

12 if (a==1 && m==1){

13 sum+=i;

14 }

15 }

16 printf("%d\n",sum);

17 scanf("%d",&i);

18 }

13、有关类和对象的说法中不正确的一项是（C）

 A、对象是类的一个实例    B、任何一个对象只能是属于一个具体的类   C、一个类只能有一个对象  D、类与对象的关系与数据模型和变量的关系类似

 14、答案（-29）

1 #include<stdio.h>

2 #define fire(X,Y) (X)--;(Y)++;(X)\*(Y);

3 int main(){

4 int a=3,b=4;

5 for(int i=0;i<5;i++){fire(a,b);}

6 printf("%d%d",a,b);

7 scanf("%d",&a);

8 }

15、如下下列程序的答案（011122）

1 #include<stdio.h>

2 void main(){

3 int i;

4 for(i=0;i<3;i++){

5 switch(i){

6 case 1:printf("%d",i);

7 case 2:printf("%d",i);

8 default :printf("%d",i);

9 }

10 }

11 scanf("%d",&i);

12 }

16、下列网络协议在协议栈中排位最高的是（A）？？？

A、TCP   B、SMTP   C、IEEE 802.11a    D、Bluetooth

 17、C#中下列语句不能用于将Person类型转换为Student类型的是（AD）

A、Person is Student B、 Person as Student  C、Student stu1=（Student）Person  D、Studnet.Convert(Person)

 18、写出下面程序运行的结果（false）

1 public class exam {

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

3 String s1 = "Hello World";

4 String s2 = new String("Hellow World");

5 System.out.println(s1 == s2);

6 }

7 }

19、下列关于数据库概念“关系”的陈述中 错误的是（B）

A 表中任意两行的值不能相同　　B 表中任意两列的值不能相同     C 行在表中的顺序无关紧要　　D 列在表中的顺序无关紧要

20、队列是一种（A）的线性表

 A、先进先出   B、先进后出  C、只能插入  D、只能删除

21、常用的数据传输速率单位有：Kbps、Mbps、Gbps，1Gbps等于（A）

A、1\*10^3Mbps   B、1\*10^3Kbps  C、1\*10^6Mbps D、1\*10^9Kbps

 22、 下列关于类的构造函数说法不正确的是（D   ）。

A、构造函数名与类名相同   B、构造函数在说明类变量时自动执行 C、构造函数无任何函数类型 D、构造函数有且只有一个

 23、关于delete运算符的下列描述中，（ C）是错误的。

A、它必须用于new返回的指针； B、使用它删除对象时要调用析构函数； C、对一个指针可以使用多次该运算符； D、指针名前只有一对方括号符号，不管所删除数组的维数。

 24、用链表表示线性表的优点？（C）

A 便于随机存取 B 花费的存储空间比顺序表少  C 便于插入与删除 D 数据元素的物理顺序与逻辑顺序相同

 25、模块A直接访问模块B的内部数据，则模块A和模块B的耦合类型为 (D) 。

A．数据耦合     B．标记耦合       C．公共耦合     D．内容耦合

 笔记：当一个模块直接修改或操作另一个模块的数据,或者直接转入另一个模块时，就发生了内容耦合。此时，被修改的模块完全依赖于修改它的模块。如果发生下列情形，两个模块之间就发生了内容耦合。

26、写出下列程序运行的结果（BAAAABCAAABCDAABCD）

1 #include<stdio.h>

2 #include <string.h>

3 int main(){

4 char str1[100]="BCD";

5 char str2[100]="AAAAA";

6 int i=0;

7 for(i=1;i<=4;i++){

8 memcpy(str2,str1,i);

9 printf("%s",str2);

10 }

11 return 0;

12 }

27、写出下列程序运行的结果（compute）

1 #include<stdio.h>

2 void main(){

3 static char a[]="computer";

4 static char b[]="computer";

5 char \*p=a;

6 char \*q=b;

7 int j;

8 for(j=0;j<7;j++){

9 if(\*(p+j)==\*(q+j)){

10 printf("%c",\*(p+j));

11 }

12 }13 }

28、写出下列程序运行的结果（4）

1 class exam{

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

3 int[] circle = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

4 int counter=0;

5 int i=0,j=0;

6 while(counter<circle.length){

7 if(circle[i]!=-1){

8 j++;

9 if(j==3){

10 counter++;

11 if(counter==circle.length){

12 System.out.println(circle[i]);

13 break;

14 }

15 circle[i]=-1;

16 j=0;

17 }

18 }

19 i++;

20 if(i>=circle.length){

21 i=0;

22 }

23 }

24

25 }

26 }

 29、（A）

A、the application complies but doesn't print anything

B 、the application complies and print "DONE WAITING";

C 、the application complies and print "FINALLY";

D 、the application complies and print "ALL DONE";

1 public class Z{

2 public static void main(String[] args){new Z();}

3 private Z(){

4 Z alias1=this;

5 Z alias2=this;

6 synchronized(alias1){

7 try{

8 alias2.wait();

9 System.out.println("DONE WAITING");

10 }

11 catch(InterruptedException e){

12 System.out.println("INTERR UPTED");

13 }

14 catch(Exception e){

15 System.out.println("OTHER EXCEPTION");

16 }

17 finally{

18 System.out.println("FINALLY");

19 }

20 }

21 System.out.println("ALL DONE");

22 }

23 }

30、输出712的N次方,结果后三位为696的N的个数，0<N<24669（246）

1 #include<stdio.h>

2 int main(){

3 int sum=712,count=0;

4 for(int i=0;i<24669;i++)

5 {

6 sum=sum\*712%1000;

7 if(sum==696)

8 count++;

9 }

10 printf("输出712的N次方,结果后三位为696的N的个数为：%d （0<N<24669）",count);

11 scanf("%d",&count);

12 }

31、输出下面的第三行的结果（1）

public class exam{

public static void main(String []args){

String overview="This chapter contains a description of convertions used in this manual";

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

int num=0;

for(int j=0;j<overview.length();j++){

if(overview.charAt(i)==overview.charAt(j))

num++;

}

System.out.println(num);

}

}

}

32、输出下面程序的结果：

dog   
cat   
boy   
apple

1 public class exam{

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

3 String overivew="apple boy cat dog";

4 String[] arr=overivew.split(" ");

5 int len=arr.length;

6 for(int i=1;i<=len;i++){

7 System.out.println(arr[len-i]+" ");

8 }

9 }

10 }

33、为了使索引键的值在基本表中唯一，在建立索引语句中应使用保留字（A）

A、 UNIQUE    B、COUNT      C、DISDINCT    D、UNION

34、苹果ios操作系统上的应用开发语言是（C）

A、C    B、C++   C、Objective  c    D、java    E、Python

 35、Android的IPC（进程通讯）主要采用以下哪个？（C）

A、Socket    B、Pipe    C、Binder   D、Signal

36、不属于构造函数特征的是（D）

A、构造函数的函数名与类名同名  B、构造函数可以重载  C、构造函数可以设置缺省（默认）参数  D、构造函数必须指定类型说明

 37、以下数据结构中哪一个是非线性结构（）

A、

**1、叉树的先序遍历序列**和后序遍历序列正好相反,则该二叉树满足的条件是(D)

A、空或只有一个结点 B、高度等于其结点数 C、该二叉树是完全二叉树 D、所有结点无右孩子

应该是二叉树的每个结点都只有一个（左或右）子树时，先序和和后序正好相反

**2、假设X是这样的自然数**,X能被4整除或者X的个位数是3,那么在3000以内(小于1000)，求这样的X的累加之和。（1575900）

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/moubenmao_jun/article/details/11714379)

1. #include "stdafx.h"
3. #include
5. #include
7. #include
9. #include "math.h"
11. #include
13. #include


17. **using** **namespace** std;




23. **int** main(**int** argc, \_TCHAR\*argv[])
25. {
27. **int** i,sum=0;


31. **for**(i=0; i<=3000;i++)
33. {
35. **if**(i%4==0 || i==3)
37. {
39. sum = sum+i;
41. }
43. }
45. printf("%d\n", sum);
47. **return** 0;
49. }

**3、某公司数据库密码规定为5位组成的字符串**，存储之前，需要将其加密，加密算法为：依次将每个字符依次将每个字符的ASC码值乘以2，再加上10，若计算到的新字符的值等于128，则继续将其除以3，否则不进行除法运算。最后将该得到的新字符串中所有字符前后互换（第一位和最后一位，第二位和倒数第二位交换，依次交换），编程求字符串“efvrg”加密后的字符串。（HORGF）

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/moubenmao_jun/article/details/11714379)

1. #include "stdafx.h"
3. #include
5. #include
7. #include
9. #include "math.h"
11. #include
13. #include
15. **using** **namespace** std;
17. **int** main(**int** argc, \_TCHAR\*argv[])
19. {
21. **char** pwd[]= "efvrg",temp;
23. **int** len =strlen(pwd);


27. **for**(inti=0;i
29. {
31. printf("%c\n",pwd[i]);
33. **if**((pwd[i]\*2+10)>=128)
35. {
37. pwd[i]=(pwd[i]\*2+10)/3;
39. printf("新字符的值等于128%c\n",pwd[i]);
41. }
43. **else**
45. {
47. pwd[i]=(pwd[i]\*2+10);
49. printf("小于128%c\n",pwd[i]);
51. }
53. }


57. **for**(intj=0;j
59. {
61. temp=pwd[j];
63. pwd[j]=pwd[len-j-1];
65. pwd[len-j-1]=temp;
67. }


71. **for**(intj=0;j
73. {
75. printf("加密后的字符串%c\n",pwd[j]);
77. }


81. return0;
83. }

5、为了有效的捕获系统需求，应采用（C）

A、瀑布模型 B、V 模型 C、原型模型 D、螺旋模型

**12、2到38642之间个位数为1的素数(只能被自己和1整除的数)的和是（18343110）**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/moubenmao_jun/article/details/11714379)

1. #include "stdafx.h"
3. #include
5. #include
7. #include
9. #include "math.h"
11. #include
13. #include


17. **using** **namespace** std;




23. **int** main(**int** argc, \_TCHAR\*argv[])
25. {
27. **int** i=0,sum=0;


31. **for**(i=2; i<=38642; i++)
33. {
35. **int** a,b,m = i/2;


39. a=i;


43. **while** (i%m!=0  && m>1)
45. {
47. m--;
49. }


53. **if** (a==1  && m==1)
55. {
57. sum += i;
59. }
61. }
63. printf("%d\n",sum);


67. **return** 0;
69. }
71. 13、有关类和对象的说法中不正确的一项是（C）
73. A、对象是类的一个实例 B、任何一个对象只能是属于一个具体的类 C、一个类只能有一个对象 D、类与对象的关系与数据模型和变量的关系类似


77. 26、写出下列程序运行的结果（BAAAABCAAABCDAABCD）
79. #include
81. #include
83. intmain(){
85. charstr1[100]="BCD";
87. **char** str2[100]="AAAAA";
89. **int** i=0;
91. **for**(i=1;i<=4;i++){
93. memcpy(str2,str1,i);
95. printf("%s",str2);
97. }
99. **return** 0;
101. }

15、输出712的N次方,结果后三位为696的N的个数，0（246）

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/moubenmao_jun/article/details/11714379)

1. #include "stdafx.h"
3. #include
5. #include
7. #include
9. #include "math.h"
11. #include
13. #include


17. **using** **namespace** std;


21. **int** main(**int** argc, \_TCHAR\* argv[])
23. {
25. intsum=712,count=0;


29. **for**(inti=0;i<24669;i++)
31. {
33. sum=sum\*71200;
35. **if**(sum==696)
37. count++;
39. }
41. printf("输出712的N次方,结果后三位为696的N的个数为：%d （0）",count);
43. scanf("%d",&count);


47. return0;
49. }

16、输出下面的第三行的结果（7）

public class Hello

{

     public static void main(String args[])

     {

            String overview = "This chapter contains a description ofconvertions used in this manual";

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

            {

                   int num = 0;

                   for (int j = 0; j < overview.length();j++)

                   {

                          if (overview.charAt(i) == overview.charAt(j))

                                 num++;

                   }

                   System.out.println(num);

            }

     }

}

**17、不属于构造函数特征的是（D）**

A、构造函数的函数名与类名同名 B、构造函数可以重载 C、构造函数可以设置缺省（默认）参数 D、构造函数必须指定类型说明

**18到5635468724之间的最大素数是？1340501419**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/moubenmao_jun/article/details/11714379)

1. #include "stdafx.h"
3. #include
5. #include
7. #include
9. #include "math.h"
11. #include
13. #include


17. **using** **namespace** std;




23. **int** main(**int** argc, \_TCHAR\*argv[])
25. {
27. unsigned longa=5635468724,c,i,d;
29. **int** b=1;
31. **while**(b)
33. {
35. a--;
37. c=sqrt(a\*1.0);
39. **for**(i=2;i<=c;i++)
41. {
43. d = a%i;
45. **if**(!d) **break**;
47. }
49. **if**(i>c)b=0;
51. }
53. printf("%ld",a);
55. **return** 1;


59. **return** 0;
61. }