**1、大端模式和小端模式区别**

区别：大端模式中字数据的高字节存储在低地址中，而字数据的低字节则存放在高地址中；而与大端存储模式相反，在小端存储模式中，低地址中存放的是字数据的低字节，高地址存放的是字数据的高字节。

**2、堆和栈的区别**

**3、static readonly和const的区别**

**4、结构体和共同体的区别**

1）变量长度不同

结构体变量所占内存长度是各成员占的内存长度之和，每个成员分别占有其自己的内存单元。

共用体变量所占的内存长度等于最长的成员的长度。

2）占用空间不同

结构体是同时存在的，并一次占用一段连续的内存空间；

而共用体则是多个共用成员占用同一个开始的内存地址，同时他们只能存在一个，所以空间大小就是最大那个所需的空间，如果单从一个共用体来讲，我们是不知道里面存的是什么内容，需要根据程序上下文才能确定。

**5、线程调度中sleep和wait区别**

（1）属于不同的两个类，sleep()方法是线程类（Thread）的静态方法，wait()方法是Object类里的方法。

（2）sleep()方法不会释放锁，wait()方法释放对象锁。

（3）sleep()方法可以在任何地方使用，wait()方法则只能在同步方法或同步块中使用。

（4）sleep()必须捕获异常，wait()方法、notify()方法和notiftAll()方法不需要捕获异常。

（5）sleep()使线程进入阻塞状态（线程睡眠），wait()方法使线程进入等待队列（线程挂起），也就是阻塞类别不同。

  (6)  它们都可以被interrupted方法中断。

**6、完全二叉树和满二叉树的区别**

满二叉树——除了叶结点外每一个结点都有左右子女且叶结点都处在最底层的二叉树,。（这个似乎很好想像出来）完全二叉树——只有最下面的两层结点度小于2，并且最下面一层的结点都集中在该层最左边的若干位置的二叉树；（这个，就说从满二叉树里，最下一层的叶子，如果是从右往左拿掉叶子，不论多少，都是完全的，如果不是从右往左拿，而是在中间拿掉了一个，就是不完全的）

7、什么是AVL树？

**8、const和define的区别**

1）就起作用的阶段而言： #define是在编译的预处理阶段起作用，而const是在编译、运行的时候起作用。

2）就起作用的方式而言： #define只是简单的字符串替换，没有类型检查。而const有对应的数据类型，是要进行判断的，可以避免一些低级的错误。

3）就存储方式而言：#define只是进行展开，有多少地方使用，就替换多少次，它定义的宏常量在内存中有若干个备份；const定义的只读变量在程序运行过程中只有一份备份。

4）从代码调试的方便程度而言： const常量可以进行调试的，define是不能进行调试的，因为在预编译阶段就已经替换掉了。

**const优点**

1）const常量有数据类型，而宏常量没有数据类型。编译器可以对前者进行类型安全检查。而对后者只进行字符替换，没有类型安全检查，并且在字符替换可能会产生意料不到的错误。

2）有些集成化的调试工具可以对const常量进行调试，但是不能对宏常量进行调试。

3）const可节省空间，避免不必要的内存分配，提高效率；define – 不分配内存，给出的是立即数，有多少次使用就进行多少次替换，在内存中会有多个拷贝，消耗内存大，const – 在静态存储区中分配空间，在程序运行过程中内存中只有一个拷贝；

**9、虚函数和纯虚函数**

**10、引用传递和指针传递**

指针参数传递本质上是值传递，它所传递的是一个地址值。值传递的特点是，被调函数对形式参数的任何操作都是作为局部变量进行的，不会影响主调函数的实参变量的值（形参指针变了，实参指针不会变）。

　　引用参数传递过程中，被调函数的形式参数也作为局部变量在栈中开辟了内存空间，但是这时存放的是由主调函数放进来的实参变量的地址。被调函数对形参（本体）的任何操作都被处理成间接寻址，即通过栈中存放的地址访问主调函数中的实参变量（根据别名找到主调函数中的本体）。因此，被调函数对形参的任何操作都会影响主调函数中的实参变量。

　　引用传递和指针传递是不同的，虽然他们都是在被调函数栈空间上的一个局部变量，但是任何对于引用参数的处理都会通过一个间接寻址的方式操作到主调函数中的相关变量。而对于指针传递的参数，如果改变被调函数中的指针地址，它将应用不到主调函数的相关变量。如果想通过指针参数传递来改变主调函数中的相关变量（地址），那就得使用指向指针的指针或者指针引用。

**11、快排和冒泡排序**