# 一、C/C++中struct和class的区别

## ★C中struct和class的区别:

struct不能定义成员函数，只能定义变量，没有继承、没有封装。

## ★C++中struct和class的区别:

### ① 默认成员权限区别

struct的成员默认权限是public，而class的成员默认权限是private。

### ② 默认继承方式

struct的默认继承方式为public，而class的默认继承为private。

### ③ 用于定义模板参数

模板为C++语言新增特性，C语言没有，只有class可用于定义参数，而struct不可以。

总：建议用法，在字面上struct与class的含义不一样，struct更适合看成是一个数据结构的实现体，class更适合看成是一个对象的实现体。

# 二、C++中union介绍

union内定义多种不同的数据类型，数据共享同一段内存，以达到节省空间的目的。union变量所占用的内存长度等于最长的成员的内存长度。

## Union可以用来测试CPU大小端

在小端模式中，低位字节放在低地址，高位字节放在高地址；

1.在c中，联合体（共用体）的数据成员都是从低地址开始存放。

2. 若是小端模式，由低地址到高地址c.a存放为0x01 00 00 00，c.b被赋值为0x01；

3. 若是大端模式，由低地址到高地址c.a存放为0x00 00 00 01，c.b被赋值为0x0；  
//若处理器是Big\_endian的，则返回0；若是Little\_endian的，则返回1。

int checkCPU()   
{   
      union w   
       {     
           int  a;   
           char b;   
       } c;   
    c.a = 1;   
   return (c.b == 1);   
}

# 三、堆溢出和栈溢出原因分析

1. 堆溢出: 不断的new 一个对象，一直创建新的对象，

2. 栈溢出：死循环或者是递归太深 。

# 四、为什么TCP需要三次握手和四次挥手

TCP协议是一种面向连接的、可靠的、基于字节流的传输层通信协议，采用全双工通信。

#### 那为什么需要三次握手呢？请看如下的过程：

1. A向B发起建立连接请求：A——>B；
2. B收到A的发送信号，并且向A发送确认信息：B——>A；
3. A收到B的确认信号，并向B发送确认信号：A——>B。

三次握手大概就是这么个过程。  
通过第一次握手，B知道A能够发送数据。通过第二次握手，A知道B能发送数据。结合第一次握手和第二次握手，A知道B能接收数据。结合第三次握手，B知道A能够接收数据。

至此，完成了握手过程，A知道B能收能发，B知道A能收能发，通信连接至此建立。三次连接是保证可靠的最小握手次数，再多次握手也不能提高通信成功的概率，反而浪费资源。

#### 那为什么需要四次挥手呢？请看如下过程：

1. A向B发起请求，表示A没有数据要发送了：A——>B；
2. B向A发送信号，确认A的断开请求请求：B——>A；
3. B向A发送信号，请求断开连接，表示B没有数据要发送了：B——>A；
4. A向B发送确认信号，同意断开：A——>B。

B收到确认信号，断开连接，而A在一段时间内没收到B的信号，表明B已经断开了，于是A也断开了连接。至此，完成挥手过程。

可能有捧油会问，为什么2、3次挥手不能合在一次挥手中？那是因为此时A虽然不再发送数据了，但是还可以接收数据，B可能还有数据要发送给A，所以两次挥手不能合并为一次。

**挥手次数比握手多一次，是因为握手过程，通信只需要处理连接。而挥手过程，通信需要处理数据+连接。**

# 五、进程间的通信方式

**管道、信号、信号量、共享内存、消息队列、套接字**

# 六、引用和指针的区别

★相同点：

①都是地址的概念；指针指向一块内存，它的内容是所指内存的地址；而引用则是某块内存的别名。

★不同点：

①指针是一个实体，而引用仅是个别名；

②引用只能在定义时被初始化一次，之后不可变；指针可变；

③引用没有const，指针有const，const的指针不可变；

④引用不能为空，指针可以为空；

⑤“sizeof 引用”得到的是所指向的变量(对象)的大小，而“sizeof 指针”得到的是指针本身的大小；

⑥指针和引用的自增(++)运算意义不一样；

⑦引用是类型安全的，而指针不是 (引用比指针多了类型检查)。

**参考：**

**1. http://blog.csdn.net/nocky/article/details/6195556**

**4.** [**https://www.jianshu.com/p/bbb6261cb13e**](https://www.jianshu.com/p/bbb6261cb13e)

**5.** **http://blog.csdn.net/c15522627353/article/details/52972941**

# 1、图搜索

## ★深度优先搜索

class solution {

static 记录; // List<Integer> result = new ArrayList<Integer>();

// static 路径;

// static 剪枝变量;

void dfs(...状态变量，全局输入) {

if( 边界条件 ) {

具体记录方法; // result.add();

return;

}

for ( 所有分支 ) {

// 剪枝

// 剪枝变量设置

dfs(...状态变量，全局输入);

// 剪枝变量还原(most important)

}

// 有可能多分支dfs有组合逻辑

return;

}

优化算法：迭代加深+启发式（IDA\*）

LeetCode: 51, 78

## ★广度优先搜索

void bfs(int x)

visited[x] true

Queue.push(x)

While(!Queue.empty())

V = Queue.pop()

For (v的每个邻接点w)

If（!visited[w]）

Visited[w] = true

Queue.push(w)

Leetcode: 200, 130

经典问题：洪水填充法

**参考：**

http://www.julyedu.com/video/play/51/263