第二讲

C/C++基础要点复习

主讲人：Jesse

2017-09-21

1

CHAPTER1

基础语法考点

2

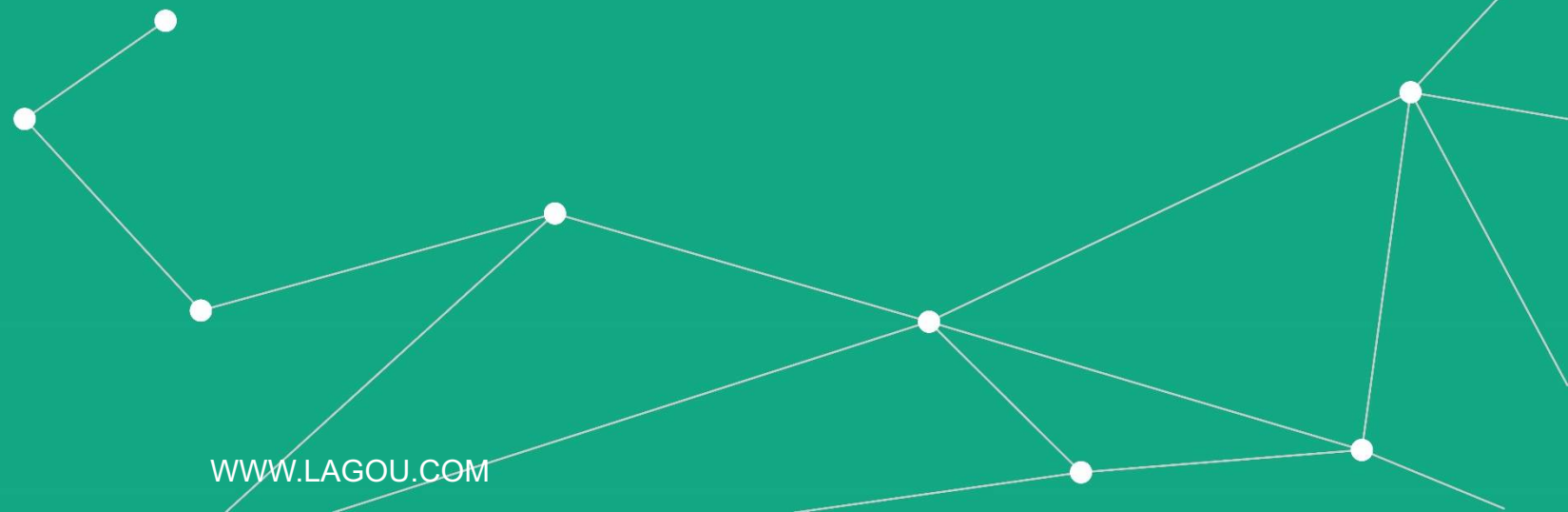
CHAPTER2

面向对象考点（C++特性）

CHAPTER1

基础语法考点

运算符优先级、内存占用、const、static、extern、堆栈、大小端 ……



基础语法考点

运算符的优先级规则

10 * a++ + 1 >> 2 ??

- **操作符**：近50种
- **优先级**：15种
- **结合性**：两种（从左到右，从右到左）
- **最常考**：自增++，移位

阅读材料 http://www.cnblogs.com/ywl925/p/3710246.html

运算符的优先级规则

下列哪项是正确的？

逻辑"或"（即运算符||）的运算级别比算术运算要高

C语言的关系表达式：0<x<10完全等价于：（0<x）&&（x<10）

逻辑"非"（即运算符!）的运算级别是最低的

由&&构成的逻辑表达式与由||构成的逻辑表达式都有"短路"现象

运算符的优先级规则

表达式是true还是false？

C语言中，若有int a[5]={12,34,56,78,90}，*p=a；则*p++==13.

位运算

补码与溢出：计算机中数值一律用补码来表示和存储。无符号数的补码跟原码相同，负数的补码由负数的原码各位取反（除符号位）后加1得到。有符号数补码符号位产生进位和接受进位时表示数据溢出。

运算符：& | << >> ^ ~

考点：运算、优先级、整数溢出

阅读材料

http://www.360doc.com/content/14/0903/11/15077656_406715772.shtml

内存占用与sizeof

- **基本类型**：int / char / float / double 内存占用
- **数组指针**：int a[10]; char * p = new char[10]; sizeof(a) sizeof(p)
- **结构体、类、联合体**：内存对齐原则
- **注意**：64位与32位系统的区别

阅读材料 http://blog.csdn.net/oktears/article/details/19352577

http://blog.csdn.net/microsues/article/details/6140329

内存占用与sizeof

若有以下说明和定义
union dt
{
int a; char b; double c;
}data;
以下叙述中错误的是（ ）。

A. data的每个成员起始地址都相同

B. 变量data所占内存字节数与成员c所占字节数相等

C. 程序段：data.a=5;printf("%f\n",data.c);输出结果为5.000000

D. data可以作为函数的实参

正确答案：C

宏定义与typedef

- **宏定义**：在编译时将定义的符号直接进行替换，可以写成函数形式
- **typedef**：声明一种新类型来等价于已有的类型，在语法上就是一种类型
- **常见考点**：是否正确处理typedef出来的类型，#define的符号与运算符优先级结果的考察

阅读材料 http://blog.csdn.net/luoweifu/article/details/41630195

宏定义与typedef

下列正确的是？

用typedef可以定义各种类型名，但不能定义变量

用typedef只是将已存在的类型用一个新的名称代替

用typedef可以增加新类型

使用typedef便于程序的通用

static与extern

static：可以修饰局部变量、全局变量、函数
static数据存放在静态区，程序启动时存放，程序结束时释放
修饰局部变量和正常情况没有区别，修饰全局变量或函数限制在该源文件中访问
c++类中static修饰表示为该类所有对象共有（仅一份）

extern：修饰变量或函数，表面是在别处定义的，这里进行引用，但不能引用别处的static

阅读材料

http://www.cnblogs.com/dolphin0520/archive/2011/04/20/2022701.html

栈区、堆区、全局区

- **栈区**：程序自动维护存放与释放，函数的参数、局部变量，声明周期为函数调用过程
- **堆区**：开发者维护数据的存放与释放，存在于整个程序进程；malloc/new的数据
- **全局区**：全局变量、静态变量，生命周期为整个程序进程

阅读材料 http://blog.csdn.net/hyqsong/article/details/42006637

const关键字

- **用法**：修饰变量、函数参数与返回值、函数；修饰成员函数；修饰对象、对象指针、引用
- **作用**：定义常量，防止修改；类型检查；节省空间；提高运行效率
- **常见考点**：与char*结合 const char * / char * const / const char * const

阅读材料 http://blog.csdn.net/Eric_Jo/article/details/4138548

const关键字

下列哪两个是等同的

int b;

1.const int *a = &b;

2.const * int a = &b;

3.const int* const a = &b;

4.int const* const a = &b;

int const *a 和 const int *a 意义相同，作用等价 同理，本题3、4意义相同

const int *a 这里const 修饰的是int，而int定义的是一个整值

int *const a 这里const修饰的是 a ，a代表的是一个指针地址 因此不能赋给a其他的地址值，但可以修改a指向的值

const int * const a 这个代表a所指向的对象的值以及它的地址本身都不能被改变

参数传递

- 值传递（拷贝，对象会调用复制构造函数）
- 指针传递
- 引用传递

阅读材料 http://www.cnblogs.com/yanlingyin/archive/2011/12/07/2278961.html

大端和小端对齐

- **大端 Big-Endian**：高位字节排放在内存的低地址端，低位字节排放在内存的高地址端
- **小端 Little-Endian**：高位字节排放在内存的高地址端，低位字节排放在内存的底地址端
- **常见考点**：大小端转换、int数组转char数组

阅读材料 http://www.cnblogs.com/ciaos/p/4622165.html

大端和小端对齐

unsigned int a= 0x1234; unsigned char b=*(unsigned char *)&a; 在32位大端模式处理器上变量b等于()?

unsigned int a= 0x1234的32位完全表示是0x00001234，在大端（低地址存储高位）处理器上的存储方式为：

由低地址到高地址依次为（假设低地址为0x4000）：

0x4000	0x4001	0x4002	0x4003
00	00	12	34

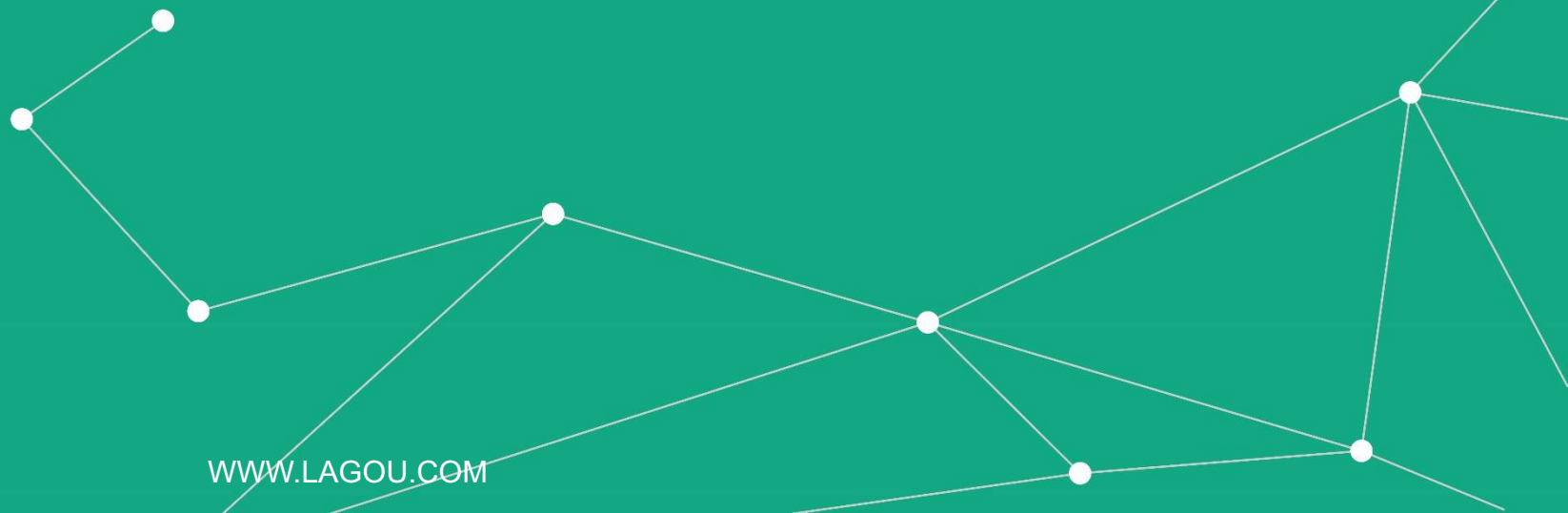
则&a的值为0x4000， char占一个字节，即b最终所取的值为0x4000地址内存储的内容，故为0x00。

若处理器为小端 （低地址存储低位） 模式，则b的值为0x34。

CHAPTER2

面向对象考点（C++特性）

继承、多态、虚函数、重载、对象模型、STL



malloc与new的区别

- **malloc**：是标准库的一个函数，仅仅是分配了内存，对应free函数，用于释放内存
- **new**：是c++的关键字，与delete对应，不仅分配内存，还能执行构造函数
- **delete与delete[]**：new出来的对象指针，delete可以释放内存和执行析构，而new [] 出来的对象数组指针，delete仅仅释放空间和析构第一个元素，delete[]释放内存且析构所有元素

阅读材料 http://blog.csdn.net/hazir/article/details/21413833

Offer工场

继承与多态

- **访问控制**：protected, public, private
- **覆盖、重载、重写**：同一个类或继承中，重名函数的处理
- **单继承、多继承、虚继承**
- **静态多态、动态多态、虚函数**

阅读材料 http://www.cnblogs.com/ChenZhongzhou/p/5682776.html

overwrite、override、overload

- **overwrite(重写)**：在继承中，若派生类函数和基类同名（可以不同参），基类函数被隐藏
- **override(覆盖)**：派生类覆盖基类中同名同参且为virtual的方法
- **overload(重载)**：同一个类中，函数名相同参数不同

阅读材料 http://blog.csdn.net/penzo/article/details/6001193

虚函数与虚函数表的原理

- **虚函数**：能在派生类中进行覆盖，从而实现多态的函数
- **纯虚函数**：在基类中不进行实现，而在派生类中必须进行实现的虚函数
- **虚函数表**：用于动态维护对象中成员函数指针指向的函数表（面试常考题）

阅读材料 http://blog.csdn.net/sanfengshou/article/details/4574604

http://www.cnblogs.com/jin521/p/5602190.html

析构与虚析构函数

- **析构函数调用顺序**：从派生类执行到基类；一个对象被析构时，该对象中成员对象也会被析构
- **虚析构函数的作用**：基类指针指向派生类，delete基类指针，也能够调用派生类的析构函数

阅读材料 http://blog.csdn.net/bresponse/article/details/6914155

深拷贝与浅拷贝

- **对象拷贝**：对象直接赋值，会调用复制构造函数；默认复制构造函数直接复制对象中所有成员
- **浅拷贝**：若对象成员中存在指针，复制过程仅仅复制了指针地址
- **深拷贝**：若对象成员中存在指针，复制过程中为该成员重新申请空间并拷贝内容

阅读材料 http://blog.csdn.net/lpp0900320123/article/details/39007047

C++对象模型

- **考察方式**：通常面试综合性考察对C++理解深度
- **主要知识点**：对象内存排布、继承与多态的实现方式、虚函数表的原理
- **掌握情况**：看懂资料，能够理解原理，能够大致讲出原理即可

阅读材料 http://www.cnblogs.com/skynet/p/3343726.html

C++对象模型

sizeof(A) = ?

```
class A {
public:
    virtual void funa();
    virtual void funb();
    void func();
    static void fund();
    static int si;
private:
    int i;
    char c;
};
```

STL： map、hash_map、set

- **map**：采用红黑树实现，插入、删除、查询的时间复杂度都是O(logN)
- **hash_map**：采用哈希表实现，理论上插入、删除、查询的时间复杂度都是O(1)
- **set**：与map底层原理相同，采用红黑树实现。hash_set与hash_map类似

阅读材料 http://blog.csdn.net/sdnu111111111/article/details/38658929

Thank You