

C/C++面试高频知识点八股文

最近有小伙伴找我问面试都问些啥，这不我就把高频被问的——列出来。

目录

- C语言必考知识点
 - 含参数的宏与函数的不同点
 - scanf格式化输入的注意事项
 - 指针
 - 数组
 - 数组与指针的区别
 - sizeof关键字
 - sizeof关键字与strlen函数的区别
 - 大小端
 - 使用库函数需要知道什么
 - 数值数组与字符数组的区别
 - strcpy和memcpy区别
 - 内存五大分区(超级重要)
 - 三个变量
 - 表达式计算通用规则(优先级表要背会)
 - 字节对齐规则
 - extern与static区别
 - 断言
 - 链表和数组的区别
 - const和define的区别
- C++必考知识点
 - 高频被问(41条)
 - new与malloc的区别
 - C++与C的区别
 - 静态链接和动态链接有什么区别？
 - 指针和引用的区别
 - 类和结构体的区别
 - define与inline区别
 - 前置++与后置++区别
 - static在C和c++的区别
 - 友元 friend

C语言必考知识点

含参数的宏与函数的不同点

函数	带参数的宏
函数调用时，先求出实表达式的值，然后带入形参。	使用带参的宏只是进行简单的字符替换。
函数调用是在程序运行时进行的，分配临时的内存单元；	而宏替换则是在编译时进行的，在展开时不分配内存单元，不进行值的传递处理，也没有“返回值”的概念。
对函数中的实参和形参都要定义类型，二者的类型数量要求一致；	宏没有类型检查，宏名无类型，它的参数也无类型，只是一个符号代表，展开时带入指定的字符即可。
调用函数只可得到一个返回值，	而用宏可以得到几个结果。
函数调用不使源程序变长。	使用宏时，宏替换后源程序增长
函数调用占运行时间（分配单元、保留现场、值传递、返回）。	宏替换不占运行时间，只占编译时间；

scanf格式化输入的注意事项

- 1. 输入数据时采用的分隔符应与格式控制中的分隔符一致；
- 2. 输入数据时可指定宽度，但不可规定精度；
- 3. 输入格式符带有*，表示该数据不赋值；
- 4. 无符号数可用%d、%o、%x格式输入；
- 5. 用c格式输入字符时，空格、转义字符均为有效字符；
- 6. 遇空格、回车、Tab键，遇宽度以及非法输入可认为数据输入结束；

指针

英文名pointer，也叫地址，就是内存块的首地址

指针变量（地址变量）

- 1. 指针变量就是存放指针数据的变量。
- 2. int*p; //p只能存放int类型内存块的地址
- 3. 所有指针变量都是4字节(32环境)
- 4. 未赋初值的指针变量禁止使用(访问未初始化的指针的值程序直接崩溃。访问NULL指针的值也会崩溃)

指针常见含义：

*p的含义： 根据p里面的地址，找对应类型的内存块。
p+n的含义：• p地址向后偏移n个 存储单元 ，得到一个新地址（在一连续空间中两个指针相减等于间隔的内存空间个数）
p[n]的含义：• 表示p地址第n+1个内存块
指针支持的运算 算术运算、自增自减、关系运算、逻辑运算 、赋值运算、条件运算(三目)、逗号运算、sizeof关键字、& *

数组

- 1、数目固定，类型相同，连续存放的一组有序数的集合；
- 2、C语言规定，数组名就是内存块的首地址，它是一个常量指针；
- 3、定义数组时，长度应是常量或常量表达式，不能用变量，而在使用时，a[i]表示a之后第i+1个内存块；
- 4、a + n表示a地址后向右偏移n个储存单元得到新的地址，a[n]表示a地址后第n + 1个内存块；
- 5、数组中每个元素都是变量；
- 6、int a[10]={1, 2, 3}; 部分元素赋初值，其余未赋值为0； int a[10]; 一个值都不赋，数组元素均为乱值。

数组与指针的区别

数组	指针
数组是一种数据结构	指针是一个地址
数组名是该数组的首地址	
sizeof测量数组即为所占内存空间的大小	sizeof测量指针在win32为四字节.win64为八字节

sizeof关键字

sizeof作用：能测试常量，变量，数组、类型 所占 应占 内存大小的单位字节数

测试字符串 就是字符个数 + 1
printf(“%d”, sizeof(“lovechina”)); 有\0
测试变量 就是变量对照类型分配的空间大小
测试数组 就是数组所占的空间大小，元素占的空间 * 数组长度
所有指针变量 所占4字节(32 系统)

sizeof关键字与strlen函数的区别

Sizeof:编译器在编译时就计算出了sizeof的结果，而strlen函数必须在运行时才能计算出来。并且sizeof计算的是数据类型占内存的大小，而strlen计算的是字符串实际的长度
strlen只能测量字符串 计算字符串 str 的长度，直到空结束字符，但不包括空结束字符

大小端

大端模式：是指数据的高字节保存在内存的低地址中；
小端模式：是指数据的高字节保存在内存的高地址中；

使用指针强转

```
57
58 int check_sys()
59 {
60     int a = 1;
61     char* p = (char*)&a;
62
63     return *p; //0-大端    1小端
64 }
65
```

CSDN @林夕07

使用共用体

```
66
67
68 int check_sys()
69 {
70     union Un
71     {
72         char c;
73         int i;
74     };
75     union Un u = { 0 };
76     u.i = 1; //如果是小端存储，就会把c也修改成1。
77     return u.i; //0-大端    1-小端
78 }
```

CSDN @林夕07

使用库函数需要知道什么

- 1. 知道函数的功能及函数名
- 2. 需要导入使用的包含头文件
- 3. 参数的数目和顺序，以及每个参数的意义及类型
- 4. 返回值的意义及类型

数值数组与字符数组的区别

	数值数组	字符型数组 (1表示用来存放数值 2表示用来存放字符)
定义及赋初值	int a[5]	1. char str[5] = {1, 2, 3} 2.char str[5] = {"abc"}
输入和输出:	循环读入和循环输出	1. 用%d %c 循环读入(数值型数组特点) 2. 用%s可以整体读入和整体输出(字符串使用)
二维数组:	int a[3][5] = {0};	3行5列数字矩阵 char str[3][5] 1. 3行5列一个字节整型数矩阵 2. 3个最大长度为4的字符串
作为函数参数	int fun(int a[], int n);	1. int fun(char str[], int n); //数值型数组需要传n 2. int fun(char str[]);//字符型数组不需要传n

	数值数组	字符型数组 (1表示用来存放数值 2表示用来存放字符)
处理函数(专有)	无	1. 无(数值数组无) 2. strstr、strcmp、strcpy、strncpy、strtok、strlwr、strupr、strcat、strlen
处理函数(公共)	qsort、memset、memcpy、memcmp、memmove、malloc	qsort、memset、memcpy、memcmp、memmove、malloc

strcpy和memcpy区别

1.复制的内容不同。

strcpy只能复制字符串，
memcpy可以复制任意内容，例如字符数组、整型、结构体、类等。

2.复制的方法不同。

strcpy不需要指定长度，它遇到被复制字符的串结束符“\0”才结束，如果空间不够，就会引起踩内存。
memcpy则是根据其第3个参数决定复制的长度。

3.用途不同。

通常在复制字符串时用strcpy，而需要复制其他类型数据时则一般用memcpy，由于字符串是以“\0”结尾的，所以对于在数据中包含“\0”的数据只能用memcpy。

内存五大分区(超级重要)

内存分区	生存周期	相关	管理	空间大小	产生问题	效率
栈	运行时分配，结束就释放	函数、auto、register(寄存器)	系统	1M	栈溢出(内存溢出)	高
堆(内存是链表结构)	程序员主动分配主动释放	malloc.h/stdlib.h/memory.h。 malloc/calloc/realloc/free库函数	用户	最大可到G级别	内存泄露、野指针、内存碎片	
全局/静态	程序运行前分配，程序运行结束释放	1、static、extern 2、作用域(全局)	系统	很小		
常量文本						
程序代码						CSDN @赫夕07

内存总共分为五大分区：栈区，堆区，全局静态区，常量文本区，程序代码区。
从生存周期来说：栈区是运行时分配，结束就释放。堆区是程序员主动分配和释放。全局静态区是程序运行前分配，程序结束释放。常量文本区也是程序运行前分配，程序结束释放
从管理角度来说：只有堆区是由程序员自己来管理，其他四个区都是由系统来管理的。
从可能产生的问题来说：栈区可能会造成栈溢出(1、深度递归，栈帧太多 2、只分配不释放，栈耗尽)。堆区可能造成内存泄漏(就是申请空间，然后忘记释放了)，野指针（指向了一块已经被释放的空间），内存碎片。
从效率来说：栈区比堆区效率高
从内存大小来说：栈区大小m，堆区大小和内存有关。最大可1G。其他三个区都很小
代码区：存放程序体的二进制代码。比如我们写的函数，都是在代码区的。

三个变量

等号左边必须是变量
自增自减必须是变量
形参必须是变量

表达式计算通用规则(优先级表要背会)

从左向右运算符俩俩比较，左边运算符优先级高于右边运算符时，先算左边的。
左边运算符优先级低于右边的继续向右比较，直到找到一个相对最高的。（之后的的运算符低，或者到末尾了），进行运算。
如果左右优先级相同看结合性，(单目运算右结合，双目运算左结合。三目运算嵌套是右结合。三目运算是左结合)，

计算完一个运算符再重复前面所有步骤。（遇到||和&&注意短路）

优先级	运算符
1	[] () . ->
2	-(负号) ++ -- *(取值) & (取址) ! ~ sizeof 函数
3	/ * (乘) %
4	+ - (减)
5	<< >>
6	> >= < <=
7	== !=
8	&(按位与)
9	^
10	
11	&&
12	
13	?: 右结合
14	= += -= *= /= %= &= ^= = <<= >>=
15	,(逗号运算)

CSDN @林夕07

字节对齐规则

- 结构体各成员的起始位置相对于结构体变量的起始位置的偏移量，应该为该结构体成员类型所占字节数与pack(n)的n取最小值的倍数
- 结构体变量所占字节数应该是结构体各成员所占字节数的最大值与pack(n)的n取最小值

extern与static区别

static

作用于局部变量时： 叫静态局部变量，在函数调用时，只有在该函数第一次调用时才对其分配空间和初始化。在函数调用结束时，不对该变量的内存进行释放，值仍然保留。这也是于自动变量的区别。

作用于全局变量时： 叫静态全局变量。表示该变量是私有的，只能在该文件使用。不能通过extern关键字对其引用。

作用于函数时： 叫静态函数，表示该函数是私有的，只能在本文件中使用，不能通过extern关键字对其引用

extern

本文件： 定义在本文件下面的全局变量，想要在上面使用时需要使用extern关键字对其声明

其他文件： 定义在其他文件的全局变量想要在本文件使用时，若该变量未被static修饰时可通过extern关键字在本文件对其声明。即可使用

断言

assert(src != NULL); //断言 括号内部成立上面事情不发生，否则报错

头文件： #include <assert.h>

作用： 解决预防性编程的问题，例如参数传入一个指针为NULL时，程序就会奔溃时，我们可以增加assert来防御这种问题。

在我们联调中assert会显示崩溃的信息，加快联调速度，也能对参数问题进行判断。assert只能在debug版起作用，发布版不生效。

综上所述： assert就是预防性编程一个重要的宏，能加快联调速度。

链表和数组的区别

链表

逻辑上相邻的元素在物理位置上不一定相邻。

优点： 插入、删除效率高，不需要一个连续的很大的内存

缺点： 查找某一个位置的元素效率低。

数组

优点： 存取速度快

缺点：

- 1.整块连续空间， 占很大内存。
- 2.插入或删除数据效率低、不方便

const和define的区别

1.编译器处理方式

define – 在预处理阶段进行替换

const – 在编译时确定其值

2.类型检查

define – 无类型， 不进行类型安全检查， 可能会产生意想不到的错误

const – 有数据类型， 编译时会进行类型检查

3.内存空间

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

4.其他

宏只作替换， 不做计算， 不做表达式求解。

总结

const比define安全

define	const
在预处理阶段进行替换	在编译时确定其值
无类型， 不进行类型安全检查， 可能会产生意想不到的错误	有数据类型， 编译时会进行类型检查
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	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
宏只作替换， 不做计算， 不做表达式求解。	const比define安全

C++必考知识点

高频被问(41条)

- 1. 空类占用内存空间： 1字节
- 2. explicit作用： 关闭函数的类型自动转换（防止隐式转换）
- 3. 当初始化列表时， 被初始化的顺序是声明是的顺序不是列表顺序。
- 4. 命名空间 作用： 解决同名冲突， 使用
 - a. 方法一： 使用命名空间名称::标识符的方式来访问
 - b. 方法二： 使用命名using namespace 命名空间名称;的方式作前置声明， 在声明之后， 可以直接使用标符来访问。
 - c. 特点： 允许不连续的命名空间、 允许嵌套类： 具有相同的属性和行为的对象的集合。 从访问权限： public（共有的） protected(保护的) private(私有的)
- 5. 函数缺省参数注意事项：
 - a.如果某个位置已经有了默认参数， 那么从这个位置往后， 从左往右都必须有默认值 如:func(int a, int b = 20, int c)错误
 - b.缺省值放在声明位置
- 6. 拷贝构造函数（一般类中有指针才需要自己实现）
三种场景：
 - 1. 无须实现
 - a. 无指针
 - 2. 需要实现
 - a. 一般类中有指针才需要自己实现

3. 无法实现

- a. `File(const File& other) = delete;` //表示禁止使用 拷贝构造

7. 浅拷贝：简单的赋值拷贝操作

深拷贝：在堆区重新申请空间，进行拷贝操作

浅拷贝常见问题：堆区数据重复释放

8. 成员变量的初始化

1. 构造函数体内
2. 构造函数的初始化列表 (速度快)
3. 在成员变量声明时

9. 类对象作为类成员

c++类的成员可以是另一个类的对象，我们称该成员为 对象成员

例如：

```
1 | class A{  
2 | class B  
3 | {  
4 |     A a;  
5 | }
```

B类中有对象A作为成员，A为对象成员

创建规则：**先创建A，在创建B 先释放b再释放a**

10. 静态成员、函数

概念：成员变量和成员函数前加一个static，称为静态成员

目的：为了实现一个类的不同对象之间的数据和函数共享。

静态成员变量特点：

- 所有对象共享同一份数据
 - 在编译阶段分配内存
 - 类内的声明，类外初始化
- a. **语法：**类型 类名::静态变量 = 表达式
 - b. **访问：**对象.静态成员变量 类名.静态成员变量

静态成员函数：

- a. 所有对象共享同一个函数
 - b. 静态成员函数只能访问静态成员变量
11. 在C++中，类内的成员变量和成员函数分开存储、只有非静态成员变量才属于类的对象上

12. this指针：当形参和成员变量同名时，可用this指针来区分(解决同名冲突)、在类的非静态成员函数中返回对象本身，可使用 `return * this`

13. const修饰成员函数

常函数：

- a. 成员函数后加const，叫常函数
- b. 常函数内不可以修改成员属性
- c. 成员属性声明时加关键词mutable后，在常函数依然可以修改

常对象：

- a. 声明对象前加const,叫常对象
- b. 常对象只能调用常函数

14. 静态多态

重载(在编译时期就可以通过函数名和参数确定需要调用那个函数)

模板

动态多态

虚函数(通过运行阶段才能知道需要调用那个对象)

纯虚函数

虚析构函数

虚函数表

15. **虚析构作用：**使用父类指针释放子类对象时可以让子类的析构函数和父类的析构函数同时被调用到。

16. 虚析构和纯虚析构共性：

- 可以解决父类指针释放子类对象
- 都需要具体的函数实现

17. **虚析构语法**: virtual ~类名 () {};

纯虚析构语法: virtual ~类名 () = 0;

纯虚析构实现 类名: : ~类名 () {}

18. 继承方法

访问权限

- public: 在子类和外部可以访问
- protected: 在子类中可以访问, 外部不可以访问
- private: 在子类和外部都不可以访问

继承方式:

子类权限 => 父类权限与继承权限中取严谨值

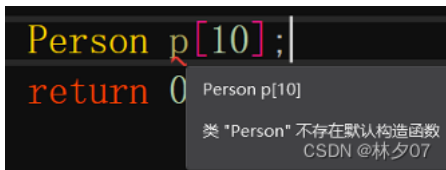
影响的是外面对该子类的访问权限以及当该子类当父类时, 别的子类的访问权限。

19. 创建对象时: String s1;

a. 分配空间

b. 调用构造函数

20. 一个类可以没有缺省构造函数, 但最多有一个 无缺构造函数时, 无法创建对象数组



21. 匿名对象 A() 生命周期只有本行(当前行执行结束后, 系统会立即回收掉)。

22. ****函数重载作用**: **达到行为标识符统一, 减少程序中标识符的个数

23. 父类中所有非静态成员属性都会被子类继承下去

父类中私有属性 是被编译器隐藏了 因此访问不到 但是的确继承下去了

24. **多继承初始化顺序**: 从左向右

```
class B : virtual public A
{
public:
};

class C : virtual public A
{
}
```

CSDN @林夕07

25. **菱形继承**: 一个类被两个类继承 这两个又被一个类继承

26. ****无法继承的内容**: **构造函数、析构函数、友元、运算符重载

27. 子类对象可以赋给父类的指针(引用)

28. **final关键字** 用处: 当前我这个类就是最终类, 我不想让别的类再继承我自己。

```
1 class Super final
2 {
3     //.....
4 };
```

31. **多态的优点**: (提倡)

- 代码组织结构清晰
- 可读性强
- 利与前期和后期的扩展和维护

32. **继承语法**: class 子类 : public 父类 目的: 减少重复代码(代码复用) 实现方式: 单继承、多继承

33. **重载**: 作用域相同函数名相同

34. **重写**: 父类的方法, 子类重写, 要求父类的该方法必须是虚函数或者纯虚函数virtual

35. **隐藏**: 父类的方法, 子类重写, 要求父类的该方法不能被virtual修饰

36. 在当前类的初始化列表调用基类的构造函数

37. 类是具有相同属性和行为的一组对象的集合

38. 封装是将抽象出的属性成员、行为成员相结合，将它们视为一个整体。
目的是增强安全性和简化编程，使用者不必了解具体的实现细节，而只需要通过外部接口（函数），以特定的访问权限，来使用类的成员。
39. 多态是同一名称，不同的功能实现方式。
目的：达到行为标识符统一，减少程序中标识符的个数
实现：重载函数、虚函数和模板
40. 浅拷贝和深拷贝有什么区别？
浅拷贝只复制指向某个对象的指针，而不复制对象本身，新旧对象还是共享一块内存；而深拷贝会创建一个相同的对象，新对象与原对象不共享内存，修改新对象不会影响原对象。
41. **虚继承 作用**：解决多继承时产生的菱形继承问题

new与malloc的区别

特性:new/delete是C++关键字，需要编译器支持。malloc/free是库函数，需要头文件支持。

参数:使用new操作符申请内存分配时无须指定内存块的大小，编译器会根据类型信息自行计算。而malloc则需要显式地指出所需内存的尺寸。

返回类型:new操作符内存分配成功时，返回的是对象类型的指针，类型严格与对象匹配，无须进行类型转换，故new是符合类型安全性的操作符。而malloc内存分配成功则是返回void *，需要通过强制类型转换将void*指针转换成我们需要的类型。

分配失败: new内存分配失败时，会抛出bad_alloc异常。malloc分配内存失败时返回NULL。

特性:new/delete是C++关键字，需要编译器支持。malloc/free是库函数，需要头文件支持。

参数:使用new操作符申请内存分配时无须指定内存块的大小，编译器会根据类型信息自行计算。而malloc则需要显式地指出所需内存的尺寸。

C++与C的区别

- C是面向过程的语言，而C++是面向对象的语言
- C只能写面向过程的代码，而C++既可以写面向过程的代码，也可以实现面向对象的代码
 - C和强制类型转换上也不一样 const_cast static_cast reinterpret_cast dynamic_cast
 - C和C++的输入输出方式也不一样
 - C++引入 new/delete 运算符，取代了C中的 malloc/free 库函数；
 - C++引入引用用的概念
 - C++引入类的概念
 - C++引入函数重载的特性

静态链接和动态链接有什么区别？

静态链接
静态链接是在编译链接时直接将需要的执行代码拷贝到调用用处；
优点: 在于程序在发布时不需要依赖库，可以独立执行，
缺点: 在于程序的体积会相对较大，而而且如果静态库更新之后，所有可执行文件需要重新链接；

动态链接
动态链接是在编译时不直接拷贝执行代码，而是通过记录一系列符号和参数，在程序运行行或加载时将这些信息传递给操作系统，操作系统负责将需要的动态库加载到内存中，然后程序在运行行到指定代码时，在共享执行内存中寻找已经加载的动态库可执行代码，实现运行时链接；
优点 在于多个程序可以共享同一个动态库，节省资源；
缺点 在于由于运行时加载，可能影响程序的前期执行性能

指针和引用的区别

指针	引用(优先使用)
指针是地址，有存储空间	就是别名
指针可以有多个级	引用只能是一级
指针可以指向NULL	引用不可以为NULL
指针可以在定义的时候不初始化	引用必须在定义的时候初始化
指针初始化之后可以再改变指向	引用初始化后不可以再改变指向
sizeof 的运算结果不同,指针就是4/8字节	而引用是被引用对象的大小
指针作为函数参数时，指针需要检查是否为空。	引用作为函数参数时，引用不需要检查是否为空

指针	引用(优先使用)
指针使用自增运算是指针指向向后偏移一个存储单元	引用自增是将对应的值+1
	引用比指针多了类型检查

类和结构体的区别

类和结构体的区别： struct默认访问权限公有 class默认访问权限私有
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。
C++之所以要引入结构体，是为了保持和C程序的兼容性。

define与inline区别

相同点：拿空间换时间，提高程序的执行效率

inline	define
内联函数在编译时展开，	宏是由预处理器对宏进行展开
内联函数会检查参数类型，所以内联函数更安全	宏定义不检查函数参数。
inline是函数	宏不是函数
程序员设置的内联函数编译器不一定会满足要求，这取决于函数大小或者函数是否调用自身	宏在定义时要小心处理宏参数，（一般情况是把参数用括弧括起来）

前置++与后置++区别

后置++中tmp是一个临时对象，会造成一次构造函数和一次析构函数的额外开销
效率高：前置++，不产生临时对象

static在C和c++的区别

在C和C++都可以作用于局部变量 叫静态局部变量
在函数调用时，只有在该函数第一次调用时才对其分配空间和初始化。在函数调用结束时，不对该变量的内存进行释放，值仍然保留。这也是于自动变量的区别


在C和C++中都可以作用于全局变量和全局函数
表示该变量或者函数是私有的，只能在该文件使用。不能通过extern关键字对其引用。


修饰成员变量时，所有的对象都只维持一份拷贝，可以实现不同对象间的数据共享；不需要实例化对象即可访问；不能在类内部初始化，一般在类外部初始化，并且初始化时不加 static

修饰成员函数时，该函数不接受 this 指针，只能访问类的静态成员；不需要实例化对象即可访问

友元 friend

友元：让一个函数或者类，访问另一个类的私有成员(打破封装)
三种实现：
• 全局函数做友元
• 类做友元(友元类)
• 成员函数做友元

 技术咨询

 QQ名片 >