直击招聘

229道面试真题

直击招聘

程序员面试笔试
C++语言深度解析

◎ 李春葆 李筱驰 主编

清華大學出版社

直击招聘

直击招聘

程序员面试笔试
C++语言深度解析

◎ 李春葆 李筱驰 主编

清华大学出版社
北京

内 容 简 介

本书汇总国内外众多著名 IT 企业近几年的 C++面试笔试真题并予以解析，按知识点类型对常见的C++语言难点和疑点进行了系统归纳和透彻剖析，并提供了一定数量的自测题以便于读者自我检验。

全书逻辑清晰，通俗易懂，适合参加 IT 企业校园招聘和笔试面试环节的同学复习，也适合 C++语言编程爱好者和在校学生阅读与提高。

图书在版编目（CIP）数据

直击招聘：程序员面试笔试 C++语言深度解析 / 李春葆，李筱驰主编. —北京：清华大学出版社，2018
（直击招聘）
ISBN 978-7-302-48797-5

Ⅰ. ①直… Ⅱ. ①李… ②李… Ⅲ. ①C++语言-程序设计-资格考试-自学参考资料
Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字（2017）第 272570 号

责任编辑： 魏江江　王冰飞
封面设计： 刘　键
责任校对： 胡伟民
责任印制： 王静怡

出版发行： 清华大学出版社
网　　址：http://www.tup.com.cn, http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者： 三河市金元印装有限公司
经　　销： 全国新华书店
开　　本： 185mm×240mm　　**印　张：** 19.75　　**字　数：** 435 千字
版　　次： 2018 年 3 月第 1 版　　**印　次：** 2018 年 3 月第 1 次印刷
印　　数： 1～2000
定　　价： 69.50 元

产品编号：077558-01

出版说明

国内外许多著名的IT企业都采用“企业网申→测评→笔试→面试→发放录用意向书”的招聘流程，如阿里巴巴校园招聘网站“https://campus.alibaba.com/process.htm”发布了详细的招聘信息，申报各种技术类岗位（如研发工程师、算法工程师、前端开发工程师和测试开发工程师等）的同学必须经过笔试和面试环节。

尽管不同IT公司的笔试和面试方式存在差异，但考查点基本相同，除了语言表达能力、自我控制能力、人际关系处理能力和生活情趣等非工作技能外，从专业角度看，无非就是如下几点：

❶ 基本知识点掌握情况：面试者至少需要熟练掌握一门计算机语言，常用的有C/C++、Java等，这是作为程序员的基本功。所谓“熟练掌握”，就是不仅仅限于基本语法，还要理解语言的实现与运行时情况，如C语言中的指针、变量存储类别和函数执行过程等。

❷ 基本程序设计情况：考查面试者是否具有编程能力，包含编写代码、调试代码和测试代码的整个过程，能够做到“让代码说话”，不能只会夸夸其谈却不会编程。

❸ 算法设计能力：考查面试者求解问题的算法设计和实现能力，采用的算法策略是否合适，算法是否涵盖所有的测试用例。

❹ 计算逻辑思维能力：考查面试者是否具有一定的分析问题和解决问题的能力，用计算机求解问题时思维是否缜密，能否抓住问题的本质，采用的思路是否得当，条理是否清晰。

本丛书是面向参加笔试和面试的同学编写的，不仅对接大学计算机课程体系，而且结合IT企业招聘人才的基本行规，涵盖的内容如下：

- 程序员面试笔试C语言深度解析
- 程序员面试笔试C++语言深度解析
- 程序员面试笔试数据结构深度解析
- 程序员面试笔试算法设计深度解析

本丛书以C/C++语言为工具、数据结构为基础、算法设计为目标，其中《直击

招聘——程序员面试笔试 C 语言深度解析》和《直击招聘——程序员面试笔试 C++语言深度解析》侧重 C/C++语言的核心概念和深层次用法，《直击招聘——程序员面试笔试数据结构深度解析》侧重以常用数据结构为核心的算法设计，《直击招聘——程序员面试笔试算法设计深度解析》侧重通用的算法设计。本丛书具有如下特点。

❶ 定位准确：面向企业应聘人才，面向编程技术提高者。

❷ 答疑解惑：解析相关课程中的难点、疑点和热点，许多都是目前各大网站上的热门讨论话题。

❸ 实战性强：收集近些年的笔试和面试题目，涵盖常见考点。

本丛书虽然经过精心的编写与校订，但仍然难免有疏漏和不足之处，需要不断地补充、修订和完善，编者热情欢迎读者提出宝贵意见和建议，使之更臻成熟。

前　言

C++语言是 C 语言的扩展，是一种基础的面向对象的编程语言，提供了类、模板、函数重载和运算符重载设计等功能，充分支持抽象、继承和多态等面向对象程序设计的特征，方便大型软件的开发。

相比 C 语言，C++语言中的概念众多，掌握难点更高。实际上 C++对象在内存中的存储组织结构是核心，只要抓住了这个纲，对其他概念的理解就会迎刃而解。本书就是以它提纲挈领，系统归纳 C++语言常见的知识要点，汇总国内外众多著名 IT 企业近几年的 C++面试笔试真题并予以解析，透彻剖析了难点和疑点。

本书不是面向初学者，而是以知识点提纲挈领，章节之间难免会出现要点重复的现象，敬请读者谅解。书中侧重 C++语言的语法，相关算法设计在本丛书的其他书中讨论。另外，为了方便阅读，对于部分企业面试笔试中的文字和代码在格式上做了调整。书中程序在 VC++ 6.0 环境中调试通过（个别程序在 Dev C++中调试）。

在编写过程中参考了众多网站和博客，对此无法一一列出，编者对其作者表示衷心感谢。

限于编者水平，书中难免存在遗漏，恳请读者批评指正。

编　者

2017 年 10 月

目 录

·第1章·
C++中的C

常见考点

- C++语言的类型安全问题。
- const关键字和volatile关键字的作用。
- C++的显式转换，包括dynamic_cast、const_cast、static_cast和reinterpret_cast 4个显式转换的作用与区别。
- 内存管理：malloc/free和new/delete的区别、new/delete和new[]/delete[]的区别。
- C++函数设计：传引用参数、返回引用、默认参数值和函数重载。
- extern "C"链接指示符的作用。
- 断言及其应用。

1.1 类型系统和类型安全

1.1.1 要点归纳

⋙ 1. 类型系统

类型系统（type system）是一门编程语言最核心也是最基础的部分。一门计算机语言无论基于何种编程范式，都必须首先对类型系统作出明确的定义。

类型系统用于定义如何将编程语言中的数值和表达式归为许多不同的类型，如何操作这些类型，这些类型如何互相作用。在每一个编程语言中都有一个特定的类型系统，保证程序的行为良好，并且排除违规的行为。例如，C++语言中的类型如下。

- 数据类型：一个数据值的类型，如int、bool、char、double等。
- 类：一个对象的类型。
- 模板：一个类的类型。

类型系统在各种语言之间有非常大的不同，最主要的差异是编译阶段的类型检查以及执行阶段的操作实现方式。

1 强类型定义和弱类型定义语言

强类型定义语言（或者强类型语言）：在该类型语言中一旦一个变量被指定了某个数据类型，如果不经过强制转换，那么它就永远是这个数据类型了。例如，在C/C++语言中，如果定义了一个整型变量n，那么程序根本不可能将n当作字符串类型处理，如"hello"+n是错误的。强类型语言通常只允许以不丢失信息为前提的自动类型转换（如果采用强制转换可能丢失信息）。

弱类型定义语言（或者弱类型语言）：在该类型的语言中，在定义变量时可以忽略数据类型。它与强类型定义语言相反，一个变量可以赋不同数据类型的值，如JavaScript、PHP就是弱类型定义语言。

强类型定义语言在速度上可能略逊色于弱类型定义语言，但是强类型定义语言带来的严谨性能够有效地避免许多错误。

关于强类型和弱类型语言标准有不同的观点，有的从程序错误类别的角度出发认为C/C++是弱类型语言，这里采用上述大众化的观点，认为C/C++属于强类型语言。

2 动态类型语言和静态类型语言

静态类型语言：其程序中的数据类型是在编译阶段检查的，大多数静态类型语言都要求在使用变量之前定义它们的数据类型。由于在编译时就能发现数据类型错误，因此通常能增强最终程序的可靠性，程序有比较高的执行效率。C/C++是静态类型语言的典型代表，静态类型语言还有C#、Java等。

静态类型语言的优点是其程序结构非常规范，便于调试；向编译器提供更多有用的类型检查信息，便于优化，例如，如果一个类型指明其值必须以4的倍数对齐，编译器就可以使用更有效率的机器指令；另外，由于类型强制定义，很多静态类型语言提供了具有智能感知的集成开发环境，方便软件开发。

动态类型语言：与静态类型语言刚好相反，动态类型语言是指在执行期间才去做类型检查的语言。在动态类型语言中经常在执行阶段进行类型标记的检查，因为变量所约束的值可经由执行路径获得不同的标记。其优点是在用动态类型语言编程时人们不需要分心去考虑程序编程问题，而集中精力思考业务逻辑的实现。

静态类型语言为了达到多态性会采取一些类型鉴别手段，如继承、接口等，所以缺点是为此需要编写更多的类型相关代码，导致不便于阅读、不清晰明了。动态类型语言却不需要这样做，但也有缺点，就是不方便调试，当命名不规范时会造成读不懂、不利于理解等。

动态类型语言并非一定是弱类型语言，二者不是等价的，相反，动态类型语言一般都是强类型语言。C++是一种强类型语言，也是一种静态类型语言。

2. 类型安全

类型安全在很大程度上可以等价于内存安全，类型安全的代码不会试图访问自己没被授权的内存区域。

类型安全常被用来形容编程语言，其依据在于该门编程语言是否提供保障类型安全的机制，如果编程语言不允许导致错误的运算或转换，就认为该语言是类型安全的。实际上，静态类型语言并非就是类型安全的，因为有多少的类型错误发生以及有多少比例的错误能被静态类型所捕捉仍有争论。另外，是不是动态类型语言与这门语言是不是类型安全的完全不相干，不要将它们联系在一起。

有的时候也用类型安全来形容某个程序，判别的标准在于该程序是否隐含类型错误。

类型安全的编程语言与类型安全的程序之间没有必然联系。好的程序员可以使用类型不那么安全的语言写出类型相当安全的程序，相反，差一点儿的程序员可能使用类型相当安全的语言写出类型不太安全的程序。目前暂时还没有绝对类型安全的编程语言。

C/C++语言在局部上下文中表现出类型安全，例如试图从一种结构体的指针转换成另一种结构体的指针时编译器将会报告错误，除非使用显式类型转换。相对而言，C 比 C++存在更多不安全的隐式转换操作。例如：

```
int main()
{   printf("%f\n",10);
    return 0;
}
```

其中%f浮点数格式与整数 10 并不匹配。上述代码作为 C 程序可以编译通过，执行也没报错，但是输出结果却是 0.000000。上述代码作为 C++程序可以编译通过，但执行时出现程序崩溃。

所以，C/C++都不是类型安全的语言。

3. C++和 C 的差别

从类型系统角度看，C++的类型系统包含 C 的类型系统，增加了类和模板类型，所以 C 是一种结构化编程语言，而 C++是面向对象的编程语言。

1.1.2 面试真题解析

【面试题 1-1】C++是类型安全的语言吗？

A．是　　B．不是

答：C++不是类型安全的语言。例如可以将 0 作为 false、非零作为 true。一个函数即

使是 bool 类型的，也可以返回 int 类型的，并且自动将 0 转换成 false、非零转换成 true。答案为 B。

【面试题 1-2】弱类型语言是指不需要进行变量/对象类型声明的语言。（　　）属于弱类型语言。

A. Java　　B. C/C++　　C. Python　　D.C#

答：Java、C/C++和 C#都是强类型语言。答案为 C。

【面试题 1-3】C++和 C 有什么不同？

答：就语言本身而言，C 是 C++的一个子集，C++在 C 的基础上增加了类和模板类型。一方面 C++加强了 C 的过程化功能，引入了重载、异常处理等，另一方面更是扩展了面向对象设计的内容，如类、友元、继承、虚函数和模板等。

从编程角度看，C 是一种结构化编程语言，重点在于算法和数据结构，C 程序设计首要考虑的是如何通过一个过程（包含函数和参数等）对输入进行运算处理得到输出；而 C++是面向对象的编程语言，C++程序设计首要考虑的是如何构造一个对象模型，包括数据封装、类、消息、对象接口和继承等，让这个模型能够契合与之对应的问题域，这样就可以通过获取对象的状态信息得到输出或实现过程控制，所以两者的区别在于解决问题的思想方法不一样。之所以说 C++比 C 更先进，是因为“设计这个概念已经被融入 C++之中”。

1.2 const 和 volatile

1.2.1 要点归纳

1. const

const 关键字用于修饰普通变量和指针变量。如果 const 修饰变量名，表示该变量为常变量，不能修改常变量值；如果 const 修饰*，表示该指针变量指向的是常量，不能通过该指针变量修改指向的内容，但该指针变量的值可以修改。例如：

```
const int n=10;                 //const 修饰普通变量，表示 n 为常量，不能修改 n
int const n=10;                 //与 const int n=10 相同
int * const p=&n;               //const 修饰指针变量名，表示 p 为常量，不能修改 p 的值
int const *q=&n;                //const 修饰*，表示 q 指向常量，不能修改 q 指向的内容
const int *q=&n;                //与 const int *q=&n 相同
const int const *r=&n;          //具有前面 p、q 的特性
```

const 关键字用于修饰指针变量时，如果 const 位于*的左侧，则 const 就是用来修饰指针变量指向的内容，即指针变量指向的内容为常量；如果 const 位于*的右侧，const 就是修饰指针变量本身，即指针变量值是常量。

需要注意的是，const 在 C 和 C++中略有不同。在 C 中，C 编译器不把 const 常量看成一个编译期间的常量，为其分配内存空间，但 C 编译器不知道它在编译时的值，因此以下 C 程序会出现“cannot allocate an array of constant size 0”的错误：

```
const int n=10;
int a[n];              //编译时不能确定 n 值，将其看成 0
```

在 C++中上述程序没有错误，说明 C++编译器在编译时知道 const 常量的值。

2. volatile

volatile 关键字的含义是“易变的”，它告诉编译器 volatile 变量是随时可能发生变化的，与 volatile 变量有关的运算不要进行编译优化，以免出错，因为一般编译器会进行编译优化。例如：

```
volatile int i=10;
int j=i;
...
int k=i;
```

其中，i 是 volatile 变量，告诉编译器 i 是随时可能发生变化的，每次使用它的时候必须从 i 的地址中读取，因此编译器生成的可执行代码会重新从 i 的地址读取数据放在 k 中。如果没有 volatile 关键字，编译器的优化做法是由于发现两次从 i 读数据的代码之间的代码没有对 i 进行过操作，它会自动把上次读的数据放在 k 中，而不是重新从 i 里面读。这样一来，如果 i 是一个寄存器变量或者表示一个端口数据就容易出错，所以说 volatile 可以保证对特殊地址的稳定访问，不会出错。

例如，下面的函数有什么错误？

```
int square(volatile int *ptr)
{
    return *ptr * *ptr;
}
```

上述函数的功能是返回指针变量 ptr 指向值的平方，但是由于*ptr 指向一个 volatile 型参数，编译器将产生类似下面的代码：

```
int square(volatile int *ptr)
```

```
{   int a,b;
    a=*ptr;
    b=*ptr;
    return a*b;
}
```

由于 ptr 指向的值可能被意外地修改，因此 a 和 b 可能是不同的。结果，这段代码可能返回的不是所期望的平方值，正确的代码如下：

```
int square(volatile int *ptr)
{   int a;
    a=*ptr;
    return a*a;
}
```

在嵌入式编程中经常用到 volatile 关键字，主要用法可以归结为以下两点：告诉编译器不能做任何优化；由于用 volatile 定义的变量会在程序外被改变，每次都必须从内存中读取，所以不能把它放在 Cache 或寄存器中重复使用。

1.2.2 面试真题解析

【面试题 1-4】说明 const char *p 和 char * const p 两个定义的区别。

答：在定义 const char *p 中，const 位于星号的左侧，表示指针变量 p 指向的内容是常量，不能用 p 变量修改常量值，如*p='a'或者 p[1]='a'都是错误的，但 p 值可以修改，如 p=&n。在定义 char * const p 中，const 位于星号的右侧，表示 const 修饰指针变量 p 本身，p 指向的内容可以修改，但 p 值不能修改，如 p=&n 是错误的。

【面试题 1-5】总结 const 的应用和作用。

答：const 的应用和作用如下。

❶ 若要阻止一个变量被改变，可以使用 const 关键字。在定义该 const 变量时通常需要对它进行初始化，因为以后就没有机会再去改变它了。

❷ 对指针来说，可以指定指针本身为 const，也可以指定指针所指的内容为 const，或两者同时指定为 const。

❸ 在一个函数定义中，const 可以修饰形参，表明它是一个输入参数，在函数内部不能改变其值。

❹ 对于类的成员函数，若指定其为 const 类型，则表明其是一个常成员函数，不能修改类的数据成员。

❺ 对于类的成员函数，有时候必须指定其返回值为 const 类型，以使其返回值不为“左值”。

【面试题 1-6】说明 volatile 关键字的作用。

答：用 volatile 关键字修饰的变量确保编译器不对其代码进行优化，且要求每次直接从内存读值。

【面试题 1-7】一个指针可以是 volatile 吗？

答：可以，因为指针和普通变量一样，有时也可能会被意想不到地改变。例如，中断服务子程序修改一个指向 buffer 的指针时需要用 volatile 来修饰这个指针。

【面试题 1-8】给出几个使用 volatile 关键字的示例。

答：一个定义为 volatile 的变量是说这个变量可能会被意想不到地改变，在用到这个变量时必须每次都小心地重新从内存中读取这个变量的值，而不是使用保存在 Cache 或者寄存器里的备份。使用 volatile 变量的几个示例如下：

❶ 并行设备的硬件寄存器（如状态寄存器）。

❷ 一个中断服务子程序中会访问到的非自动变量。

❸ 多线程应用中被几个任务共享的变量。

1.3 C++的显式类型转换

1.3.1 要点归纳

C++中的类型转换（cast）就是告诉编译器“忘记类型检查，把它看成是其他类型”，也就是说在 C++类型系统中引入一个漏洞，并阻止编译器报告在该类型方面的错误。

标准 C++包括一个显式的转换语法，使用它可以完全替代旧式 C 风格的转换。显式转换语法使得程序员很容易发现它们，因为通过它们的名字就能找到。

⋙ 1. 静态转换（static_cast）

static_cast 全部用于明确定义的转换，包括编译器允许不用强制转换的“安全”转换和不太安全但清楚定义的转换，如窄化转换（可能有信息丢失）、使用 void *的强制转换和隐式类型转换等。例如：

```
int i=0x7fff;
long l;
float f;
```

```
//(1)非强制转换：可以不用 static_cast，但使用时突出转换行为
l=i;
f=i;
```