直击招聘

302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 编著

定位准确 面向企业应聘人才，面向编程技术提高者。

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

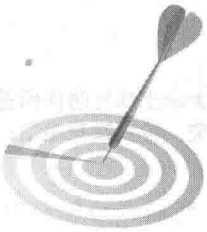
实战性强 收集近些年笔试和面试题目，涵盖常见考点。

清华大学出版社

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 编著

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语言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企业校园招聘和面试笔试环节的同学复习，也适合C语言编程爱好者和在校学生阅读和提高。

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，李筱驰编著. —北京：清华大学出版社，2018
（直击招聘）

ISBN 978-7-302-48798-2

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272572号

责任编辑： 魏江江　王冰飞
封面设计： 刘　键
责任校对： 焦丽丽
责任印制： 杨　艳

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： 北京鑫海金澳胶印有限公司
经　　销： 全国新华书店
开　　本： 185mm×240mm　　**印　张：** 19.25　　**字　数：** 418千字
版　　次： 2018年2月第1版　　**印　次：** 2018年2月第1次印刷
印　　数： 1～2000
定　　价： 69.80元

产品编号：077570-01

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，但考察点基本相同，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，包括目前各大网站上热门讨论的话题。

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，编者热切欢迎读者提出宝贵的意见和建议，使之更臻成熟。

前　言

C 语言是计算机及相关专业的必修课程，是许多面试者学习的第一门计算机语言，也是绝大多数 IT 企业面试笔试的内容之一。很多面试笔试题看起来简单，实际上却隐含着奥秘和某个深入的知识点，这些往往是在课堂上难以学到的。本书系统归纳 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 语言面试笔试真题并予以解析，透彻剖析了难点和疑点。

由于 IT 企业面试笔试的编程环境一般采用 C++，本书主要在 VC++ 6.0（之所以采用这种“古老”的编译器，是考虑程序代码的兼容性，通常高版本的编译器是兼容低版本的）中调试程序（个别程序在 Dev C++中调试），并且仅仅涵盖 C 语言部分。在 VC++6.0 中，源程序文件可以采用.c 扩展名，也可以采用.cpp 扩展名，前者遵循 C 语言语法，后者遵循 C++语法。尽管 C++语言支持 C 语言，但二者略有差异，例如 C 语言中包含更多的隐式转换，所有变量必须在执行语句之前定义。书中的程序调试除特别说明外主要采用后者。

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。

在编写过程中参考了众多网站和博客的有关内容，无法一一列出，编者在此表示衷心感谢。

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·
程序设计基础——变量

常见考点

- 变量的作用域。
- 变量的存储类别。
- 变量在内存中的存储方式。
- 变量声明和变量定义的区别。
- 运算符的优先级和结合性及其应用。
- 不同类型数据之间的转换方式。
- 表达式求值的副作用。

1.1 变量定义和声明

1.1.1 要点归纳

⋙ 1. 定义变量就是使用内存

计算机的主要资源是内存和CPU，程序员通过定义变量来使用内存。假设内存按字节编码，内存大小为4GB（4GB/1B=1G=2^{32}个），则内存地址编号为0～2^{32}-1，用32个二进制位（4B）表示。现在定义如下变量n：

```
int n=10;
```

系统为变量n分配存储空间，若int类型占用4个字节，其起始字节的内存地址为十六进制0x0000000C，如图1.1所示，这个存储单元中存放值10。存储单元是可读和可写的，这样程序员就可以通过n变量名来操作这个存储单元，这种逻辑化方式大大简化了内存的使用，称之为“透明性”。

如果一个变量占用内存空间的多个内存字节，其第1个字节地址就是它的存储地址。

简单地说，计算机中的透明性是指计算机中存在的，但对于程序员而言又不需要了解的东西。例如编写C程序的程序员，不需要了解加法指令是如何工作的，只要会用就可以

了，即加法指令对 C 程序员是透明的。试想象一下，如果不是通过变量名使用内存，而是要记住内存地址，那是多么痛苦的事情。

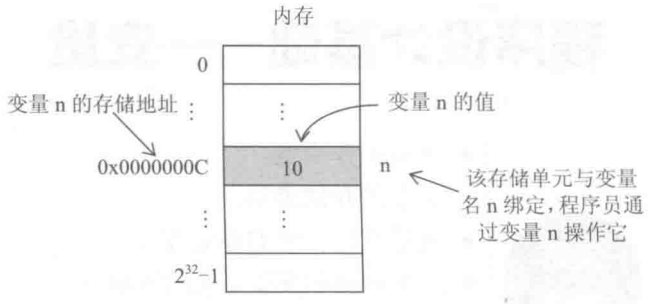


图 1.1　变量 n 的定义

因此，可以简单地理解为变量就是内存中的一个箱子，箱子的名称就是变量名。用户可以向其中放入数据，也可以取出其中的数据。

要想知道计算机中的每个 int 类型数据占用几个字节，一个简单的程序就搞定了：

```
void main()
{   int n=10;
    int *p=&n;
    printf("%d,%x\n",sizeof(p),p);
}
```

例如，上述程序输出“4,12ff44”，表示每个 int 类型数据占用 4 个字节，变量 n 分配的内存地址为 0x12ff44（这 4 个字节中存放整数 10），准确地说是 0x0012ff44（共 4×8=32 位，输出时省略了前面的两个 0）。

▷▷▷ 2. C 语言数据类型

变量用于存储数据，而数据有各种类型，有的占用一个内存单元，有的占用多个内存单元。如果让程序员在定义变量时指定内存单元的个数，那是十分麻烦的。因此，在 C 语言中预先设置好了一些“模板”，即数据类型，程序员只需要指定变量的数据类型，系统就为其分配合适的存储空间。图 1.2 所示为 C 语言的数据类型，其中基本类型也称为内置数据类型。

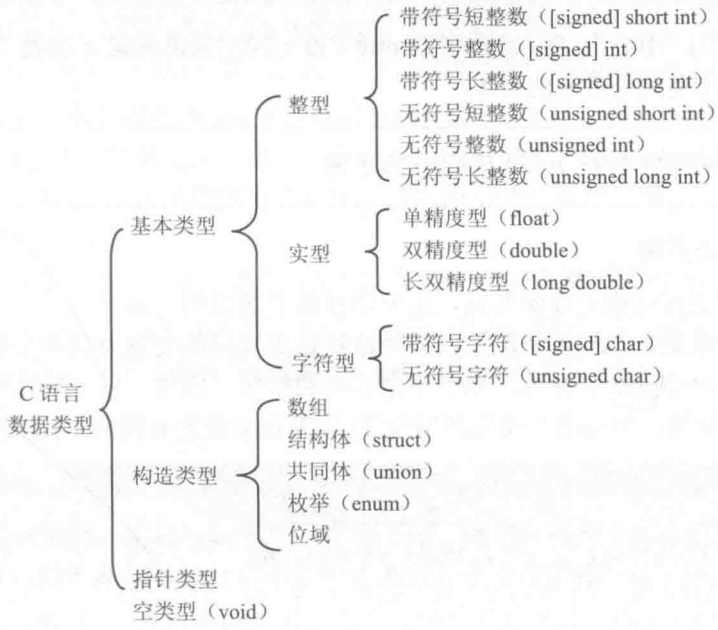
因此，在定义变量时至少需要指定变量的数据类型和变量名，可以赋初值（称为初始化），也可以不赋初值。

▷▷▷ 3. 变量的作用域和变量在内存中的存储方式

1 局部变量和全局变量

变量的作用域指变量的有效范围。在一个程序中可能定义多个变量，根据其定义位置

分为局部变量和全局变量。



说明：

- long 和 short 修饰符用于改变数据类型的最大值和最小值。
- signed 和 unsigned 修饰符告诉编译器如何使用整数和字符的符号位，后者能够存储比前者大两倍的正数。signed 是默认的。

图 1.2　C 语言数据类型

- **局部变量**：在函数内部定义的变量为内部变量，它只在本函数范围内有效，在该函数以外不能使用这些变量，称之为局部变量。所以局部变量的作用域仅限于定义它的函数。
- **全局变量**：在函数之外定义的变量为外部变量，它的作用域为从定义变量的位置开始到本源程序文件结束，称之为全局变量。所以全局变量的作用域仅限于定义它的源程序文件，在本源程序文件的所有函数中都可以使用其中定义的全局变量。

例如：

```
int n=1;                    //全局变量 n
void fun()
{   int n=10;               //局部变量 n
    printf("%d,",n);        //输出局部变量 n 的值 10
}
void main()
{   fun();
```

```
    printf("%d\n",n);    //输出全局变量 n 的值 1
}
```

上述程序输出为“10,1”。fun 函数中的 printf 语句输出局部变量 n 的值 10，而主函数中的 printf 语句输出全局变量 n 的值 1。

在相同作用域内不允许出现相同的变量名。

2 变量的存储类别

在定义变量时还可以指定存储类别，主要的存储类别说明符如下。

❶ auto：自动变量，在缺省情况下，编译器默认所有局部变量为自动变量，这种变量的存储空间由系统自动分配和释放，系统不会自动初始化。例如，以下程序执行时会输出 n 中没有意义的垃圾值，如输出“-858993460”，因为自动变量 n 没有初始化。

```
void main()
{   int n;                    //n 默认为自动变量
    printf("%d\n",n);         //输出：-858993460
}
```

❷ register：寄存器变量，变量值存放在 CPU 的内部寄存器中，存取速度最快，通常只将需要频繁读的变量定义为寄存器变量，这类变量不能进行取变量地址操作（有的编译器会做优化处理，将计算量不大的寄存器变量也作为自动变量处理，如 Visual C++ 6.0）。由于寄存器个数有限，所以一般不能在程序中定义很多寄存器变量。另外，只有局部自动变量和函数形参才可以定义为寄存器变量。

❸ extern：外部变量，外部变量和全局变量是对同一类变量的两种不同角度的提法。全局变量是从作用域提出的，外部变量是从存储类别提出的。

❹ static：静态变量，在函数内部用 static 关键字定义的变量称为静态局部变量，在函数外部用 static 关键字定义的变量称为静态全局变量。

在程序执行期间，静态局部变量在内存的静态存储区中占据着永久性的存储单元。即使退出函数以后，下次再进入该函数时，静态局部变量仍使用原来的存储单元。对于在定义时初始化的静态局部变量，初始化仅仅执行一次；对于未初始化的静态局部变量，C 编译系统自动给它赋初值 0。

例如：

```
void fun()
{   static int n;             //定义静态局部变量 n，并自动初始化为 0
    n++;
    printf("%d\n",n);
}
void main()
```

```
{   fun();                  //调用时输出 1
    fun();                  //调用时输出 2
    fun();                  //调用时输出 3
}
```

在主函数中，第 1 次调用 fun()时，初始化静态局部变量 n 为 0，执行 n++，输出为 1。第 2 次调用 fun()时，静态局部变量 n 仍然存在，其值为 1，执行 n++，输出为 2。第 3 次调用 fun()时，输出为 3。在多次调用 fun()时，可以简单地理解仅仅第 1 次执行“static int n;”语句。

在一般情况下，全局变量默认是外部的，即定义全局变量 n 的如下两种方式是等价的：

```
int n=1;
extern int n=1;   //如果不初始化，就变成声明变量 n 了
```

如下方式定义的全局变量 n 为静态全局变量：

```
static int n=1;
```

静态全局变量的作用域只限于本源程序文件。静态全局变量和普通全局变量的区别是静态全局变量只能初始化一次，以防止在其他源程序文件中被访问。

归纳起来，auto 和 register 变量属于动态存储类别的变量，而 extern 和 static 变量属于静态存储类别的变量。

3 内存组织结构

尽管内存空间很大，但程序只能直接访问其中的一部分空间，内存的低地址空间被操作系统占用，其余的高地址空间划分为 4 个部分，大致结构如图 1.3 所示，其说明如下。

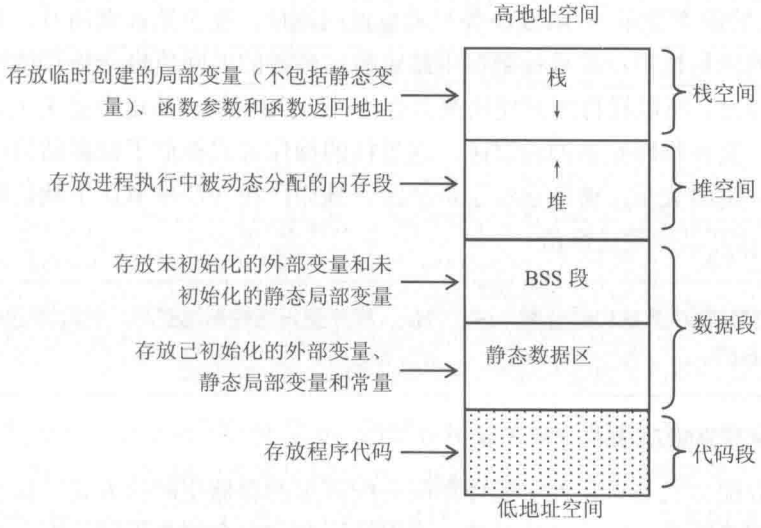


图 1.3　内存组织结构

❶ **代码段**（code segment）：用来存放程序执行代码（编译后的代码），又称为文本段（text segment）。这部分区域的大小在程序执行前就已经确定，并且该内存区域属于只读。

❷ **数据段**（data segment）：用于存放程序执行中使用的那些变量的值，包含以符号开始的块（Block Started by Symbol，BSS）段和静态数据区两个部分。BSS 段用来存放程序中未初始化的外部变量和未初始化的静态局部变量。在执行程序时，BBS 段会预先清空，所以存放在 BSS 段中的变量均默认初始化为 0。这就是为什么外部变量和静态局部变量可以不初始化，但是会被赋予默认值 0；另外的静态数据区用来存放程序中已初始化的外部变量、静态局部变量和常量。

❸ **堆空间**（heap segment）：堆是用于存放进程（进程简单理解为程序的一次执行）执行中被动态分配的内存段，它的大小并不固定，可动态扩张或缩减。当进程调用 malloc()等函数分配内存时，新分配的内存就被动态添加到堆上（堆被扩张）；当利用 free()等函数释放内存时，被释放的内存从堆中被剔除（堆被缩减）。从堆分配的内存仅能通过指针访问。这里的堆与数据结构中队列的概念是不同的。

一般来讲在 32 位系统中堆内存可以达到 4GB 的空间，从这个角度来看堆内存几乎是没有什么限制的。对于堆空间频繁地 malloc/free 会造成堆空间的不连续，从而造成大量的碎片，使程序效率降低。

堆空间是由动态分配函数分配的内存空间，一般速度比较慢，而且容易产生内存碎片，通常堆空间是向高地址方向生长的。程序员可以方便地管理堆空间。

❹ **栈空间**（stack segment）：栈又称堆栈，存放程序中临时创建的局部变量（但不包括用 static 定义的静态变量）、函数参数和函数返回地址。在函数被调用时，其参数也会被压入发起调用的进程栈中，并且待到调用结束后，函数的返回值也会被存放回栈中。由于栈的先进后出特点，所以栈特别方便用来保存/恢复调用现场。从这个意义上讲，可以把栈看成一个存放、交换临时数据的内存区。这里栈的操作方式类似于数据结构中的栈。一般地，栈空间有一定的大小，通常远小于堆空间。例如，在 VC++ 6.0 中默认的栈空间大小是 1MB，程序员可以修改这个值。

栈空间由系统自动分配，速度较快，程序员无法控制栈空间。通常栈空间是向低地址方向生长的。

4 变量静态分配和动态分配方式

变量静态分配方式是指在程序编译期间分配固定的存储空间的方式。该存储分配方式通常是在变量定义时就分配存储单元并一直保持不变，直至整个程序结束。例如：