

精选380余个C/C++技术面试真题，详解应聘C/C++程序员的常见考点
揭秘知名IT企业用人的核心准则，解读应聘IT企业的成功要诀



C/C++程序员 面试宝典

另赠超值
学习视频



16.5小时多媒体教学视频

梁镇宇 等编著

清华大学出版社

第8章 指针

在 C++ 中，很多程序员都痴迷于指针的运用，很多公司在招聘较高级职位的时候，也会考察到指针的相关知识，对指针的运用能看出程序员编写程序的严谨性。指针是用来控制对象的内存地址，它功能非常强大，可以直接访问和操作系统内存，合理地运用指针也会让程序的性能得到很好的优化。

8.1 指针概述

在 C++ 中，语言可以在运行时获得变量的地址，并且具有操作地址的能力。这种功能在其他的语言中可能都不如 C++ 中如此的重要，被用来操作变量地址的特殊类型的变量就是指针变量。指针可以用于数组，或者作为函数的参数，用来访问内存和对内存的操作。由于指针的作用，使得 C++ 的功能非常强大，它可以使程序变得非常高效，但是指针使用起来又比较危险，使用不当会导致程序出现比较严重的问题。本节将介绍指针的相关使用方法，以及指针与数组、字符串之间的紧密联系。

面试题 83 什么是指针

【出现频率】★★★

【关键考点】

□ 指针的概念

【考题分析】

程序中的所有变量和常量都存在一个内存地址中。这个内存地址表示变量或者常量在内存中存储的位置，同样，函数也有对应的内存地址。内存地址的不同会导致程序执行时有所不同，指针就是用来存储内存地址的变量。

编程者所知道的所有基本数据类型，例如，`int`、`float`、`double` 等，每一种基本数据类型都有相应的指针类型，编程者可以建立对应类型的指针来处理基本数据类型。

“*”在运算符中是表示乘法，它也被用来定义指针。指针变量的定义语句，由数据类型后跟星号，再跟指针变量名组成。定义指针的示例代码如下：

```
int * ip;
const int * ip2
```

上面所示 `ip` 和 `ip2` 都是指针变量名，`int` 表示该指针变量的类型是整型，*表示是指针变量。指向整型数的指针是包含该整型数地址的变量或常量。另外 C++ 还提供了一种特殊

的指针类型 `void*`，它可以保存任何类型对象的地址。示例代码如下：

```
double obj = 3.14;
double *pd = &obj;           //double 类型指针
void *pv = &obj;              //void 类型指针，在这里保存了 double 类型指针地址
pv = pd;
```

【答案】

指针是用来存储内存地址的变量，它指向单个对象的地址，除了 `void` 指针类型以外，指针的数据类型与所指向地址的变量数据类型须保持一致。

面试题 84 如何初始化指针并对其赋值

【出现频率】★★★★

【关键考点】

- ☐ 如何初始化指针；
- ☐ 如何对指针进行赋值。

【考题分析】

建立指针包括定义指针变量和给指针变量赋初值，第一次给指针变量赋值就是指针的初始化。用 `&` 符号可以获取变量的地址，指针变量用来存储变量的地址，基本的指针变量初始化示例代码如下：

```
int * ipstr;
int counta = 20;
ipstr = &counta;
int *p = 0;           //初始化指针为 0
```

以上代码完成了提取一个变量的地址，并把它存储在一个指针变量中。如果 `counta` 的地址是 `0000:F233`，这个时候指针变量 `ipstr` 就赋值了 `counta` 的地址 `0000:F233`。一般来说，编程者需要尽量避免使用没有初始化的指针，因为这样很容易导致不可预料的运行错误。如果可能，除非所指向的对象已经存在，否则不要先定义指针，这样可以避免定义一个未初始化的指针。

如果编程者没有可以给指针初始化的地址值，编程者可以把指针初始化为 `0`（在 C 语言中为 `NULL`），这样指针不会指向任何实体，可以避免指针未初始化的问题。没有初始化的指针指向是随机的，它有可能导致随机修改了程序的值。

指针的赋值和初始化一样，可以通过赋值运算符“`=`”来完成，同样是通过 `&` 符号来获取变量的地址，并且把它赋值给指针变量，变量的数据类型需要和指针的数据类型兼容，示例代码如下：

```
long test = 123;
long * testip;
testip = &test;       //获取变量的地址
```

这样的赋值语句不是正确的，虽然它也给指针变量进行了赋值，但并不是编程者需要达到的目的，代码如下：

```
long test = 123;
long * testip;
testip = 123;           //赋值为123
```

这样给 `testip` 指向的内存为 123，而不是变量 `test` 的内存地址。这样程序在编译的时候不会发生错误，但是很可能导致程序会修改未知区域的内存。

【答案】

指针的初始化就是给指针赋初值，&符号可以用来获取对象的内存地址，并且赋值给指针变量。指针变量的初始化和赋值都可以通过运算符“=”来实现。

面试题 85 是否可以确定指针指向一个对象

【出现频率】★★★

【关键考点】

□ 指针的用途

【考题分析】

指针用于指向对象。与迭代器一样，指针提供对其所指对象的间接访问，只是指针结构更通用一些。与迭代器不同的是，指针用于指向单个对象，而迭代器只能用于访问容器内的元素。具体来说，指针保存的是另一个对象的地址。示例代码如下：

```
string s("hello world");
string *sp = &s;           // sp 保存了 s 的地址
```

【答案】

指针用于指向对象，一个指针只指向一个对象的内存地址。

面试题 86 如何使用指针操作数组

【出现频率】★★★★

【关键考点】

□ 指针对数组的操作

【考题分析】

在 C++ 语言中，指针和数组的关系很密切。特别是如果在表达式中使用数组名时，该数组名会自动转换为指向数组第一个元素的指针。示例代码如下：

```
int ia[] = {0,2,4,6,8};
int *ip = ia;           //指针 ip 指向数组第一个元素 ia[0] 的地址
```

如果希望让指针指向数组中的另一个元素，则可以先使用下标操作符给某个元素定位，然后使用地址操作符&获取该元素的存储地址。示例代码如下：

```
int ia[] = {0,2,4,6,8};
int *ip = &ia[4];        //指针 ip 指向数组第 5 个元素 ia[4] 的地址
int *ip1 = &ia[3];
int *ip2 = &ia[1];
int diff= ip1-ip2;
```


上面代码中对指针进行了一个相减的计算，diff 的值为 2，因为地址之间的差是由元素而不是字节来决定的。

编程者还可以对多维数组使用指针，使用指针存储一维数组相对来说是比较简单的，但是用来存储多维数组就有些复杂了，声明的示例代码如下：

```
double beans[5][5];  
double pbeans = *beans[0][0];
```

这样把指针设置为了 double 类型的第一个元素的地址，还可以把指针设置为数组第一行的地址：

```
double * pbeans = beans[0];
```

 **注意：**在 C++ 中，建议尽量避免使用指针和数组，指针和数组容易产生不可预料的错误。

其中一部分是概念上的问题，因为指针用于低级操作，容易产生与繁琐细节相关的错误，而其他错误则源于使用指针的语法规则，特别是声明指针的语法。许多有用的程序都可以不使用数组或指针实现，现代 C++ 程序采用 vector 类型和迭代器取代一般的数组、采用 string 类型取代 C 风格字符串。

【答案】

在 C/C++ 中，指针对于数组的操作是通过将数组的地址，通常是第一个数的地址赋值给指针来进行操作的。指针可以操作一维和 multidimensional 数组。

面试题 87 const 对象的指针和 const 指针的区别

【出现频率】★★

【考点】

- ☐ const 对象的指针概念；
- ☐ const 指针的概念。

【考题分析】

指针和 const 限定符之间的两种交互类型：指向 const 对象的指针和 const 指针。指向 const 对象的指针是指指针指向 const 对象的地址，编程者使用指针来修改其所指对象的值。但是如果指针指向 const 对象，则不允许用指针来改变其所指的 const 值，因为 const 值是不可以进行修改的。为了保证这个特性，C++ 语言强制要求指向 const 对象的指针也必须具有 const 特性，指向 const 对象的指针示例代码如下：

```
const double *cptr; //cptr 指针可能指向一个类型为 double 的常量值
```

在这里的 cptr 是一个指向 double 类型的 const 对象的指针，const 限定了 cptr 指针所指向的对象类型，而并非 cptr 本身。cptr 本身并不是 const，在定义时不需要对它进行初始化，也可以给 cptr 重新赋值，使其指向另一个 const 对象。但不能通过 cptr 修改它所指的 const 对象的值。

另外，不可以使用 void* 指针保存 const 对象的地址，而必须使用 const void* 类型的指针来保存 const 对象的地址。示例代码如下：

```
const int universe = 42;
const void *cpv = &universe;    //正确: cpv 是一个常量
void *pv = &universe;           //错误: universe 是一个常量
```

除指向 `const` 对象的指针外, C++语言还提供了 `const` 指针。`const` 指针自身的值不能被修改。示例代码如下:

```
int a = 0;
int *const cura = &a;           //cura 是一个 const 指针
```

上面所示 `cura` 是指向 `int` 型对象的 `const` 指针。与其他 `const` 变量一样, `const` 指针的值不能修改, 这就意味着不能使 `cura` 指向其他对象。任何企图给 `const` 指针赋值的行为(即使给 `cura` 赋回同样的值)都是不合法的。`const` 指针和其他 `const` 变量一样, 必须在定义的时候进行初始化。

【答案】

`const` 指针的值不可以被修改, 但是可能可以使用该指针修改它所指向对象的值。指针所指对象的值能否修改完全取决于该对象的类型, 而指向 `const` 变量的指针不可以修改所指向的 `const` 变量的值, 但是自身可以被重新赋值。

面试题 88 数组指针与指针数组的区别

【出现频率】★★★★

【关键考点】

- ☐ 数组指针的概念;
- ☐ 指针数组的概念。

【考题分析】

指向一个数组的指针就是数组指针。定义数组指针的示例代码如下:

```
int (*ap)[2];
```

以上代码定义了一个指向包含有两个元素的数组的数组指针。

而如果一个数组的每一个元素都是指针, 则这个数组是一个指针数组。定义指针数组的示例代码如下:

```
char *chararr[] = {"Fortan", "C", "C++", "Basic"}
```

以上代码定义了一个指针数组并且对其进行了初始化, `chararr` 数组的每个元素都存放着一个字符指针, 初始化时每个值都是一个字符串常量, 而对应的字符指针存储了 3 个字符第一个字母在内存中的位置。

很多时候, 使用指针数组来控制程序可以节约内存空间, 也可以节约时间。一个使用指针数组的示例代码如下:

```
#include <stdlib.h>
#include <stdio.h>
main ( )
{
    char *pchar1[4]={ "china", "chengdu", "sichuang", "chongqin" };
    //指针数组 pchar1 的 4 个指针分别依此指向 4 个字符串
```

```
int i,* pchar2[3],a[3]={1,2,3},b[3][2]={1,2,3,4,5,6};
for(i=0;i<4;i++)
{
    printf("\n%s", pchar1 [i]);
    //输出 pchar1 数组 4 个指针指向的 4 个字符串
    printf("\n");
}
for(i=0;i<3;i++)
{
    pchar2 [i]=&a[i];    //将整型一维数组 a 的 3 个元素的地址传递给指针数组 pchar2
    for(i=0;i<3;i++)    //依次输出 pchar2 所指向的 3 个整型变量的值
    {
        printf("%4d",* pchar2 [i]);
        printf("\n");
    }
}
for(i=0;i<3;i++)
{
    pchar2 [i]=b[i];    //传递二维数组 b 的每行首地址给指针数组的 4 个指针/
    for(i=0;i<3;i++)    //按行输出
        printf("%4d%4d\n",* pchar2 [i],* pchar2 [i]+1);
}
}
```

数组指针和指针数组是比较难的问题，很多考官借此类问题来考查程序员的编程思维和严谨度。

【答案】

数组指针是一个指针变量，它指向一个数组。而指针数组是一个只包含指针元素的数组，它的元素可以指向相同类型的不同对象。

8.2 函数指针

在程序运行中，函数是程序的算法指令部分，它们和数组一样也占用存储空间，也都有相应的地址。编程者可以使用指针变量指向数组的首地址，同样，也可以使用指针变量指向函数代码的首地址，指向函数代码首地址的指针变量称为函数指针。

面试题 89 什么是函数指针？如何使用函数指针

【出现频率】★★★

【关键考点】

- ☐ 函数指针的概念；
- ☐ 函数指针的使用。

【考题分析】

函数指针就是指向函数的指针。像其他指针一样，函数指针也指向某个特定的类型。函数类型由其返回类型及形参表确定，而与函数名无关。函数指针的示例代码如下：

```
int (*f)(int x);
double (*ptr)(double x);
```

由于“（）”运算符的优先级高于“*”，所以指针变量名外的括号必不可少，后面的“形参列表”表示指针变量指向的函数所带的参数列表。函数指针和它指向的函数的参数个数和类型必须保持一致，函数指针的类型和函数的返回值类型也必须保持一致。

函数指针的使用主要包括函数指针的赋值和通过函数指针调用函数，函数名和数组名一样代表了函数代码的首地址，因此在赋值时，是直接将函数指针指向函数名。函数指针的赋值示例代码如下：

```
Int func(int x);      //声明一个函数
int (*f) (int x);     //声明一个函数指针
f = func;             //将 func() 函数的首地址赋给指针 f
```

赋值时函数 `func` 不带括号，也不带参数，由于 `func` 代表函数的首地址，因此赋值以后，指针 `f` 就指向函数 `func(x)` 的代码的首地址。

函数指针是通过函数名及有关参数对函数进行调用的。与其他指针变量相类似，如果指针变量 `pi` 是指向某整型变量 `i` 的指针，则 `*p` 等于它所指的变量 `i`；如果 `pf` 是指向某浮点型变量 `f` 的指针，则 `*pf` 就等价于它所指的变量 `f`。同样，如果 `f` 是指向函数 `func(x)` 的指针，则 `*f` 就代表它所指向的函数 `func`。所以在执行了 `f=func` 之后，`(*f)` 和 `func` 就代表同一个函数。由于函数指针是指向存储区中的某个函数，因此可以通过函数指针调用相应的函数。通过函数指针调用函数的示例代码如下：

```
main()
{
    int f();
    int i, a, b;
    int (*p) ();           //定义函数指针
    scanf("%d", &a);
    p=f;                   //给函数指针 p 赋值，使它指向函数 f
    for(i=1;i<9;i++)
    {
        scanf("%d", &b);
        a=(*p) (a, b);     //通过指针 p 调用函数 f
    }
    printf("The Max Number is:%d", a)
}

f(int x, int y)           //定义比较函数
{
    int z;
    z=(x>y)?x:y;
    return(z);
}
```

上面代码的运行结果如下：

```
343 -45 4389 4235 1 -534 988 555 789
The Max Number is: 4389
```

【答案】

函数指针就是指向函数的存储空间地址的指针。可以对函数指针进行赋值并且通过函

数指针来调用函数。

面试题 90 指针函数和函数指针的区别

【出现频率】★★★

【关键考点】

- 指针函数概念;
- 指针函数的使用。

【考题分析】

函数不仅可以返回整型、字符型等数据类型的数据,还可以返回指针类型的数据,使其指向某个地址单元。返回指针的函数称为指针函数。指针函数的定义示例代码如下:

```
Int *test(x, y);
```

上面代码中 `x, y` 是形式参数, `test()` 是函数名,调用函数后会返回一个指向整型数据的地址指针。`test(x, y)` 是函数,它的值是指针。

指针函数可以返回指针值,它的使用示例代码如下:

```
main()
{
    char *ch(char *, char *);           //调用函数
    char str1[]="I am glad to meet you!";
    char str2[]="Welcom to study C!";
    printf("%s", ch(str1, str2));       //打印结果
}
char *ch(char *str1, char *str2)       //定义返回指针的函数
{
    int i;
    char *p;                           //定义指针
    p=str2
    if(*str2==NULL) exit(-1);
    do
    {
        *str2=*str1;
        str1++;
        str2++;
    }
    while(*str1!=NULL);
    return(p);                          //返回指针
}
```

上面代码中,函数 `char *ch()` 表示的就是一个返回字符型指针的函数。

【答案】

函数指针是一个指向函数的指针。它的本质是一个指针,而指针函数只是说明它是一个返回值为指针的函数,它的本质是一个函数。

8.3 this 指针

this 指针是面向对象程序设计中的一重要概念，在 C++ 中，它表示当前运行的对象。在实现对象的方法时，可以使用 this 指针来获得该对象自身的引用。

面试题 91 什么是 this 指针

【出现频率】★★★

【关键考点】

□ this 指针的概念

【考题分析】

this 指针是一个隐含的指针，它是指向对象本身的，表示当前对象的地址。

在一个非静态的成员里面，this 关键字就是一个指针，指向该函数的这次调用所针对的那个对象。在类 a 的非 const 成员函数里，this 的类型是 a*，但是 this 不是一个常规变量，所以不可以获取 this 的地址或者给它赋值。在类 a 的 const 成员函数里，this 的类型是 const a*，不可以对这个对象本身进行修改。

this 指针的一个示例代码如下：

```
void Date::setMonth( int mn )
{
    month = mn;
    this->month = mn;           //this 指针
    (*this).month = mn;
}
```

以上代码中，函数花括号内的 3 个语句是等价的，说明了 this 表示当前对象的地址。

【答案】

在调用成员函数时，编译器会隐含地插入一个参数，这个参数就是 this 指针。this 指针指向当前对象本身，表示当前对象的地址。

面试题 92 何时使用 this 指针

【出现频率】★★★

【关键考点】

□ this 指针的使用

【考题分析】

当对一个对象调用成员函数时，编译程序先将对象的地址赋给 this 指针，然后调用成员函数，每次成员函数存取数据成员时，由隐含作用 this 指针。而通常不去显式地使用 this 指针来引用数据成员。同样也可以使用*this 来标识调用该成员函数的对象。this 指针的示例代码如下：

```

class A
{
public:
    A()
    {
        a=b=0;
    }
    A(int a, int b)
    {
        this.a=a;
        this.b=b;
    }
    void copy(A &aa);           //对象引用作函数参数
    void print()
    {
        cout<<a<<" "<<b<<endl;
    }
private:
    int a, b;
};
void A::copy(A &aa)
{
    if (this == &aa) return;    //这个this 是操作对象 a1 的地址
    *this = aa;                 // *this 操作该成员函数的对象，在这里是对象 a1
    //对象 aa 赋给 a1, aa 具有的数据成员的值赋给 a1 的数据成员
}
void main()
{
    A a1, a2(3, 4);
    a1.copy(a2);
    a1.print();
}

```

以上代码运行结果如下：

```
3 , 4
```

【答案】

当对一个对象调用成员函数时，编译程序先将对象的地址赋给 `this` 指针，然后调用成员函数，每次成员函数存取数据成员时，由隐含作用 `this` 指针。而通常不去显式地使用 `this` 指针来引用数据成员。同样也可以使用 `*this` 来标识调用该成员函数的对象。

8.4 引用与值传递

C++语言中，函数的参数和返回值的传递方式有3种：引用传递、值传递和指针传递。指针编程者已经在上面进行了介绍，接下来介绍C++中引用和值传递的相关内容。

面试题 93 什么是值传递

【出现频率】★★★★★

【关键点】

☐ 值传递的概念

【考题分析】

在 C++ 中，值传递是指将要传递的值作为一个副本传递。值传递过程中，被调函数的形参作为被调函数的局部变量处理，在内存的堆栈中开辟空间以存放由主调函数放进来的实参的值，从而成为了实参的一个副本。值传递的特点是被调函数对形参的任何操作都是作为局部变量进行，不会更改主调函数的实参变量的值。值传递的示例代码如下：

```
void Func1(int x)                //参数为值传递的函数
{
    x = x + 10;
}

:

int n = 0;
Func1(n);                        //值传递
cout << "n = " << n << endl;
```

运行结果如下：

```
n = 0
```

由于 Func1 函数体内的 x 采用的是值传递的方式，它只是外部变量 n 的一个副本，改变 x 的值不会影响 n 的值，所以 n 的值仍然是 0。

【答案】

值传递将要传递的值作为一个副本传递，在函数调用时，实参把它的值传递给对应的形参，方法执行中形参值的改变不影响实参，实参的值不会发生改变。

面试题 94 引用与值传递的区别

【出现频率】★★★★★**【关键考点】**

- ☐ 引用传递的概念；
- ☐ 类的实例化。

【考题分析】

引用传递传递的是引用对象的内存地址。在地址传递过程中，被调函数的形参也作为局部变量在堆栈中开辟了内存空间，但是这时存放的是由主调函数放进来的实参变量的地址。被调函数对形参的任何操作都被处理成间接寻址，即通过堆栈中存放的地址访问主调函数中的实参变量。所以，被调函数对形参做的任何操作都会影响主调函数中的实参变量。引用传递的示例代码如下：

```
void Func2(int &x)
{
    x = x + 10;
}

:

int n = 0;
Func3(n);                        //引用传递
```

```
cout << "n = " << n << endl;
```

结果如下:

```
n = 10
```

由于 Func2() 函数体内的 x 采用的是引用传递的方式, x 是外部变量 n 的引用, x 和 n 表示的是相同的对象, 所以改变 x 也就改变了 n, 所以 n 的值更改为 10。

引用传递和值传递是面试时考查频率非常高的问题。这个问题比较简单, 所以更加需要掌握清楚, 不要出现回答错误的情况。

【答案】

值传递传递的是一个值的副本。函数对形参的操作不会影响实参的值, 而引用传递传递的引用对象的内存地址, 函数对形参的操作会影响实参的值, 实参的值将会随着形参值的更改而同样进行更改。

面试题 95 指针和引用有什么区别

【出现频率】★★★

【考点】

□ 指针和引用的区别

【考题分析】

指针和引用都是关于地址的概念, 指针指向一块内存, 它的内容是所指内存的地址。而引用是某块内存的别名。指针是作为一个真正的实体而存在的。

指针的功能非常强大, 指针能够毫无约束地操作内存中的任何东西, 由于指针功能强大, 所以导致它比较危险。如果使用不当的话会对程序运行造成很大的影响, 如果一些场合只需要借用一下某个对象的别名, 那么就可以使用引用, 而避免使用指针, 以免发生意外。程序员可以根据程序的需要来灵活选择方案。

【答案】

- 指针是一个实体, 而引用仅是个别名;
- 引用使用时无需解引用(*), 指针需要解引用;
- 引用只能在定义时被初始化一次, 之后不可变; 指针可变;
- 引用没有 const, 指针有 const;
- 引用不能为空, 指针可以为空;
- “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(所指向的变量或对象的地址)的大小;
- 和引用的自增(++) 运算意义不一样;
- 在内存分配上, 程序为指针变量分配内存区域, 而引用不需要分配内存区域。