第 4 章 指　　针

指针毫无疑问是C语言最强大和最流行的功能之一，但它也成了很多初学者的梦魇。当指针指向不应该指向的位置时，后患无穷。（实际上，很多与指针相关的问题与内存分配的关系更为密切。参见第7章。）

基本的指针应用

4.1

问：指针到底有什么好处？

答：它的好处太多了，比如：

- 动态分配的数组（参见问题6.14和6.16）；
- 对多个相似变量的一般访问；
- （模拟）按引用传递函数参数（参见问题4.8和20.1）；
- 各种动态分配的数据结构，尤其是树和链表；
- 遍历数组（例如，解析字符串）；
- 高效地、按引用“复制”数组和结构，特别是作为函数参数的时候。

（请注意，这并非一个完整的列表！）

4.2

问：我想声明一个指针并为它分配一些空间，但却不行。下面的代码有什么问题呢？

```
char *p;
*p = malloc(10);
```

答：你声明的指针是p，而不是*p，当操作指针本身时（例如当你对其赋值，使之指向别处时），只需要使用指针的名字即可：

```
p = malloc(10);
```

当操作指针所指向的内存时，才需要使用*作为间接操作符：

```
*p = 'H';
```

然而，如果像下边这样在局部变量的声明中使用malloc调用作为初始式：

```
char *p = malloc(10);
```

则很容易会犯下问题中的错误。

在把一个初始化的指针声明分成一个声明和一个后续赋值的时候，要记得去掉*号。

总之，在表达式中，p是指针，*p是它指向的内容（在这个例子中是一个char）。参见问题1.21、7.1、7.5和8.3。

参考资料：[22, Sec. 3.1 p. 28]

4.3

问： *p++自增p还是p所指向的变量？

答： 后缀++和--操作符本质上比前缀一元操作符的优先级高，因此*p++和*(p++)等价，它自增p并返回p自增之前所指向的值。要自增p指向的值，则使用(*p)++，如果副作用的顺序无关紧要也可以使用++*p。

参考资料：[18, Sec. 5.1 p. 91]
[19, Sec. 5.1 p. 95]
[8, Sec. 6.3.2, Sec. 6.3.3]
[11, Sec. 7.4.4 pp. 192-193, Sec. 7.5 p. 193, Secs. 7.5.7, 7.5.8 pp. 199-200]

指针操作

4.4

问： 我用指针操作int数组的时候遇到了麻烦。下边的代码有什么问题？

```
int array[5], i, *ip;
for(i = 0; i < 5; i++) array[i] = i;
ip = array;
printf("%d\n", *(ip + 3 * sizeof(int)));
```

我以为最后一行会打印出3，但它打印了一堆垃圾信息。

答： 你做了一些无用功。C语言中的指针算术总是自动地采纳它所指向的对象的大小。你所需要的就是：

```
printf("%d\n", *(ip + 3));  /*or ip[3]-----See Q 6.3*/
```

这就可以打印出数组的第三个元素。在类似的代码中，你无需考虑按指针指向的元素的大小进行计算。如果你那样计算，会不经意地访问并不存在的数组元素。（根据你的机器上sizeof(int)的大小，也许是array[6]，也许是array[12]。）

参考资料：[18, Sec. 5.3 p. 94]
[19, Sec. 5.4, p. 103]
[35, Sec. 3.3.6]
[8, Sec. 6.3.6]
[11, Sec. 7.6.2 p. 204]

4.5

问：我有一个char *型指针碰巧指向一些int型变量，我想跳过它们。为什么((int *)p)++;这样的代码不行？

答：在C语言中，类型转换操作符并不意味着“把这些二进制位看作另一种类型，并作相应的处理”。这是一个转换操作符，根据定义它只能生成一个*右值*（rvalue）。而右值既不能赋值，也不能用++自增。（如果编译器接受这样的表达式，那要么是一个错误，要么是有意作出的非标准扩展。）要达到你的目的可以用：

```
p = (char *)((int *)p + 1);
```

或者，因为p是char*型，直接用

```
p += sizeof(int);
```

要想真正明白无误，你得用

```
int *ip = (int *)p;
p = (char *)(ip + 1);
```

但是，可能的话，你还是应该一开始就选择适当的指针类型，而不是一味地试图李代桃僵。

参考资料：[19, Sec. A7.5 p. 205]
[35, Sec. 3.3.4 esp. footnote 44]
[8, Sec. 6.3.4]
[14, Sec. 3.3.2.4]
[11, Sec. 7.1 pp. 179-180]

4.6

问：为什么不能对void *指针进行算术操作？

答：参见问题11.26。

4.7

问：我有些解析外部结构的代码，但是它却崩溃了，显示出了“unaligned access”（未对齐的访问）的信息。这是什么意思？

答：参见问题16.8。

作为函数参数的指针

4.8

问：我有个函数，它应该接受并初始化一个指针：

```
void f(int *ip)
{
    static int dummy = 5;
    ip = &dummy;
}
```

但是当我如下调用时：

```
int *ip;
f(ip);
```

调用者的指针没有任何变化。

答：你确定函数初始化的是你希望它初始化的东西吗？请记住在C语言中，参数是通过值传递的。

上述代码中被调函数仅仅修改了传入的指针副本。为了达到期望的效果，你需要传入指针的地址（函数变成接受指向指针的指针）：

```
void f(ipp)
int **ipp;
{
    static int dummy = 5;
    *ipp = &dummy;
}

...

int *ip;
f(&ip);
```

这里，实际上在模拟通过引用传递参数。另一种方法是让函数返回指针：

```
int *f()
{
    static int dummy = 5;
    return &dummy;
}

...

int *ip = f();
```

参见问题4.9和4.11。

4.9

问：能否像下边这样用void **通用指针作为参数，使函数模拟按引用传递参数?

```
void f(void **);
double *dp;
f((void **)&dp);
```

答：不可移植。这样的代码可能有效，而且有时鼓励这样用，但是它依赖一种假设——所有指针的内部表示都是一样的（这很常见但并不一定如此。参见问题5.17。）

C语言中没有通用的指针类型。void *可以用作通用指针只是因为当它和其他类型相互赋值的时候，如果需要，它可以自动转换成其他类型。但是，如果试图这样转换所指类型为void *之外的类型的void **指针时，就不会自动转换了。当你使用void **指针的时候（例如，使用*操作符访问void **指针所指的void *值的时候），编译器无从知道void *值是否是从其他类型的指针转换而来的。从而，编译器只能认为它仅仅是个void *指针，不能对它进行任何隐式的转换。

换言之，你使用的任何void **值必须的确是某个位置的void *值的地址。(void **)&dp 这样的类型转换虽然可以让编译器接受，但却不能移植，而且也可能不会达到你想要的目的。参见问题13.9。如果void **指针指向的值不是void *类型，而且它的大小或内部表示和void *也不相同，则编译器就不能正确地访问它。

要使上边的代码正确工作，你需要使用一个中间void *变量：

```
double *dp;
void *vp = dp;
f(&vp);
dp = vp;
```

对/从vp的赋值使编译器有机会在需要的时候进行适当的类型转换。

目前，我们的讨论假设的是不同类型的指针可能具有不同的大小和内部表示。这种情况现在已经很少见到，但也并非完全没有。为了进一步弄清void **的问题，让我们看一个类似的情形，比如类型int和double。它们可能大小不一样，而且肯定内部表示也不一样。

假如有这样的函数：

```
void incme(double *p)
{
    *p += 1;
}
```

可以这样做：

```
int i = 1;
double d = i;
incme(&d);
i = d;
```

很明显，i增加了1。（这跟使用辅助变量vp的void **指针的正确代码类似。）但是，假如想这样：

```
int i = 1;
incme((double*)&I);  /*WRONG*/
```

跟问题中的代码类似，这段代码肯定不会正确运行。

4.10

问：我有一个函数extern intf(int *);，它接受指向int型的指针。我怎样用引用方式传入一个常量？调用f(&5);似乎不行。

答：在C99中，你可以使用“复合字面量”：

```
f((int[]){5});
```

在C99之前，你不能直接这样做，必须先定义一个临时变量，然后把它的地址传给函数：

```
int five = 5;
f(&five);
```

在C语言中，接受指针而不是值的函数可能往往希望修改指针指向的值，因此传入一个常数指针可能不是一个好主意。事实上，如果f被定义成接受int *，则向它传入const int的指针的时候需要诊断。（如果函数能够保证不修改传入指针指向的值，则它可以定义成接受const int *参数。）参见问题2.11、4.8和20.1。

4.11

问：C语言可以“按引用传参”吗？

答：真的没有。严格地讲，C语言总是按值传参。你可以自己模拟按引用传参，定义接受指针的

函数，然后在调用时使用&操作符。事实上，当你向函数传入数组（传入指针的情况参见问题6.4及其他相关问题）时，编译器本质上就是在模拟按引用传参。但是C没有任何真正等同于按引用传参或C++引用参数的东西。另一方面，类似函数的预处理宏可以提供一种“按名称传参”的形式。

参见问题4.8和20.1。

参考资料：[18, Sec. 1.8 pp. 24-25, Sec. 5.2 pp. 91-93]
[19, Sec. 1.8 pp. 27-28, Sec. 5.2 pp. 95-97]
[8, Sec. 6.3.2.2]
[11, Sec. 9.5 pp. 273-274]。

其他指针问题

4.12

问：我看到了用指针调用函数的不同语法形式。到底怎么回事？

答：最初，函数指针必须用*操作符（和一对括号）“转换为”一个“真正的”函数才能调用：

```
int r, (*fp)(), func();
fp = func;
r = (*fp)();
```

最后一行的解释很明确：fp是一个函数的指针，因此*fp是个函数。在括号内加上函数参数列表（再在*fp外加上一对括号用于使运算的优先级正确），就完成了一个完整的函数调用。

而函数总是通过指针进行调用的，所有“真正的”函数名在表达式和初始化中，总是隐式地退化为指针。参见问题1.36。这个推论表明，无论fp是函数名还是函数的指针，r = fp();都是合法的且能正确工作。（这种用法没有任何歧义，使用函数指针后跟参数列表的方法，除了调用它所指的函数之外，别的什么也做不了。）使用显式的*号依然允许，而且为了保证在较老的编译器上的可移植性，这也是推荐的用法。

参见问题1.36。

参考资料：[18, Sec. 5.12 p. 116]
[19, Sec. 5.11 p. 120]
[8, Sec. 6.3.2.2]
[14, Sec. 3.3.2.2]
[11, Sec. 5.8 p. 147, Sec. 7.4.3 p. 190]

4.13

问：通用指针类型是什么？当我把函数指针赋向void *类型的时候，编译通不过。

答：没有什么“通用指针类型”。void *指针只能保存对象（也就是数据）指针。将函数指针转换为void *指针是不可移植的。（在某些机器上，函数指针可能很大——比任何数据指针都大。）

但是，可以确保的是，所有的函数指针类型都可以相互转换，只要在调用之前转回了正确的类型即可。因此，可以使用任何函数类型（通常是int (*)()或void (*)()，即未指明参数、返回int或void的函数）作为通用函数指针。如果你需要一个既能容纳对象指针又能容纳函数指针的地方，可移植的解决方案是使用包含void *指针和通用函数指针（任何类型都可以）的联合。

参见问题1.22和5.8。

参考资料：[35, Sec. 3.1.2.5, Sec. 3.2.2.3, Sec. 3.3.4]
[8, Sec. 6.1.2.5, Sec. 6.2.23, Sec. 6.3.4]
[14, Sec. 3.2.2.3]
[11, Sec. 5.3.3 p. 123]

4.14

问：怎样在整型和指针之间进行转换？能否暂时把整数放入指针变量中，或者相反？

答：曾经有一段时间，可以确保能将指针转换为整数（尽管谁也不知道究竟是需要int还是long型），将整数转换为指针。同时可以确保指针在转换为（足够大的）整数及转换回来的时候值不会改变，而且转换（及任何映射）都不应该“让那些知道机器寻址结构的人感到惊奇”。换言之，有整数/指针转换的先例和支持，但这总是和机器相关的，因此不具可移植性。而且总是需要显式的类型转换（但是就算你忘了转换，早期的编译器也几乎不会报警。）

为了使C语言广泛地可实现，ANSI/ISO C标准削弱了这些早期的保证。指针到整数和整数到指针的转换变成了实现定义的（参见问题11.35），因此也就没有了指针和整数可以无需修改就相互转换的保证。

强制将指针转换为整数和将整数转换为指针从来都不是什么好的实践。当需要同时保存两种类型数据的存储结构的时候，使用联合是一个更好的办法。

参见问题5.18和19.30。

参考资料：[18, Sec. A14.4 p. 210]
[19, Sec. A6.6 p. 199]
[35, Sec. 3.3.4]
[8, Sec. 6.3.4]

[14, Sec. 3.3.4]
[11, Sec. 6.2.3 p. 170, Sec. 6.2.7 pp. 171-172]

*4.15

问：我怎样把一个int变量转换为char *型？我试了类型转换，但是不行。

答：这取决于你希望做什么。如果你的类型转换不成功，你可能是企图把整数转换为字符串，这种情况参见问题13.1。如果你试图把整数转换为字符，参见问题8.6。如果你试图让一个指针指向特定的内存地址，参见问题19.30。