Interoperate Swift with C

C中的基本类型在Swift中是如何表示的

(https://www.boxueio.com/series/interoperate-swift-with-c/ebook/245)

C中的struct和union是如何桥接到Swift的

(https://www.boxueio.com/series/interoperate-swift-with-c/ebook/247)

C中的简单函数是如何桥接到Swift的

Back to series (/series/interoperate-swift-with-c)

字号
字号
默认主题
金色主题
暗色主题

在这一节，我们来分享和函数有关的桥接工作。这是一个有些复杂的过程，根据函数参数和返回值的不同，桥接的方式也有差异。作为第一部分，我们先来看简单函数的桥接过程。所谓简单函数，就是指那些参数和返回值都是简单类型，并且不使用指针的函数。

固定参数个数的函数

固定参数个数的简单函数桥接到Swift时很简单，几乎会“原封不动”的桥接到Swift。例如，在*traditional_oc.h*中，声明一个全局函数：

```
int add(int m, int n);
```

并在*traditional_oc.m*中定义它：

```
int add(int m, int n) {
    return m + n;
}
```

在Swift里， `add` 会变成这样：

```
func add(_ m: Int32, _ n: Int32) -> Int32 {
    return m + n
}
```

其中有两点需要注意：

- C中桥街过来的函数默认都是省略external name的；
- C中的 `int` 会自动转换成 `Int32` ，因此默认是不能传递Swift `Int` 类型的，只能使用 `CInt` 类型；

不固定参数个数的函数

接下来，我们了解不固定参数个数的函数。在C里，我们有两种方法定义这样的函数，例如，先在*traditional_oc.h*中添加下面的声明：

```
int sum(int count, ...);
int vsum(int count, va_list numbers);
```

这两个函数的第一个参数表示后续不确定参数的个数，然后在*traditional_oc.m*中实现它们：

```
int sum(int count, ...) {
    va_list ap;
    int s = 0;

    va_start(ap, count);
    vsum(count, ap);
    va_end(ap);

    return s;
}

int vsum(int count, va_list numbers) {
    int s = 0;
    int i = 0;

    for (; i < count; ++i) {
        s += va_arg(numbers, int);
    }

    return s;
}
```

它们都是很简单的C代码，如果你还不熟悉C的可变参数函数，可以先在这里(http://en.cppreference.com/w/c/variadic)简单了解下，我们就不重复了。这两个函数会如何桥接到Swift呢？遗憾的是，Swift只能接受 `vsum` ，而不能接受 `sum` 。也就是说，无论如何都无法在Swift中直接调用 `sum` 函数。而 `vsum` 桥接到Swift之后，是这样的：

```
func vsum(count: Int32, numbers: CVaListPointer) -> Int32
```

因此，在Swift里，我们不能像 `vsum(6, 1, 2, 3, 4, 5, 6)` 这样调用 `vsum` ，那么这个 `CVaListPointer` 是什么呢？简单来说，它就是C中 `va_list` 桥接到Swift后对应的类型。为了得到这个对象，我们有两种方法。

第一种，是调用 `getVaList` 方法，并把要传递的可变参数作为一个数组传递给它：

```
let vaListPointer = getVaList([1, 2, 3, 4, 5, 6])
let sum = vsum(6, vaListPointer)
```

这样，我们就可以把 `vaListPointer` 作为 `vsum` 的第二个参数了。

第二种，是调用 `withVaList` 方法，它的第一个参数是一个数组，我们像调用 `getVaList` 一样把所有可变参数传递给它；第二个参数是一个closure， `withVaList` 会根据第一参数中的所有成员，生成一个对应的 `CVaListPointer` 对象，并传递给这个closure。因此，我们只要在closure里调用 `vsum` 就好了：

```
let sum = withVaList([1, 2, 3, 4, 5, 6]) {
    vaListPointer in
    vsum(6, vaListPointer)
}
```

当然，这两种方法的结果，是完全一样的。

What's next?

以上，就是两种简单函数桥接到Swift时的处理方式，在进一步了解C函数和Swift的交互前，还要做不少铺垫工作，因此，我们先把这个话题放放。下一节，我们来看Swift如何处理C中的 `struct` 和 `union` 。

⏮ C中的基本类型在Swift中是如何表示的
(https://www.boxueio.com/series/interoperate-swift-with-c/ebook/245)

C中的struct和union是如何桥接到Swift的 ⏭
(https://www.boxueio.com/series/interoperate-swift-with-c/ebook/247)

职场漂泊的你，每天多学一点。

从开发、测试到运维，让技术不再成为你成长的绊脚石。我们用打磨产品的精神去传播知识，把最新的移动开发技术，通过简单的图表， 清晰的视频，简明的文字和切实可行的例子一一向你呈现。让学习不仅是一种需求，也是一种享受。

泊学动态

一个工作十年PM终创业的故事（二） (https://www.boxueio.com/after-the-full-upgrade-to-swift3)
Mar 4, 2017

人生中第一次创业的"10有" (https://www.boxueio.com/founder-chat)
Jan 9, 2016

猎云网采访报道泊学 (http://www.lieyunwang.com/archives/144329)
Dec 31, 2015

What most schools do not teach (https://www.boxueio.com/what-most-schools-do-not-teach)
Dec 21, 2015

一个工作十年PM终创业的故事（一） (https://www.boxueio.com/founder-story)
May 8, 2015

泊学相关

关于泊学 ›

加入泊学 ›

泊学用户隐私以及服务条款 (HTTPS://WWW.BOXUEIO.COM/TERMS-OF-SERVICE)

版权声明 (HTTPS://WWW.BOXUEIO.COM/COPYRIGHT-STATEMENT)

联系泊学

Email: 10[AT]boxue.io (mailto:10@boxue.io)

QQ: 2085489246

友情链接 SwiftV (http://www.swiftv.cn) | Seay信息安全博客 (http://www.cnseay.com) | Swift.gg (http://swift.gg/) | Laravist (http://laravist.com/) | SegmentFault (https://segmentfault.com) | 靛青K的博客 (http://blog.dianqk.org/)