指针和引用

基本概念

指针是所指内存的地址。
引用是别名。
引用必须初始化， 并且初始化后不能重新引用其它变量。

> 那么到底引用在汇编层面和指针有什么区别呢?

没区别。
是的，引用会被 C++ 编译器当做 const 指针来进行操作。

原版函数

```
// 指针版
void swap(int *a, int *b) {
int temp = *a;
*a = *b;
*b = temp;
}

// 引用版
void swap(int &a, int &b) {
int temp = a;
a = b;
b = temp;
}
```

```
gcc -S
```

输入以上命令，查看汇编

总结

1. 引用只是c++语法糖，可以看作编译器自动完成取地址、解引用的常量指针
2. 引用区别于指针的特性都是编译器约束完成的，一旦编译成汇编就喝指针一样
3. 由于引用只是指针包装了下，所以也存在风险，比如如下代码:

```
int *a = new int;
int &b = *a;
delete a;
b = 12; // 对已经释放的内存解引用
```

4. 引用由编译器保证初始化，使用起来较为方便(如不用检查空指针等)
5. 尽量用引用代替指针
6.
    - 引用没有顶层 const (引用本身不可变) 即

```
int & const p;
```

因为引用本身就不可变， 所以在加顶层 const 也没有意义；

- 但是可以有底层 const() 即

```
const int & p;
```

这表示引用所引用的对象本身是常量。

7.
   - 指针既有顶层 const (引用本身不可变) 即

```
int * const p;
```

   - 也有底层 const(指针所指向的对象不可变)

```
const int * p;
```

8.
   - 有指针引用----绑定指针的引用
   - 没有引用指针--指向引用的指针

> 因为很多时候指针存在的意义就是间接改变对象的值。 但是引用本身的值我们上面说过了是所引用对象的地址，但是引用不能更改所引用的对象，也就当然不能有引用指针了。

9. 指针和引用的自增(++）和自减含义不同
   - 指针是指针运算
   - 而引用是代表所指向的对象对象执行++或--