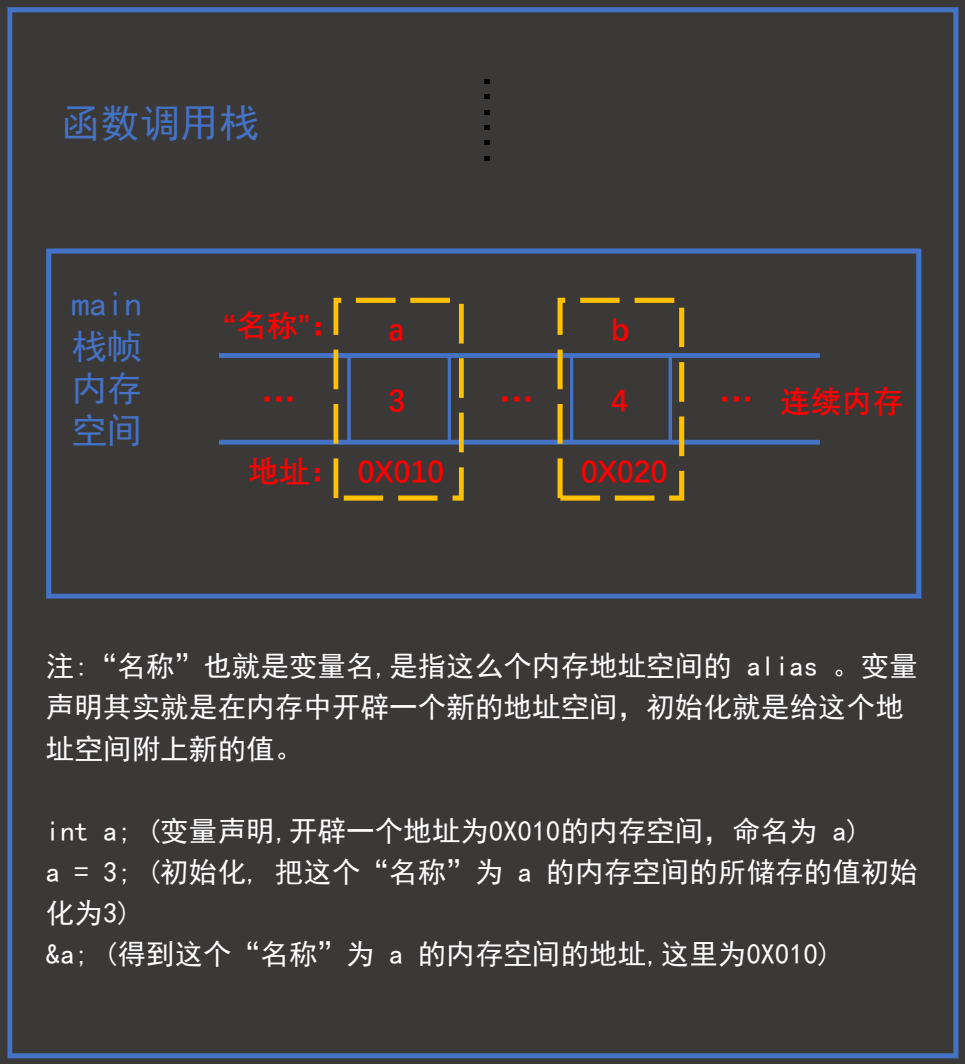
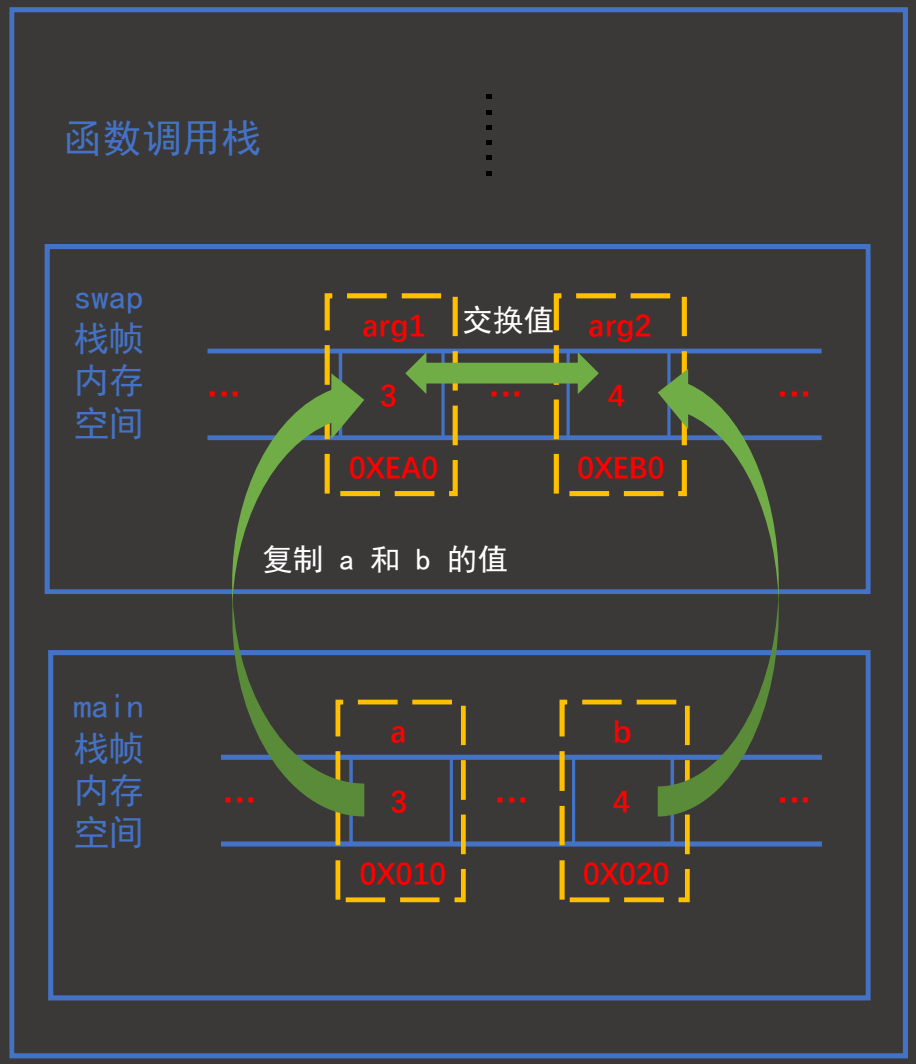


注:“名称”也就是变量名,是指这么个内存地址空间的 alias 。变量声明其实就是在内存中开辟一个新的地址空间，初始化就是给这个地址空间附上新的值。

int a; (变量声明,开辟一个地址为0X010的内存空间，命名为 a)
a = 3; (初始化，把这个“名称”为 a 的内存空间的所储存的值初始化为3)
&a; (得到这个“名称”为 a 的内存空间的地址,这里为0X010)

C++ 变量初始化

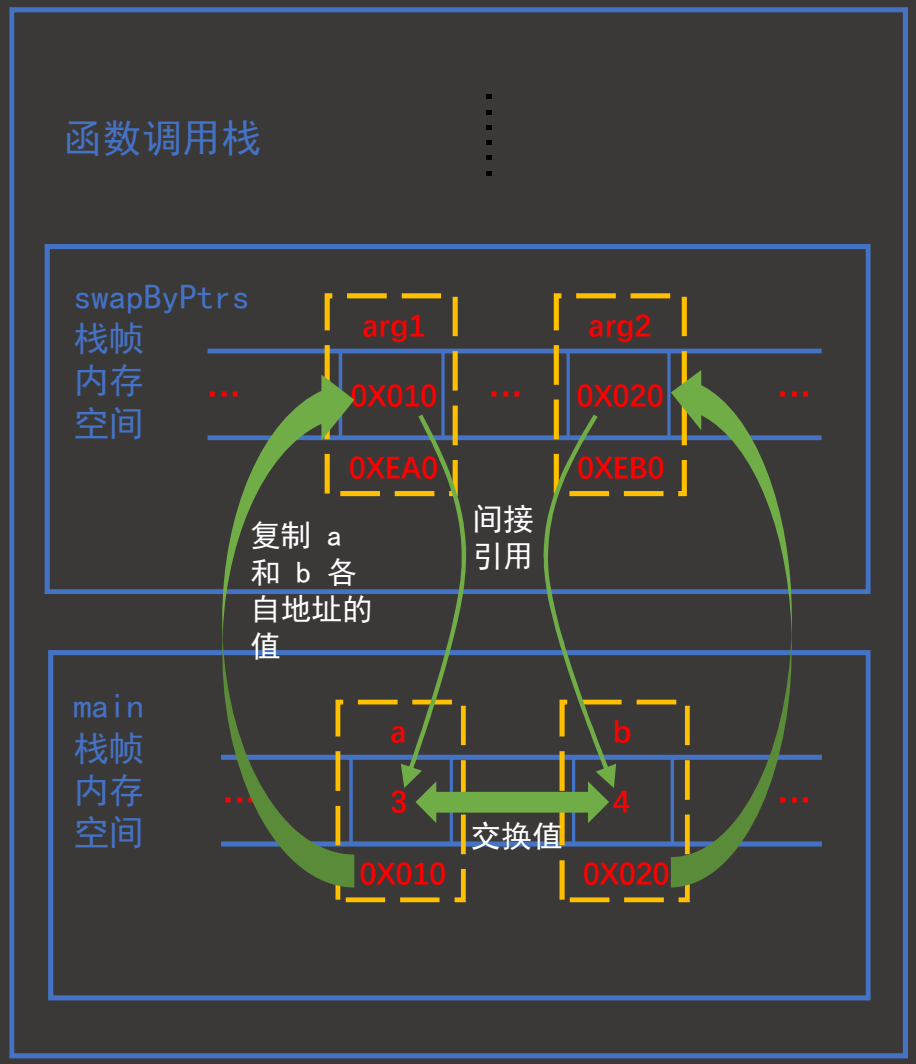
```
#include <stdio.h>
void swap(int, int);

int main() {
    int a = 3;
    int b = 4;
    printf("Address of a: %p\n", &a);
    printf("Address of b: %p\n", &b);
    printf("a = %d\n", a);
    printf("b = %d\n", b);

    swap(a, b);

    printf("After Swap: Address of a: %p\n", &a);
    printf("After Swap: Address of b: %p\n", &b);
    printf("After Swap: a = %d\n", a);
    printf("After Swap: b = %d\n", b);
}

void swap(int arg1, int arg2) {
    printf("\nEnter Swap Function:\n");
    printf("Address of arg1: %p\n", &arg1);
    printf("Address of arg2: %p\n", &arg2);
    printf("arg1 = %d\n", arg1);
    printf("arg2 = %d\n", arg2);
    int temp = arg1;
    arg1 = arg2;
    arg2 = temp;
    printf("After Swap: arg1 = %d\n", arg1);
    printf("After Swap: arg2 = %d\n", arg2);
    printf("Leaf Swap Function.\n\n");
}
```



C++ 值传递

注：swap 函数接受 main 函数传递过来的 a 和 b 时，首先会在自己独立的内存空间中划分出两个地址，并命名为 arg1 和 arg2 如左图所示，地址分别为 0XEA0和0XEB0，然后各自拷贝 a 和 b 的值。最后的交换操作也只是在 swap 自己的内存空间中交换 arg1 和 arg2 的值，对 main 函数中的 a 和 b 没有任何影响。

```
#include <stdio.h>
void swap(int, int);

int main() {
    int a = 3;
    int b = 4;
    printf("Address of a: %p\n", &a);
    printf("Address of b: %p\n", &b);
    printf("a = %d\n", a);
    printf("b = %d\n", b);

    swap(a, b);

    printf("After Swap: Address of a: %p\n", &a);
    printf("After Swap: Address of b: %p\n", &b);
    printf("After Swap: a = %d\n", a);
    printf("After Swap: b = %d\n", b);
}

void swap(int arg1, int arg2) {
    printf("\nEnter Swap Function:\n");
    printf("Address of arg1: %p\n", &arg1);
    printf("Address of arg2: %p\n", &arg2);
    printf("arg1 = %d\n", arg1);
    printf("arg2 = %d\n", arg2);
    int temp = arg1;
    arg1 = arg2;
    arg2 = temp;
    printf("After Swap: arg1 = %d\n", arg1);
    printf("After Swap: arg2 = %d\n", arg2);
    printf("Leaf Swap Function.\n\n");
}
```



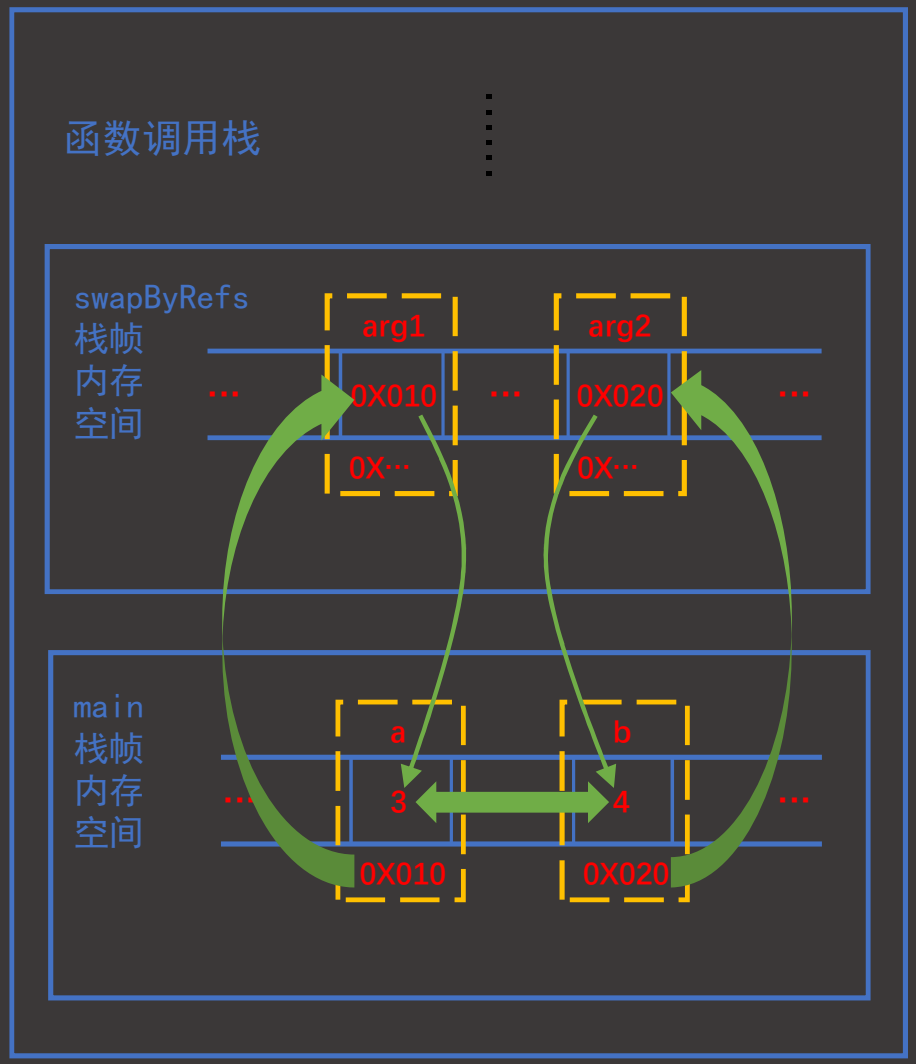
C++ 指针传递

注：swapByPtrs 函数接受 main 函数传递过来的指向 a 和 b 的指针时，首先会在自己独立的内存空间中划分出两个地址，并命名为 arg1 和 arg2 如左图所示，地址分别为0XEA0和0XEB0，然后各自拷贝 指向 a 和 b 指针的值(即 a 和 b 的地址0X010, 0X020)。

swapByPtrs 函数通过对这两个引用 dereference （*arg1 和 *arg2）来交换 main 内存中 a 和 b 的值。

如果不 dereference 直接进行交换，那么就只会交换 arg1 和 arg2 中所储存的地址值，不会改变 main 函数的a 和 b 的值。

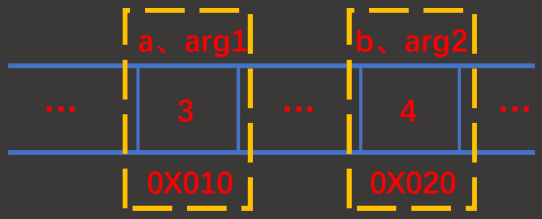
```
#include <stdio.h>
void swapByPtrs(int *, int *);

int main() {
    int a = 3;
    int b = 4;
    printf("Address of a: %p\n", &a);
    printf("Address of b: %p\n", &b);
    printf("a = %d\n", a);
    printf("b = %d\n", b);

    swap(&a, &b);

    printf("After Swap: Address of a: %p\n", &a);
    printf("After Swap: Address of b: %p\n", &b);
    printf("After Swap: a = %d\n", a);
    printf("After Swap: b = %d\n", b);
}

void swapByPtrs(int *arg1, int *arg2) {
    printf("\nEnter swapByPtrs Function!\n");
    printf("Address of arg1: %p\n", &arg1);
    printf("Address of arg2: %p\n", &arg2);
    printf("arg1 = %p\n", arg1);
    printf("arg2 = %p\n", arg2);
    int temp = *arg1;
    *arg1 = *arg2;
    *arg2 = temp;
    printf("After Swap: arg1 = %p\n", arg1);
    printf("After Swap: arg2 = %p\n", arg2);
    printf("Leaf swapByPtrs Function.\n\n");
}
```



C++ 引用传递

注:通过引用传递函数参数时，在编译阶段，编译器会创建临时指针来协助完成操作。底层实现大体上和指针传递相同。只是不需要通过显式的“* 和“&”来实现。另外，如果对某个引用执行“&”操作例如 &arg1 得到的是 a 的地址 0X010。arg1 = arg2 将会改变 a 的值（编译器通过临时指针操作，对程序员是透明的）。

实际上，图方便来看，可以把引用看成是给变量又创建一个新的 alias 。本例中，在执行 swap 函数时，给 main 内存空间的 a 和 b 分别又添加了新的“名称”：arg1、arg2。执行完成后又抹去了这两个新的名称。




```
#include <stdio.h>
void swapByRefs(int &, int &);

int main() {
    int a = 3;
    int b = 4;
    printf("Address of a: %p\n", &a);
    printf("Address of b: %p\n", &b);
    printf("a = %d\n", a);
    printf("b = %d\n", b);

    swap(a, b);

    printf("After Swap: Address of a: %p\n", &a);
    printf("After Swap: Address of b: %p\n", &b);
    printf("After Swap: a = %d\n", a);
    printf("After Swap: b = %d\n", b);
}

void swapByRefs(int &arg1, int &arg2) {
    printf("\nEnter swapByRefs Function!\n");
    printf("Address of arg1: %p\n", &arg1);
    printf("Address of arg2: %p\n", &arg2);
    printf("arg1 = %d\n", arg1);
    printf("arg2 = %d\n", arg2);
    int temp = arg1;
    arg1 = arg2;
    arg2 = temp;
    printf("After Swap: arg1 = %d\n", arg1);
    printf("After Swap: arg2 = %d\n", arg2);
    printf("Leaf swapByRefs Function.\n\n");
}
```