```
    methodAddress     = getMethodAddress(methodName);
    InstructionAddress = getInstructionAddress(methodAddress, offset);
    changeInstruction(InstructionAddress);
}
```

2. 获得 C++方法地址和指令地址

正如上面提到的，在计算指令地址，必须先获得函数地址，本例中，函数为 C++的方法。通过使用指向 C++方法的指针，我们可以得到该方法的首地址；另外，由于 C++编译器最终产生的了 C 风格的函数符号，因此使用一般的函数指针也能获得方法地址，只不过这种情况下需要知道编译后的 C++方法的符号名。下面分别介绍这两种方法。

2.1 使用 C++方法指针

C++方法指针的用法与 C 函数指针稍有不同，***C++ Primer*** 一书中进行了详细介绍。

在本例中，通过方法指针获得方法地址的代码如下

```
    // Function pointer points to method to change
    int (YLPhraseCandidates::*methodPointer)(unsigned short*) = NULL;
    // Get method address
    methodPointer = &YLPhraseCandidates::GetPhrases;
    // Seek target instruction by method address
    instruction    =    (unsigned    char*)(_phraseEngine->*methodPointer)    +
Instructionoffset;
```

在上面代码中，首先声明一个方法指针，然后将该指针与指定的方法关联，最后通过计算指令与方法首地址的偏移量获得指令地址。

需要注意的是，使用方法指针在获得方法地址时与 C 语言函数指针颇有些不同，因为方法指针实际上被 g++编译器作为结构体来处理。

使用 C++方法指针的好处主要是不用知道额外的其他信息，处理起来比较“干净”，但是方法指针用起来没有函数指针方便。

2.2 使用 C 风格的函数符号

使用 C 语言的函数指针也能获得 C++方法地址，这是因为通常情况下，C++方法最后都被编译器处理成了 C 函数的形式，只是名字符号和参数形式会有些改变，但这并不影响获得其地址。

通过反汇编，我们发现 ***YLPhraseCandidates::GetPhrases()*** 方法的最终符号是 ***_ZN18YLPhraseCandidates10GetPhrasesEPt***，为了能使用该符号，我们必须先将此符号声明为 ***extern "C"***。然后再使用一个指向该函数的指针，即可得到该符号的地址。