# 1.**c, c++函数名编译符号修饰符说明**

资料一： 无论 \_\_cdecl，\_\_fastcall还是\_\_stdcall调用方式，函数修饰都是以一个“?”開始，后面紧跟函数的名字。 再后面是參数表的開始标识和 依照參数类型代号拼出的參数表。

资料二：

对于C++的类成员函数（其调用方式是thiscall）。

函数的名字修饰与非成员的C++函数稍有不同，首先就是在函数名字和參数表之间插入以“@”字 符引导的类名。

资料三：

其次是參数表的開始标识不同，公有（public）成员函数的标识是“@@QAE”,保护（protected）成员函数的标识是 “@@IAE”,私有（private）成员函数的标识是“@@AAE”，

假设函数声明使用了constkeyword，则对应的标识应分别为 “@@QBE”，“@@IBE”和“@@ABE”。

资料四：

參数表的拼写代号例如以下所看到的：

X--void

D--char

E--unsigned char

F--short

H--int

I--unsigned int

J--long

K--unsigned long（DWORD）

M--float

N--double

\_N--bool

U--struct

....

指针的方式有些特别。用PA表示指针，用PB表示const类型的指针。

资料五：

參数表后以“@Z”标识整个名字的结束。假设该函数无參数，则 以“Z”标识结束。

<https://blog.csdn.net/liweigao01/article/details/78351464>

Wp参考

<https://blog.csdn.net/weixin_43876357/article/details/108087660>

private: char \* \_\_thiscall R0Pxx::My\_Aut0\_PWN(unsigned char \*)

->

?My\_Aut0\_PWN@R0Pxx@@AAEPADPAE

查看转换结果的另一种方式

#include <iostream>

class R0Pxx {

public:

R0Pxx() {

My\_Aut0\_PWN((unsigned char\*)"hello");

}

private:

char\* \_\_thiscall My\_Aut0\_PWN(unsigned char\*);

};

char\* \_\_thiscall R0Pxx::My\_Aut0\_PWN(unsigned char\*) {

std::cout << \_\_FUNCDNAME\_\_ << std::endl;

return 0;

}

int main()

{

R0Pxx A;

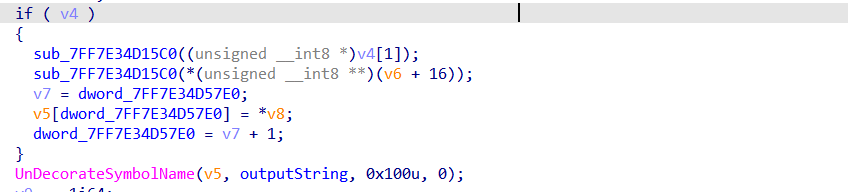
system("PAUSE");

return 0;

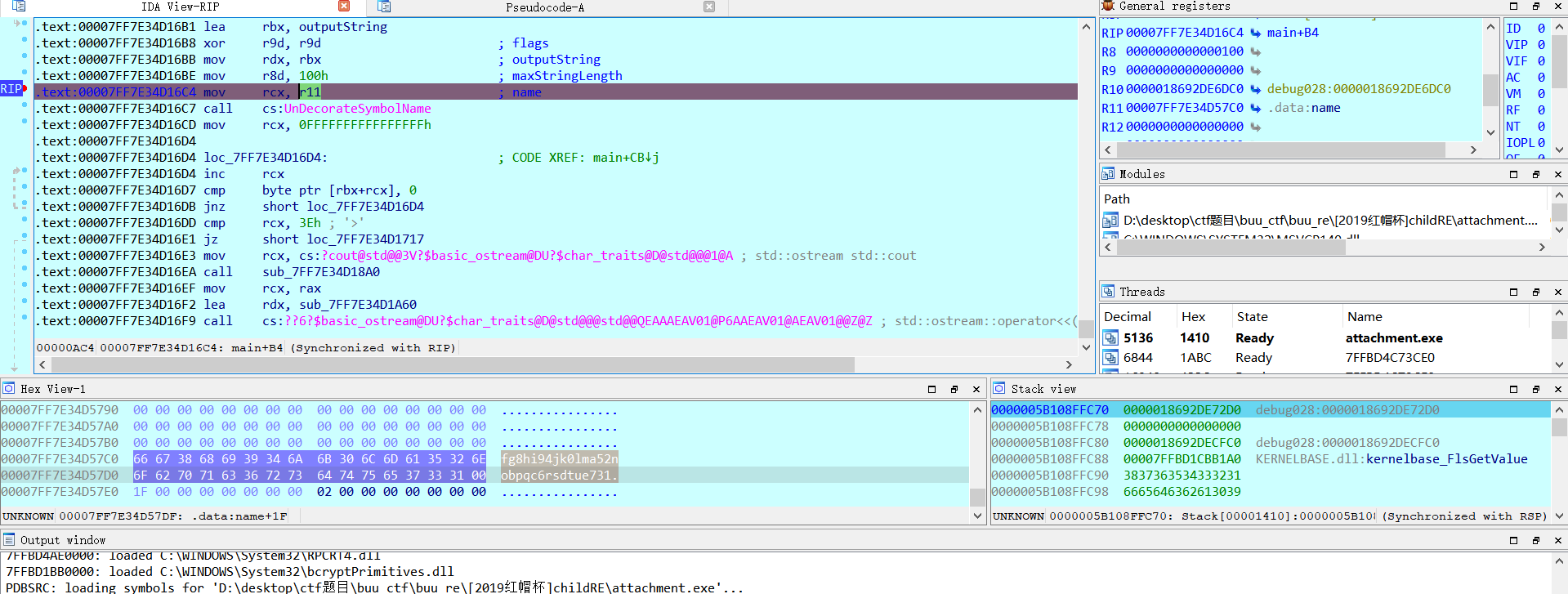
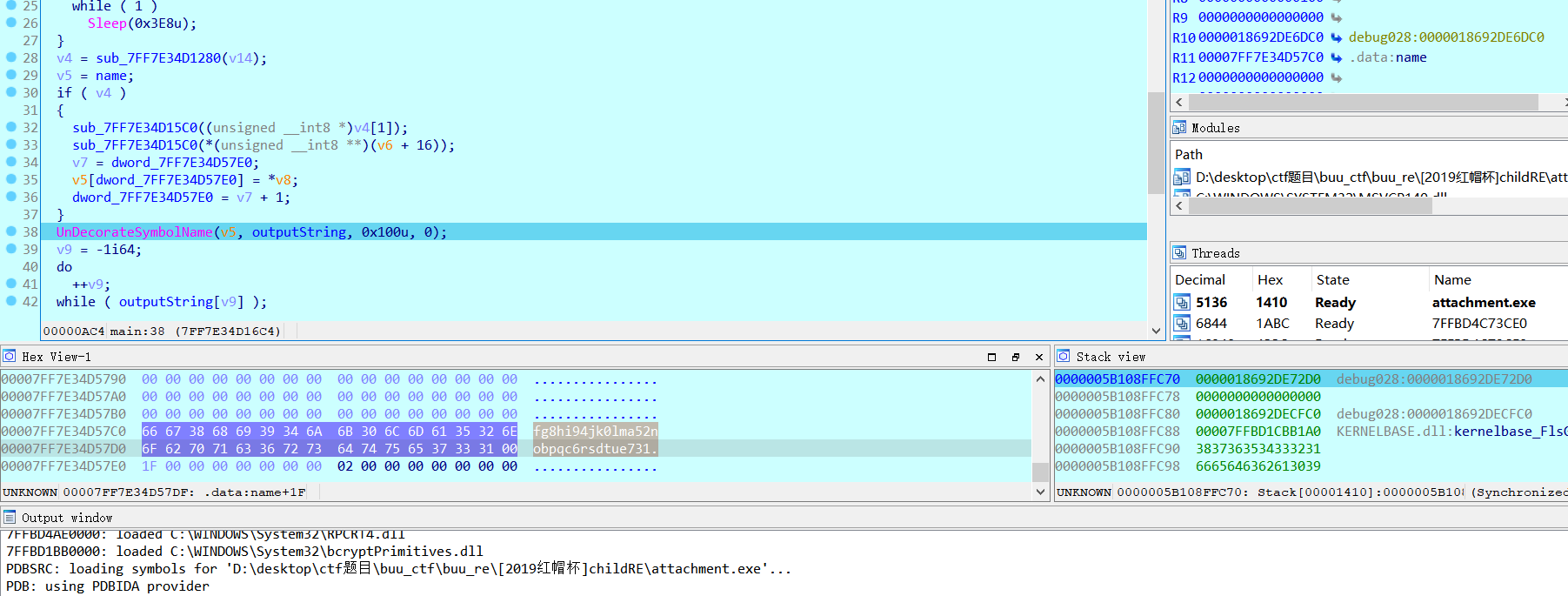
}

1. 乱序排列

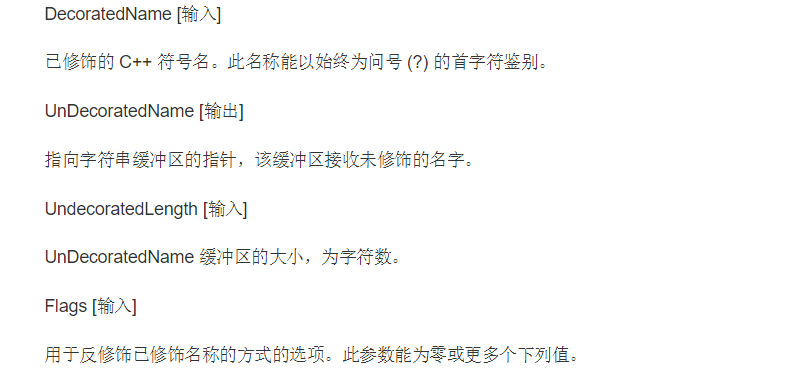
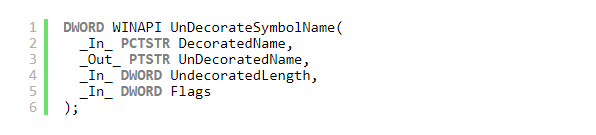
在程序中一开始我始终没理解dword\_7FF7E34D57E0到底是啥，后来想明白了就是一个不知道值的变量，所以就很清晰了，下面这一段代码即执行对输入字符串执行乱序排列，但其实本质上是固定位置互换的



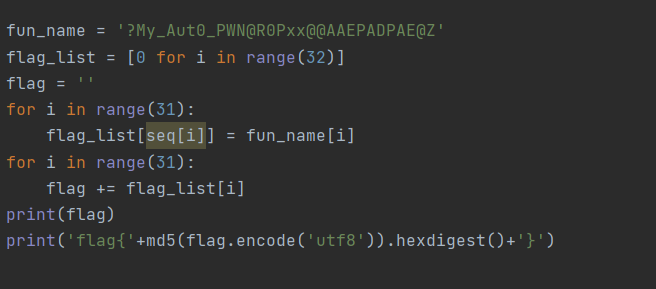
故输入1234567890abcdefghijklmnopqrstu进行动态调试，直接断在UnDecorateSymbolName()函数执行前，查看v5对应的寄存器值，去数据区查对应数据，一一对应即可查出顺序。



3.UnDecorateSymbolName()函数



1. md5py函数的使用



<https://blog.csdn.net/m0_38080253/article/details/78838489>