2021/05/13_x86逆向C++_第2课_全局对象、堆对象的识别

笔记本： x86逆向-C++
创建时间： 2021/5/13 星期四 15:07
作者： ileemi

- 课前会议
- 堆对象数组（局部）
  - 对象数组构造时需要的信息
  - 对象数组析构时需要的信息
  - 对象调用方法
- 析构代理标志
- 全局对象
  - 构造的调用
  - atexit
  - 析构的调用
- 全局对象数组
- 堆对象数组（全局）

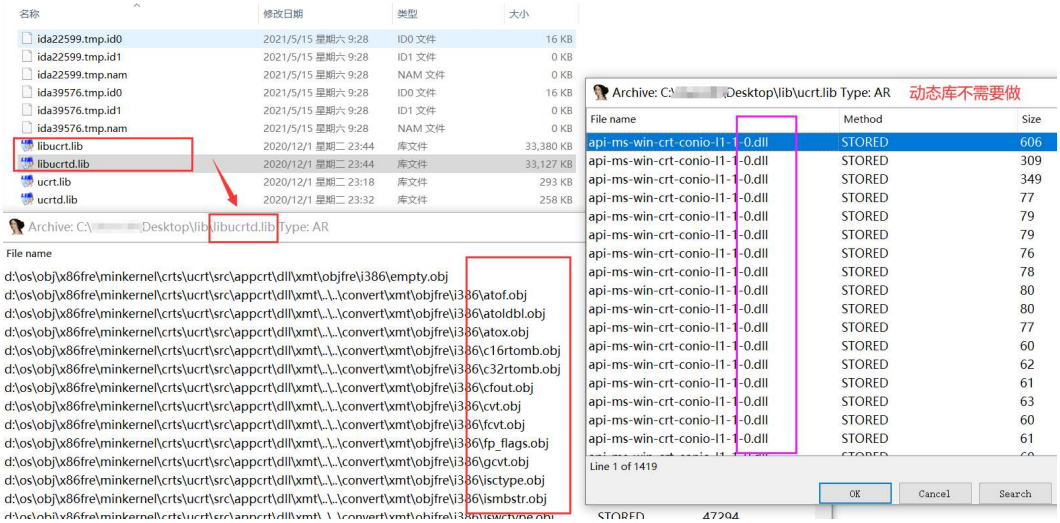
课前会议

C++类程序的还原：先结构后代码

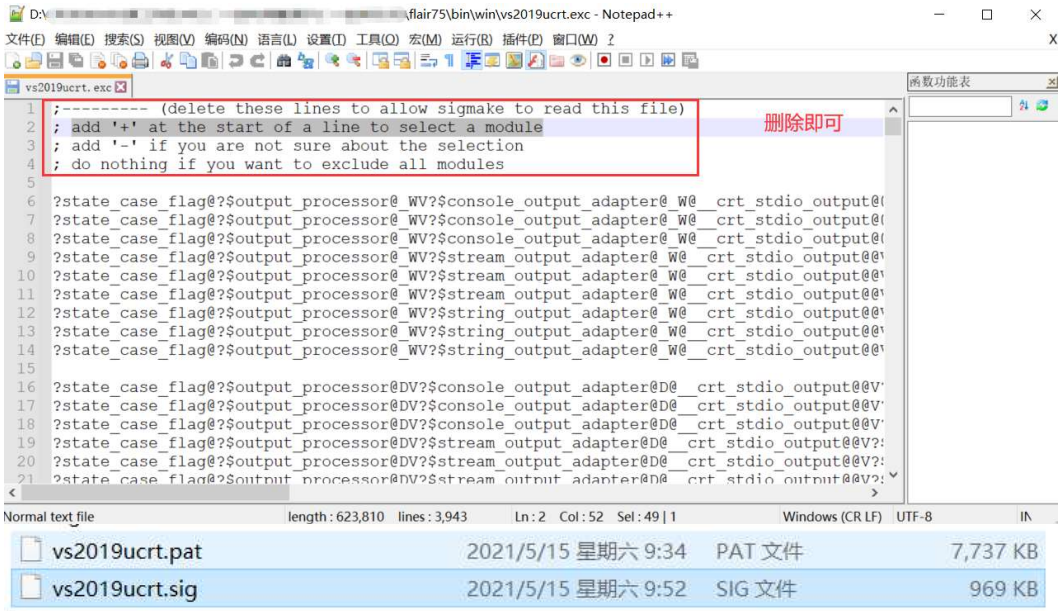
类构造前的判断不需要还原，之后的根据的情况进行还原。

sig文件的制作，库函数需要静态链接，动态链接有导入函数，IDA中可以进行识别。

在制作sig文件时（32、64为的静态库分开做），lib库做的多的情况下，提取到的特征会和IDA自动识别有一定的冲突，会导致识别失败（出现unknown）。

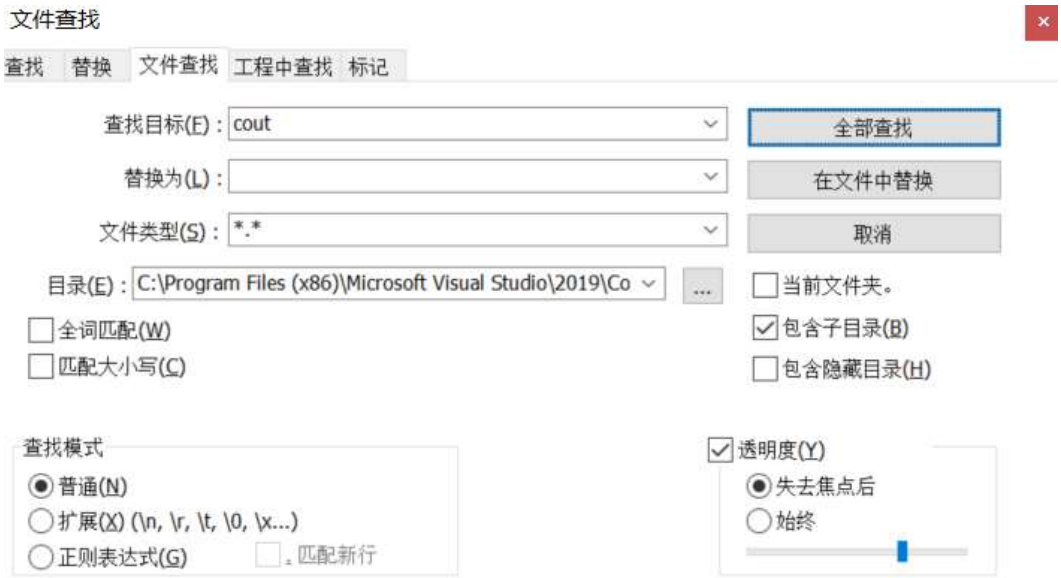


'.exc' 文件中会显示出使用到冲突特征码的具体函数，将该文件顶部的注释删除，默认使用第一个：



> 定位函数所在库，以stl中的cout为例：

定位stl源代码位置：
Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29910\crt\src\stl
定位静态库位置：
Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29910\lib\x86
通过工具进行快速搜索：



将找到的静态库做成对应的

堆对象数组（局部）

对象数组构造时需要的信息

1. 数组首地址
2. 对象数量

3. 对象的大小
4. 构造函数（new 不同对象的构造函数的数组行为一致，编译器会封装一个函数自动生成代码）
5. 析构函数（new 对象失败时需要，一个失败，需要之前申请成功的全部调用析构释放资源）

数组对象的申请和释放编译器有对应的函数对其进行管理。

对象的地址在申请堆空间+4的位置（add ecx, 4），前4个字节存放对象的个数：

```
push    124             ; Size
call    new
add     esp, 4
mov     [ebp+var_EC], eax
mov     [ebp+var_4], 0
cmp     [ebp+var_EC], 0
jz      short loc_417721
mov     eax, [ebp+var_EC]
mov     dword ptr [eax], 10
push    offset sub_411429 ; void (__thiscall *)(void *)
push    offset sub_4112B2 ; void (__thiscall *)(void *)
push    0Ah             ; unsigned int
push    0Ch             ; unsigned int
mov     ecx, [ebp+var_EC]
add     ecx, 4
```

会多申请4个字节，用来存放new对象的个数

> eh vector constructor iterator(void* ary, int obj_size, int count, pfnConstructor, pfnDestructor);

1.
```
push    offset CPerson_Destructor ; void (__thiscall *)(void *)
push    offset CPerson_Constructor ; void (__thiscall *)(void *)
push    10              ; unsigned int
push    12              ; unsigned int
mov     ecx, [ebp+var_EC]
add     ecx, 4
push    ecx             ; void *
call    j_??_L@YGXPAXIIP6EX0@Z1@Z ; `eh vector constructor iterator'(void *,uint,uint,void (*)
mov     edx, [ebp+var_EC]
add     edx, 4
mov     [ebp+var_100], edx
jmp     short loc_41772B
```

析构
构造
对象个数
对象大小
存储对象的数组首地址

2.
```
;   __try { // __finally(loc_41391D)
                mov     [ebp+ms_exc.registration.TryLevel], 0
                jmp     short loc_4138DD
; ---------------------------------------------------------------------------

loc_4138D4:                             ; CODE XREF: `eh vector constructor iterator'(void *,uint,uint
                mov     eax, [ebp+i]
                add     eax, 1
                mov     [ebp+i], eax

loc_4138DD:                             ; CODE XREF: `eh vector constructor iterator'(void *,uint,uint
                mov     ecx, [ebp+i]
                cmp     ecx, [ebp+arg_8] ; 判断对象的个数
                jz      short loc_41390B
                mov     edx, [ebp+arg_C] ; 构造
                mov     [ebp+var_28], edx
                mov     eax, [ebp+var_28]
                mov     [ebp+var_24], eax
                mov     ecx, [ebp+var_24]
                call    ds:___guard_check_icall_fptr
                mov     ecx, [ebp+arg_0]
                call    [ebp+var_24]
```

这里会注册SEK异常，创建对象失败会通过抛异常进行析构释放资源

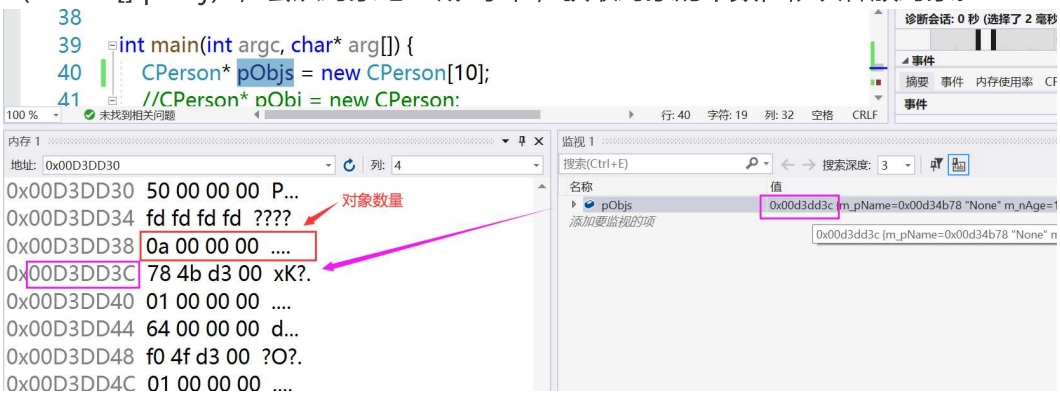
3.
```
loc_41391D:                             ; CODE XREF: `eh vector constructor iterator'(void *,uint,uint,void (*)(void
                                        ; DATA XREF: .rdata:stru_41D620↓o
;   __finally // owned by 4138CB
                movzx   edx, [ebp+var_19]
                test    edx, edx
                jnz     short loc_41393A
                mov     eax, [ebp+arg_10]
                push    eax             ; void (__thiscall *)(void *)
                mov     ecx, [ebp+i]
                push    ecx             ; unsigned int
                mov     edx, [ebp+arg_4]
                push    edx             ; unsigned int
                mov     eax, [ebp+arg_0]
                push    eax             ; void *
                call    j_?__ArrayUnwind@@YGXPAXIIP6EX0@Z@Z ; __ArrayUnwind(void *,uint,uint,void (*)(void *))

loc_41393A:                             ; CODE XREF: `eh vector constructor iterator'(void *,uint,uint,void (*)(void
                retn
```

异常展开，进行析构

对象数组析构时需要的信息

1.数组首地址
2.对象数量
3.对象的大小
4.析构函数

删除数组对象时（delete pObj），没有获取到对象的个数，导致程序会出现崩溃（delete ((int*)pObj - 1) 可解决程序的崩溃问题），而删除数组对象数组时（delete[] pObj），会从对象地址减4字节，获取对象的个数，依次释放对象。



IDA快捷键：
ctrl + 展开折叠的汇编代码

对象调用方法

> 通过对象在数组中的下标进行指定对象访问：

构造迭代器

```
                call    j_??_L@YGXPAXIIP6EX0@Z1@Z ; `eh vector constructor iterator'(void *,uint,uint,void (*)(voi
                mov     edx, [ebp+var_EC]
                add     edx, 4
                mov     [ebp+var_100], edx
                jmp     short loc_41772B
; ---------------------------------------------------------------------------

loc_417721:                             ; CODE XREF: _main_0+65↑j
                mov     [ebp+var_100], 0

loc_41772B:                             ; CODE XREF: _main_0+9F↑j
                mov     eax, [ebp+var_100]
                mov     [ebp+var_E0], eax
                mov     [ebp+var_4], 0FFFFFFFFh
                mov     ecx, [ebp+var_E0]
                mov     [ebp+var_14], ecx
                mov     eax, 12
                imul    ecx, eax, 0
                add     ecx, [ebp+var_14]
                call    CPerson_GetAge
                push    eax
                mov     ecx, 12
                imul    ecx, 0
                add     ecx, [ebp+var_14]
                call    CPerson_SetAge
                push    eax             ; char
                push    offset aNameSAgeD ; "name:%s age:%d\n"
                call    printf
                add     esp, 12
```

通过下标寻址根据对应的对象访问对应的方法

析构代理标志

- 00（0）：一个类对象释放，不需要delete。pObj->~CTest();
- 01（1）：一个类对象释放，需要delete。delete pObj;

- 11（3）：数组对象释放。delete[] pObjs;

Debug：

```
                push    3
                mov     ecx, [ebp+var_F8]
                call    Proxy_Destructor ; 代理析构
                mov     [ebp+var_100], eax
                jmp     short loc_4177A7
; ---------------------------------------------------------------------------

loc_41779D:                             ; CODE XREF: _main_0+106↑j
                mov     [ebp+var_100], 0

loc_4177A7:                             ; CODE XREF: _main_0+11B↑j
                xor     eax, eax
                mov     ecx, [ebp+var_C]
                mov     large fs:0, ecx
                pop     ecx
                pop     edi
                pop     esi
                pop     ebx
                add     esp, 100h
                cmp     ebp, esp
                call    j___RTC_CheckEsp
                mov     esp, ebp
                pop     ebp
                retn
_main_0         endp
```

析构迭代器：

```
                and     eax, 2
                jz      short loc_41326B
                push    offset CPerson_Destructor ; void (__thiscall *)(void *)
                mov     eax, [ebp+var_14]
                mov     ecx, [eax-4]    ; this指针 - 4, 获取对象个数
                push    ecx             ; unsigned int
                push    12              ; unsigned int
                mov     edx, [ebp+var_14] ; 数组首地址
                push    edx             ; void *
                call    j_??_M@YGXPAXIIP6EX0@Z@Z ; `eh vector destructor iterator'(void *,uint,uint,void (*)(void *
                mov     eax, [ebp+arg_0]
                and     eax, 1
                jz      short loc_413263
                mov     eax, [ebp+var_14]
                imul    ecx, [eax-4], 0Ch
                add     ecx, 4
                push    ecx
                mov     edx, [ebp+var_14]
                sub     edx, 4
                push    edx             ; void *
                call    sub_411456      ; delete
                add     esp, 8

loc_413263:                             ; CODE XREF: sub_4131E0+67↑j
                mov     eax, [ebp+var_14]
                sub     eax, 4
                jmp     short loc_41328C
loc_413263:                             ; CODE XREF: sub_4131E0+67↑j
                mov     eax, [ebp+var_14]
                sub     eax, 4
                jmp     short loc_41328C
; ---------------------------------------------------------------------------

loc_41326B:                             ; CODE XREF: sub_4131E0+48↑j
                mov     ecx, [ebp+var_14] ; void *
                call    CPerson_Destructor
                mov     eax, [ebp+arg_0]
                and     eax, 1
                jz      short loc_413289
                push    0Ch
                mov     eax, [ebp+var_14]
                push    eax             ; void *
                call    delete
                add     esp, 8

loc_413289:                             ; CODE XREF: sub_4131E0+99↑j
                mov     eax, [ebp+var_14]

loc_41328C:                             ; CODE XREF: sub_4131E0+89↑j
                mov     ecx, [ebp+var_C]
                mov     large fs:0, ecx
                pop     ecx
                pop     edi
                pop     esi
                pop     ebx
                add     esp, 0D8h
```

析构代理 delete pObj;

显示析构 pObj->~CTest();

Release：

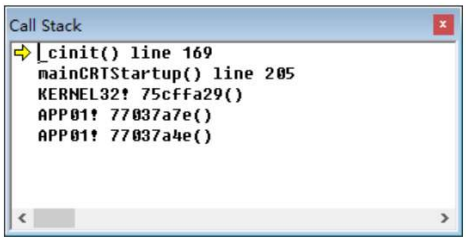
析构代理可能会被优化掉（受是否开启内联有关），判断构造，析构可根据特征进行判断。

全局对象

> 全局对象的构造函数和析构函数的调用时机：

- 类的全局对象会在main函数之前（C++在第二个initterm里）进行构造初始化并注册代理析构；
- exit调用析构。

```
    /*
     * do initializations
     */
    _initterm( __xi_a, __xi_z );

    /*
     * do C++ initializations
     */
    _initterm( __xc_a, __xc_z );

}
```



构造的调用

Debug：

```
                call    j__initterm_e
                add     esp, 8
                test    eax, eax
                jz      short loc_413F5F
                mov     [ebp+var_38], 0FFh
;   } // starts at 413EFF
                mov     [ebp+ms_exc.registration.TryLevel], 0FFFFFFFEh
                mov     eax, [ebp+var_38]
                jmp     loc_4140B2
; ---------------------------------------------------------------------------

loc_413F5F:                             ; CODE XREF: __scrt_common_main_seh(void)+97↑j
                push    offset dword_41A30C ; Last
                push    offset dword_41A000 ; First
                call    j__initterm
                add     esp, 8
                mov     dword_41D3EC, 2
                jmp     short loc_413F81
; const _PVFV dword_41A000
dword_41A000    dd 41h dup(0)           ; DATA XREF: __scrt_common_main_seh(void)+B4↑o
                dd offset ?pre_cpp_initialization@@YAXXZ ; pre_cpp_initialization(void)
                db 100h dup(0)
                dd offset sub_411910
                db 100h dup(0)
; const _PVFV dword_41A30C
dword_41A30C    dd 0                    ; DATA XREF: __scrt_common_main_seh(void):loc_413F5F↑o
                db    0
                db    0
                db    0
sub_411910      proc near               ; DATA XREF: .rdata:0041A208↓o

var_C0          = byte ptr -0C0h

                push    ebp
                mov     ebp, esp
                sub     esp, 0C0h
                push    ebx
                push    esi
                push    edi
                lea     edi, [ebp+var_C0]
                mov     ecx, 30h
                mov     eax, 0CCCCCCCCh
                rep stosd
                mov     ecx, offset unk_41F03F
                call    j_@__CheckForDebuggerJustMyCode@4 ; __CheckForDebuggerJustMyCode(x)
                push    0Ch             ; Size
                mov     ecx, offset g_obj
                call    sub_41107D
                mov     ecx, offset g_obj
                call    sub_41129E      ; 构造
                push    offset sub_418240 ; void (__cdecl *)()
                call    j__atexit
                add     esp, 4
                pop     edi
                call    j_@__CheckForDebuggerJustMyCode@4 ; __CheckForDebuggerJustMyCode(x)
                push    offset aCpersonCperson ; "CPerson::CPerson()\n"
```

调用构造函数

Release：

```
loc_40185F:                             ; CODE XREF: __scrt_common_main_seh(void)+97↑j
                push    offset dword_4030E0 ; Last
                push    offset dword_4030D4 ; First
                call    _initterm
; const _PVFV dword_4030D4
dword_4030D4    dd 0                    ; DATA XREF: __scrt_common_main_seh(void)+B4↑o
                dd offset ?pre_cpp_initialization@@YAXXZ ; pre_cpp_initialization(void)
                dd offset sub_401000

sub_401000      proc near               ; DATA XREF: .rdata:004030DC↓o
                push    ecx
                call    sub_401120
                call    sub_401070
                push    offset sub_402A10 ; void (__cdecl *)()
                call    _atexit
                pop     ecx
                retn
sub_401000      endp

sub_401070      proc near               ; CODE XREF: sub_401000+6↑p
                push    offset Format   ; "CPerson::CPerson()\n"
                call    sub_401050
                push    64h ; 'd'       ; Size
                mov     dword ptr qword_404400+4, 1
                mov     dword_404408, 64h ; 'd'
                call    unknown_libname_2 ; Microsoft VisualC 14/net runtime
                push    offset Source   ; "None"
                push    64h ; 'd'       ; SizeInBytes
                push    eax             ; Destination
                mov     dword ptr qword_404400, eax
                call    ds:strcpy_s
                add     esp, 14h
                mov     eax, offset qword_404400
                retn
sub_401070      endp
```

atexit

atexit(FunName); // FunName -- 函数名，注册终止函数（C语言，即main执行结束后调用的函数）。

```
extern "C" int __cdecl atexit(_PVFV const function)
{ 已用时间 <= 1ms
    return _onexit(reinterpret_cast<_onexit_t>(function)) != nullptr
        ? 0
        : -1;
}
```

析构的调用

> main函数结束前调用 atexit 注册析构函数方法不可取，this指针无法传递。全局对象的析构需要在构造代理中注册析构代理（传递this指针）。

Release：

```
Prox_Construct  proc near               ; DATA XREF: .rdata:004030DC↓o
                push    ecx
                call    sub_401120
                call    Constructor
                push    offset Proxy_Destructor ; void (__cdecl *)()
                call    _atexit
                pop     ecx
                retn
Prox_Construct  endp
```

```
; FUNCTION CHUNK AT .text:004010C0 SIZE 00000027 BYTES

                mov     ecx, offset qword_404400    传递 this 指针
                jmp     loc_4010C0
Proxy_Destructor endp

loc_4010C0:                             ; CODE XREF: Proxy_Destructor+5↓j
                push    esi             ; ArgList
                push    offset aCpersonCperson_0 ; "CPerson::~CPerson()\n"
                mov     esi, ecx
                call    sub_401050
```

> 全局对象的构造必须注册析构代理。所以通过分析 atexit 的参考引用可以快速定位程序中全局对象的个数。

Release：

```
; const _PVFV dword_4030D4
dword_4030D4    dd 0                    ; DATA XREF: __scrt_common_main_seh(void)+B4↑o
                dd offset ?pre_cpp_initialization@@YAXXZ ; pre_cpp_initialization(void)
                dd offset Class1
                dd offset Class2
; =============== S U B R O U T I N E =======================================


Class1          proc near               ; DATA XREF: .rdata:004030DC↓o
                push    ecx
                mov     ecx, offset g_obj1
                call    sub_401130
                call    Proxy_Construct
                push    offset Cleass1_Proxy_Destructor ; void (__cdecl *)()
                call    _atexit
                pop     ecx
                retn
Class1          endp

; ---------------------------------------------------------------------------
                align 10h

; =============== S U B R O U T I N E =======================================


Class2          proc near               ; DATA XREF: .rdata:004030E0↓o
                push    ecx
                mov     ecx, offset g_obj2
                call    sub_401130
                call    Proxy_Construct
                push    offset Cleass2_Proxy_Destructor ; void (__cdecl *)()
                call    _atexit
                pop     ecx
                retn
Class2          endp
```

全局对象数组

> 构造时会调用构造迭代器，**在构造中必须注册析构迭代器**，Release版分析如下：

```
; void __cdecl Proxy_Destructor()
Proxy_Destructor proc near              ; DATA XREF: sub_401000+18↑o
                push    offset Destructor ; void (__thiscall *)(void *)
                push    10              ; unsigned int
                push    12              ; unsigned int
                push    offset dword_404400 ; void *
                call    ??_M@YGXPAXIIP6EX0@Z@Z ; `eh vector destructor iterator'(void *,uint,uint,void (*)(void *)
                retn
Proxy_Destructor endp


; const _PVFV dword_4030D4
dword_4030D4    dd 0                    ; DATA XREF: __scrt_common_main_seh(void)+B4↑o
                dd offset ?pre_cpp_initialization@@YAXXZ ; pre_cpp_initialization(void)
                dd offset sub_401000
```

构造迭代

```
sub_401000      proc near               ; DATA XREF: .rdata:004030DC↓o
                push    offset Destructor ; void (__thiscall *)(void *)
                push    offset Constructor ; void (__thiscall *)(void *)
                push    10              ; unsigned int
                push    12              ; unsigned int
                push    offset dword_404400 ; void *
                call    ??_L@YGXPAXIIP6EX0@Z1@Z ; `eh vector constructor iterator'(void *,uint,uint,void (*)(void
                push    offset Proxy_Destructor ; void (__cdecl *)()
                call    _atexit
                pop     ecx
                retn                                    析构代理
sub_401000      endp
```

析构迭代

```
; void __cdecl Proxy_Destructor()
Proxy_Destructor proc near              ; DATA XREF: sub_401000+18↑o
                push    offset Destructor ; void (__thiscall *)(void *)
                push    10              ; unsigned int
                push    12              ; unsigned int
                push    offset dword_404400 ; void *
                call    ??_M@YGXPAXIIP6EX0@Z@Z ; `eh vector destructor iterator'(void *,uint,uint,void (*)(void *)
                retn
Proxy_Destructor endp
```

堆对象数组（全局）

> **不存在构造中注册析构代理**，delete 对象数组时，在delete处会调用析构迭代器。

CPerson* g_pObjs = new CPerson[10];

Release：

```
; ---------------------------------------------------------------------------

loc_4016AF:                             ; CODE XREF: __scrt_common_main_seh(void)+97↑j
                push    offset dword_4030E4 ; Last
                push    offset dword_4030D8 ; First
                call    _initterm
; const _PVFV dword_4030D8
dword_4030D8    dd 0                    ; DATA XREF: __scrt_common_main_seh(void)+B4↑o
                dd offset ?pre_cpp_initialization@@YAXXZ ; pre_cpp_initialization(void)
                dd offset sub_401000
```

保存对象this指针：

```
; __unwind { // SEH_401000
                push    ebp
                mov     ebp, esp
                push    0FFFFFFFFh
                push    offset SEH_401000
                mov     eax, large fs:0
                push    eax
                push    ecx
                push    esi
                mov     eax, ___security_cookie
                xor     eax, ebp
                push    eax
                lea     eax, [ebp+var_C]
                mov     large fs:0, eax
                push    124             ; Size
                call    unknown_libname_2 ; Microsoft VisualC 14/net runtime
                add     esp, 4
                mov     [ebp+var_10], eax
                push    offset Destructor ; void (__thiscall *)(void *)
                push    offset Constructor ; void (__thiscall *)(void *)
                push    10              ; unsigned int
                lea     esi, [eax+4]
;   try {
                mov     [ebp+var_4], 0
                push    12              ; unsigned int
                push    esi             ; void *
                mov     dword ptr [eax], 10
                call    ??_L@YGXPAXIIP6EX0@Z1@Z ; `eh vector constructor iterator'(void *,uint,uint,void (*)(void *
                mov     dword_404400, esi ; 保存this指针，方便调用delete时调用析构迭代器
```

delete 全局堆对象，判断堆对象（不为空，调用自购迭代器）：