# 리버스 엔지니어링 바이블

ch03. C++ 클래스와 리버스 엔지니어링

저자 : 강병탁

김영철

2015. 10. 28.

# 클래스 뼈대

```
class Employee
{
        public :
                int number;
                char* name;
                long pay;
                void ShowData();
                void Test();
};

void Employee::ShowData()
{
        printf("number : %d\n", number);
        printf("name : %s\n", name);
        printf("pay : %ld\n", pay);
        Test();
}

void Employee::Test()
{
        printf("Test function\n");
        return;
}
```

```
int main(int argc, char* argv[])
{
        Employee kim;

        printf("size : %X\n", sizeof(Employee));

        kim.number = 0x1111;
        kim.name = "김영철";
        kim.pay = 0x100;

        kim.ShowData();
        return 0;
}
```

# 클래스 뼈대

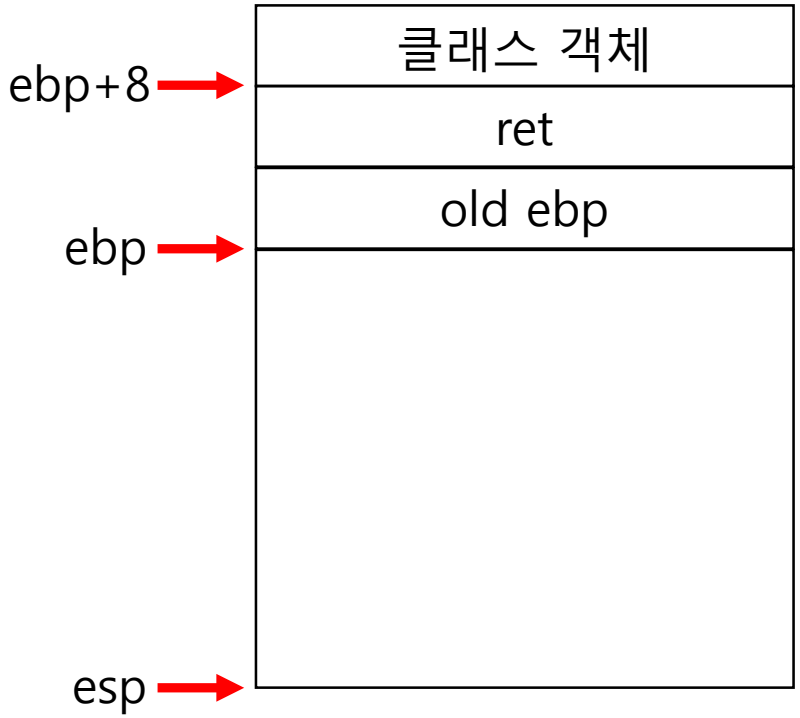
```
Dump of assembler code for function main:
   0x080484b7 <+0>:     push   ebp
   0x080484b8 <+1>:     mov    ebp,esp
   0x080484ba <+3>:     and    esp,0xfffffff0
   0x080484bd <+6>:     sub    esp,0x20
   0x080484c0 <+9>:     mov    DWORD PTR [esp+0x4],0xc
   0x080484c8 <+17>:    mov    DWORD PTR [esp],0x80485c1
   0x080484cf <+24>:    call   0x8048310 <printf@plt>
   0x080484d4 <+29>:    mov    DWORD PTR [esp+0x14],0x1111
   0x080484dc <+37>:    mov    DWORD PTR [esp+0x18],0x80485cc
   0x080484e4 <+45>:    mov    DWORD PTR [esp+0x1c],0x100
   0x080484ec <+53>:    lea    eax,[esp+0x14]
   0x080484f0 <+57>:    mov    DWORD PTR [esp],eax
   0x080484f3 <+60>:    call   0x804844e <_ZN8Employee8ShowDataEv>
   0x080484f8 <+65>:    mov    eax,0x0
   0x080484fd <+70>:    leave
   0x080484fe <+71>:    ret
End of assembler dump.
```

객체의 멤버에 값을 넣는 부분

thiscall 방식
=> 클래스의 메모리 위치를 알 수 있다.

# 클래스 뼈대

```
Dump of assembler code for function _ZN8Employee8ShowDataEv:
   0x0804844e <+0>:     push   ebp
   0x0804844f <+1>:     mov    ebp,esp
   0x08048451 <+3>:     sub    esp,0x18
   0x08048454 <+6>:     mov    eax,DWORD PTR [ebp+0x8]
   0x08048457 <+9>:     mov    eax,DWORD PTR [eax]
   0x08048459 <+11>:    mov    DWORD PTR [esp+0x4],eax
   0x0804845d <+15>:    mov    DWORD PTR [esp],0x8048590
   0x08048464 <+22>:    call   0x8048310 <printf@plt>
   0x08048469 <+27>:    mov    eax,DWORD PTR [ebp+0x8]
   0x0804846c <+30>:    mov    eax,DWORD PTR [eax+0x4]
   0x0804846f <+33>:    mov    DWORD PTR [esp+0x4],eax
   0x08048473 <+37>:    mov    DWORD PTR [esp],0x804859d
   0x0804847a <+44>:    call   0x8048310 <printf@plt>
   0x0804847f <+49>:    mov    eax,DWORD PTR [ebp+0x8]
   0x08048482 <+52>:    mov    eax,DWORD PTR [eax+0x8]
   0x08048485 <+55>:    mov    DWORD PTR [esp+0x4],eax
   0x08048489 <+59>:    mov    DWORD PTR [esp],0x80485a8
   0x08048490 <+66>:    call   0x8048310 <printf@plt>
   0x08048495 <+71>:    mov    eax,DWORD PTR [ebp+0x8]
   0x08048498 <+74>:    mov    DWORD PTR [esp],eax
   0x0804849b <+77>:    call   0x80484a2 <_ZN8Employee4TestEv>
   0x080484a0 <+82>:    leave
   0x080484a1 <+83>:    ret
End of assembler dump.
```



# 클래스 뼈대

```
Dump of assembler code for function _ZN8Employee8ShowDataEv:
   0x0804844e <+0>:     push   ebp
   0x0804844f <+1>:     mov    ebp,esp
   0x08048451 <+3>:     sub    esp,0x18
   0x08048454 <+6>:     mov    eax,DWORD PTR [ebp+0x8]
   0x08048457 <+9>:     mov    eax,DWORD PTR [eax]
   0x08048459 <+11>:    mov    DWORD PTR [esp+0x4],eax
   0x0804845d <+15>:    mov    DWORD PTR [esp],0x8048590
   0x08048464 <+22>:    call   0x8048310 <printf@plt>
   0x08048469 <+27>:    mov    eax,DWORD PTR [ebp+0x8]
   0x0804846c <+30>:    mov    eax,DWORD PTR [eax+0x4]
   0x0804846f <+33>:    mov    DWORD PTR [esp+0x4],eax
   0x08048473 <+37>:    mov    DWORD PTR [esp],0x804859d
   0x0804847a <+44>:    call   0x8048310 <printf@plt>
   0x0804847f <+49>:    mov    eax,DWORD PTR [ebp+0x8]
   0x08048482 <+52>:    mov    eax,DWORD PTR [eax+0x8]
   0x08048485 <+55>:    mov    DWORD PTR [esp+0x4],eax
   0x08048489 <+59>:    mov    DWORD PTR [esp],0x80485a8
   0x08048490 <+66>:    call   0x8048310 <printf@plt>
   0x08048495 <+71>:    mov    eax,DWORD PTR [ebp+0x8]
   0x08048498 <+74>:    mov    DWORD PTR [esp],eax
   0x0804849b <+77>:    call   0x80484a2 <_ZN8Employee4TestEv>
   0x080484a0 <+82>:    leave
   0x080484a1 <+83>:    ret
End of assembler dump.
```

eax = 객체의 주소값

객체의 주소가 가리키는 값은 첫 번째 멤버

객체의 주소+4가 가리키는 값은 두 번째 멤버

객체의 주소+8이 가리키는 값은 세 번째 멤버

# 클래스의 수명과 전역변수

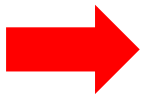
- 지역변수로 선언

```
int main(int argc, char* argv[])
{
        Employee kim;

        printf("size : %X\n", sizeof(Employee));

        kim.number = 0x1111;
        kim.name = "김영철";
        kim.pay = 0x100;

        kim.ShowData();
        return 0;
}
```

# 클래스의 수명과 전역변수

- 지역변수로 선언

```
Dump of assembler code for function main:
   0x080484b7 <+0>:     push   ebp
   0x080484b8 <+1>:     mov    ebp,esp
   0x080484ba <+3>:     and    esp,0xfffffff0
   0x080484bd <+6>:     sub    esp,0x20
   0x080484c0 <+9>:     mov    DWORD PTR [esp+0x4],0xc
   0x080484c8 <+17>:    mov    DWORD PTR [esp],0x80485c1
   0x080484cf <+24>:    call   0x8048310 <printf@plt>
   0x080484d4 <+29>:    mov    DWORD PTR [esp+0x14],0x1111
   0x080484dc <+37>:    mov    DWORD PTR [esp+0x18],0x80485cc
   0x080484e4 <+45>:    mov    DWORD PTR [esp+0x1c],0x100
   0x080484ec <+53>:    lea    eax,[esp+0x14]
   0x080484f0 <+57>:    mov    DWORD PTR [esp],eax
   0x080484f3 <+60>:    call   0x804844e <_ZN8Employee8ShowDataEv>
   0x080484f8 <+65>:    mov    eax,0x0
   0x080484fd <+70>:    leave
   0x080484fe <+71>:    ret
End of assembler dump.
```



스택공간에 저장
=> 함수 호출 종료 후, 제거

# 클래스의 수명과 전역변수

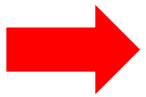
- 전역변수로 선언

```
Employee kim;
int main(int argc, char* argv[])
{
        printf("size : %X\n", sizeof(Employee));

        kim.number = 0x1111;
        kim.name = "김영철";
        kim.pay = 0x100;

        kim.ShowData();
        return 0;
}
```

# 클래스의 수명과 전역변수

- 전역변수로 선언

```
Dump of assembler code for function main:
   0x080484b7 <+0>:     push   ebp
   0x080484b8 <+1>:     mov    ebp,esp
   0x080484ba <+3>:     and    esp,0xfffffff0
   0x080484bd <+6>:     sub    esp,0x10
   0x080484c0 <+9>:     mov    DWORD PTR [esp+0x4],0xc
   0x080484c8 <+17>:    mov    DWORD PTR [esp],0x80485d1
   0x080484cf <+24>:    call   0x8048310 <printf@plt>
   0x080484d4 <+29>:    mov    DWORD PTR ds:0x804a028,0x1111
   0x080484de <+39>:    mov    DWORD PTR ds:0x804a02c,0x80485dc
   0x080484e8 <+49>:    mov    DWORD PTR ds:0x804a030,0x100
   0x080484f2 <+59>:    mov    DWORD PTR [esp],0x804a028
   0x080484f9 <+66>:    call   0x804844e <_ZN8Employee8ShowDataEv>
   0x080484fe <+71>:    mov    eax,0x0
   0x08048503 <+76>:    leave
   0x08048504 <+77>:    ret
End of assembler dump.
```



데이터 영역에 저장
=> 함수 호출 후에도 사용 가능

# 객체의 동적 할당

```
int main(int argc, char* argv[])
{
        Employee *kim;
        kim = new Employee;

        printf("size : %X\n", sizeof(Employee));

        kim->number = 0x1111;
        kim->name = "김영철";
        kim->pay = 0x100;

        kim->ShowData();
        return 0;
}
```

# 객체의 동적 할당

```
Dump of assembler code for function main:
   0x08048597 <+0>:     push   ebp
   0x08048598 <+1>:     mov    ebp,esp
   0x0804859a <+3>:     and    esp,0xfffffff0
   0x0804859d <+6>:     sub    esp,0x20
   0x080485a0 <+9>:     mov    DWORD PTR [esp],0xc
   0x080485a7 <+16>:    call   0x8048410 <_Znwj@plt>
   0x080485ac <+21>:    mov    DWORD PTR [esp+0x1c],eax
   0x080485b0 <+25>:    mov    DWORD PTR [esp+0x4],0xc
   0x080485b8 <+33>:    mov    DWORD PTR [esp],0x80486c1
   0x080485bf <+40>:    call   0x8048400 <printf@plt>
   0x080485c4 <+45>:    mov    eax,DWORD PTR [esp+0x1c]
   0x080485c8 <+49>:    mov    DWORD PTR [eax],0x1111
   0x080485ce <+55>:    mov    eax,DWORD PTR [esp+0x1c]
   0x080485d2 <+59>:    mov    DWORD PTR [eax+0x4],0x80486cc
   0x080485d9 <+66>:    mov    eax,DWORD PTR [esp+0x1c]
   0x080485dd <+70>:    mov    DWORD PTR [eax+0x8],0x100
   0x080485e4 <+77>:    mov    eax,DWORD PTR [esp+0x1c]
   0x080485e8 <+81>:    mov    DWORD PTR [esp],eax
   0x080485eb <+84>:    call   0x804852e <_ZN8Employee8ShowDataEv>
   0x080485f0 <+89>:    mov    eax,0x0
   0x080485f5 <+94>:    leave
   0x080485f6 <+95>:    ret
End of assembler dump.
```

객체의 크기만큼 할당하여 new 함수를 호출
[esp+0x1c]에 객체가 할당된 주소 저장

객체의 주소값을 이용하여 멤버에 값을 저장

# 생성자와 소멸자

```
class Employee
{
        public :
                int number;
                char* name;
                long pay;
                Employee();
                ~Employee();
};

Employee::Employee()
{
        printf("constructor!!\n");
}

Employee::~Employee()
{
        printf("destructor!!\n");
}
```

```
int main(int argc, char* argv[])
{
        Employee kim;
        kim.number = 0x111;
        kim.name = "김영철";
        kim.pay = 0x2222;
        return 0;
}
```

# 생성자와 소멸자

```
Dump of assembler code for function main:
   0x08048446 <+0>:     push   ebp
   0x08048447 <+1>:     mov    ebp,esp
   0x08048449 <+3>:     push   ebx
   0x0804844a <+4>:     and    esp,0xfffffff0
   0x0804844d <+7>:     sub    esp,0x20
   0x08048450 <+10>:    lea    eax,[esp+0x14]
   0x08048454 <+14>:    mov    DWORD PTR [esp],eax
   0x08048457 <+17>:    call   0x804841e <_ZN8EmployeeC2Ev>
   0x0804845c <+22>:    mov    DWORD PTR [esp+0x14],0x111
   0x08048464 <+30>:    mov    DWORD PTR [esp+0x18],0x804853b
   0x0804846c <+38>:    mov    DWORD PTR [esp+0x1c],0x2222
   0x08048474 <+46>:    mov    ebx,0x0
   0x08048479 <+51>:    lea    eax,[esp+0x14]
   0x0804847d <+55>:    mov    DWORD PTR [esp],eax
   0x08048480 <+58>:    call   0x8048432 <_ZN8EmployeeD2Ev>
   0x08048485 <+63>:    mov    eax,ebx
   0x08048487 <+65>:    mov    ebx,DWORD PTR [ebp-0x4]
   0x0804848a <+68>:    leave
   0x0804848b <+69>:    ret
End of assembler dump.
```

# 캡슐화 분석

```
class Employee
{
        public :
                int number;
                char* name;
                long pay;
                void functionTest();
        private :
                void privateFunction();
};

void Employee::functionTest()
{
        privateFunction();
}

void Employee::privateFunction()
{
        printf("private Function Test.\n");
}
```

```
int main(int argc, char* argv[])
{
        Employee *kim;
        kim = new Employee;

        kim->number = 0x111;
        kim->name = "김영철";
        kim->pay = 0x2222;

        kim->functionTest();

        delete kim;
        return 0;
}
```

# 캡슐화 분석

```
Dump of assembler code for function _ZN8Employee12functionTestEv:
   0x0804852e <+0>:     push   ebp
   0x0804852f <+1>:     mov    ebp,esp
   0x08048531 <+3>:     sub    esp,0x18
   0x08048534 <+6>:     mov    eax,DWORD PTR [ebp+0x8]
   0x08048537 <+9>:     mov    DWORD PTR [esp],eax
   0x0804853a <+12>:    call   0x8048542 <_ZN8Employee15privateFunctionEv>
   0x0804853f <+17>:    leave
   0x08048540 <+18>:    ret
End of assembler dump.
```

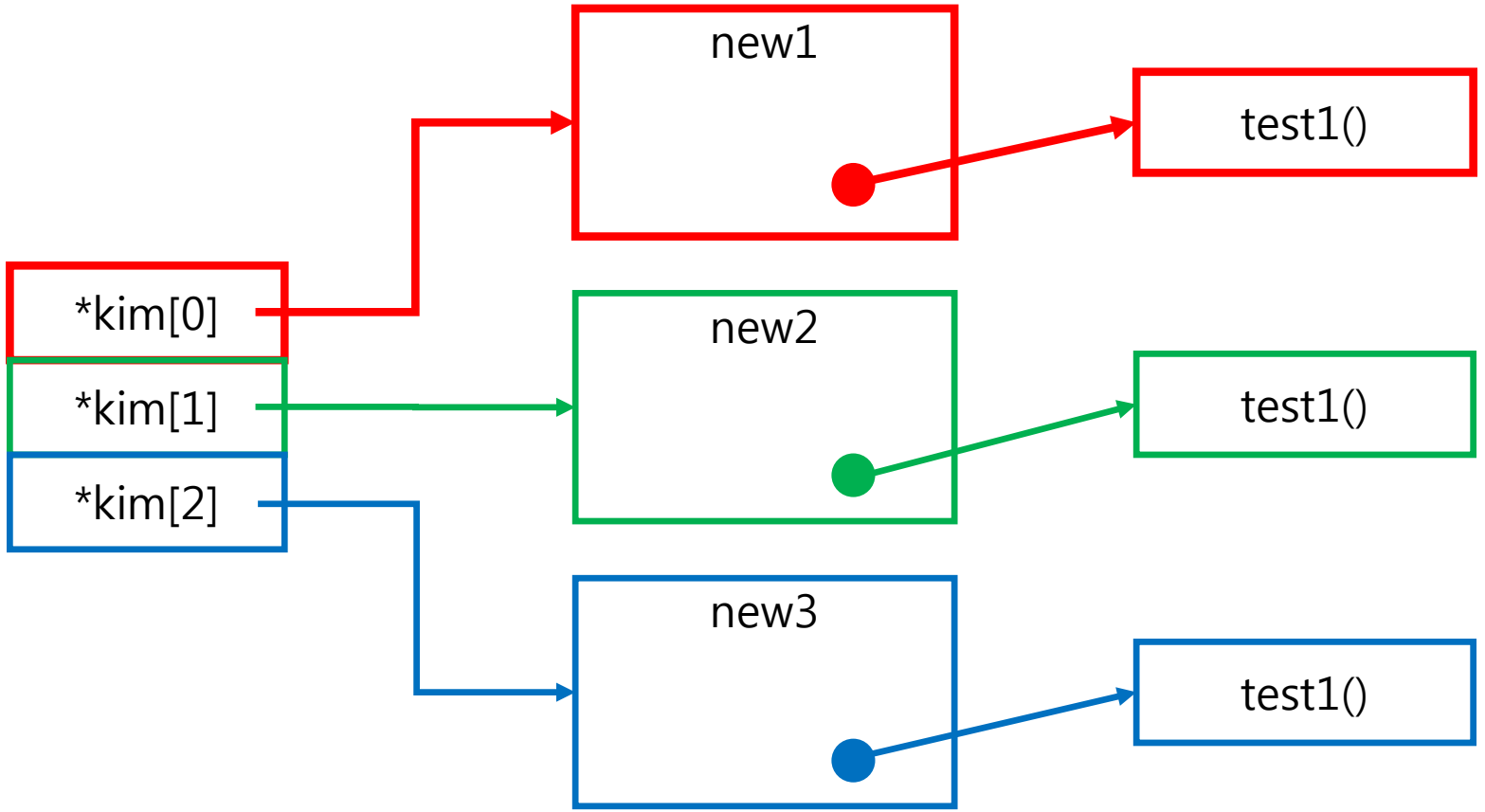
```
Dump of assembler code for function _ZN8Employee15privateFunctionEv:
   0x08048542 <+0>:     push   ebp
   0x08048543 <+1>:     mov    ebp,esp
   0x08048545 <+3>:     sub    esp,0x18
   0x08048548 <+6>:     mov    DWORD PTR [esp],0x8048640
   0x0804854f <+13>:    call   0x8048420 <puts@plt>
   0x08048554 <+18>:    leave
   0x08048555 <+19>:    ret
End of assembler dump.
```

# 다형성 구조 파악

```
class Employee
{
        public :
                int number;
                char* name;
                long pay;
                void virtual test1()=0;
};

class new1 : public Employee
{
        public :
                virtual void test1() { printf("new 1\n"); }
};

class new2 : public Employee
{
        public :
                virtual void test1() { printf("new 2\n"); }
};

class new3 : public Employee
{
        public :
                virtual void test1() { printf("new 3\n"); }
};
```

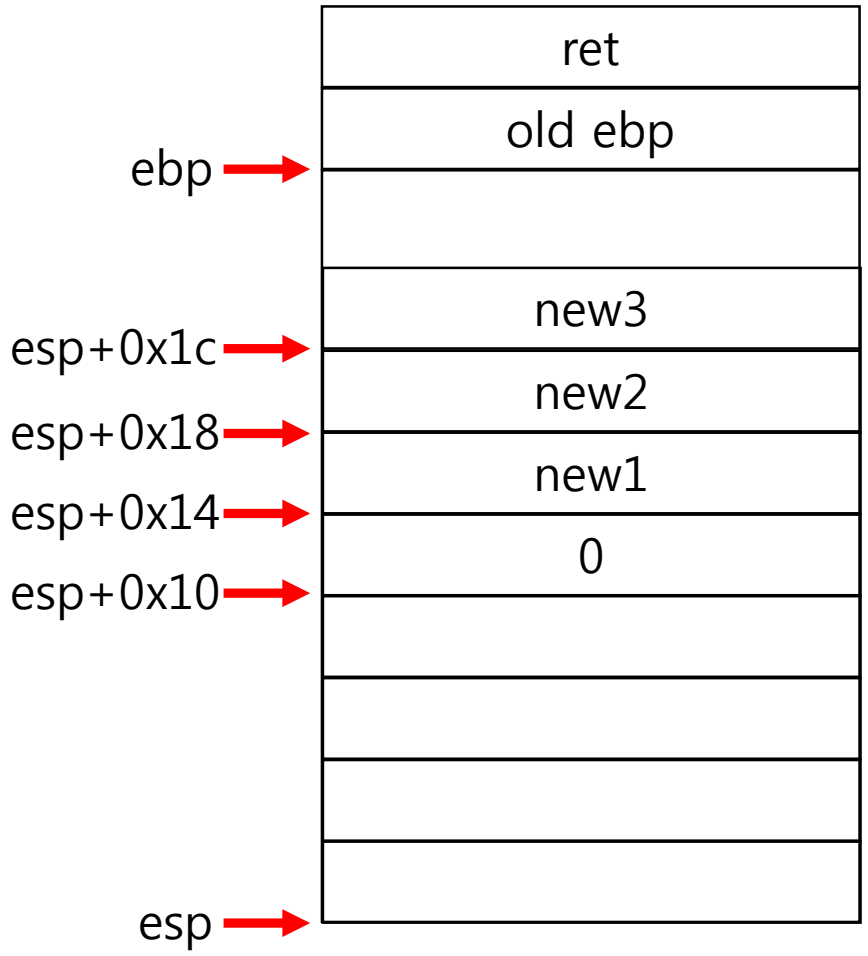
```
int main(int argc, char* argv[])
{
        Employee *kim[3];
        int i;

        kim[0] = new new1;
        kim[1] = new new2;
        kim[2] = new new3;

        for(i=0; i<3; i++)
                kim[i]->test1();

        for(i=0; i<3; i++)
                delete kim[i];

        return 0;
}
```

# 다형성 구조 파악



# 다형성 구조 파악

```
0x08048675 <+88>:    mov    DWORD PTR [esp+0x10],0x0
0x0804867d <+96>:    jmp    0x804869d <main+128>
0x0804867f <+98>:    mov    eax,DWORD PTR [esp+0x10]
0x08048683 <+102>:   mov    eax,DWORD PTR [esp+eax*4+0x14]
0x08048687 <+106>:   mov    eax,DWORD PTR [eax]
0x08048689 <+108>:   mov    eax,DWORD PTR [eax]
0x0804868b <+110>:   mov    edx,DWORD PTR [esp+0x10]
0x0804868f <+114>:   mov    edx,DWORD PTR [esp+edx*4+0x14]
--Type <return> to continue, or q <return> to quit---
0x08048693 <+118>:   mov    DWORD PTR [esp],edx
0x08048696 <+121>:   call   eax
0x08048698 <+123>:   add    DWORD PTR [esp+0x10],0x1
0x0804869d <+128>:   cmp    DWORD PTR [esp+0x10],0x2
0x080486a2 <+133>:   jle    0x804867f <main+98>
0x080486a4 <+135>:   mov    DWORD PTR [esp+0x10],0x0
0x080486ac <+143>:   jmp    0x80486c3 <main+166>
0x080486ae <+145>:   mov    eax,DWORD PTR [esp+0x10]
0x080486b2 <+149>:   mov    eax,DWORD PTR [esp+eax*4+0x14]
0x080486b6 <+153>:   mov    DWORD PTR [esp],eax
0x080486b9 <+156>:   call   0x80484d0 <_ZdlPv@plt>
0x080486be <+161>:   add    DWORD PTR [esp+0x10],0x1
0x080486c3 <+166>:   cmp    DWORD PTR [esp+0x10],0x2
0x080486c8 <+171>:   jle    0x80486ae <main+145>
0x080486ca <+173>:   mov    eax,0x0
0x080486cf <+178>:   mov    ebx,DWORD PTR [ebp-0x4]
0x080486d2 <+181>:   leave
0x080486d3 <+182>:   ret
```



# 다형성 구조 파악

```
0x08048675 <+88>:    mov    DWORD PTR [esp+0x10],0x0
0x0804867d <+96>:    jmp    0x804869d <main+128>
0x0804867f <+98>:    mov    eax,DWORD PTR [esp+0x10]
0x08048683 <+102>:   mov    eax,DWORD PTR [esp+eax*4+0x14]
0x08048687 <+106>:   mov    eax,DWORD PTR [eax]
0x08048689 <+108>:   mov    eax,DWORD PTR [eax]
0x0804868b <+110>:   mov    edx,DWORD PTR [esp+0x10]
0x0804868f <+114>:   mov    edx,DWORD PTR [esp+edx*4+0x14]
--Type <return> to continue, or q <return> to quit---
0x08048693 <+118>:   mov    DWORD PTR [esp],edx
0x08048696 <+121>:   call   eax
0x08048698 <+123>:   add    DWORD PTR [esp+0x10],0x1
0x0804869d <+128>:   cmp    DWORD PTR [esp+0x10],0x2
0x080486a2 <+133>:   jle    0x804867f <main+98>
0x080486a4 <+135>:   mov    DWORD PTR [esp+0x10],0x0
0x080486ac <+143>:   jmp    0x80486c3 <main+166>
0x080486ae <+145>:   mov    eax,DWORD PTR [esp+0x10]
0x080486b2 <+149>:   mov    eax,DWORD PTR [esp+eax*4+0x14]
0x080486b6 <+153>:   mov    DWORD PTR [esp],eax
0x080486b9 <+156>:   call   0x80484d0 <_ZdlPv@plt>
0x080486be <+161>:   add    DWORD PTR [esp+0x10],0x1
0x080486c3 <+166>:   cmp    DWORD PTR [esp+0x10],0x2
0x080486c8 <+171>:   jle    0x80486ae <main+145>
0x080486ca <+173>:   mov    eax,0x0
0x080486cf <+178>:   mov    ebx,DWORD PTR [ebp-0x4]
0x080486d2 <+181>:   leave
0x080486d3 <+182>:   ret
```

```
(gdb) x/x $esp+0x14
0xbffff6b4:     0x0804b008
(gdb) x/x 0x804b008
0x804b008:      0x08048850
(gdb) x/x 0x8048850
0x8048850 <_ZTV4new1+8>:        0x080486d4
(gdb) x/i 0x80486d4
   0x80486d4 <_ZN4new15test1Ev>:        push   ebp
```