# 리버스 엔지니어링

-3장 : C++클래스와 리버스 엔지니어링

김 정 우

## 이때까지 배운 것 들..

01 기본 어셈블리어

02  C언어와 리버스 엔지니어링

03

## C++클래스와 리버스엔지니어링

## Intro

클래스와 멤버변수

전역변수

객체의 동적할당

생성자와 소멸자

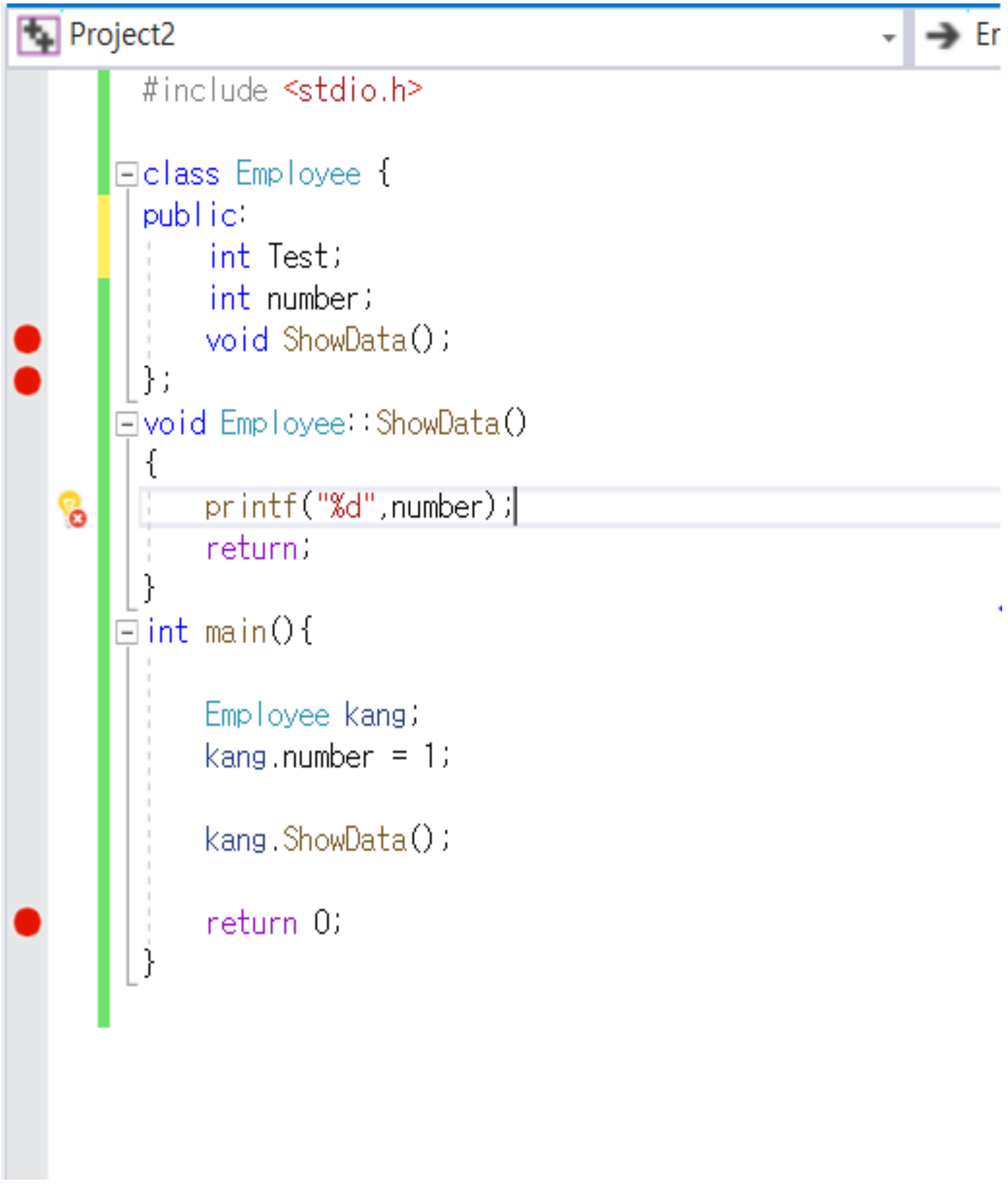
캡슐화

다형성 구조

**이번 시간에 확인할 키워드!**

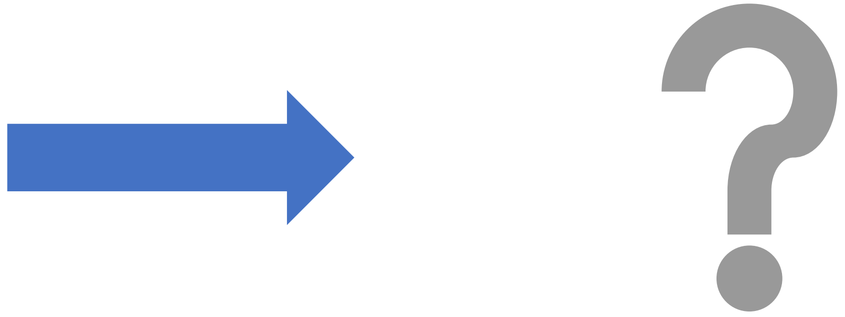
Body

## 클래스의 뼈대



Project2

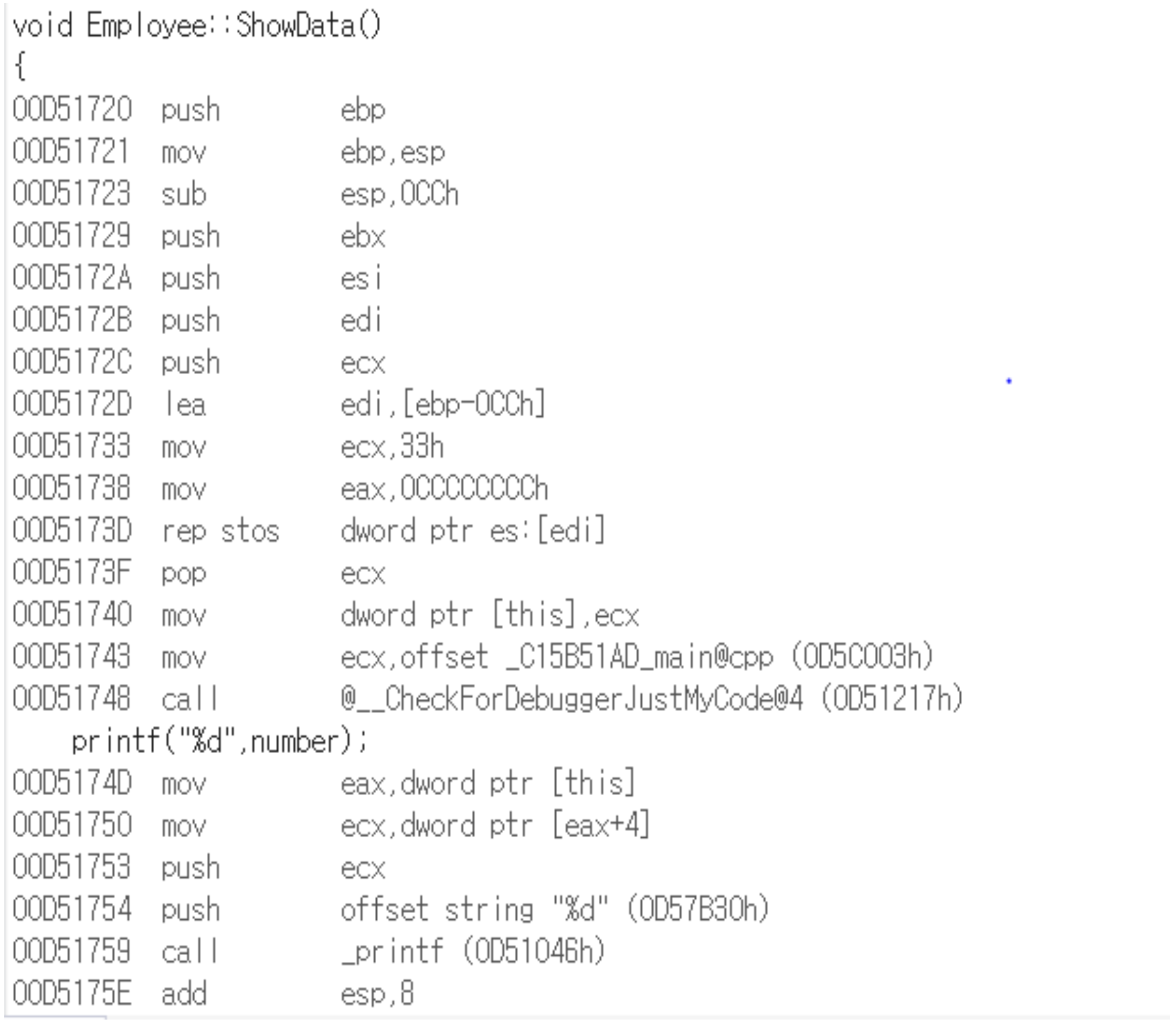
```
#include <stdio.h>

class Employee {
public:
    int Test;
    int number;
    void ShowData();
};
void Employee::ShowData()
{
    printf("%d",number);
    return;
}
int main(){

    Employee kang;
    kang.number = 1;

    kang.ShowData();

    return 0;
}
```



Body

## 클래스의 뼈대

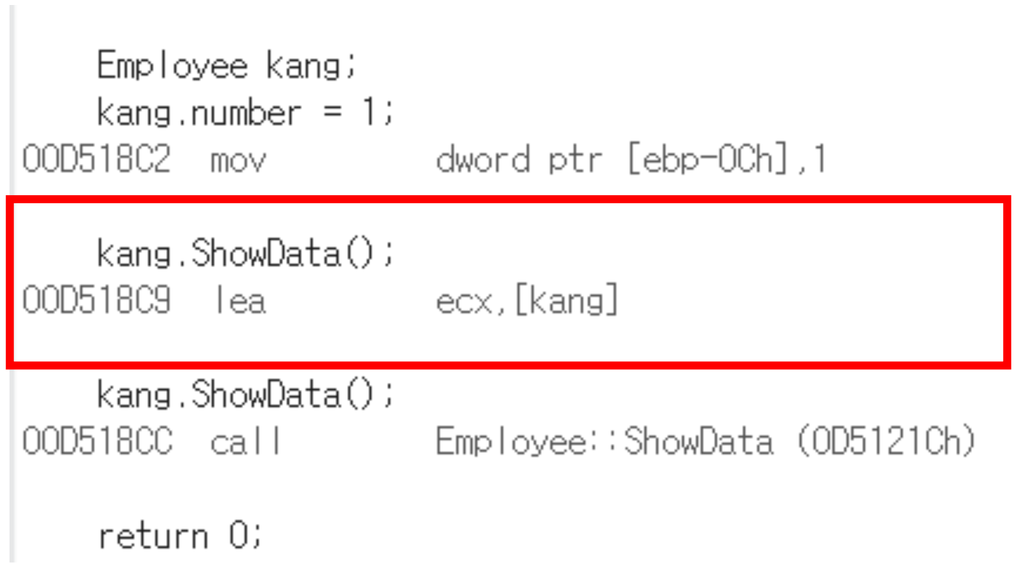
1. 클래스에 구현되는 함수부터 빌드!



```
void Employee::ShowData()
{
00D51720  push        ebp
00D51721  mov         ebp,esp
00D51723  sub         esp,0CCh
00D51729  push        ebx
00D5172A  push        esi
00D5172B  push        edi
00D5172C  push        ecx
00D5172D  lea         edi,[ebp-0CCh]
00D51733  mov         ecx,33h
00D51738  mov         eax,0CCCCCCCCh
00D5173D  rep stos    dword ptr es:[edi]
00D5173F  pop         ecx
00D51740  mov         dword ptr [this],ecx
00D51743  mov         ecx,offset _C15B51AD_main@cpp (0D5C003h)
00D51748  call        @__CheckForDebuggerJustMyCode@4 (0D51217h)
    printf("%d",number);
00D5174D  mov         eax,dword ptr [this]
00D51750  mov         ecx,dword ptr [eax+4]
00D51753  push        ecx
00D51754  push        offset string "%d" (0D57B30h)
00D51759  call        _printf (0D51046h)
00D5175E  add         esp,8
```

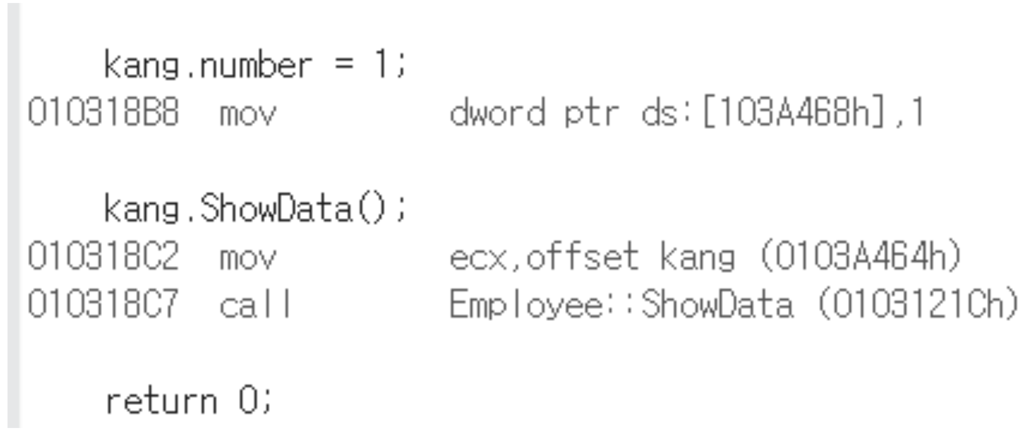
## Body

1. main()



```
    Employee kang;
    kang.number = 1;
00D518C2  mov         dword ptr [ebp-0Ch],1

    kang.ShowData();
00D518C9  lea         ecx,[kang]

    kang.ShowData();
00D518CC  call        Employee::ShowData (0D5121Ch)

    return 0;
```

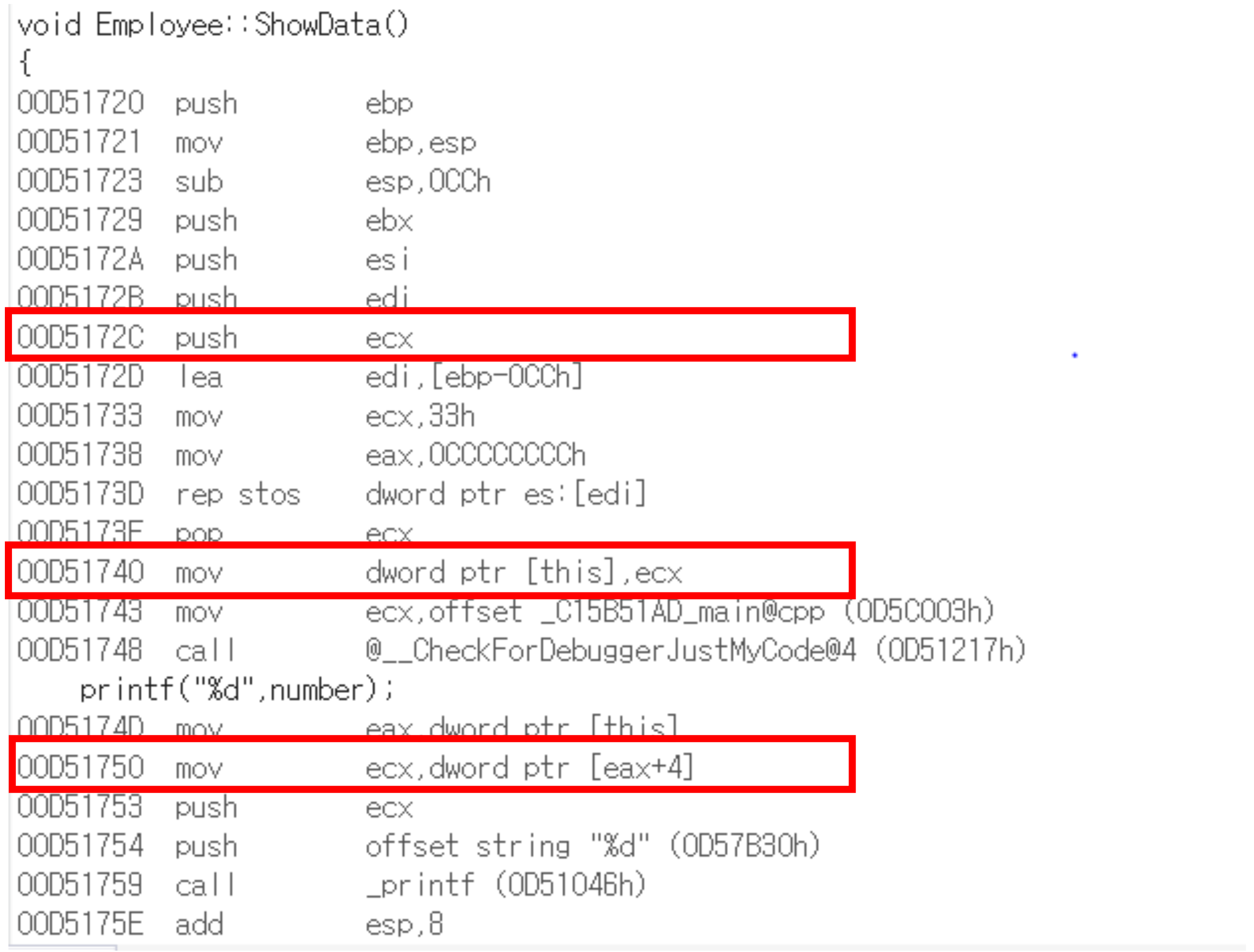
2. 클래스를 전역으로 선언 시



```
    kang.number = 1;
010318B8  mov         dword ptr ds:[103A468h],1

    kang.ShowData();
010318C2  mov         ecx,offset kang (0103A464h)
010318C7  call        Employee::ShowData (0103121Ch)

    return 0;
```

Body

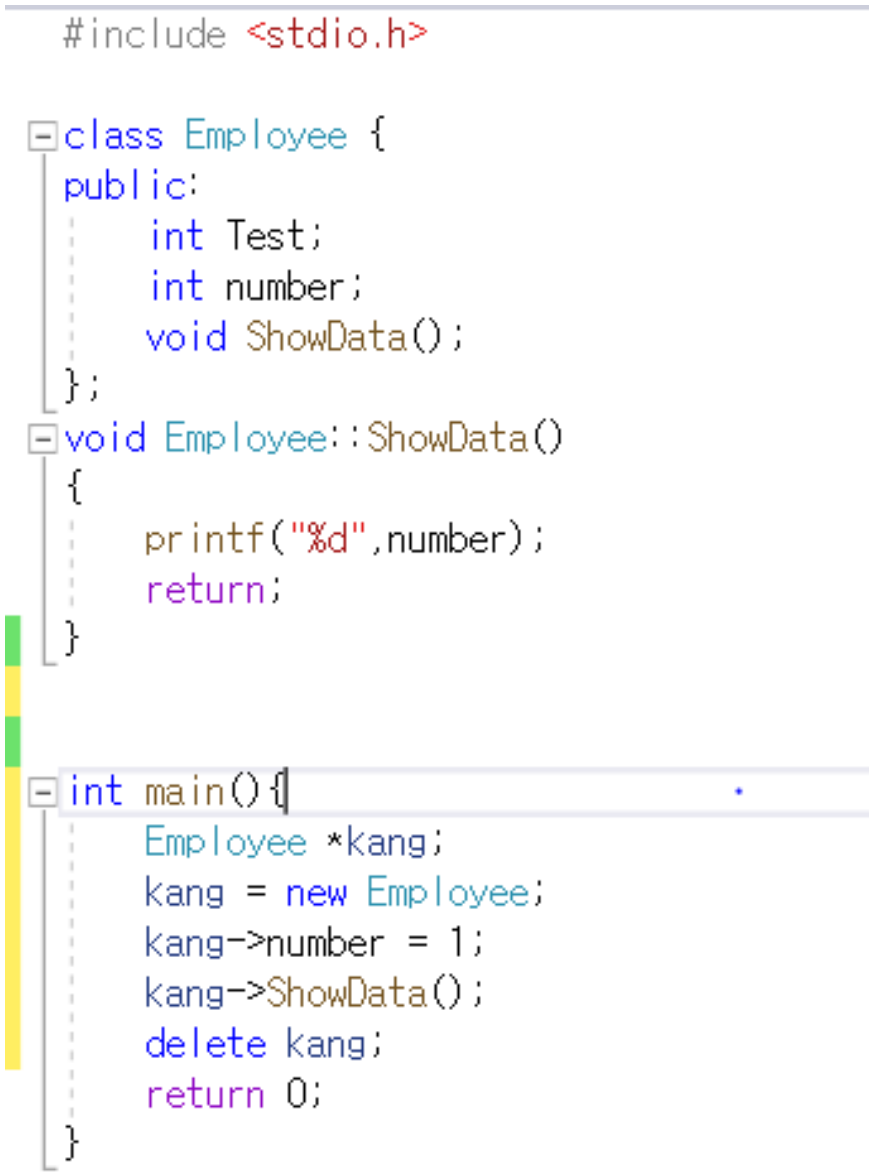
## 클래스의 뼈대



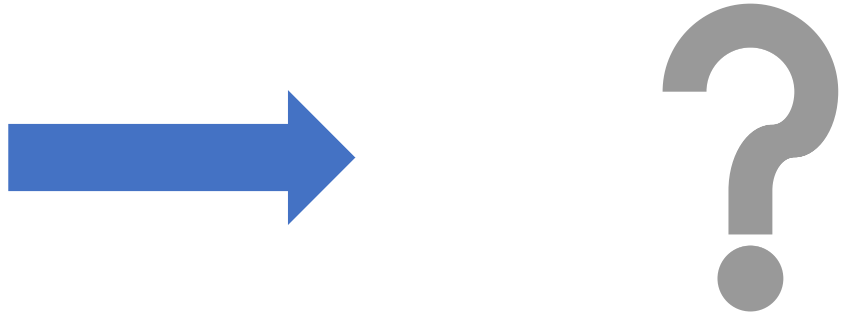
```
void Employee::ShowData()
{
00D51720  push        ebp
00D51721  mov         ebp,esp
00D51723  sub         esp,0CCh
00D51729  push        ebx
00D5172A  push        esi
00D5172B  push        edi
00D5172C  push        ecx
00D5172D  lea         edi,[ebp-0CCh]
00D51733  mov         ecx,33h
00D51738  mov         eax,0CCCCCCCCh
00D5173D  rep stos    dword ptr es:[edi]
00D5173F  pop         ecx
00D51740  mov         dword ptr [this],ecx
00D51743  mov         ecx,offset _C15B51AD_main@cpp (0D5C003h)
00D51748  call        @__CheckForDebuggerJustMyCode@4 (0D51217h)
    printf("%d",number);
00D5174D  mov         eax,dword ptr [this]
00D51750  mov         ecx,dword ptr [eax+4]
00D51753  push        ecx
00D51754  push        offset string "%d" (0D57B30h)
00D51759  call        _printf (0D51046h)
00D5175E  add         esp,8
```

Body

## 객체의 동적할당과 해제

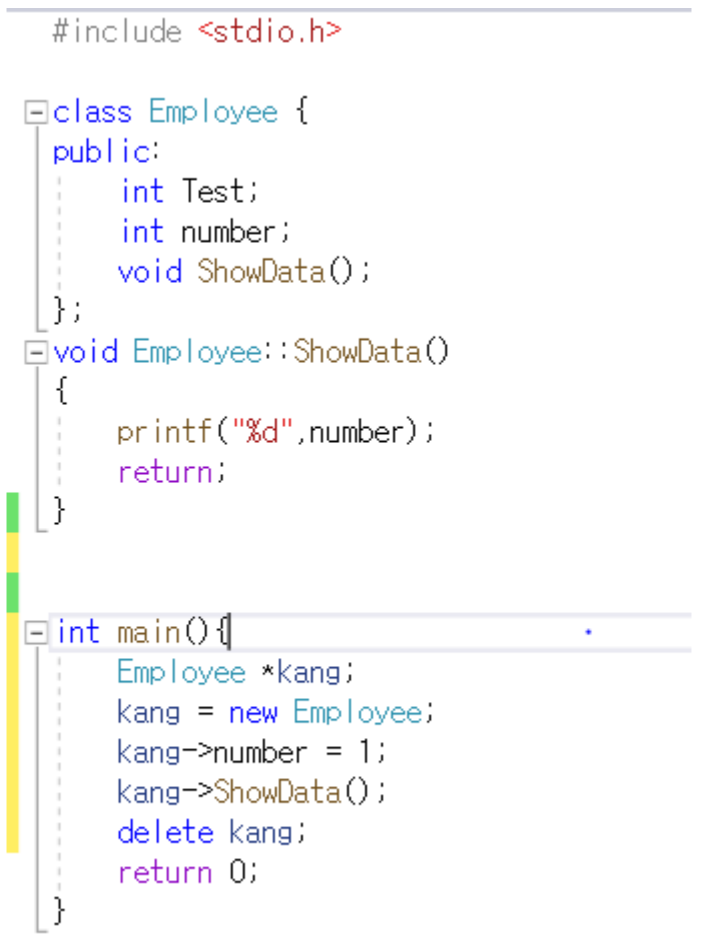


```
#include <stdio.h>

class Employee {
public:
    int Test;
    int number;
    void ShowData();
};
void Employee::ShowData()
{
    printf("%d",number);
    return;
}


int main(){
    Employee *kang;
    kang = new Employee;
    kang->number = 1;
    kang->ShowData();
    delete kang;
    return 0;
}
```

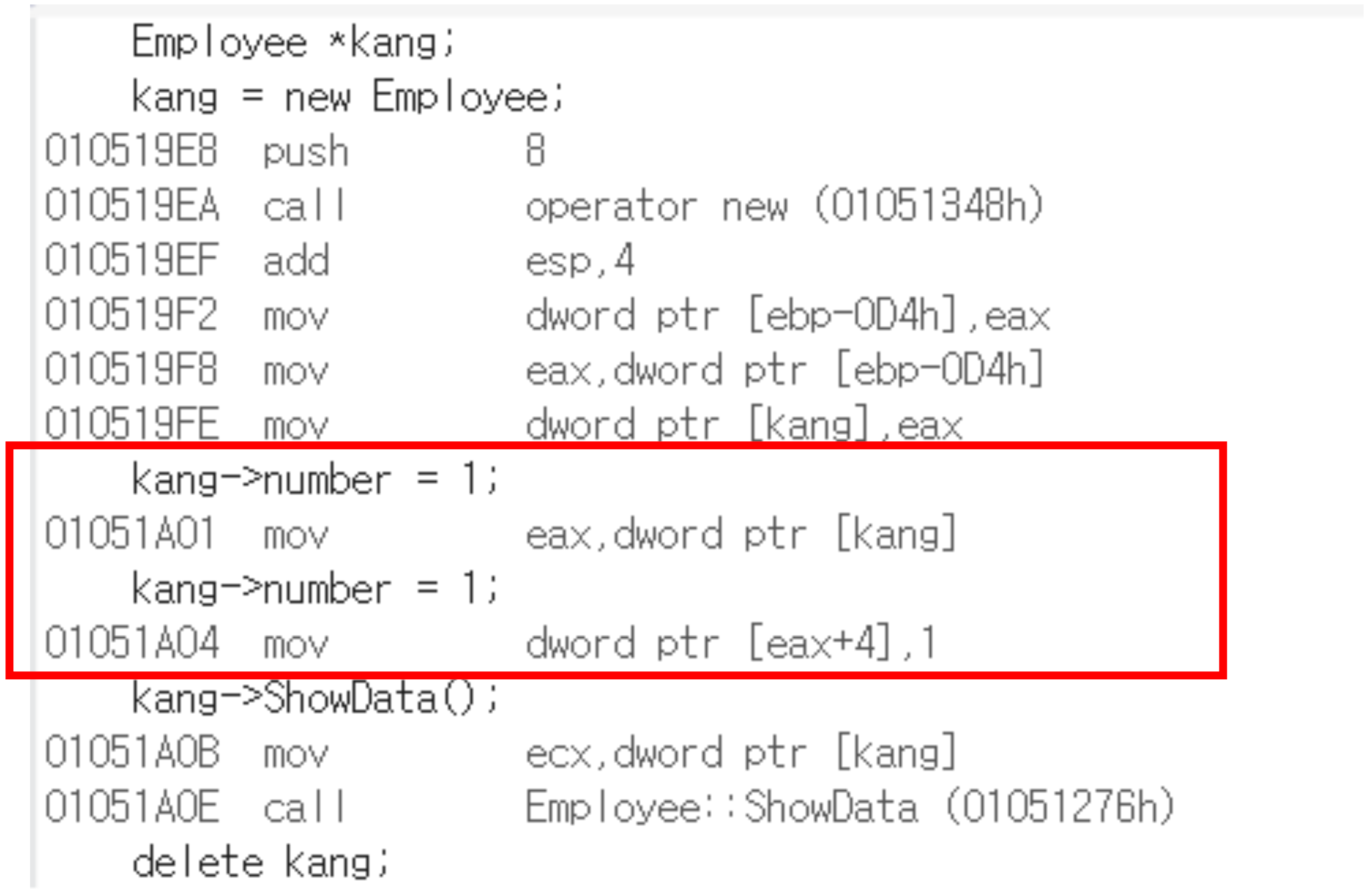


Body

## 객체의 동적할당과 해제



```
#include <stdio.h>

class Employee {
public:
	int Test;
	int number;
	void ShowData();
};
void Employee::ShowData()
{
	printf("%d",number);
	return;
}


int main(){
	Employee *kang;
	kang = new Employee;
	kang->number = 1;
	kang->ShowData();
	delete kang;
	return 0;
}
```



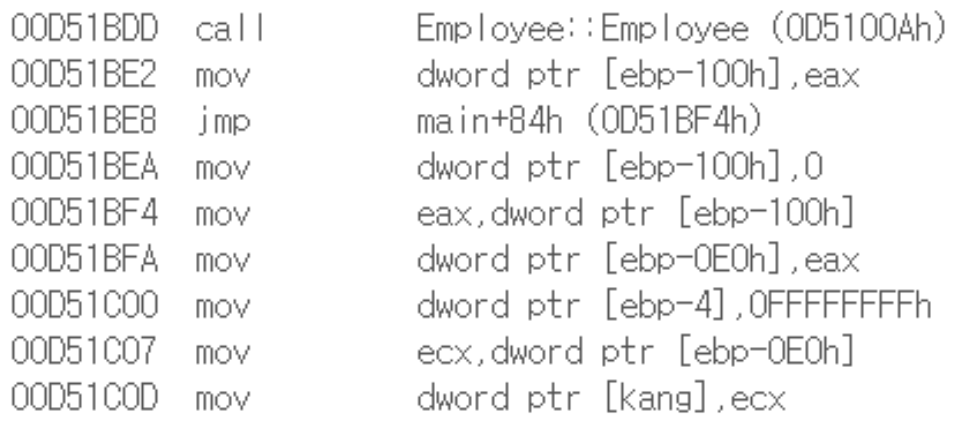
```
	Employee *kang;
	kang = new Employee;
010519E8  push        8
010519EA  call        operator new (01051348h)
010519EF  add         esp,4
010519F2  mov         dword ptr [ebp-0D4h],eax
010519F8  mov         eax,dword ptr [ebp-0D4h]
010519FE  mov         dword ptr [kang],eax
	kang->number = 1;
01051A01  mov         eax,dword ptr [kang]
	kang->number = 1;
01051A04  mov         dword ptr [eax+4],1
	kang->ShowData();
01051A0B  mov         ecx,dword ptr [kang]
01051A0E  call        Employee::ShowData (0105127Bh)
	delete kang;
```

스택이 아닌 객체가 가리키는 번지에 offset에 해당하는 위치에 값을 삽입!

Body

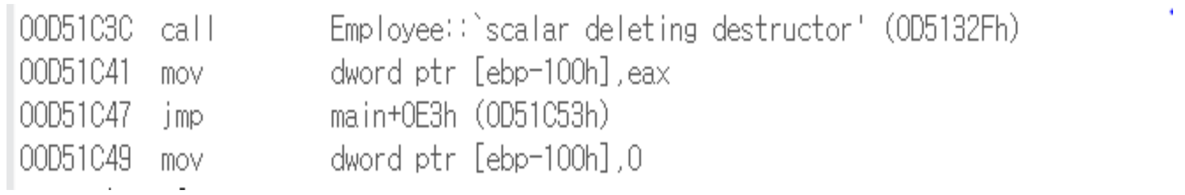
## 생성자와 소멸자

### 생성자 호출 코드



```
00D51BDD  call          Employee::Employee (0D5100Ah)
00D51BE2  mov           dword ptr [ebp-100h],eax
00D51BE8  jmp           main+84h (0D51BF4h)
00D51BEA  mov           dword ptr [ebp-100h],0
00D51BF4  mov           eax,dword ptr [ebp-100h]
00D51BFA  mov           dword ptr [ebp-0E0h],eax
00D51C00  mov           dword ptr [ebp-4],0FFFFFFFFh
00D51C07  mov           ecx,dword ptr [ebp-0E0h]
00D51C0D  mov           dword ptr [kang],ecx
```

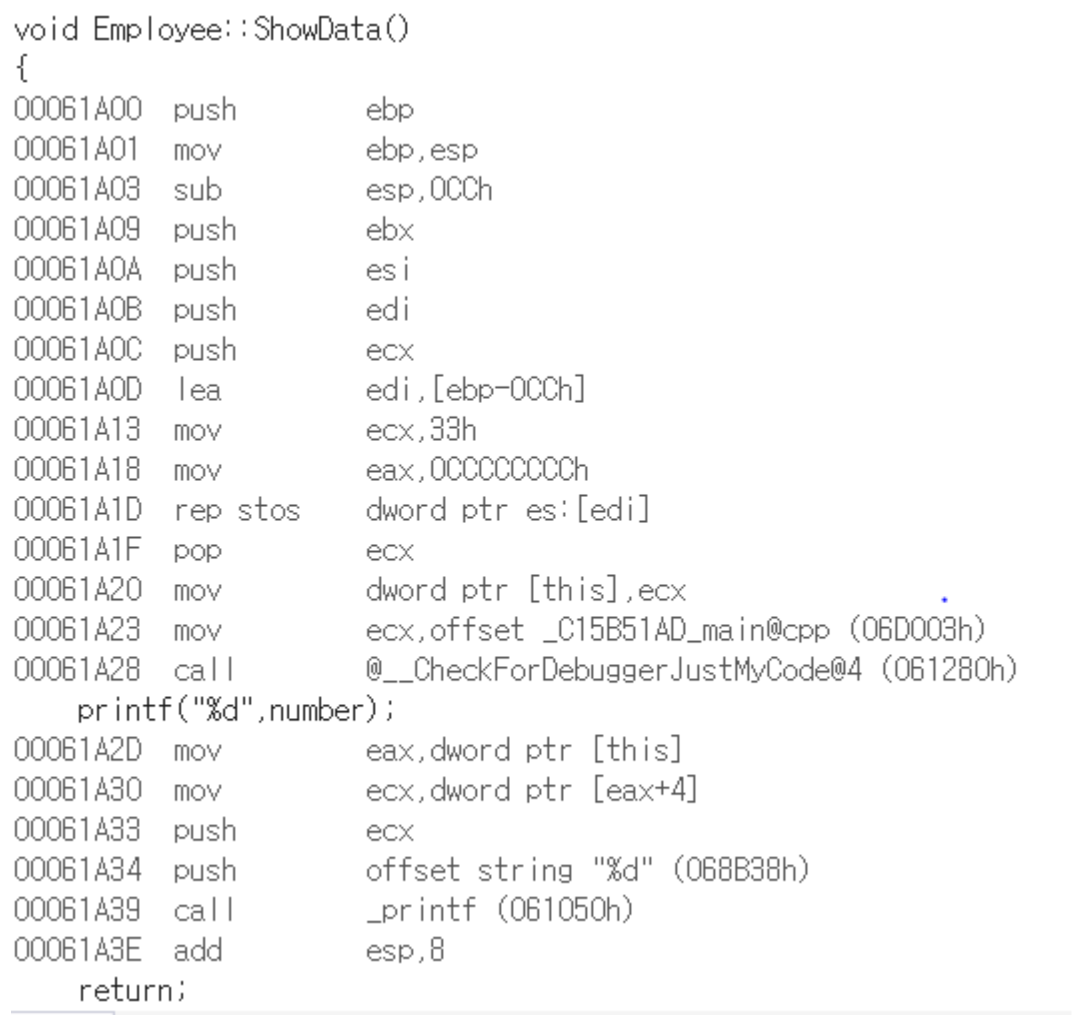
### 소멸자 호출 코드



```
00D51C3C  call          Employee::`scalar deleting destructor' (0D5132Fh)
00D51C41  mov           dword ptr [ebp-100h],eax
00D51C47  jmp           main+0E3h (0D51C53h)
00D51C49  mov           dword ptr [ebp-100h],0
```

객체 호출, 객체 삭제 이후 위와 같은 코드를 확인할 수 있음!

## 캡슐화



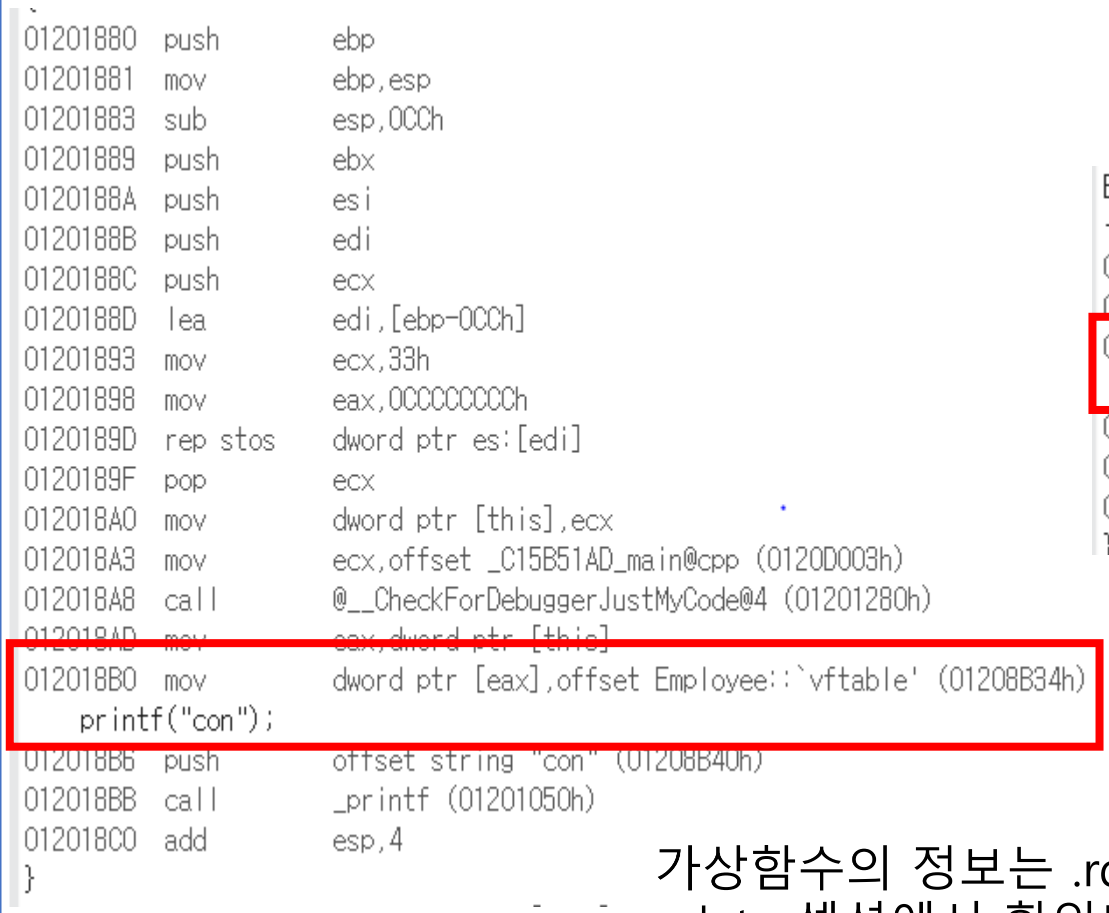
```
void Employee::ShowData()
{
00061A00  push        ebp
00061A01  mov         ebp,esp
00061A03  sub         esp,0CCh
00061A09  push        ebx
00061A0A  push        esi
00061A0B  push        edi
00061A0C  push        ecx
00061A0D  lea         edi,[ebp-0CCh]
00061A13  mov         ecx,33h
00061A18  mov         eax,0CCCCCCCCh
00061A1D  rep stos    dword ptr es:[edi]
00061A1F  pop         ecx
00061A20  mov         dword ptr [this],ecx
00061A23  mov         ecx,offset _C15B51AD_main@cpp (06D003h)
00061A28  call        @__CheckForDebuggerJustMyCode@4 (061280h)
    printf("%d",number);
00061A2D  mov         eax,dword ptr [this]
00061A30  mov         ecx,dword ptr [eax+4]
00061A33  push        ecx
00061A34  push        offset string "%d" (068B38h)
00061A39  call        _printf (061050h)
00061A3E  add         esp,8
    return;
```

바이너리 상에서는 캡슐화된 부분을 찾기에 한계가 있다!

**Body**

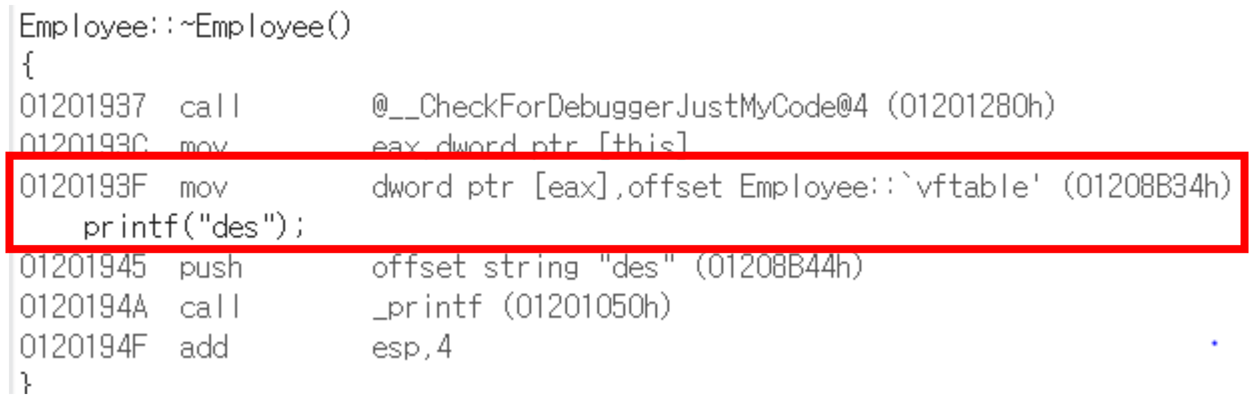
## 다형성 구조 파악

생성자 호출 코드



```
01201880  push      ebp
01201881  mov       ebp,esp
01201883  sub       esp,0CCh
01201889  push      ebx
0120188A  push      esi
0120188B  push      edi
0120188C  push      ecx
0120188D  lea       edi,[ebp-0CCh]
01201893  mov       ecx,33h
01201898  mov       eax,0CCCCCCCCh
0120189D  rep stos  dword ptr es:[edi]
0120189F  pop       ecx
012018A0  mov       dword ptr [this],ecx
012018A3  mov       ecx,offset _C15B51AD_main@cpp (0120D003h)
012018A8  call      @__CheckForDebuggerJustMyCode@4 (01201280h)
012018AD  mov       eax,dword ptr [this]
012018B0  mov       dword ptr [eax],offset Employee::`vftable' (01208B34h)
    printf("con");
012018B6  push      offset string "con" (01208B40h)
012018BB  call      _printf (01201050h)
012018C0  add       esp,4
}
```

소멸자 호출 코드



```
Employee::~Employee()
{
01201937  call      @__CheckForDebuggerJustMyCode@4 (01201280h)
0120193C  mov       eax,dword ptr [this]
0120193F  mov       dword ptr [eax],offset Employee::`vftable' (01208B34h)
    printf("des");
01201945  push      offset string "des" (01208B44h)
0120194A  call      _printf (01201050h)
0120194F  add       esp,4
}
```

가상함수의 정보는 .rdata 섹션에 저장되어 있음!
.rdata 섹션에서 확인되는 함수포인터는 가상함수라고 추측할 수 있음!

# Q & A