

6장 응용 프레임워크와 가상함수의 만남

변영철 교수

(ycb@jejunu.ac.kr)

1. 응용 프로그램들이 비슷한 이유

- 윈도우 특징
 - GUI (Graphic User Interface)
 - 멀티 태스킹
- 응용 프로그램들이 서로 모양이 비슷
 - 겉 모습
 - 내부 이벤트 처리
- 중복되는 코드가 많이 존재
 - 기본적으로 혹은 라이브러리로... (빠대)



1. 응용 프로그램들이 비슷한 이유

- MFC 응용 프로그램 작성의 예
 - 기본적으로 많은 코드가 작성되어 있다.
 - 파일(F) | 새로 만들기 | 프로젝트
 - MFC 앱 선택 후 만들기 | 마침
 - 컴파일 후 실행해보자.
- 여러분이 할 일은?
 - 기존에 작성되어 있는 코드를 **조금 바꾸거나**
 - 새로운 코드를 **추가하거나**
 - 뼈대에 살을 붙이는 것



```
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    cs.cx = 100;
    cs.cy = 200;
```

이미 많은 코드(응용 프레임워크)가
준비되어 있다.

여기에 **살**만 붙이자.

이제부터 우리가 공부할 내용은?

2. Console4 프로젝트 생성

- CBase 클래스 작성하기

```
#include <stdio.h>
```

```
class CBase
{
public:
    void WhoAreYou()
    {
        printf("CBase!\n");
    }
};
```

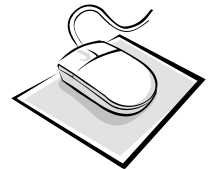
CBase gildong;

```
void main()
{
    gildong.WhoAreYou();
}
```



3. CBase를 항상 작성해야 한다면

- 반복해서 CBase 클래스를 작성하는 것이 귀찮다면 **미리 만들어 두고 사용**하라. (**라이브러리**)
 - CBase 클래스를 라이브러리로 생각하자.
 - 라이브러리는 여러 사람이 필요할 때마다 사용하는 것이므로 수정할 수 없다.
 - 라이브러리는 (일반적으로) *.dll, *.lib 형태로 저장되어 소스 코드가 보이지 않는다.
 - “수정할 수도 없고 보이지도 않는다.”



```
#include <stdio.h>
```

```
class CBase  
{  
public:  
    void WhoAreYou()  
    {  
        printf("CBase!\n");  
    }  
};
```

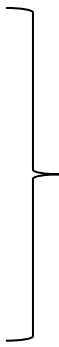
라이브러리 코드
(기존에 있었던 코드)



CBase gildong;

```
void main()  
{  
    gildong.WhoAreYou();  
}
```

라이브러리가 아닌 코드
(여러분이 작성하는 코드)



4. 마음에 들지 않으면?

- 이 프로그램을 실행하면??
- 화면에 표시되는 문자열이 마음에 들지 않으면?
- CBase는 라이브러리라고 가정하였으므로 코드가 마음에 들지 않아도 수정할 수 없다.
- OOP에서 마음에 들지 않은 코드를 직접 고치지 않으면서 고치는 세련된 방법 : 함수 재정의 (5장에서 공부한 내용)




```
#include <stdio.h>
```

```
class CBase {  
public:  
    void WhoAreYou()  
    {  
        printf("CBase!\\n");  
    }  
};
```

```
class CDerived : public CBase {  
public:  
    void WhoAreYou()  
    {  
        printf("-----\\n");  
        printf("Hello, I am a CBase!\\n");  
        printf("-----\\n");  
    }  
};
```

```
CDerived gildong;
```

```
void main()  
{  
    gildong.WhoAreYou();  
}
```

마음에 들지 않으니 재정의

마음에 들었다면? 재정의하
지 않았을 것!

5. 상상력 발휘하기

- main 함수가 없는 프로그램은 없다.
 - 따라서 main는 반드시 존재해야 함
 - 반드시(항상) 존재해야(작성해야) 한다면 main 함수도 라이브러리로 되어 있으면 좋을 듯

```
void main()  
{  
    gildong.WhoAreYou();  
}
```

```
#include <stdio.h>
```

```
class CBase  
{  
public:  
    void WhoAreYou()  
    {  
        printf("CBase!\n");  
    }  
};
```

라이브러리 코드
(기존에 있었던 코드)

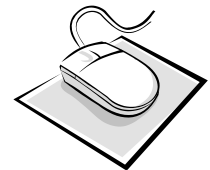
CBase gildong;

라이브러리가 아닌 코드
(여러분이 작성하는 코드)

```
void main()  
{  
    gildong.WhoAreYou();  
}
```

5. 상상력 발휘하기

- 라이브러리로 존재하는 코드 상상하기
 - CBase 클래스
 - main 함수



5. 상상력 발휘하기

- gildong이라는 변수 이름은 누가 정하나?
- 만일 변수를 cheolsu라고 하면?
- main 함수도 그에 따라 수정해야...
- 따라서 프로그래머가 정하는 변수이름이 달라지면 main 함수 코드도 달라져서 라이브러리화할 수 없음.
- 어떤 변수 이름을 쓰더라도 main 함수 코드가 바뀌지 않도록 하려면? → 변수 이름을 직접 언급하지 않으면 됨!

```
#include <stdio.h>
```

```
class CBase  
{  
public:  
    void WhoAreYou() {  
        printf("CBase!\n");  
    }  
};
```

```
CBase gildong;
```

```
void main() {  
    CBase* p;  
    p = &gildong;  
    p->WhoAreYou();  
}
```

```
#include <stdio.h>
```

```
class CBase  
{  
public:  
    void WhoAreYou() {  
        printf("CBase!\n");  
    }  
};
```

```
CBase gildong;
```

```
Cbase* AfxGetApp() {  
    return &gildong;  
}
```

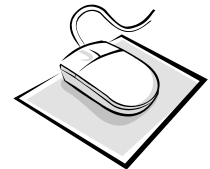
```
void main() {  
    CBase* p;  
    p = AfxGetApp();  
    p->WhoAreYou();  
}
```

5. 상상력 발휘하기

- 포인터 이용하기

Afx = Application Framework

- AfxGetApp 함수 : CBase 클래스로 만든 전역변수 (객체) 주소 값을 알아내는 함수



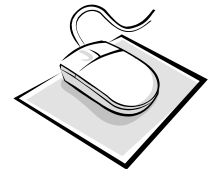
5. 상상력 발휘하기

- main 함수에서 포인터를 활용한 이유는?

변수 이름을 직접 사용하지 않으려고
main 함수를 라이브러리로 제공하려고...

5. 상상력 발휘하기

- 라이브러리로 존재하는 코드 상상하기
 - CBase 클래스
 - main 함수
 - AfxGetApp 함수



6. Is-a 관계

```
class Ellipse : public Circle
{
}

```

```
class Man : public Human
{
}

```

- 타원과 원 관계

- 타원은 원이다. Ellipse is a Circle.
- 남자는 사람이다. Man is a Human.
- 왼쪽 것은 오른쪽 것이다(클래스에서).
- 아래 것은 위의 것이다(UML에서).

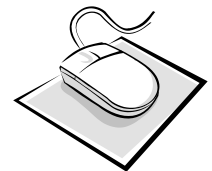
```
class Derived : public Base
{
}

```

- 특화(specialization)와 일반화(generalization)

```
void main()
{
    CBase* p;
    p = AfxGetApp();
    p->WhoAreYou();
}

```



7. 가상함수의 위력

- 라이브러리화 하였지만 실행되지 않는다.



- 여기에 **virtual 키워드**를 붙이는 순간!
- 아주 **놀라운 한 줄짜리 프로그램**

`CBase gildong;`

– 눈에 보이지 않는 코드는?

```

#include <stdio.h>
-
class CBase
{
public:
    void WhoAreYou() {
        printf("CBase!\n");
    }
};

```

CBase gildong;

```

Cbase* AfxGetApp() {
    return &gildong;
}

```

```

void main() {
    CBase* p;
    p = AfxGetApp();
    p->WhoAreYou();
}

```

구성(C): 활성(Debug)플랫폼(P): 활성(Win32)

구성 관리자(O)...

▲ 구성 속성

일반

고급

디버깅

VC++ 디렉터리

▷ C/C++

▷ 링커

▷ 매니페스트 도구

▷ XML 문서 생성기

▷ 찾아보기 정보

▷ 빌드 이벤트

▷ 사용자 지정 빌드 단계

▷ 코드 분석

공용 언어 런타임 지원

공용 언어 런타임 지원 안 함

.NET 대상 프레임워크 버전

관리되는 증분 빌드 사용

아니요

▼ 고급 속성

대상 파일 확장명

.exe

정리할 때 삭제할 확장명

.cdf;.cache;*.obj;*.obj.enc;*.ilk;*.ipdb;*.iobj;*.resources;*.t

빌드 로그 파일

\$(IntDir)\$(MSBuildProjectName).log

기본 설정 빌드 도구 아키텍처

기본값

디버깅 라이브러리 사용

예

Unity(JUMBO) 빌드 사용

아니요

OutDir에 콘텐츠 복사

아니요

OutDir에 프로젝트 참조 복사

아니요

OutDir에 프로젝트 참조의 기호 복사

아니요

OutDir에 Cpp 런타임 복사

아니요

MFC 사용

공유 DLL에서 MFC 사용

문자 집합

표준 Windows 라이브러리 사용

전체 프로그램 최적화

정적 라이브러리에서 MFC 사용

MSVC 도구 세트 버전

공유 DLL에서 MFC 사용

MFC 사용

구성에서 MFC를 사용하는 방법을 지정합니다.

확인

취소

적용(A)

구성(C): 활성(Debug)플랫폼(P): 활성(Win32)

구성 관리자(O)...

▲ 구성 속성

일반

고급

디버깅

VC++ 디렉터리

▶ C/C++

▲ 링커

일반

입력

매니페스트 파일

디버깅

시스템

최적화

포함 IDL

Windows 메타데이터

고급

모든 옵션

명령줄

▶ 매니페스트 도구

▶ XML 문서 생성기

▶ 찾아보기 정보

▶ 빌드 이벤트

하위 시스템

창(/SUBSYSTEM:WINDOWS)

필요한 최소 버전

설정 안 함

힙 예약 크기

콘솔(/SUBSYSTEM:CONSOLE)

힙 커밋 크기

창(/SUBSYSTEM:WINDOWS)

스택 예약 크기

네이티브(/SUBSYSTEM:NATIVE)

스택 커밋 크기

EFI 애플리케이션(/SUBSYSTEM:EFI_APPLICATION)

큰 주소 처리

EFI 부트 서비스 드라이버(/SUBSYSTEM:EFI_BOOT_SERVICE_DRIVER)

터미널 서버

EFI ROM(/SUBSYSTEM:EFI_ROM)

CD에서 스왑 실행

EFI 런타임(/SUBSYSTEM:EFI_RUNTIME_DRIVER)

네트워크에서 스왑 실행

POSIX(/SUBSYSTEM:POSIX)

드라이버

<부모 또는 프로젝트 기본값에서 상속>

하위 시스템

/SUBSYSTEM 옵션을 사용하면 운영 체제에서 .exe 파일을 실행하는 방법을 지시할 수 있습니다. 선택한 하위 시스템에 따라 링커가 선택하는 진입점 기호(또는 진입점 함수)가 결정됩니다.

확인

취소

적용(A)

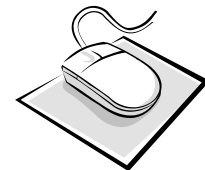
7. 가상함수의 위력

- 한 줄짜리 놀라운 MCF 프로그램 코드

```
#include <afxwin.h>  
CWinApp gildong;
```

– 눈에 보이지 않는 코드는?

- 코드가 마음에 들지 않으면? 무엇인가를 표시하려면?



```
#include <afxwin.h>
```

CBase

```
class CDerived : public CWinApp
```

```
{
```

```
public: WhoAreYou
```

```
    BOOL InitInstance() {
```

```
        AfxMessageBox(_T("Hello, World!"));
```

```
        return TRUE;
```

```
    }
```

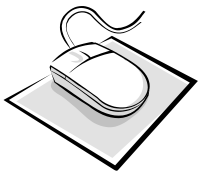
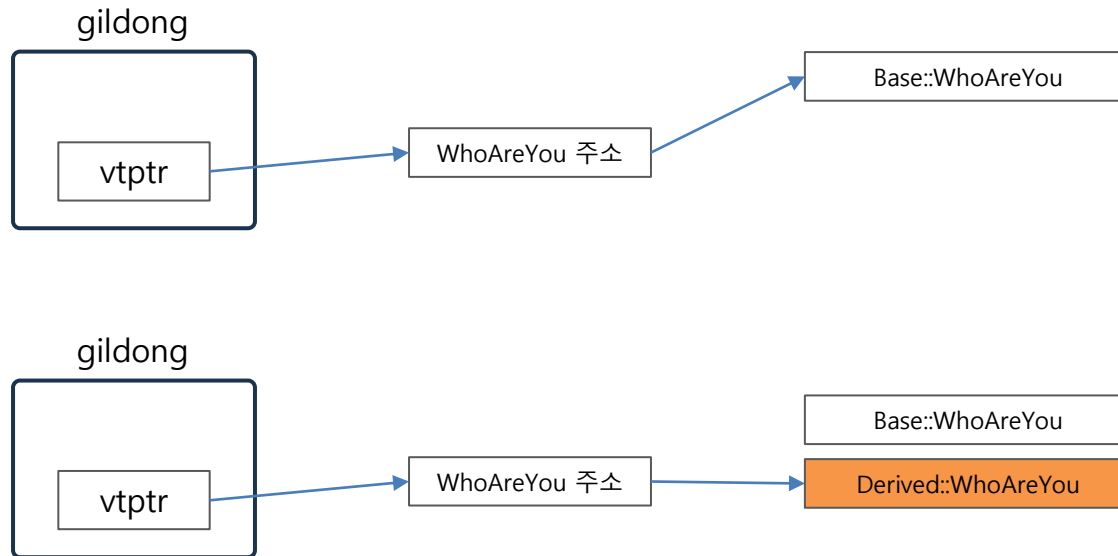
```
};
```

```
CDerived gildong;
```



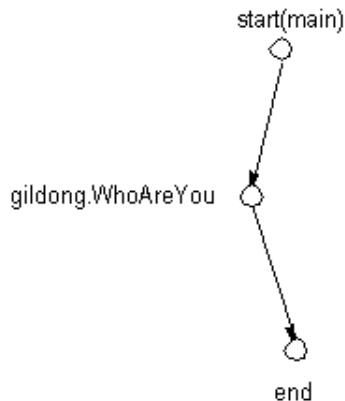
8. 가상함수 구현방법

- 가상함수 테이블



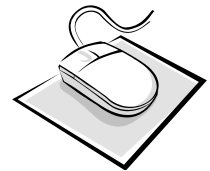
9. 가상함수와 응용 프레임워크

- 응용 프레임워크는
 - 한 줄짜리 프로그램에 숨겨져 있는(!) 거대한 코드 덩어리
 - 가상함수 재정의로 응용 프레임워크 실행 흐름을 원하는 대로 바꿀 수 있다.



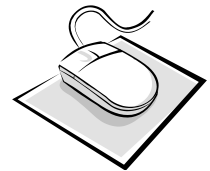
10. 코드 분할 및 정리

- Lib.h
 - CBase 클래스
- Lib.cpp
 - main
 - AfxGetApp
- Console5.cpp
 - 한 줄짜리 코드 (객체 정의)



11. MFC 라이브러리 이용하기

- MFC 라이브러리에 이미 있는 코드들
 - CWinApp
 - main
 - AfxGetApp
- 프로젝트 속성 설정하기



12. Visual C++ 프로그래밍

- MFC 라이브러리 + 코드 생성 마법사

12. 객체를 보라

- 클래스를 보지 말고 객체를 보라.
 - 클래스 지향이 아닌 객체 지향 프로그래밍