9장 템플릿과 가상 함수

변영철 교수
(ycb@jejunu.ac.kr)

제1절 함수 템플릿

- 둘 중 큰 값(정수, 실수)을 반환하는 함수, Max

```
#include <stdio.h>

void main()
{
        int a = Max (2, 3);
        double b = Max (2.1, 3.4);

        printf("%d, %f₩n", a, b);
}
```

제1절 함수 템플릿

```
int Max (int a, int b)
{
        if(a > b)
                return a;
        else
                return b;
}
```

```
double Max (double a, double b)
{
        if(a > b)
                return a;
        else
                return b;
}
```

제1절 함수 템플릿

- 두 함수 코드가 거의 비슷
- 왠지 여러 번 중복해서 작성하는 느낌
- 템플릿 = 어떤 것(함수)을 만들어 내는(찍어내는) 틀
- 자료 형을 지정하면 함수를 만들어 내는 틀 → 함수 템플릿

제1절 함수 템플릿

```
template <typename T>
T Max(T a, T b)
{
        if(a > b)
                return a;
        else
                return b;
}
```

```
#include <stdio.h>

template <typename T>
T Max(T a, T b)
{
        if(a > b)
                return a;
        else
                return b;
}

void main()
{
    printf("%d₩n", Max(2, 3));
    printf("%f₩n", Max(8.1, 8.8));

    getchar();
}
```

제2절 클래스 템플릿

- 앞의 함수를 클래스 My 로 묶어 추상화하자.
- 클래스를 찍어내는 틀
- 두 값(정수, 실수 등) 중 큰 값을 반환하는 클래스 My

```
#include <stdio.h>

template <typename T>
class My {
public:
T Max(T a, T b)
{
   if(a > b)
  return a;
   else
  return b;
}
};

void main()
{
   My<int> gildong;
   printf("%d₩n", gildong.Max(2, 3));
   My<double> cheolsu;
   printf("%f₩n", cheolsu.Max(8.1, 8.8));
   getchar();
}
```

제2절 클래스 템플릿

- 두 수 더하는 클래스 CAdder
- typename을 class로 해도 됨

```
#include <stdio.h>

template<class T1, class T2>
class CAdder
{
public:
   T2 Add(T1 a, T1 b)
   {
      T2 sum = a + b;
      return sum;
   }
};

void main()
{
   CAdder<int, long> a;

   printf("%d₩n", a.Add(2, 3));
}
```

아래 코드를 보면 어떤 생각?

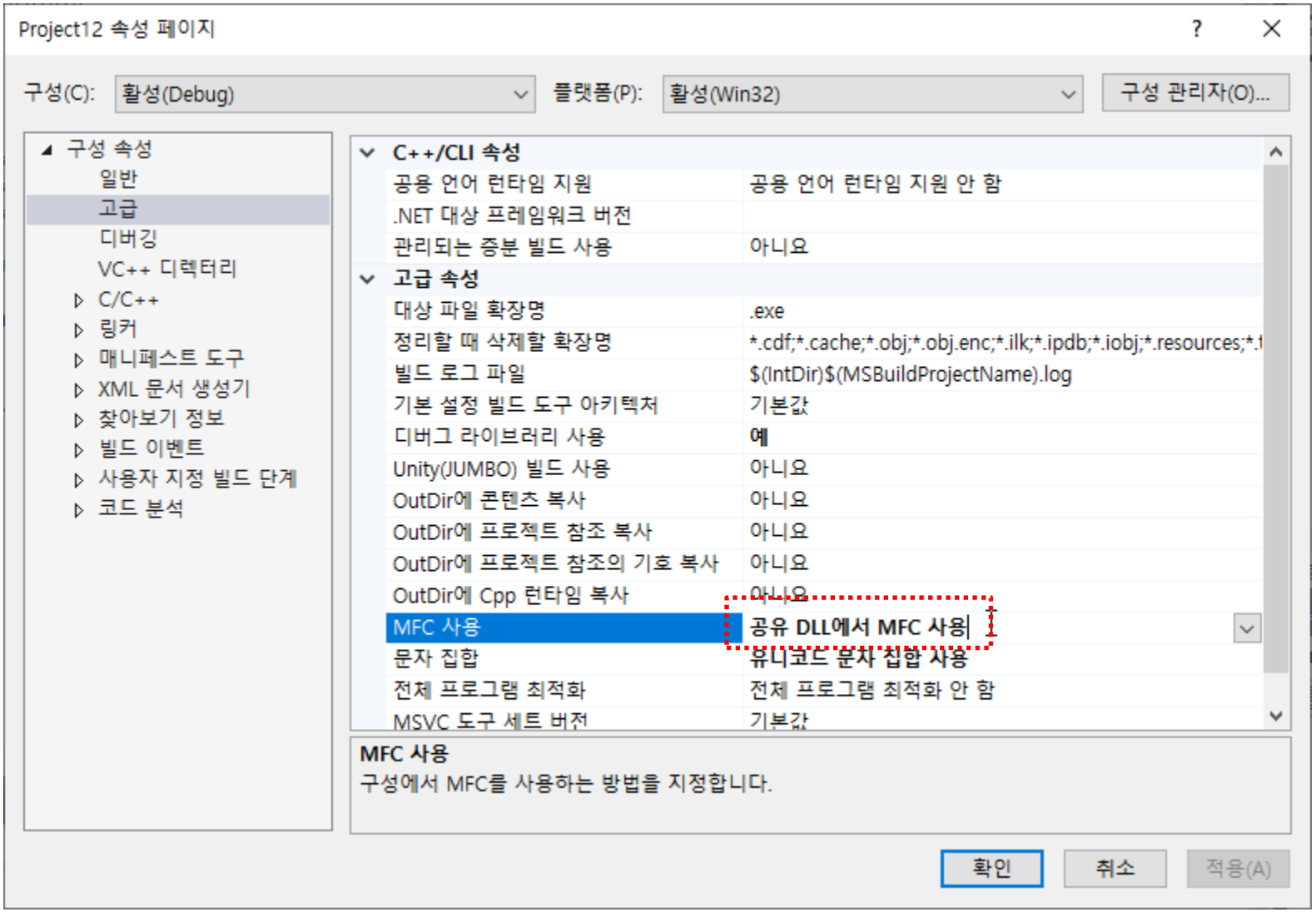
```
CAdder<int, long> a;

CList<int, int> a;
```

제2절 클래스 템플릿

- MFC 라이브러리에 있는 클래스 탬플릿 사용하기
  - 프로젝트 이름에서 오른쪽 마우스 버튼을 클릭, 속성 메뉴 항목을 선택, ' 고급 | MFC 사용'에서 '공유 DLL에서 MFC 사용' 선택

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

void main()
{
    CList<int, int> a;

    a.AddTail(20);
    a.AddTail(30);
    a.AddTail(40);

    POSITION pos = a.GetHeadPosition();

    for (int i=0;i < a.GetCount();i++) {
        printf("%d ₩n", a.GetNext(pos));
    }
    getchar();
}
```

제2절 클래스 템플릿


Project12 속성 페이지
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
C++/CLI 속성
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
디버그 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
유니코드 문자 집합 사용
전체 프로그램 최적화
전체 프로그램 최적화 안 함
MSVC 도구 세트 버전
기본값
MFC 사용
구성에서 MFC를 사용하는 방법을 지정합니다.
확인
취소
적용(A)

제3절 상속과 접근 지정자

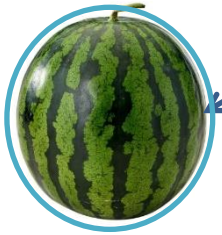
```
class Base {
public:
        int a;
protected:
        int b;
private:
        int c;
}
```

```
class Derived : protected Base
public:
```

```
protected:
```

```
private:
```

```
}
```



제4절 동적할당과 가상 소멸자

- 소멸자 → 객체가 소멸될 때 자동으로 실행되는 멤버 함수

```
CBase
~CBase
Press any key to continue
```

```
#include <stdio.h>

class CBase
{
public:
    CBase() {
        printf("CBase ₩n");
    }
    ~CBase() {
        printf("~CBase ₩n");
    }
};

void main()
{
    CBase gildong;
}
```

실행하면 어떤 결과?

→문제없음!!

제4절 동적할당과 가상 소멸자

```
#include <stdio.h>

class CBase
{
public:
    CBase() {
        printf("CBase ₩n");
    }
    ~CBase() {
        printf("~CBase ₩n");
    }
};
```

```
class CDerived : public CBase
{
public:
    CDerived() {
        printf("CDerived ₩n");
    }

    ~CDerived() {
        printf("~CDerived ₩n");
    }
};

void main()
{
    CDerived gildong;
}
```

실행하면 어떤 결과?

→문제없음!!

제4절 동적할당과 가상 소멸자

- new 연산자로 객체를 동적 할당하여 삭제할 경우 기반 클래스의 소멸자는 실행이 되지만 파생클래스의 소멸자는 실행되지 않는 문제 발생
- 가상소멸자 : 파생 클래스의 소멸자 함수도 실행되도록 하는 장치

```
void main()
{
    CBase* p = new CDerived;
    delete p; //동적 할당된 객체를 제거
}
```

```
CBase
CDerived
~CBase
Press any key to continue
```

→문제있음!!

제4절 동적할당과 가상 소멸자

- 다음과 같이 virtual로 선언하면 문제가 해결됨
- 기반 클래스의 소멸자를 가상 소멸자로 만들면 파생 클래스의 소멸자가 실행되도록 코드를 작성함

```
class CBase
{
public:
    CBase() {
        printf("CBase ₩n");
    }

    virtual ~CBase() {
        printf("~CBase ₩n");
    }
};
```

문제해결!!

제5절 순수 가상 함수와 추상 클래스

- 도형 클래스 CDrawing을 작성해보자. 면적을 구하는 함수 area도 작성해 보자.
- 멤버 함수 area를 코딩할 수 있을까?
- 함수 본체를 구현할 수 없는 함수→ 순수 가상 함수
- 순수 가상 함수를 갖는 클래스: 추상 클래스
- 실행되는 몸체가 없으므로 객체를 정의할 수 없음.

```
class CDrawing
{
public:
    double Area()
    {
        //면적을 구하는 공식
        //작성불가!!!
    }
};
```

추상클래스

```
class CDrawing
{
public:
    virtual double Area() = 0;
};
```

순수가상함수

제5절 순수 가상 함수와 추상 클래스

- 객체를 정의하려면 추상 클래스를 상속 받아 새로운 클래스를 작성해야 함 (가령 CCircle)
- 순수 가상 함수(area)는 재정의(overriding)해야 함. 그렇지 않으면 에러가 발생!
- 순수 가상 함수를 사용하는 이유
  - 상속 받는 클래스에서 반드시 재정의하도록 강제
  - 순수 가상 함수를 기술함으로써 클래스를 보다 잘 추상화

```
class CDrawing
{
public:
   virtual double Area() = 0;
};

class CCircle : public CDrawing
{
protected:
   int iRadius;

public:
   double Area()
   {
      return 3.14 * iRadius * iRadius;
   }

   void Set(int r)
   {
      iRadius = r;
   }
};

void main()
{
   CCircle a;
   a.Set(10);
   printf("%f ₩n", a.Area());
}
```