

# 제3장 추상화에서 객체지향 프로그래밍까지

변영철 교수  
([ycb@jejunu.ac.kr](mailto:ycb@jejunu.ac.kr))

처음 다운로드 받는다면,  
git clone <https://github.com/yungbyun/cpplecturenote.git>

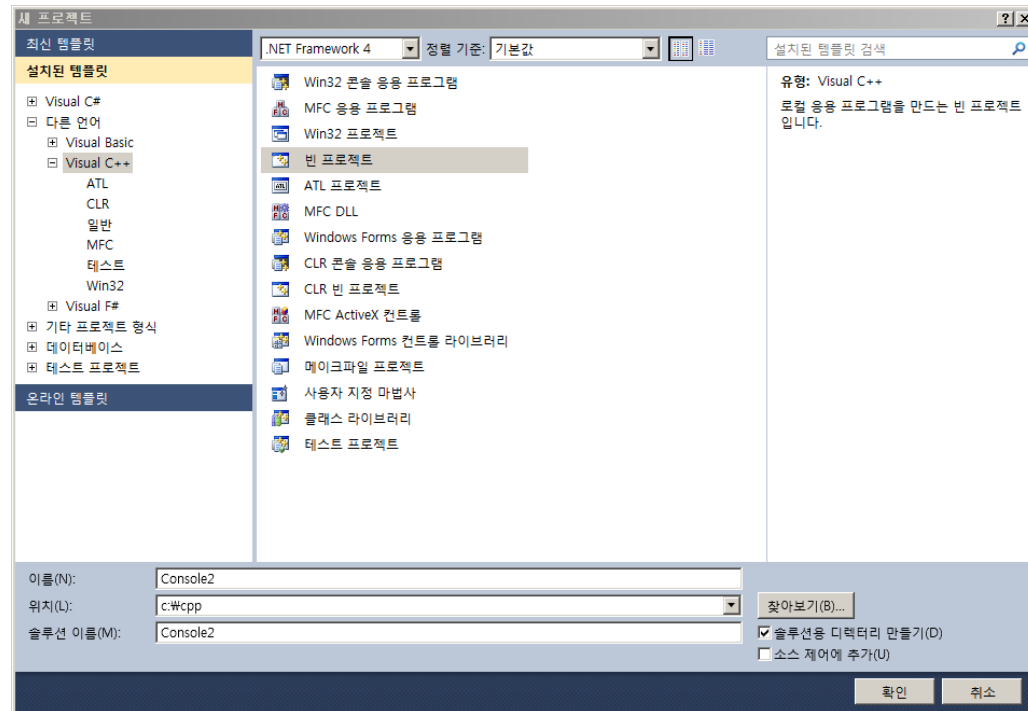
전에 받은 적이 있다면, 받은 폴더로 이동하여  
git pull

# 1. 추상화는 뭐고 데이터 추상화는 뭘까

- 추상화
  - '묶는 것' 혹은 '간략히 줄여서, 묶어서 표현하는 것'
  - 어제 한 일에 대해 말할 경우 '어제 친구랑 영화를 보았다' 등과 같이 간략히 묶어서 표현하는 것
- 코드 추상화
  - 코드를 간단히 줄여서(묶어서) 표현하는 것 → 함수
- ???
  - 코드뿐만 아니라 변수(데이터)까지 간단히 줄여서(묶어서) 표현하는 것
  - 그 결과 무엇이 만들어질까?

## 2. CONSOLE2 프로젝트 생성하기

- 파일(F) > 새로 만들기(N) > 프로젝트(P)... 메뉴 항목을 선택



- 코드 작성하기

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

```
int iX;
```

```
int iY;
```

```
void Assign(int x, int y)
```

```
{
```

```
    iX = x;
```

```
    iY = y;
```

```
}
```

```
int Add()
```

```
{
```

```
    return iX + iY;
```

```
}
```

```
void main()
```

```
{
```

```
    int iResult;
```

```
    Assign(2, 3);
```

```
    iResult = Add();
```

```
    printf("두 개의 값을 더한 결과: %d\n", iResult);
```

```
}
```

# 3. 창자 이야기

- 앞서 작성한 프로그램은...
  - 창자가 몸 밖으로 나와 있는 사람
  - 혹은 램이나 CPU 등이 밖으로 나와 있는 케이스 없는 컴퓨터
  - 케이스 없는 자판기
- 왜?
  - 아무나, 아무 데서나 막 액세스(접근)할 수 있어서...
- 어떤 문제가 있을까?
  - 쉽게 만질 수 있어서
  - 쉽게 다치거나 망가짐
  - 내용물을 분실할 수 있음

# 3. 창자 이야기

- 변수 iX는
  - 전역 변수이므로 아무 함수에서나 제한 없이 쉽게 접근할 수 있어서 잘못 접근할 수 있음
  - 가령 iX가 나이를 저장하는 변수일지라도 음수를 넣을 수 있음
  - 음수를 넣지 못하도록 하는 방법이 존재하지 않음
- 함수 Assign()은
  - ???

## 4. 변수와 함수를 묶으면 클래스가 보인다

- 문제 해결 방법
  - 창자는 몸 안으로 밀어 넣고 수술해서 봉해야 함
  - 램과 CPU 등은 단단한 하드 케이스로 싸서 조립함
  - 앞의 프로그램에서 변수 iX, iY도 마찬가지임. 단단한 케이스로 싸서 접근하지 못하도록 함
  - 즉, 관련 있는 것들을 묶어서 포장
- 관련이 있다는 것을 어떻게 알 것인가?



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

무는다 XXX  
시작

```
int iX;  
int iY;
```

```
void Assign(int x, int y)  
{  
    iX = x;  
    iY = y;  
}
```

```
int Add()  
{  
    return iX + iY;  
}  
끝
```

## 4. 변수와 함수를 묶으면 클래스가 보인다

- 데이터 추상화
  - 데이터(변수)와 코드(함수)를 묶는 것
  - 추상화하여 만든 XXX -> 추상화하여 만든 **자료형**  
Abstract Data Type, ADT)
- 우리가 알고 있는 표준 자료형(Standard Data Type)은?
  - int
  - double
  - char 등
- 자료형은 뭐하라고 있는 것?
  - 변수 만들라고 있는 것

## 4. 변수와 함수를 묶으면 클래스가 보인다

- 변수를 만든 예
  - `int a;`
  - `XXX a;`
- 여러 변수를 만든 예
  - `int a, b, c;`
  - `XXX a, b, c;`
  - `a, b, c`는 모두 `XXX` 자료형 변수들
  - `a, b, c`는 모두 `XXX` 자료형에 속하는 변수들
  - `a, b, c`는 모두 **XXX 라는 부류**(족속)에 속하는 변수
  - 따라서 `XXX`는 부류(class, 클래스)
  - 따라서 추상 자료형을 클래스라고 함

```
class XXX
{
    int iX;
    int iY;

    void Assign(int x, int y)
    {
        iX = x;
        iY = y;
    }

    int Add()
    {
        return iX + iY;
    }
}; //끝에 ;을 입력해야 문법에 맞음
```

```
class CPoint
{
    int iX;
    int iY;

    void Assign(int x, int y)
    {
        iX = x;
        iY = y;
    }


    int Add()
    {
        return iX + iY;
    }
};
```

## 4. 변수와 함수를 묶으면 클래스가 보인다

- 전역변수 vs. 멤버 변수
  - iX와 iY는 전역 변수였지만 이제는 CPoint에 속하는 멤버 변수가 됨
  - Assign, Add는 전역 함수였지만 이제는 멤버 함수
  - 멤버는 멤버를 액세스 할 수 있다.
- 길동이, 철수, 영자
  - `int a, b, c;`
  - `CPoint gildong, cheolSu, youngJa;`
- 멤버의 의미
  - 키, 몸무게, 성별 등과 같은 속성(attribute)
  - 운전하다, 걷다 등과 같은 할 수 있는 일(function)

## 4. 변수와 함수를 묶으면 클래스가 보인다

- 변수들은 서로 독립된 존재들
  - gildong, cheolSu, youngJa는 서로 독립된 존재들
  - 삼 쌍둥이와 같이 서로 몸이 붙어있는(공유하는) 존재가 아님
- 변수는 변수이나 속성과 할 줄 아는 일이 있는 변수를 객체(object)라고 함
  - 객체에게 일을 시킬 수 있음
- 따라서 클래스는 무엇 하라고 있는 것?
  - 객체 만들라고 있는 것
  - 만들어서 일 시키라고 있는 것

A close-up of a car's side-view mirror. The mirror reflects a road scene with several cars in traffic. A vertical dashed line is drawn on the mirror's surface. Korean text is overlaid on the bottom half of the mirror.

사물이 거울에 보이는 것보다  
가까이 있을.



```

#include <stdio.h>

class CPoint
{
    int iX;
    int iY;

    void Assign(int x, int y)
    {
        iX = x;
        iY = y;
    }

    int Add()
    {
        return iX + iY;
    }
};

```

CPoint gildong; //(A)

```

void main()
{
    int iResult;

    Assign(2, 3);

    iResult = Add();

    printf("두 개의 값을 더한 결과: %d\n", iResult);
}

```

허공에 소리치기, 말하기  
대상이 없어 혼자 헛소리하기

```

#include <stdio.h>

class CPoint
{
    int iX;
    int iY;

    void Assign(int x, int y)
    {
        iX = x;
        iY = y;
    }

    int Add()
    {
        return iX + iY;
    }
};

```

`CPoint gildong; //(A)`

```

void main()
{
    int iResult;

    gildong.Assign(2, 3);

    iResult = gildong.Add();

    printf("두 개의 값을 더한 결과: %d\n", iResult);
}

```

길동이에게 시키기  
길동아 Add해달라, Assign해달라.

## 4. 변수와 함수를 묶으면 클래스가 보인다

- 객체지향 프로그래밍
  - Object-Oriented Programming
  - 객체야(gildong) 무엇(Add, Assign)을 해다오~
  - 클래스 지향 프로그래밍이 아님
- 객체 지향(위주)
  - 클래스를 보지 말고 객체를 보라.
  - 객체를 만들지 않으면 클래스는 아무런 쓸모가 없음
  - 객체가 언제 만들어지는지 항상 생각

## 5. 관련된 변수와 함수를 묶는 이유

- 묶는다 = 포장한다
  - 묶는 행위를 잘 표현한 말 = 캡슐화(encapsulation)
  - 잘 포장된 것의 예: 자판기
- 묶어서 포장했을 때의 장점
  - 자판기 내부(돈 보관함, 종이컵, 커피 등)를 함부로 접근할 수 없음
  - 정보 은폐(information hiding)가 되기 때문
- 정보 은폐에서 정보가 뜻하는 것
  - 멤버 변수, 멤버 함수
  - 따라서 멤버 변수, 멤버 함수 은폐를 의미

## 6. 동전구멍과 버튼도 없는 쓸모 없는 커피자판기

- 시킨다고 다 하지는 않는다.
  - 길동아 **코딱지를 떼어내 보라.**
  - 멤버 함수는 있지만 호출할 수 없다면 정보 은폐된 멤버
  - 호출할 수 있는 멤버 함수가 하나도 없다면?
  - 이는 마치 동전 구멍과 버튼이 없는 커피 자판기와 같음. 커피를 뽑을 도리가 없어서 아무 짝에도 쓸모 없는 자판기
- Add와 Assign 멤버 함수를 호출할 수 있도록
  - public 키워드
  - 인터페이스(interface)
  - 메시지 전달(message passing)