제2장
추상화에서 OOP까지

변 영 철

1. 추상화와 데이터 추상화

- 추상화
  - 묶어서 간단히 표현하는 것
- 코드 추상화
  - 코드를 묶어서 간단히 표현하는 것

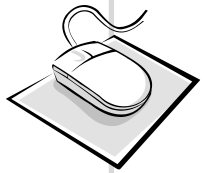
  → 그 결과 함수가 만들어짐.
- 데이터 추상화
  - 코드뿐만 아니라 데이터까지 묶어서 간단히 표현하는 것

  → 그 결과 ???가 만들어짐.

2. Console2 프로젝트

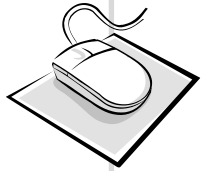
이 코드를 보면
무슨 생각이 들어야 할까?

```
#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);
}
```



2. Console2 프로젝트

```
#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);
}
```



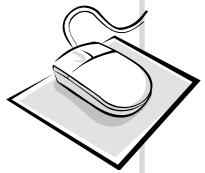
3. 창자 이야기

- 위험한 경우
  - 사고로 창자가 몸 밖으로 나오면 어떻게 해야 할까?
  - 램이나 CPU가 밖으로 나와있는 컴퓨터는?
  - 동전 통이 보이는 커피 자판기는?
- 해결 방법
  - 쉽게 접근하지 못하도록 조치를 취함.

4. 관련된 변수와 함수를 묶자

- 서로 관련된 변수와 함수 찾아서 묶기
  - 변수를 중심으로 관련된 함수 찾아서 묶자.

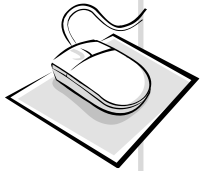
```
#include <stdio.h>

묶는다 XXX
시작
int iX;
int iY;

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

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



4. 관련된 변수와 함수를 묶자

- 묶어서 간단히 표현(추상화)한 이 **XXX를 무엇이라고 하나?**
- **자료형**
  - 추상화하여 만든 자료형, 추상 자료형
  - Abstract Data Type (ADT)
- 우리가 알고 있는 자료형
  - 기존 표준 자료형
  - Standard Data Type (SDT)

```
#include <stdio.h>

묶는다 xxx
{
int iX;
int iY;

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

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

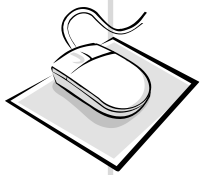


4. 관련된 변수와 함수를 묶자

- 자료형의 종류
  - 표준 자료형 int
  - 추상 자료형 XXX
- 자료형이 존재하는 이유는?
  - 변수를 만들기 위하여
  - int a, b, c;
  - XXX a, b, c;
- ‘묶는다’ 대신 class 키워드를 사용한 이유

4. 관련된 변수와 함수를 묶자

```
#include <stdio.h>

class XXX
{
int iX;
int iY;

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

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



4. 관련된 변수와 함수를 묶자

- 변수 이름 의인화 하기
  - XXX gildong, cheolsu, youngJa;
- XXX형 변수(gildong)의 멤버 의미
  - 멤버 변수 : 특성 (키, 몸무게, 나이, 전화번호, 주민번호 등)
  - 멤버 함수 : 할 줄 아는 일
- 무언가를 할 줄 아는 변수 gildong
  - 객체(object)
- 따라서 클래스가 있는 이유는
  - 객체를 만들려고…

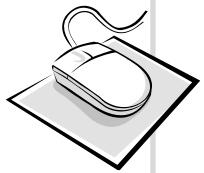
4. 관련된 변수와 함수를 묶자

- 습관적으로 생각하기
  - 클래스는 객체 만들려고 있는 것이다.
  - 객체들은 서로 독립적이다.
  - 객체가 어디에서 만들어지는지 생각하자.
  - 클래스를 보기보다는 객체를 보라(객체지향).

4. 관련된 변수와 함수를 묶자

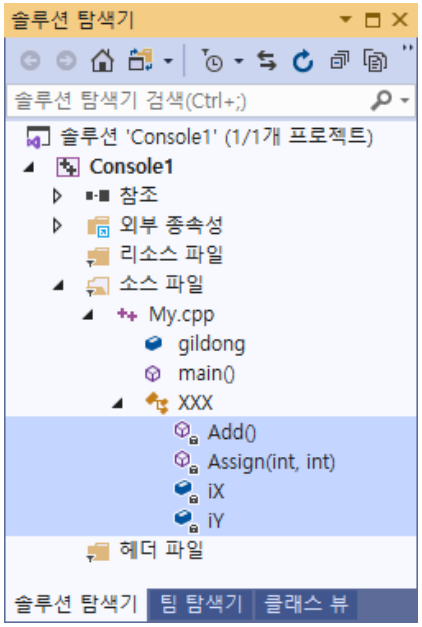
- 객체에게 일 시키기
  - '.' 연산자 이용

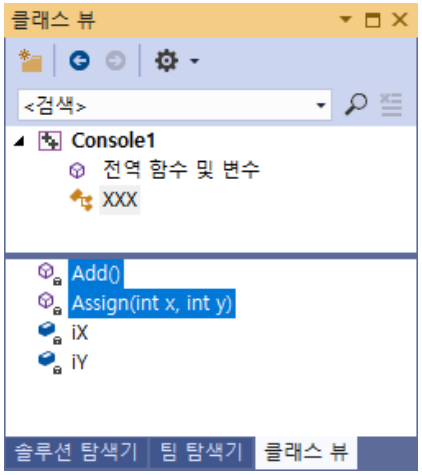
```
#include <stdio.h>

class XXX {
   int iX;
   int iY;

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

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

XXX gildong;

void main()
{
   int iResult;

   gildong.Assign(2, 3);   //gildong아 Assign해줘!
   iResult = gildong.Add(); //gildong아 Add해줘!

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



5. 관련된 변수와 함수를 묶는 이유

- 묶는 이유
  - 커피 자판기
  - 동전통과 커피를 외부로부터 감춰서 보호하기 위함.
- 정보은폐(information hiding)
  - 정보 : 멤버 변수 및 함수 등
- 캡슐화(encapsulation)
  - 묶는 행위를 잘 표현한 말





6. 쓸모 없는 자판기

- 신입사원의 예
  - 일을 시켰을 때 모두 거부하면?
  - 회사에는 전혀 도움이 안됨(쓸모 없음).
- 자판기의 예
  - 동전 구멍과 커피 배출구가 없으면
  - 전혀 쓸모가 없음.
- 쓸모 없는 이유
  - 모든 정보(멤버)가 은폐되어 있어서
  - 사용(접근)할 수 있는 방법이 없음.

6. 쓸모 없는 자판기

- public 키워드
  - 정보 공개
- 메시지 전달 (message passing)
  - public 멤버를 호출하는 것

```
#include <stdio.h>

class XXX {
    int iX;
    int iY;

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

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