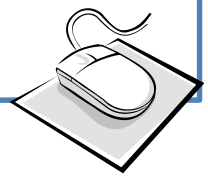
제1장
C# 프로그래밍을 위한
OOP 기본

제주대학교 컴퓨터공학과
변영철 교수

1. 아주 간단한 C 프로그램

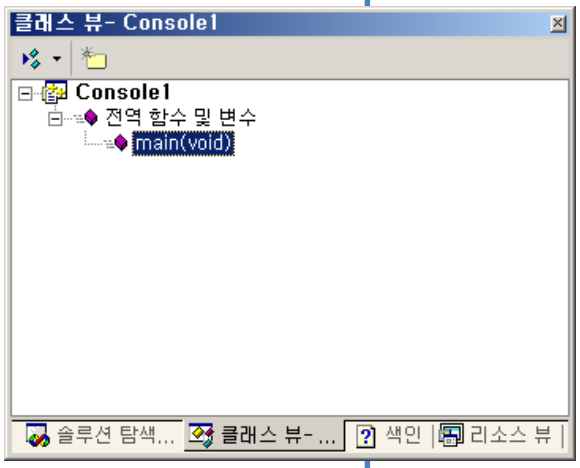
- Console1 프로젝트 생성
- 코드 작성
- 컴파일 및 실행

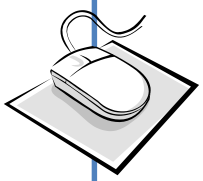
```
#include <stdio.h>

void main()
{
	printf("Hello, World!₩n");
}
```



2. 조금 복잡한 C 프로그램

```
#include <stdio.h>

void main()
{
        int iX;
        int iY;

        iX = 2;
        iY = 3;

        int iResult = iX + iY;

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





2. 조금 복잡한 C 프로그램

- 변수의 종류
  - 전역 변수 : 함수 밖에서 만든(정의한) 변수
  - 지역 변수 : 함수 안에서 만든 변수
- 생명주기(life time)
  - 전역 변수 : 프로그램이 실행될 때 만들어지고 프로그램이 종료될 때 사라짐
  - 지역 변수 : 함수가 실행될 때 만들어지고 함수가 끝날 때 사라짐
- 사용 범위(scope)
  - 전역 변수 : 모든 함수에서 접근
  - 지역 변수 : 변수를 정의한 함수 안에서만 접근

2. 조금 복잡한 C 프로그램

- 구조화 프로그램 (C, Pascal 등)
  - 순차, 조건, 반복 문으로 작성한 프로그램
  - 실행 순서가 항상 위에서 아래로 진행
  - 이해하기 쉬움
- 비구조화 프로그램 (Fortran 등)
  - goto 문을 이용한 프로그램
  - 실행 순서 흐름을 예측하기 힘듦
  - 바람직하지 않은 프로그래밍 방식

3. 함수를 이용한 구조화 프로그램

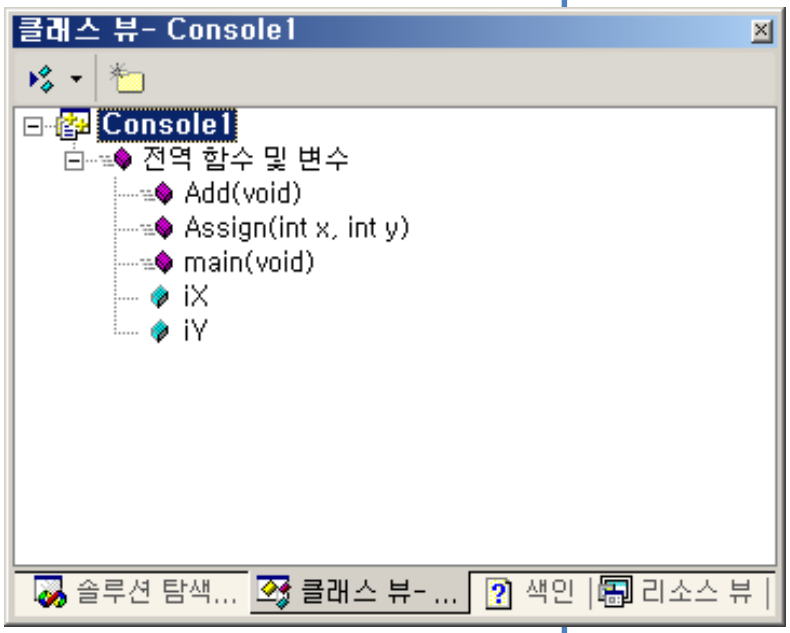
- 앞의 프로그램이 수 백 줄로 구성된다면?
  - 코드가 복잡하게 보이고 읽기(분석하기) 힘듦
  - 이를 해결할 수 있는 방법 -> 함수 이용
- 모듈의 의미
  - 함수
  - 블록 등
  - 파일(모듈별 분할 컴파일이 가능한 언어 C)
  - 클래스 (객체지향 언어 C++)

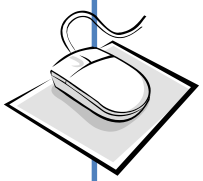
3. 함수를 이용한 구조화 프로그램

- 추상화의 의미
  - 간단히 묶어서, 줄여서 표현하는 것
- 코드를 간단히 표현하는 것 : 코드 추상화
  - 함수

3. 함수를 이용한 구조화 프로그램

```
#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);
}
```





4. 100개의 변수와 1000개의 함수

- 예상되는 문제점
  - 오류가 발생할 경우 오류 수정이 어려움
  - 잘못된 접근을 방지할 수 있는 장치가 없음
  - 코드 재사용이 어려움
  - 결국 소프트웨어 위기가 발생

4. 100개의 변수와 1000개의 함수

- 왜 이런 문제가 발생할까?
  - 어떤 함수가 어떤 변수를 액세스하는지 쉽게 알 수 없음
  - 어떤 함수가 어떤 함수를 호출하는지 쉽게 알 수 없음
  - 함수와 변수들이 기준 없이 뒤섞여 있어서 재사용시 어디까지 사용해야 하는지 쉽게 알 수 없음
  - 초기화 함수와 같이 반드시 호출해야 하는 함수를 쉽게 알 수 없음

4. 100개의 변수와 1000개의 함수

- 이를 해결하려면?
  - 관련된 변수와 함수를 묶어야 함
  - 코드 변수(데이터) 추상화
  - 데이터 추상화