모듈별 분할 컴파일 및 라이브러리화

제주대학교 컴퓨터공학과
변영철 교수
(ycb@jejunu.ac.kr)

이 장을 공부하면

- 내가 작성한 프로그램을 여러 파일(모듈)로 쪼갤 수 있다.
- 라이브러리로 만들 수 있다.
- 내가 만든 라이브러리를 쉽게 이용(재사용)할 수 있다.

가장 간단한 프로그램 작성

- 새로운 프로젝트 My 생성
- 프로젝트 | 새 항목 추가, C++ 파일 (My.cpp) 생성
- 다음 코드 작성

```
#include <stdio.h>

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

코드 추상화

- 함수로 추상화

```
#include <stdio.h>

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

void main()
{
	SayHello();
}
```

다른 파일로 작성 – Hello.cpp

- Hello.cpp 만든 후 여기로 함수 이동

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

- 발생하는 오류 수정
  - stdio.h 헤더 파일 include

헤더 파일 작성 – Hello.h

- Hello.h 만든 후 함수 선언
  - 헤더 파일에는 컴파일러에게 알리는(선언) 코드 작성

```
#pragma once

void Hello();
```

- My.cpp 오류 수정

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

void main()
{
        SayHello();
}
```

라이브러리화

- C 루트에 라이브러리 폴더(clib) 생성
- 금방 만든 Hello.cpp와 Hello.h 파일을 라이브러리 폴더로 이동
- Visual Studio에 남아 있는 Hello.cpp, Hello.h 파일 삭제

라이브러리 이용하기

- Hello.cpp 라이브러리 파일 추가
  - 프로젝트 | 기존 항목 추가...
  - c:₩clib 폴더로 이동하여 Hello.cpp 추가
- 헤더 파일 경로 추가
  - 프로젝트 | 속성 선택
  - C/C++ | 일반 선택
  - 오른쪽 맨 위의 '추가 포함(include) 디렉토리'에 라이브러리 폴더 입력 – c:₩clib
- 컴파일 및 실행

나만의 라이브러리, 왜 좋을까?

- 나중에 쉽게 재사용(함수 호출) 할 수 있다.
- 코드가 어떻게 돌아가는지 잊어버려도 문제없다(블랙박스).
- 쉽게 구현할 수 있으므로 프로그래밍이 부담스럽지 않다.

요 약

- 내가 작성한 프로그램을 여러 파일(모듈)로 쪼갤 수 있다.
- 라이브러리로 만들 수 있다.
- 내가 만든 라이브러리를 쉽게 이용(재사용)할 수 있다.