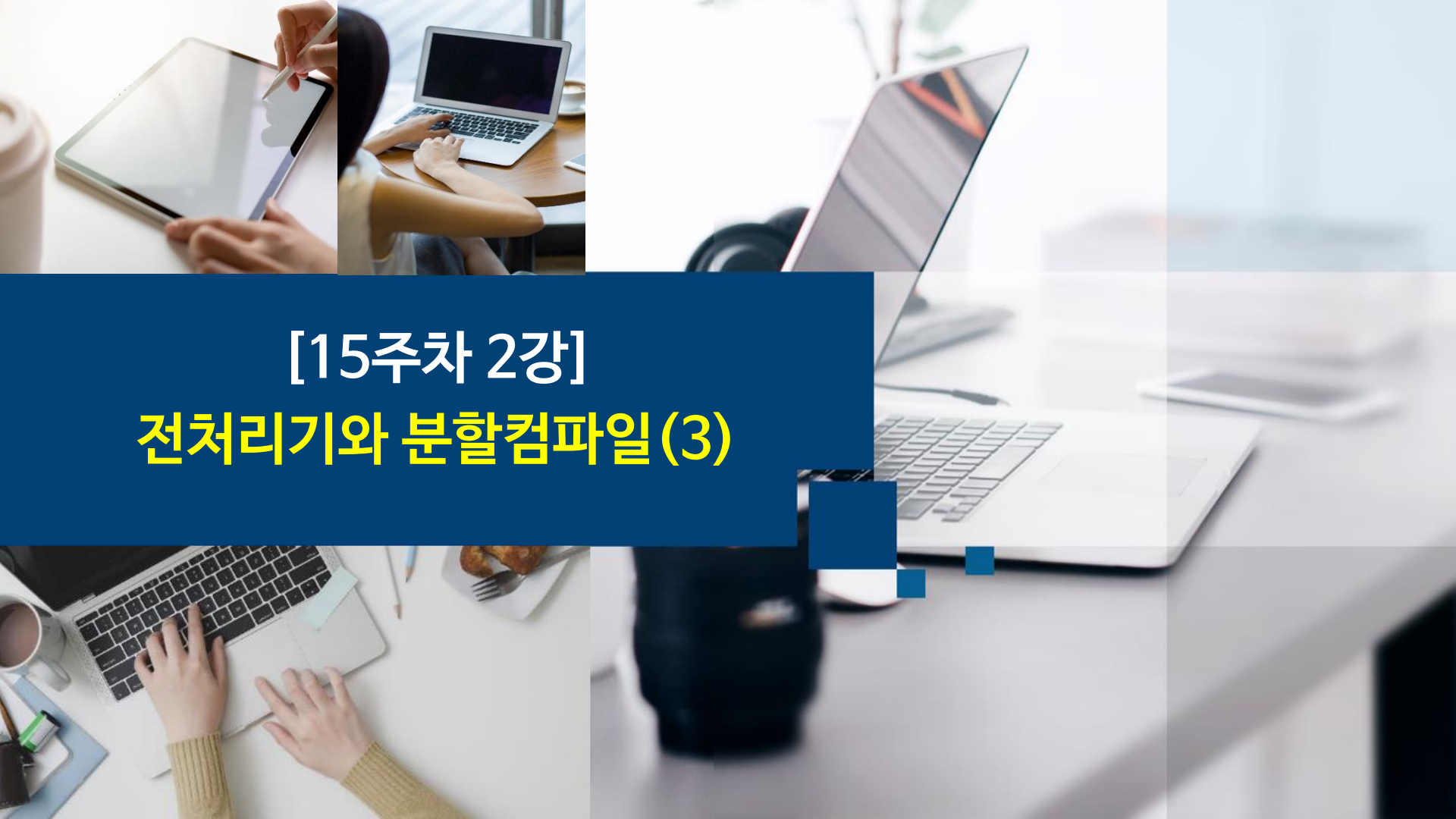

[15주차 2강]
전처리기와 분할컴파일(3)

학습 내용

- 15.2 분할 컴파일 (2)
- 15.3 변수의 사용범위와 지속기간

학습 목표

- 15.2 분할컴파일을 이해하고 활용할 수 있다.
- 15.3 변수의 사용범위와 지속기간을 이해한다.

C프로그래밍및실습

1. 전처리기
2. 분할 컴파일 (2)
3. 변수의 사용범위와 지속기간

2. 분할 컴파일

헤더 파일 (.h)

- 여러 소스 파일에서 사용하는 외부 변수나 함수 선언 등을 모아 놓은 파일로, 필요한 경우 간단히 헤더파일만 include 시키면 됨
  - ✓ 예) 문자열 관련 함수는 〈string.h〉에 선언
- 헤더파일(.h)에 포함되는 내용
  - ✓ 상수, 자료형 정의 (구조체 정의 등), 함수 원형 선언, 외부 변수 선언(extern), typedef 정의, 매크로 명령문
- 비교) 소스파일(.c)에 포함되는 내용
  - ✓ 함수 본체 정의, 전역 변수 선언(정의)
- Visual Studio에서 사용자 정의 헤더 파일을 만드는 방법
  - ✓ 소스 파일 추가와 유사

2. 분할 컴파일

main.c

```c
#include <stdio.h>
int myfunc1(int x, int y);
void myfunc2(int n);
int main() {
  int a, b, cnt = 0;
  scanf("%d %d", &a, &b);
  cnt = myfunc1(a, b);
  myfunc2(cnt);
  return 0;
}
```

myfunc.c

```c
#include <stdio.h>
int myfunc1(int x, int y) {
  return (x * y - x);
}
void myfunc2(int n) {
  int i;
  for (i = 0; i < n; i++)
    printf("%d\n", i+1);
}
```

myheader.h

```c
#include <stdio.h>
int myfunc1(int x, int y);
void myfunc2(int n);
```

main.c

```c
#include "myheader.h"
int main() {
  int a, b, cnt = 0;
  scanf("%d %d", &a, &b);
  cnt = myfunc1(a, b);
  myfunc2(cnt);
  return 0;
}
```

myfunc.c

```c
#include "myheader.h"
int myfunc1(int x, int y) {
  return (x * y - x);
}
void myfunc2(int n) {
  int i;
  for (i = 0; i < n; i++)
    printf("%d\n", i+1);
}
```

2. 분할 컴파일

헤더파일 중복삽입 문제

- 예) struct student가 두 번 정의됨

myheader1.h

```
#include <stdio.h>
#define SIZE 3
struct student{
    int id;
    double score;
};
extern struct student st[SIZE];
```

myheader2.h

```
#include "myheader1.h"
void grade(struct student *st1);
```

main.c

```
#include "myheader1.h"
#include "myheader2.h"
struct student st[SIZE];

int main() {
  int i;
  for (i=0; i<SIZE; i++) {
    scanf("%d %lf", &st[i].id, &st[i].score);
    grade(&st[i]);
  }
  return 0;
}
```

grade.c

```
#include "myheader1.h"
void grade(struct student *st1) {
    ...
}
```

헤더파일 중복삽입 문제 해결법

- 조건부 컴파일을 활용

```
#ifndef 헤더파일명
#define 헤더파일명
  (헤더파일 내용)
#endif
```

```
#ifndef __MYHEADER_H__
#define __MYHEADER_H__
  ...
#endif
```

✓ 추후에 이 헤더 파일이 다시 포함되더라도,
이미 상수 __MYHEADER_H__가 정의되어 있으므로,
헤더 파일 내용은 중복으로 포함되지 않음

2. 분할 컴파일

앞서 살펴 본 프로그램에서의 헤더파일 중복삽입 문제 해결

myheader1.h

```
#ifndef __MYHEADER1_H__
#define __MYHEADER1_H__
#include <stdio.h>
#define SIZE 3
struct student{
   ...
};
extern struct student st[SIZE];
#endif
```

①

__MYHEADER_H__가
매크로 상수로 정의됨

②

main.c

```
#include "myheader1.h"
#include "myheader2.h"
struct student st[SIZE];
int main() {
  int i;
  for (i=0; i<SIZE; i++) {
    scanf("%d %lf", &st[i].id, &st[i].score);
    grade(&st[i]);
  }
  return 0;
}
```

myheader2.h

```
#ifndef __MYHEADER2_H__
#define __MYHEADER2_H_
#include "myheader1.h"
void grade(struct student *st1);
#endif
```

__MYHEADER_H__가 이미 정의되어 있으므로
#include "myheader.h"는 중복으로 삽입되지 않음!

C프로그래밍및실습

1. 전처리기
2. 분할 컴파일 (2)
3. 변수의 사용범위와 지속기간

3. 변수의 사용범위와 지속기간

다양한 변수의 종류

- 전역변수, 지역변수, 정적 변수

변수의 종류를 구분 짓는 특징

- 변수의 **사용범위**
  - ✓ 프로그램에서 변수가 사용될 수 있는 장소의 범위
  - ✓ 변수의 공간적 특성을 나타냄
- 변수의 **지속기간**
  - ✓ 변수에 할당된 기억장소가 존재하는 기간
  - ✓ 변수의 시간적 특성을 나타냄

3. 변수의 사용범위와 지속기간

전역 변수 (사용범위와 지속기간 모두 전역적)

- 사용범위
  - ✓ 프로그램 전역에서 사용 가능
  - ✓ extern 키워드를 이용하면 다른 파일에서도 사용 가능
- 지속기간
  - ✓ 프로그램 시작 시 할당된 메모리 공간이 프로그램 종료 시까지 유지

3. 변수의 사용범위와 지속기간

각 변수 별 사용범위 : 전역 변수

main.c

```
. . . . . .
int global1;
extern func2();
int main()
{
    int local1;
    func2();
    . . . . .
}
```

func1.c

```
. . . . . .
extern int global1;
static int func1()
{
    int local2;
    . . . . .
}
```

func2.c

```
. . . . . .
static int global2;
int func2()
{
    static int local3;
    . . . . .
}
```

3. 변수의 사용범위와 지속기간

지역 변수 (사용범위와 지속기간 모두 지역적)

- 사용범위
  - ✓ 선언된 함수/블록 내부에서만 사용 가능

- 지속기간
  - ✓ 프로그램 수행 도중, 함수/블록 시작 시 메모리 할당되고, 해당 함수/블록이 끝나면 해제

3. 변수의 사용범위와 지속기간

각 변수 별 사용범위 : 지역 변수

main.c

```
. . . . . .
int global1;
extern func2();
int main()
{
    int local1;
    func2();
    . . . . .
}
```

func1.c

```
. . . . . .
extern int global1;
static int func1()
{
    int local2;
    . . . . .
}
```

func2.c

```
. . . . . .
static int global2;
int func2()
{
    static int local3;
    . . . . .
}
```

3. 변수의 사용범위와 지속기간

정적변수

- 분류
  - ✓ 정적 지역 변수: 지역 변수 앞에 static 키워드가 붙은 변수
  - ✓ 정적 전역 변수: 전역 변수 앞에 static 키워드가 붙은 변수
- 변수의 지속기간 (전역적)
  - ✓ 둘 다 전역 변수처럼 프로그램 실행 기간 전체 동안 지속
- 변수의 사용범위 (제한됨)
  - ✓ 정적 지역 변수: 선언된 함수/블록 내부에서만 사용
  - ✓ 정적 전역 변수: 선언된 파일 내부에서만 사용

3. 변수의 사용범위와 지속기간

각 변수 별 사용범위 : 정적 변수

main.c

```
. . . . . .
int global1;
extern func2();
int main()
{
    int local1;
    func2();
    . . . . .
}
```

func1.c

```
. . . . . .
extern int global1;
static int func1()
{
    int local2;
    . . . . .
}
```

func2.c

```
. . . . . .
static int global2;
int func2()
{
    static int local3;
    . . . . .
}
```

global2 변수는 func2.c 파일 내에서만 사용 가능

func1() 함수는 func1.c 파일 내에서만 호출 가능

local3 변수는 func2() 함수 내에서만 사용 가능

3. 변수의 사용범위와 지속기간

변수의 특징 정리

분류		관련 예약어	지속기간	기본 초기값	사용범위	동일 파일의 외부함수에서 사용	다른 파일의 외부함수에서 사용
전역변수		extern	프로그램 실행기간 전체	0 '\0' NULL (자료형에 따라)	프로그램 전체	O	O
정적 변수	전역	static			파일 내부	O	X
	지역				함수/ 블록 내부	X	X
지역변수		-	변수가 선언된 함수/블록의 실행 기간	쓰레기 값			

학습 정리

- **헤더파일**은 여러 소스 파일에서 사용하는 변수 및 함수에 대한 표준화 수단 제공

- **소스 파일**에는 변수 선언, 함수 정의 등이 포함되고,
  헤더 파일에는 함수의 원형 선언, 자료형 정의 등이 포함됨

- 헤더파일 중복 삽입 문제는 조건부 전처리 지시자를 사용하여 해결할 수 있음

- **전역 변수**와 **정적 변수**는 전체 프로그램 실행기간 동안 지속되고,
  지역 변수는 변수가 선언된 함수 또는 블록의 실행기간 동안만 지속됨