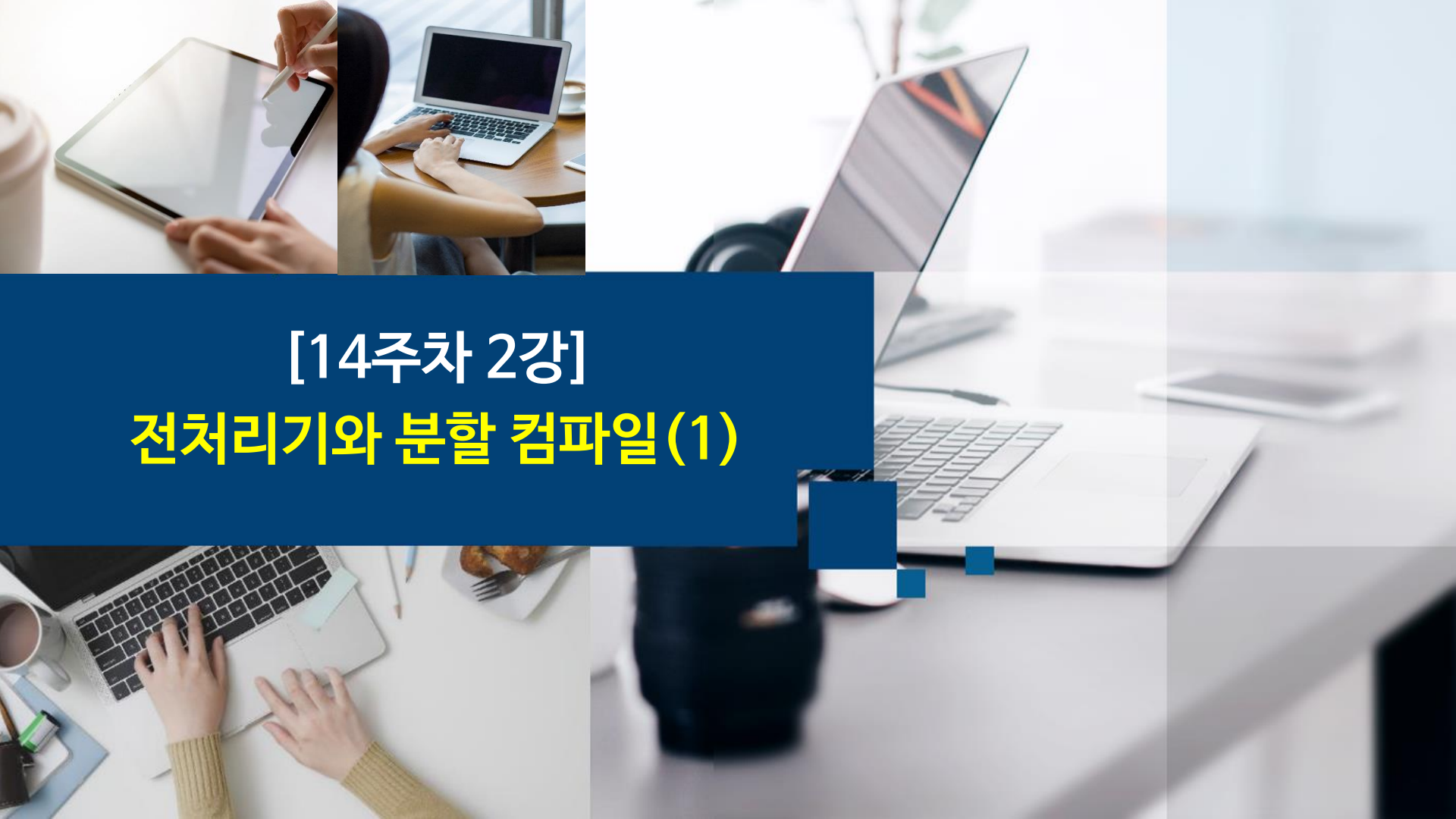


[14주차 2강]
전처리기와 분할 컴파일(1)

학습 내용

- 15.1 전처리기

학습 목표

- 15.1 전처리 개념과 전처리 지시자를 활용할 수 있다.

C프로그래밍 및 실습

1. 전처리기
2. 분할 컴파일
3. 변수의 사용범위와 지속기간

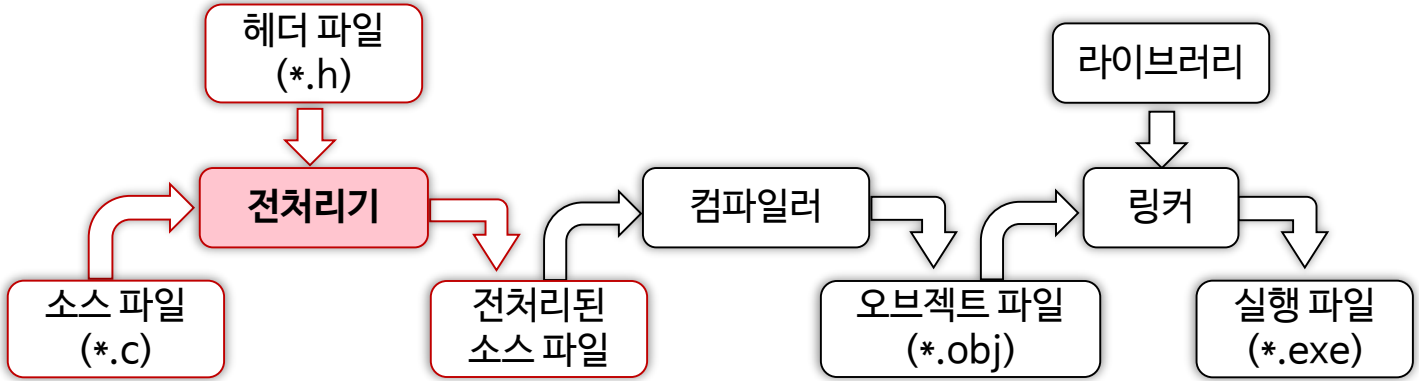
1. 전처리기

전처리란?

- 컴파일러가 소스 파일(*.c)을 컴파일하기 이전 과정
- 전처리기 지시자를 사용 (예) #include

사용하는 이유

- 프로그램을 간단히 확장하고, 가독성을 높여 유지보수에 도움



1. 전처리기

전처리 지시자

- 프로그램 선두에 위치
- #으로 시작하고 문장 끝에 세미콜론(;)을 사용하지 않음

자주 사용되는 전처리 지시자

전처리 지시자	기능
#include	프로그램 외부의 파일을 불러옴
#define	매크로(macro) 상수/함수를 정의
#undef	정의한 매크로를 취소
#if ~ (#elif ~ #else ~) #endif	조건부 컴파일
#ifdef ~ (#elif ~ #else ~) #endif	
#ifndef ~ (#elif ~ #else ~) #endif	

1. 전처리기

#include

- 프로그램 외부에 존재하는 파일을 소스에 포함시킴
- 지정된 특정 파일의 내용을 해당 지시자가 있는 위치에 삽입
- 〈 〉 : 컴파일러가 제공하는 헤더 파일을 포함시킬 때 주로 사용
- " " : 프로그래머가 직접 만든 파일을 포함시킬 때 주로 사용
  - ✓ 절대 경로 혹은 상대 경로 형식으로 사용
  - ✓ fopen()에서 파일이름을 작성할 때와는 달리, \ 만으로 경로 표현

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

```
#include "myheader.h"
#include "C:\user\mylib\lib.h"
#include "..\mylib\lib.h"
```

1. 전처리기

#define

- 특정 상수를 프로그래머가 정의한 문자열로 대체
  - ✓ 프로그램의 가독성을 높여 유지보수 용이
- 원칙적으로는 한 라인에 작성
  - ✓ 긴 경우에는 \(역 슬래시) 기호를 사용하여 다음 줄에 작성되어 있는 내용과 연결
- **매크로 상수**
  - ✓ 반복적으로 사용되는 상수를 새로운 이름으로 정의하여 사용
- **매크로 함수**
  - ✓ 반복적으로 사용되는 프로그램의 모듈을 함수로 정의하여 사용
  - ✓ 프로그램의 해독과 수정이 용이

매크로 상수

- 형식 : #define 매크로이름 상수
  - ✓ 매크로이름 : 주로 대문자로 표시, 공백 허용하지 않고 숫자로 시작하면 안됨
  - ✓ 상수 : 숫자, 문자, 문자열, 시스템 이름, 자료형 이름 등

```
#define PI 3.14
#define MAX 100
#define SUM MAX+1  // 중첩 매크로: SUM 정의 시
                   // 앞서 정의된 매크로 상수 MAX를 사용
```

매크로 상수의 필요성 및 사용법

- 아래에서 PI를 정의하지 않고, 3.14로 프로그램을 작성했다면?
  - ✓ 3.14를 3.1415로 변경하려면, 일일이 3.14를 찾아서 수정해야 함

```
#include <stdio.h>
#define PI 3.14              // 원주율을 매크로 상수로 정의

int main() {
    int r = 3;               // 원의 반지름
    double cir, area;        // 원의 둘레, 원의 면적

    cir = 2 * PI * r;        // 매크로 상수 사용
    area = PI * r * r;
    printf("cir = %f, area = %f\n", cir, area);
    return 0;
}
```

매크로 함수

- 형식 : #define 매크로함수이름(함수인자) 함수정의
  - ✓ 매크로 함수이름 : 주로 대문자로 표시

```
#define ADD(x, y) ((x) + (y))
#define MAX(x, y) ((x) > (y) ? (x) : (y))
```

- 주의사항
  - ✓ 매크로 함수이름과 (함수인자) 사이에 공백이 없어야 함
  - ✓ 정의 부분에서 각 함수 인자를 반드시 괄호 ( ) 로 묶어 주어야 함
  - ✓ 정의 부분 전체를 반드시 괄호 ( ) 로 묶어 주어야 함

매크로 함수 사용 예제

```
#include <stdio.h>

#define PI 3.14                         // 매크로 상수
#define SQUARE(x) ((x) * (x))           // 매크로 함수

int main() {
    double area;                        // 원의 면적

    area = PI * SQUARE(2 + 2);          // area = 3.14 * (4 * 4)
    printf("area = %f\n", area);
    return 0;
}
```

매크로 함수 정의에서 괄호 사용의 필요성

```
// 의도한 결과
#define SQUARE(x) ((x) * (x))
PI * SQUARE(2 + 2)
→ 3.14 * ((2 + 2) * (2 + 2))
→ 3.14 * (4 * 4)
→ 50.24
```

```
// 인자의 괄호를 생략하면?
#define SQUARE(x) (x * x)
PI * SQUARE(2 + 2)
→ 3.14 * (2 + 2 * 2 + 2)
→ 3.14 * (2 + 4 + 2)
→ 25.12
```

```
// 의도한 결과
#define ADD(x, y) ((x) + (y))
5 * ADD(2, 3)
→ 5 * ((2) + (3))
→ 5 * (5)
→ 25
```

```
// 정의부 전체의 괄호를 생략하면?
#define ADD(x, y) (x) + (y)
5 * ADD(2, 3)
→ 5 * (2) + (3)
→ 10 + 3
→ 13
```

#undef

- 정의된 매크로를 해제하는 지시자
- 해제된 매크로는 재정의 가능

- 사용법 : **#undef 매크로이름**

```
#define MAX 100

#undef MAX

#define MAX 5000
```

조건부 컴파일 전처리 지시자

- 전처리문에서 주어진 조건의 만족 여부에 따라
  코드를 선택적으로 컴파일하도록 하는 기능
- 변수, 함수, 매크로가 중복되지 않도록 하거나,
  이식성(호환성) 높은 코드를 개발할 때 유용
- 사용법은 조건문의 if 문과 유사

전처리 지시자	기능
`#if ~ #endif`	조건이 참이면 컴파일에 포함
`#ifdef ~ #endif`	매크로가 정의되어 있으면 컴파일에 포함
`#ifndef ~ #endif`	매크로가 정의되어 있지 않으면 컴파일에 포함

1. 전처리기

#if ~ (#elif) ~ (#else) ~ #endif

- 조건식이 참이면 컴파일에 포함

```
#if 조건식1
   컴파일 할 문장1;
#elif 조건식2
   컴파일 할 문장2;
…
#else
   컴파일 할 문장5;
#endif
```

```
#if   OS == 1              // Linux
   컴파일 할 문장1;
#elif  OS  == 2            // Windows
   컴파일 할 문장2;
#else                      // 그 외의 OS
   컴파일 할 문장3;
#endif
```

- 주의) 조건식에서
  - ✓ 실수 상수, 문자열 상수, 변수 등을 사용할 수 없음
  - ✓ 관계연산자, 논리연산자, 산술연산자는 사용 가능

1. 전처리기

#ifdef ~ (#elif) ~ (#else) ~ #endif

- 매크로가 정의되어 있으면 컴파일에 포함

```
#ifdef 매크로이름1
   컴파일 할 문장1;
#elif 매크로이름2
   컴파일 할 문장2;
…
#else
   컴파일 할 문장5;
#endif
```

```
#ifdef  UNIX
   #define DDIR "/usr/data"
#else
   #define DDIR "\usr\data"
#endif
```

#ifndef ~ (#elif) ~ (#else) ~ #endif

- 매크로가 정의되어 있지 않으면 컴파일에 포함

학습 정리

- **전처리기**는 컴파일러가 소스 파일을 컴파일하기 이전에, 전처리 지시자를 이용하여 특정 파일의 내용을 포함하거나 상수, 문자열 치환 등과 같은 작업을 수행함

- **#include**는 외부 파일을 포함시킬 때 사용

- **#define**을 이용하여 특정 상수를 치환 문자열로 대체하거나, **#undef**를 이용하여 정의된 매크로를 해제할 수 있음

- **#if ~ (#elif) ~ (#else) ~ #endif**는 조건에 따라 코드를 선택적으로 컴파일하는 조건부 컴파일을 위한 전처리 지시자임