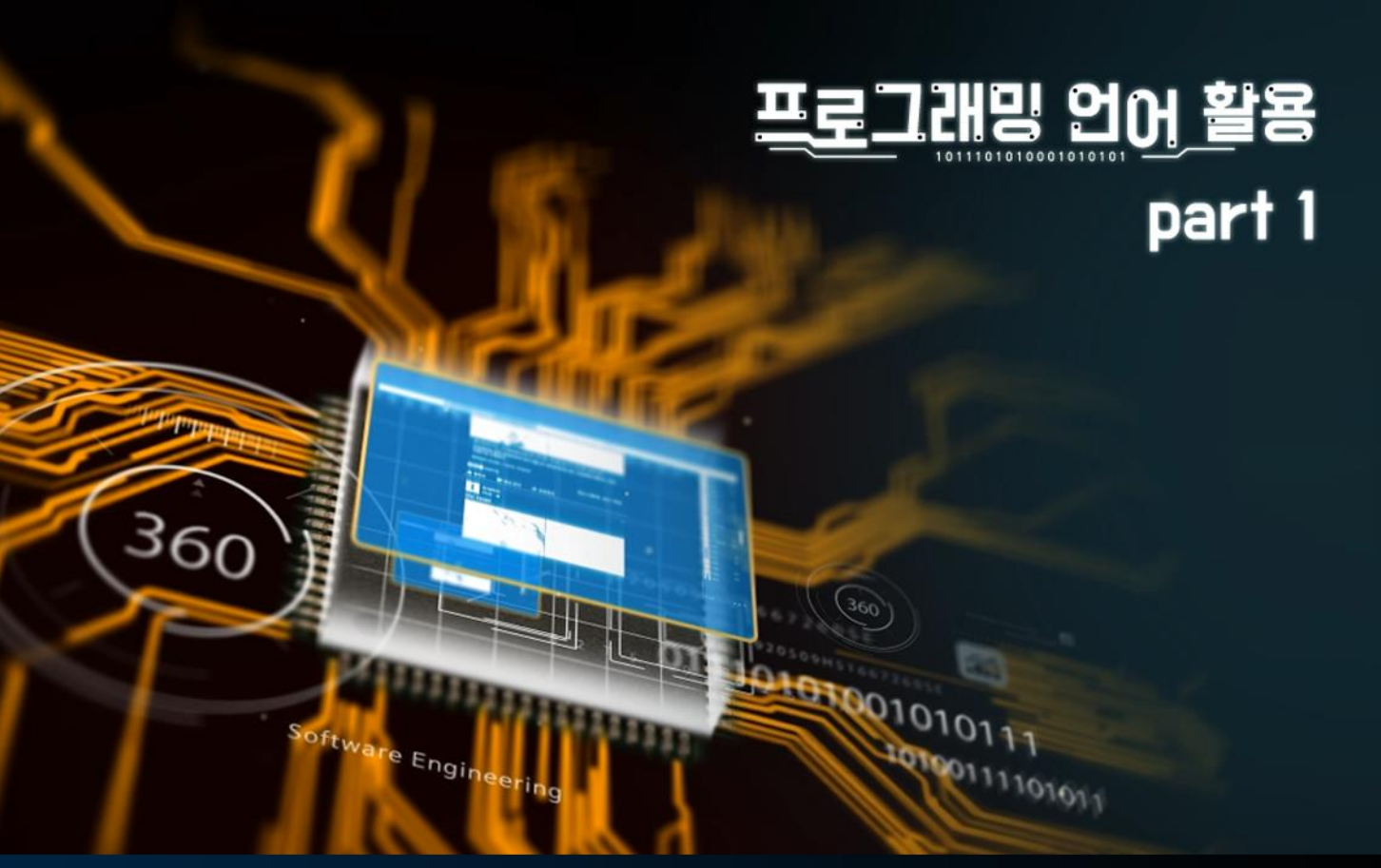


전처리기

한국기술교육대학교
온라인평생교육원
OLEI

학습내용

- 매크로
- 조건부 컴파일

학습목표

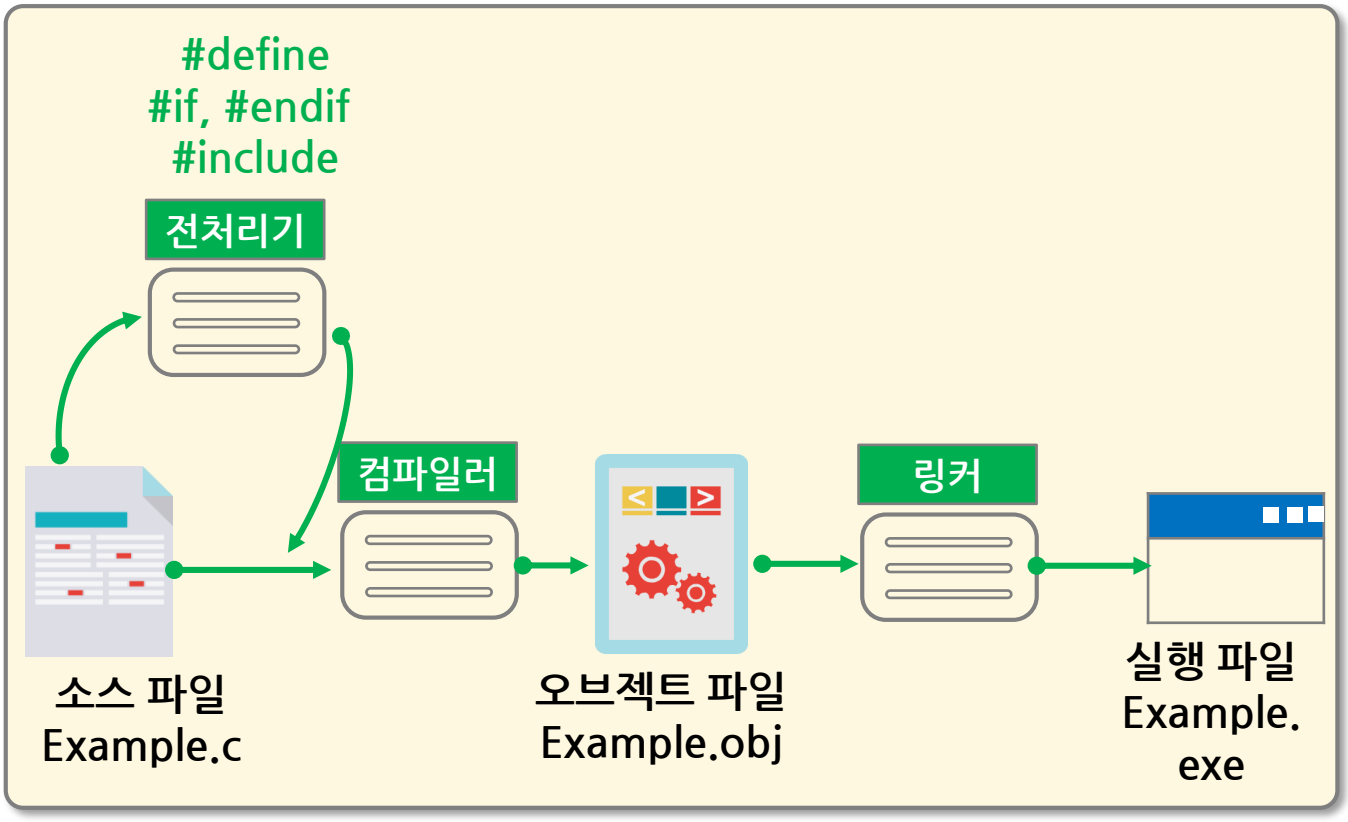
- 매크로 상수와 매크로 함수의 쓰임새를 알고 프로그램에 구현할 수 있다.
- 조건부 컴파일 구분을 이해하고 사용할 수 있다.

매크로

1 전처리기

컴파일러가 소스 파일을
컴파일하기 전에 먼저 수행되는 프로그램

1. 프로그래머가 작성한 소스 파일이 컴파일될 수 있도록 준비함
2. 전처리기 문장은 **'#'으로 시작**



매크로

1 전처리기

전처리기 문장	기능
#define	매크로를 정의
#include	헤더 파일을 포함
#if, #else, #endif	조건에 따라 컴파일
#ifdef	매크로가 정의된 경우에 컴파일
#ifndef	매크로가 정의되지 않는 경우에 컴파일

매크로

2 매크로 상수

형식 #define 매크로명 값

예제

```
#define   MAX   100
#define   MIN   -MAX
#define   PI    3.1415
#define   MSG   "잘못 입력했습니다."
```

매크로명 : MAX, MIN, PI, MSG

값 : 100, -MAX, 3.1415, "잘못 입력했습니다."

```
#define MAX 100
```
→ 매크로 상수 MAX를 정의함

전처리기 소스 파일에서 매크로 상수가 사용된 곳을 모두 찾아서 정의된 문자열로 대치

```
average = sum / MAX;
printf("MAX = %d", MAX);
```
→ average = sum / 100;으로 처리됨

→ printf("MAX = %d", 100)으로 처리됨

매크로

2 매크로 상수

```
#define PI 3.1415
area = PI * radius * radius;
```

실수형 상수를 매크로 상수로 정의

area = 3.1415 *radius *radius;로 처리

```
#define MSG "잘못 입력했습니다."
if( n < 0 )
    printf(MSG);
```

문자열 상수를 매크로 상수로 정의

printf("잘못 입력했습니다.")로 처리

```
#define PRT printf
PRT("hello");
```

함수명을 매크로 상수로 정의

printf("hello")로 처리

```
#define UINT unsigned int
UINT num;
```

데이터형을 매크로 상수로 정의

unsigned int num;으로 처리

3 매크로 함수

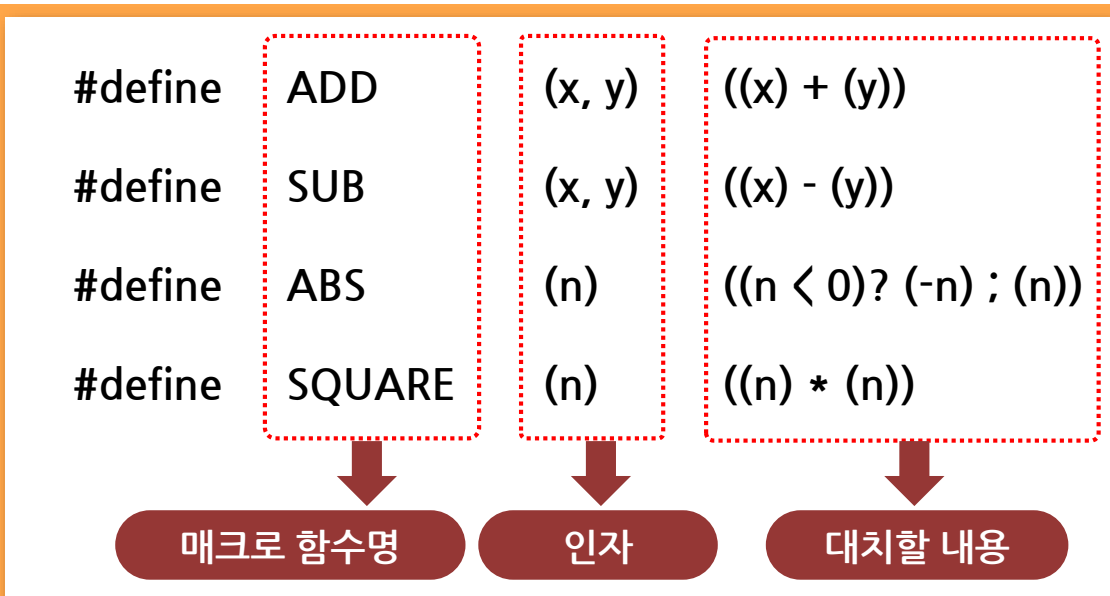
1. 매크로 함수는 함수처럼 인자를 갖는 매크로
2. 매크로 상수처럼 매크로 함수가 사용되는 곳에 문자열 대치를 통해서 코드를 확장함

매크로

3 매크로 함수

형식 #define 매크로 함수명(인자1, 인자2, …) 대치할 내용

예제

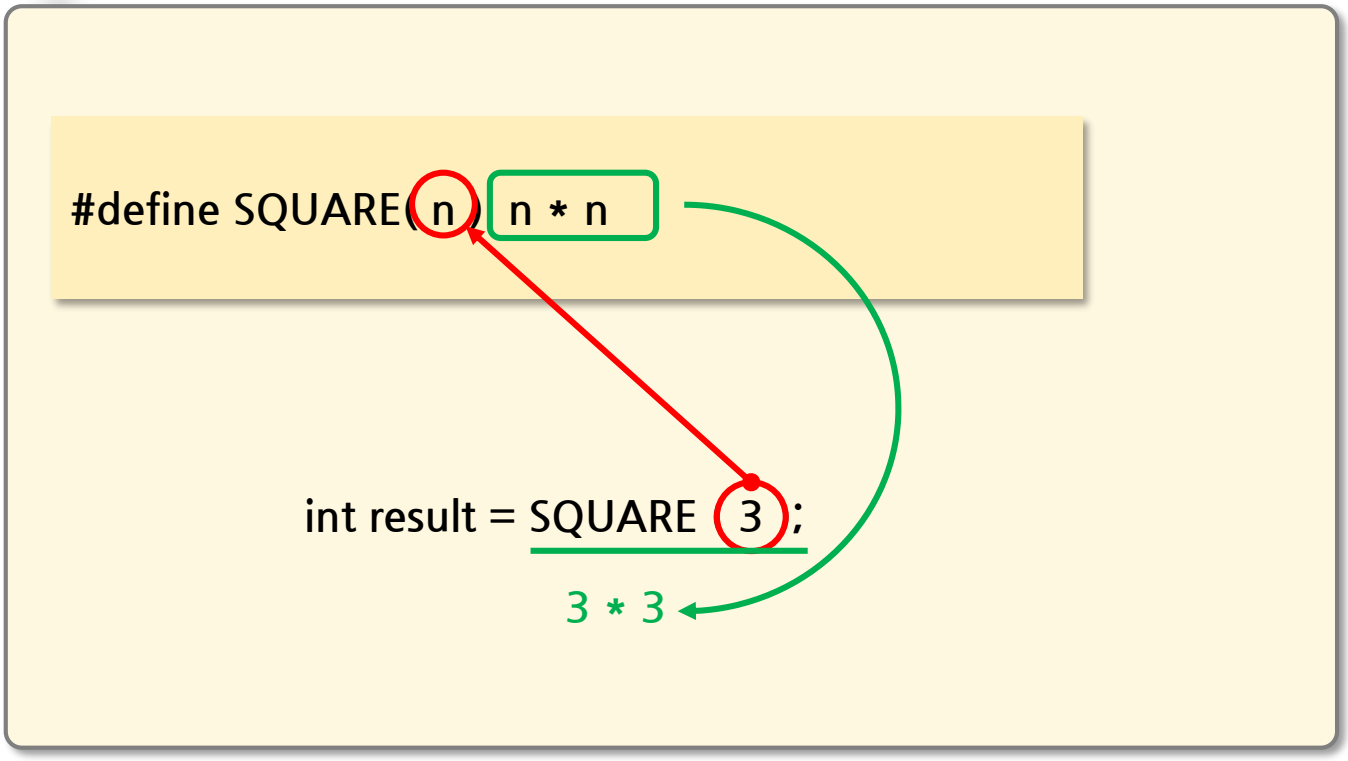



```
#define SQUARE(n) n*n
int main(void)
{
    int result = SQUARE(3);
}
```

int result = 3*3;으로 문자열 대치됨

매크로

3 매크로 함수



3 매크로 함수는 함수인 것처럼 보이지만 사실은 **함수가 아님**

4 컴파일 시에 인자의 데이터형 검사를 수행하지 않으며,
인자를 매개변수로 전달하는 **함수 호출 과정이 수행되지 않음**

5 매크로 함수 사용 시 문자열 대치 과정에서 잘못된 결과가 만들어질 수도 있음

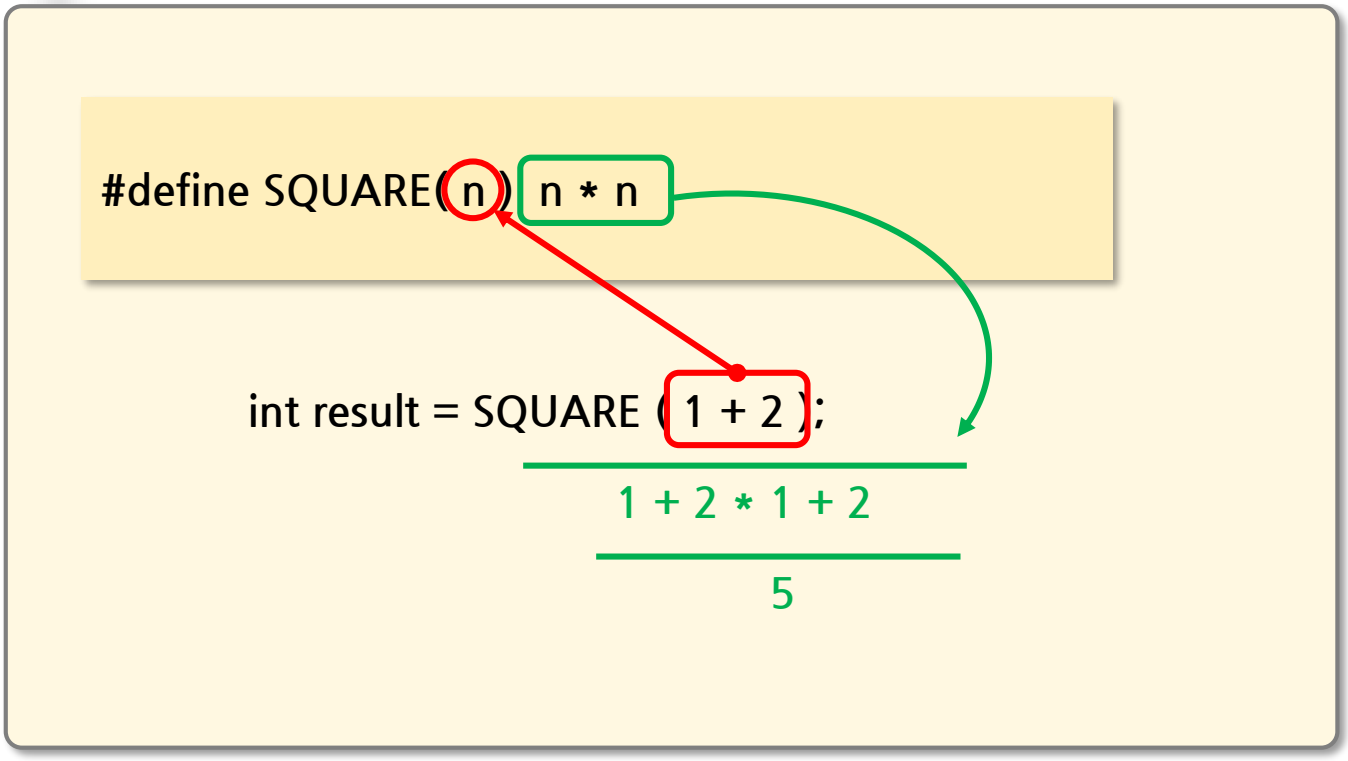
```
#define SQUARE(n) n*n
...
result = SQUARE(1+2);
```

#define SQUARE(n) ((n) * (n))

result = 1+2*1+2;로 문자열 대치됨

매크로

3 매크로 함수



매크로 함수의 장·단점

매크로 함수의 장점

- 매크로 함수를 사용하면 **프로그램의 실행 속도가 빨라짐**

  매크로 함수를 사용할 때는 함수 호출이 일어나지 않으므로 함수 호출의 오버헤드를 줄일 수 있음

매크로

3 매크로 함수

매크로 함수의 장·단점

매크로 함수의 단점

- 매크로 함수를 많이 사용하는 프로그램은 **프로그램의 크기가 커짐**
  - 일반 함수 코드는 한 번만 컴파일해서 만들어 두고, 함수 코드를 필요할 때마다 반복해서 호출함
  - 매크로 함수는 사용되는 곳마다 매크로 함수를 확장한 코드가 복사됨
- 매크로 함수를 사용하면, **코드가 알아보기 어려워짐**

4 예약 매크로

매크로	설명
__DATE__	최근에 컴파일한 날짜
__FILE__	소스파일의 이름을 절대경로와 함께함
__LINE__	소스파일에서 해당 문장이 있는 줄 번호
__TIME__	최근에 컴파일한 시각
__TIMESTAMP__	소스파일을 수정한 시각

매크로

4 예약 매크로

```
int main()
{
   printf("__DATE__ : %s :  ₩n", __DATE__ );
   printf("__FILE__ : %s :  ₩n", __FILE__ );
   printf("__LINE__ : %s :  ₩n", __LINE__ );
   printf("__TIME__ : %s :  ₩n", __TIME__ );
   printf("__TIMESTAMP__ : %s :  ₩n", __TIMESTAMP__ );

   return 0;
}
```

조건부 컴파일

1 #if, #else, #endif

01 특정 조건이 만족할 때만 코드를 컴파일함

02 상황에 따라서 특정 코드를 컴파일하게 또는 컴파일하지 않게 만들 수 있음

03 이식성 있는 코드를 개발할 때 유용함

형식

```
#if 조건식
    문장;
#endif
```

예제

```
#if WINVER >= 0x0501
    printf("윈도 XP 사용\n");
#endif
```

조건식

문장

조건부 컴파일

1 #if, #else, #endif

04 #if의 조건식에는 매크로를 정수와 비교하는 관계 연산자가 주로 사용되고, 산술 연산자, 논리 연산자 등이 사용될 수 있음

```
#if INFO__LEVEL = 1
문장;
#endif
```

INFO_LEVEL 매크로가 1이면 다음 문장을 컴파일 함

※ __ : 언더바(_)가 2개 입니다.

05 if의 조건식에서 매크로를 실수나 문자열과 비교할 수 없음

```
#if FACTOR ! = 0.5
#if ERR_MSG = “메모리 할당 실패”
```

실수와 비교할 수 없으므로 컴파일 에러

문자열과 비교할 수 없으므로 컴파일 에러

조건부 컴파일

1 #if, #else, #endif

06 #if에는 반드시 짝이 되는 #endif가 필요하며, #else를 함께 사용할 수도 있음

```
#if 조건
문장1;
#else
문장2;
#endif
```

- 조건이 참이면 문장1을 컴파일하고, 조건1이 거짓이면 문장2를 컴파일함
- #else문에 다른 조건을 다시 검사하려면 #elif를 사용함

07 #if, #endif에서는 컴파일할 문장이 여러 개여도 { }로 묶어줄 필요가 없음

```
result = x + y * 0.5 + z;
#if   INFO_LEVEL = 1
printf("result = %f \ n", result);
#elif   INFO_LEVEL = 2
printf("result = %f, x = %d, y = %d, z = %d \ n", result, x, y, z);
#else
printf("%d", x);      printf("%f", result);
#endif
```

조건부 컴파일

1 #if, #else, #endif

08 #if 안에 다른 #if를 중첩해서 사용할 수 있으며, 각각의 #if마다 #endif가 하나씩 짝을 이루어야 함

2 #ifdef

1 "if defined"라는 의미

2 #ifdef는 특정 매크로의 정의 여부에 따라 #ifdef와 #endif 사이의 문장을 컴파일할지 결정

형식

```
#indef 매크로명
        문장;
#endif
```

예제

```
#ifdef DEBUG
    printf("n = %d", n);
#endif
```

DEBUG → 매크로명

printf("n = %d", n); → 문장

조건부 컴파일

2 #ifdef

3 DEBUG 매크로 정의 시에만 함수 정보를 출력하는 경우

```
#ifdef DEBUG
    printf("GetFactorial 함수 호출:");
    printf("n = %d \ n", n);
#endif

    if( i <= 1 )
        return 1;
    return n * GetFactorial(n - 1);
}
```

DEBUG 매크로가 정의된 경우에만 출력문을 수행함

4 출력문에 수행되려면 DEBUG 매크로 정의가 필요함

```
#define DEBUG
```

DEBUG 매크로 심볼을 정의함

조건부 컴파일

3 #ifndef

"if not defined"라는 의미

#ifnded 다음의 매크로가 정의되지 않은 경우에만
#ifndef와 #endif 사이의 문장이 컴파일

```
#ifndef NAME_SIZE
        #define NAME_SIZE 20
#endif
```

학습정리

1. 매크로

- 전처리기는 컴파일 전 수행되는 프로그램임
- 매크로 상수와 함수는 컴파일 시 대치되는 문장임
- 예약 매크로는 미리 정의된 매크로 상수임

2. 조건부 컴파일

- 조건부 컴파일은 특정 조건이 만족할 때만 코드를 컴파일함
- #ifdef, #ifndef는 매크로 상수의 정의 여부에 따라 컴파일함