# 3. 전처리기(preprocessor)

## (1) 전처리문 종류

| 전처리문 | 기 능 |
|---|---|
| #include | 외부파일을 원시프로그램에 편입 |
| #define | 매크로 정의 (함수 및 상수) |
| #undef | 정의된 매크로를 취소 |
| #if-#endif | 조건에 따른 컴파일 |

## (2) 전처리(선행처리)

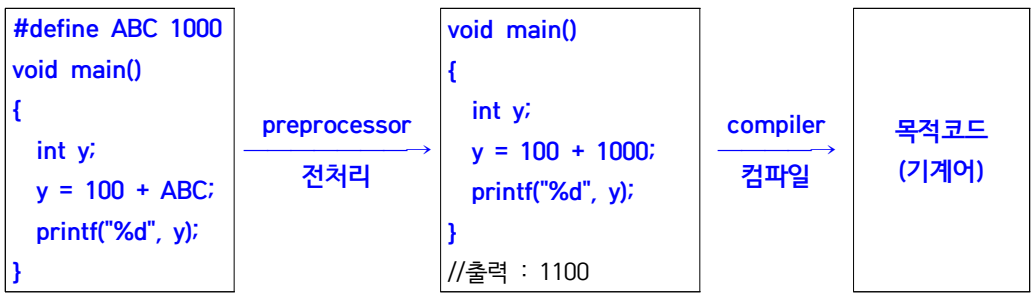
매크로 정의는 전처리문 #define을 사용한다.

```
#define   ABC   1000
#define   PI    3.141592
```

매크로 상수값

매크로 상수명(관례적으로 대문자로 기술)

위와 같은 방식으로 매크로를 정의한 후
상수명 ABC를 원시프로그램 내에 사용하면
전처리기에 의해 모든 ABC 위치에 수 1000이 치환되고 나서 컴파일 된다.



### (3) 매크로 중첩

매크로는 중첩될 수 있다.

```
#define A   5
#define B   A + 2
```
로 매크로 정의한 후

---

**수식을 "y = A * B;"처럼 기술했을 때 y의 값은?**

```
y = A * B;
  = A * A + 2;
  = 5 * 5 + 2;
  = 25 + 2;
  = 27 ──→ y의 값이 35가 아님을 주의해야 한다.
```

---

- 매크로는 **있는 모습 그대로 치환**(전개)된다.
- 매크로 정의가 수식에 적용될 때, 연산자 우선순위에 따라 값이 결정되므로 주의를 요한다.
- 해서, 매크로를 정의할 때는 매크로 명을 괄호로 묶으면 혼선을 피할 수 있다.
- 만약, #define B (A + 2)처럼 매크로를 정의하면
- y = A * B = A * (A + 2) = 5 * (5 + 2) = 5 * 7 = **35**

**[예제] 다음 C 프로그램의 출력 결과는?**

```
#include <stdio.h>
#define s1(a) (a) * (a)        //매크로 함수, 각 인수에 괄호가 있음
#define s2(b) b * b            //매크로 함수, 각 인수에 괄호가 없음
void main()
{
  printf("%d\n", s1(3+4));     //s1(3+4) = (3 + 4) * (3 + 4) = 7 * 7 = 49
  printf("%d\n", s2(3+4));     //s2(3+4) = 3 + 4 * 3 + 4 = 3 + 12 + 4 = 19
}
```

- 위의 예제는 매크로 함수를 정의할 때, **인수**에 **괄호** 존재 유무에 따른 차이를 설명하는 것이다.
- 매크로 정의에서 인수에 괄호가 없으면 단순히 눈으로 보는 **예측**과 **다르므로** 조심해야 한다.
- 해서, 매크로 정의에서 각 인수를 괄호로 묶는 것이 기본 원리이다.

## 기출문제 분석

### 1. C 프로그램의 실행 결과로 옳은 것은? [2010년 계리직]

```
---------------------------------------------------------------
#define VALUE1   1
#define VALUE2   2
void main(){
    float i;
    int j, k, m;

    i = 100 / 300;
    j = VALUE1 & VALUE2;
    k = VALUE1 | VALUE2;

    if (j && k || i) m = i + j;
    else m = j + k;
    printf("i = %.1f j = %d k = %d m = %03d\n", i, j, k, m);
}
---------------------------------------------------------------
```

① i = 0.0 j = 0 k = 3 m = 003
② i = 0.3 j = 0 k = 3 m = 000
③ i = 0.0 j = 1 k = 1 m = 001
④ i = 0.3 j = 1 k = 1 m = 001

♤ C 프로그램 연산 결과 - 매크로 상수

- i = 100 / 300 = 0.0 → 정수와 정수의 연산 결과는 정수(실수형에 대입)
- j = VALUE1 & VALUE2 = 1 & 2 = 0 → 비트 AND 연산
- k = VALUE1 | VALUE2 = 1 | 2 = 3 → 비트 OR 연산

```
    0000 0001          0000 0001
&   0000 0010      |   0000 0010
    0000 0000          0000 0011
```

if(j && k || i) → 조건은 거짓(j = 0, k = 3, i = 0.0)

m = j + k = 0 + 3 = **3**

정답 : ①