# printf

**printf**

- 고정인자
  - **format** → printf(“abcdefg%0.5d”,10) 에서 ""와 같이 출력형태를 말한다.
    - %[flag][width][.precision][type]
  - option
    - 기본 서식 지정자이며 정수, 실수, 문자, 문자열, 포인터의 메모리 주소를 출력하는 기본 서식 지정자.

| 서식 지정자 | 설명 |
|---|---|
| d, i | 부호 있는 10진 정수 |
| u | 부호 없는 10진 정수 |
| o | 부호 없는 8진 정수 |
| x | 부호 없는 16진 정수(소문자) |
| X | 부호 없는 16진 정수(대문자) |
| f | 실수를 소수점으로 표기(소문자) |
| F | 실수를 소수점으로 표기(대문자) |
| e | 실수 지수 표기법 사용(소문자) |
| E | 실수 지수 표기법 사용(대문자) |
| g | %f 와 %e 중에서 짧은 것을 사용(소문자) |
| G | %F 와 %E 중에서 짧은 것을 사용(대문자) |
| a | 실수를 16진법으로 표기(소문자) |
| A | 실수를 16진법으로 표기(대문자) |
| c | 문자 |
| s | 문자열 |
| p | 포인터의 메모리 주소 |
| n | %n 부분에 int 포인터를 넣으면 지금까지 출력한 문자 개수를 변수에 넣어줌<br>`int num1;`<br>`_set_printf_count_output(1); // Visual Studio에서 %n 활성화`<br>`printf("abcde%n\n", &num1); // num1의 주소(포인터)를 지정.`<br>`// 지금까지 출력한 문자 개수를 num1에 넣어줌`<br>`printf("%d\n", num1); // 5: abcde는 5글자이므로 5가 들어가 있음` |
| % | % 기호 출력 |
| **플래그** | **설명** |
| - | 왼쪽 정렬 |
| + | 양수일 때는 + 부호, 음수일 때는 - 부호 출력 |
| 공백 | 양수일 때는 부호를 출력하지 않고 공백으로 표시, 음수 일 때는 - 부호 출력 |
| # | 진법에 맞게 숫자 앞에 0, 0x, 0X를 붙임 |
| 0 | 출력하는 폭의 남는 공간에 0으로 채움 |
| **폭** | **설명** |
| 숫자 | 지정한 숫자만큼 폭을 지정하여 출력, 실수는 . (점), e+까지 폭에 포함됨 |
| **정밀도** | **설명** |
| .숫자 | 지정한 숫자만큼 소수점 아래 자리 출력 |

| | 서식 지정자 | | | | | | |
|---|---|---|---|---|---|---|---|
| 길이 | d i | o u x X | f F e g G a A | c | s | p | n |
| | int | unsigned int | float, double | int | char * | void * | int * |
| hh | signed char | unsigned char | | | | | signed char * |
| h | short int | unsigned short int | | | | | short int * |
| l | long int | unsigned long int | double | wint_t | wchar_t * | | long int * |
| ll | long long int | unsigned long long int | | | | | long long int * |
| j | intmax_t | uintmax_t | | | | | intmax_t * |
| z | size_t | size_t | | | | | size_t * |
| t | ptrdiff_t | ptrdiff_t | | | | | ptrdiff_t * |
| L | | | long double | | | | |

- flag
  - 출력할 값의 앞에 표시할 문자를 설정하는 영역.
  - 여러 개의 플래그를 동시에 설정 가능.
  - 문자, 문자열, 포인터, 정수 또는 부동 소수점과 같은 해당 인수를 해석하는지 여부를 지정함.
  - 왼쪽정렬일 경우 남은 공간에 0을 채우지않는다.

| 플래그 | 설명 |
|---|---|
| - | 왼쪽 정렬 |
| + | 양수일 때는 + 부호, 음수일 때는 - 부호 출력 |
| 공백 | 양수일 때는 부호를 출력하지 않고 공백으로 표시, 음수 일 때는 - 부호 출력 |
| # | 진법에 맞게 숫자 앞에 0, 0x, 0X를 붙임 |
| 0 | 출력하는 폭의 남는 공간에 0으로 채움 |

- width
  - 출력할 값의 최소 너비를 지정.
    - 양수만 저장.
    - 음수 '-'는'-' option으로 인식.
    - 0 : 0 option으로 인식.
  - width < 출력할 값 : 그대로 출력.

    - width > 출력할 값 : 자릿수를 맞추기위해 공백 또는 0을 채워넣는다.
        - 0은 왼쪽에만 붙는다(오른쪽 정렬)
    - 변환될 데이터가 어떤 길이를 갖는지 모를 경우에 * 옵션을 지정할 수 있다.
        - printf 함수에 전달된 매개 변수를 이용하기 때문에, 변환할 매개 변수 앞쪽에 별도로 매개 변수를 전달해야 한다.
        - 길이를 마음대로 지정할 수 있다.
        - 이 옵션을 사용할 경우 인자보다 매개 변수를 먼저 입력해야 한다.

- precision
    - 출력할 값의 정확도를 위한 최대 자릿수를 설정.
    - 출력할 값이 정수라면 최대 자릿수를 맞추기 위해 0을 추가한다.
    - 자릿수가 출력할 값보다 작을 경우에는 값을 자르거나 하지않고 값을 모두 출력한다.
    - 출력할 값이 실수라면 소수점 이하의 최대 자릿수를 가리키고, 자릿수에 따라 반올림이 되거나 소수점 이하에 0이 추가된다.
        - 실수 전체의 자릿수 설정은 width 옵션을 사용한다.

| 정밀도 | 설명 |
| --- | --- |
| .숫자 | 지정한 숫자만큼 소수점 아래 자리 출력 |

- type
    - 숫자, 문자, 문자열 중 하나.
    - 숫자는 정수와 실수, 정수는 부호 있는 정수와 부호 없는 정수로 나누어진다.
    - 정수는 8진수, 10진수, 16진수 형태로 출력할 수 있고, 실수는 일반적인 표기법 이외에 지수 표기법으로 출력할 수 있다.

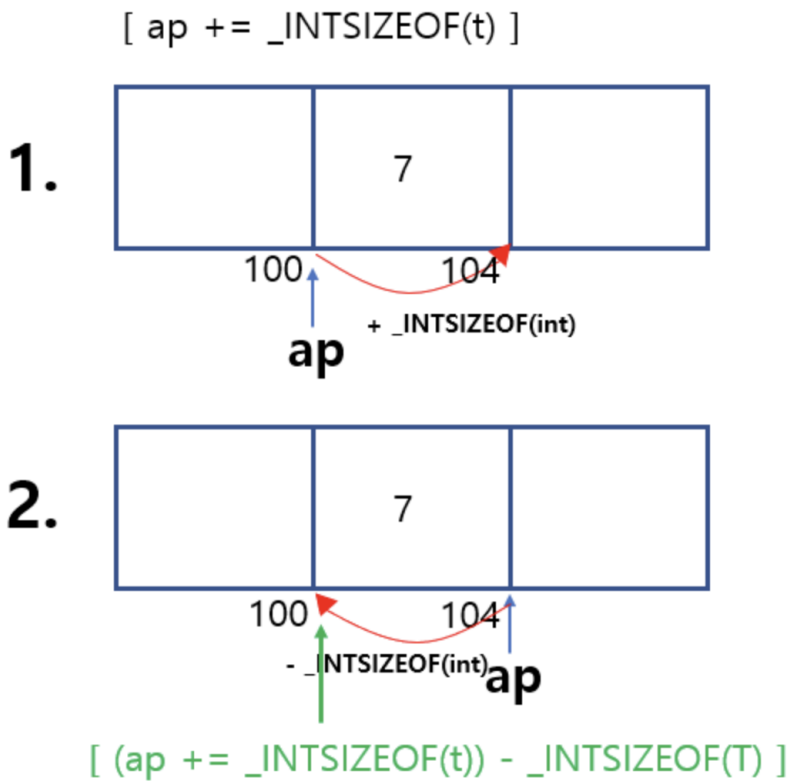
| 서식 지정자 | 설명 |
| --- | --- |
| d, i | 부호 있는 10진 정수 |
| u | 부호 없는 10진 정수 |
| o | 부호 없는 8진 정수 |
| x | 부호 없는 16진 정수(소문자) |
| X | 부호 없는 16진 정수(대문자) |
| f | 실수를 소수점으로 표기(소문자) |
| F | 실수를 소수점으로 표기(대문자) |
| e | 실수 지수 표기법 사용(소문자) |
| E | 실수 지수 표기법 사용(대문자) |
| g | %f와 %e 중에서 짧은 것을 사용(소문자) |
| G | %F와 %E 중에서 짧은 것을 사용(대문자) |
| a | 실수를 16진법으로 표기(소문자) |
| A | 실수를 16진법으로 표기(대문자) |
| c | 문자 |
| s | 문자열 |
| p | 포인터의 메모리 주소 |
| n | %n 부분에 int 포인터를 넣으면 지금까지 출력한 문자 개수를 변수에 넣어줌<br>`int num1;`<br>`_set_printf_count_output(1); // Visual Studio에서 %n 활성화`<br>`printf("abcde%n\n", &num1); // num1의 주소(포인터)를 지정.`<br>`// 지금까지 출력한 문자 개수를 num1에 넣어줌`<br>`printf("%d\n", num1); // 5: abcde는 5글자이므로 5가 들어가 있음` |
| % | % 기호 출력 |

- extend
  - extend type : l과 h에 관해서 구현.

| | 서식 지정자 | | | | | | |
|---|---|---|---|---|---|---|---|
| 길이 | d i | o u x X | f F e g G a A | c | s | p | n |
| | int | unsigned int | float, double | int | char * | void * | int * |
| hh | signed char | unsigned char | | | | | signed char * |
| h | short int | unsigned short int | | | | | short int * |
| l | long int | unsigned long int | double | wint_t | wchar_t * | | long int * |
| ll | long long int | unsigned long long int | | | | | long long int * |
| j | intmax_t | uintmax_t | | | | | intmax_t * |
| z | size_t | size_t | | | | | size_t * |
| t | ptrdiff_t | ptrdiff_t | | | | | ptrdiff_t * |
| L | | | long double | | | | |

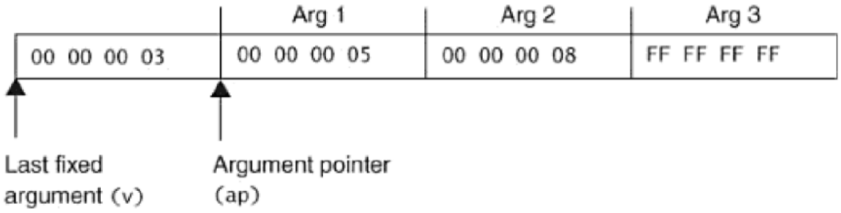
- 가변인자
  - ap(argument point) → 현재 인자를 저장하고 있는 포인터의 위치를 알려준다.
  - ...으로 들어온 가변 인자를 사용하려면 stdarg.h 헤더 파일에 정의된 매크로를 이용해야 합니다.
  - 따라서 #include로 stdarg.h 헤더 파일을 포함해줍니다.
  - stdarg.h에 정의된 가변 인자 처리 매크로는 다음과 같습니다.
    - va_list: 가변 인자 목록.
      - 가변 인자의 메모리 주소를 저장하는 포인터.
      - va_list라는 타입이 이 가변공간이 될 것이다. **va_list는 가변 인수들에 대한 정보를 홀드**하기 위한 타입.
      - 이 타입은 저장된 인수들을 쓰기위해 cstdsrg header파일에 정의된 매크로에서 매개변수로 사용된다.
    - va_start: 가변 인자를 가져올 수 있도록 포인터를 설정합니다.

```
#define va_start(ap, v) ( (ap) = (va_list)_ADDRESSOF(v) + _INTSIZEEOF
(v)//va_list 주소 값에다가 고정인수 크기를 더한 위치로 ap를 초기화한다.
```




- 두 개의 인자를 받는다. va_list 인스턴스(ap)와 고정인수를 받는다.
  - va_start는 va_list를 초기화하는 역할을 하며, va_list 타입이 선언되어 이전에 인스턴스가 필요하다.
  - 두 번째 인자인 고정인수는 첫 번째 가변인자 주소를 알기위해서 필요하다.
  - ap는 가변인수 함수에 들어오는 고정인수 다음부터 위치되어야 하기 때문에 ap를 초기화시켜준다.
- va_arg: 가변 인자 포인터에서 특정 자료형 크기만큼 값을 가져옵니다.

```
#define va_arg(ap, t) (*(t*)((ap += INTSIZEOF(t)) - _INTSIZEEOF(t)))//ap
 += INTSIZEOF(t)에서 ap 주소값이 t만큼 증가한다.//1. ap를 다음 가변인자 위치로 보낸
다.//2. ap를 증가시킨 후 원래 번지로 재이동한다.
```



    - ap 포인터가 위치한 부분의 데이터를 읽어온다. ap 주소값은 t사이즈만큼 증가한다.
        - ap 사이즈는 증가한 위치에 그대로 있지만, 실제 출력하는 부분은 기존 부분이 된다.
        - ap는 104번지로 이동했지만 실제 출력하는 데이터는 100번지에 있는 값이다.
        - 값을 호출한 후 다음 또 호출을 했을 때 104번지 값을 호출하고 ap는 다시 t만큼 증가한다.
    - ap 포인터의 위치를 증가시키고 그 전의 주소값에서 값을 읽어오는 이유는 **타입마다 사이즈가 다르기 때문에 타입별로 ap 포인터를 옮기기 위해서** 이다.
    - 읽어올 수 있는 최소 크기가 INTSIZE → int 보다 작은 사이즈는 모두 int로 받아야 한다.
- va_end: 가변 인자 처리가 끝났을 때 포인터를 NULL로 초기화합니다.

```
#define va_end(ap) (ap = (va_list)0)
```

    - va_list 타입을 null로 초기화한다.

- 출처
    - https://stdbc.tistory.com/m/59