프로그래밍 기초

4장. 전처리와 입출력

학습목표

- **전처리기와 전처리 지시자에 대하여 이해하고 설명할 수 있다.**
  - 전처리기 역할
  - 전처리 지시자 #define, #include

- **함수 printf()를 이용한 출력을 이해하고 프로그래밍 가능하다.**
  - 여러 자료형에 따른 형식지정 방식
  - 형식 제어문자 및 다양한 출력 방식
  - 출력 폭 지정과 다양한 옵션 1, –, #, 0의 사용

- **함수 scanf()를 이용한 입력을 이해하고 프로그래밍 가능하다.**
  - 여러 자료형에 따른 형식지정 방식
  - 형식 제어문자 및 다양한 입력 방식

전처리(Preprocessing) 개요

- **전처리**
  - 컴파일(compile) 이전에 소스 코드를 변환 및 수정하는 초기 단계 작업
    - 전처리 지시자(#으로 시작)를 처리하는 과정

- **전처리기의 역할**
  - 헤더 파일(예: #include <stdio.h>)을 소스 코드에 추가
  - 매크로 정의(#define BUF_SIZE 1024)를 상수로 대체
  - 조건(부) 컴파일(#ifdef, #else, #elif, #endif 등)
  - 전처리 결과인 전처리 출력 파일을 만들어 컴파일러에게 보내는 작업을 수행

- **전처리 지시자(preprocess directives)**
  - #include <헤더파일>
    - 대표적인 헤더파일인 stdio.h
      - printf(), scanf(), putchar() getchar() 등과 같은 입출력 함수의 정보가 정의
  - #define MAX 1000

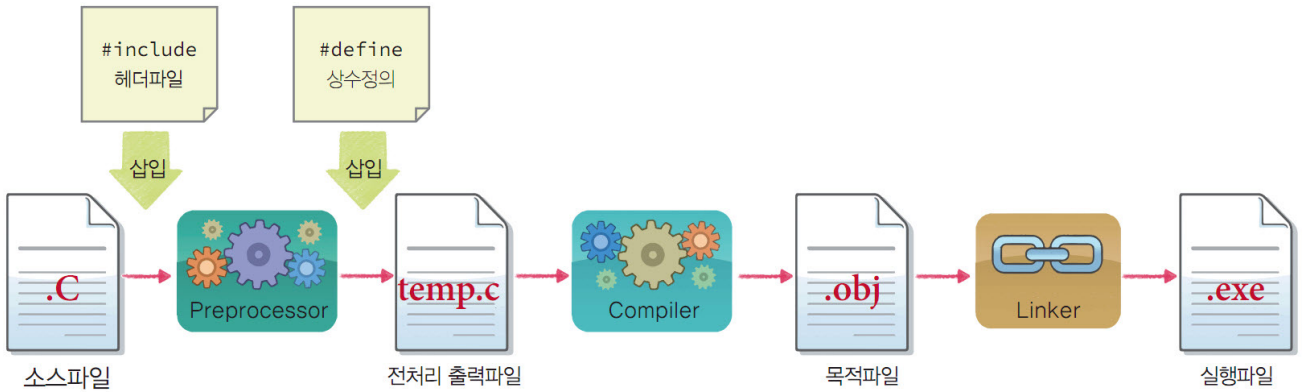


그림 4-1 전처리기와 컴파일러

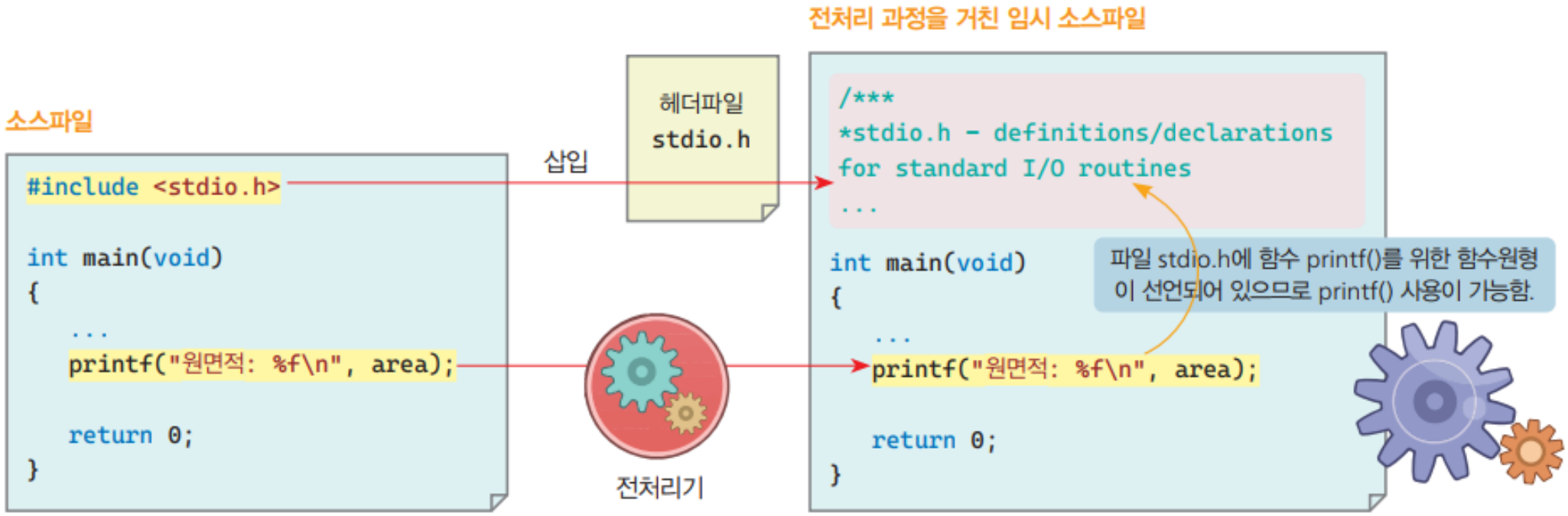
전처리 지시자 #include와 헤더 파일

- **주요 헤더파일**

헤더 파일	파일 내용
stdio.h	• 표준 입출력 함수와 상수 정의 (STanDard Input Output) • printf(), scanf(), getchar(), putchar() 등
stdlib.h	• 주요 메모리 할당 함수, 난수 생성 함수와 상수 정의 (STanDard LIBrary) • malloc()/free(), rand(), atoi(), exit() 등
string.h	• 문자열 관련 함수와 상수 정의 • stcpy(), strcat(), strcmp(), strlen() 등
math.h	• 수학 관련 함수와 상수 정의
time.h	• 시간 관련 함수와 상수 정의 • time(), clock(), localtime(), ctime() 등
ctype.h	• 문자 관련 함수와 상수 정의 • isdigit(), isalpha(), islower(), isupper(), tolower(), toupper() 등
limits.h	• 정수, 상수 등 여러 상수 정의
float.h	• 부동 소수에 관련된 각종 상수 정의

전처리 지시자 #include와 헤더 파일

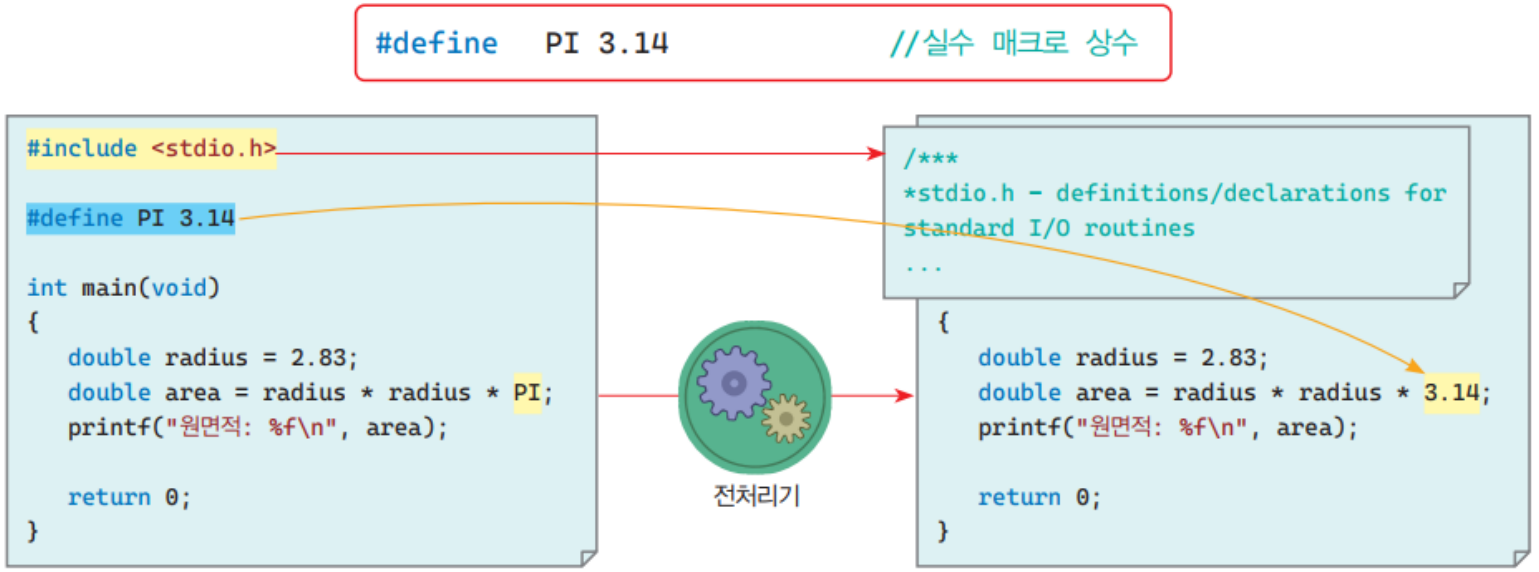
- **전처리 지시자(#include) 처리 과정**
  - 명시된 헤더 파일(stdio.h)의 내용을 선언된 위치에 추가하는 역할 수행
  - 소스 코드에서 printf() 함수 사용 가능



전처리 지시자 #define

- **매크로 상수(macro constant)**
  - #define은 매크로 상수를 정의하는 지시자
  - #define에 의한 심볼릭 상수는 주로 대문자 이름으로 정의

- **#define identifier_name [value]**
  - #define에 정의된 identifier_name은 전처리기(preprocessor)에 의해 모두 value로 대체(replace)되어 컴파일
  - 문자열 내부 또는 주석 부분에서는 대체(replace)되지 않음



#define에 의한 매크로 상수

- 01macroconst.c

```c
#include <stdio.h>

#define PI 3.14 //실수 매크로 상수

int main(void)
{
    double radius = 2.83;

    printf("원 면적: %f\n", radius * radius * PI); //PI가 3.14로 대체
    printf("원 둘레: %f\n", 2 * radius * PI); //PI가 3.14로 대체

    return 0;
}
```

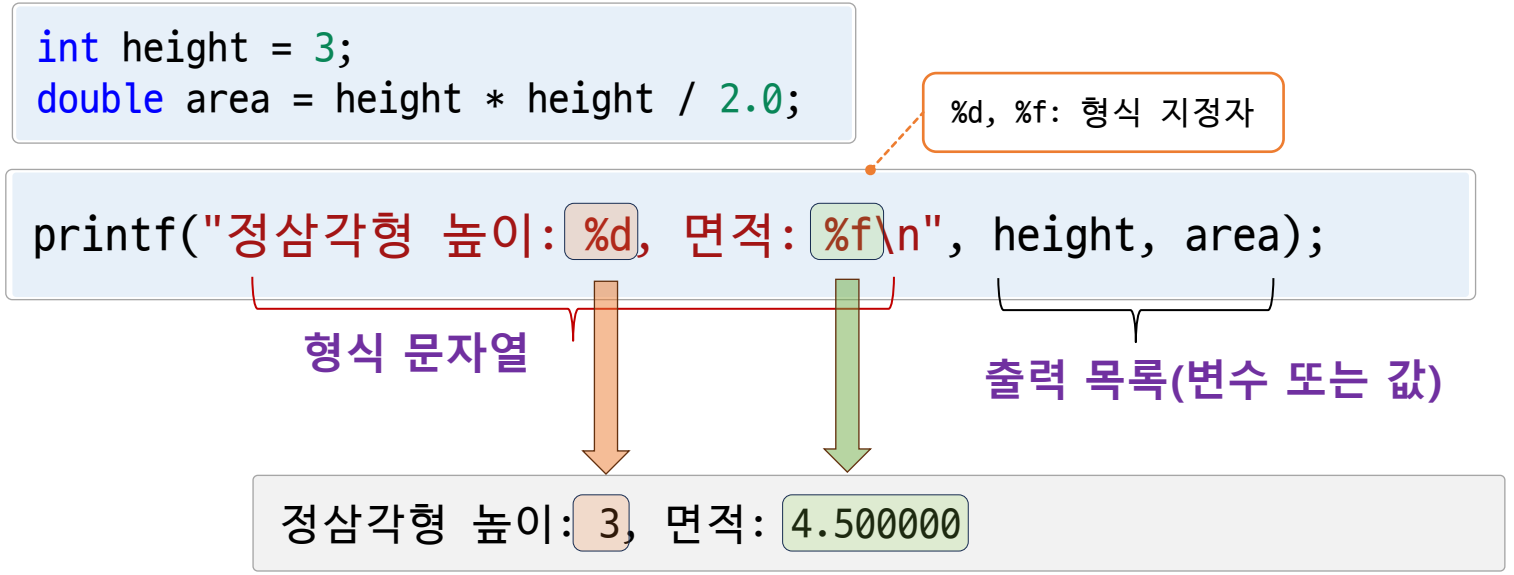
실행 결과

```
원 면적: 25.147946
원 둘레: 17.772400
```

출력 함수 printf() 개요

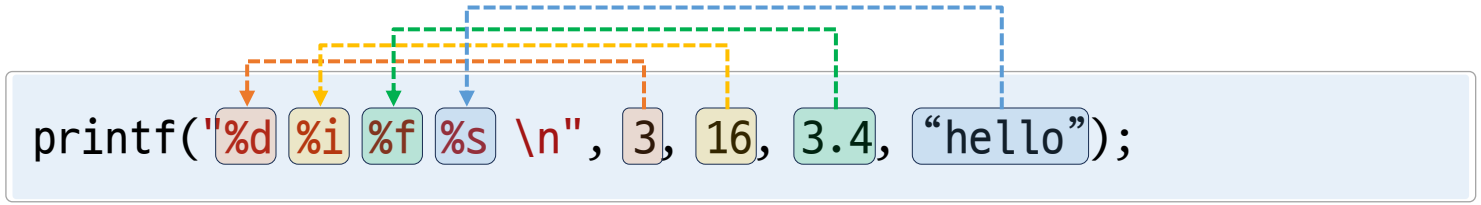
- **함수 printf("형식문자열", 출력변수(값), 출력변수(값),…)**
  - 형식 문자열과 출력할 목록으로 구성
  - 표준 출력인 명령 프롬프트 콘솔(console)에 다양한 변수와 상수를 출력하는 함수

- **첫 번째 인자인 형식 문자열(format string)**
  - 일반 문자와 이스케이프 문자 그리고 형식 지정자(format specification)로 구성
    - %로 시작하는 형식지정자(%d, %f, %s 등)는 출력하려는 값의 위치에 배치



출력 함수 printf() 다양한 형식 지정자

- **형식 지정자는 출력 내용의 자료형에 따라 %d, %i, %c, %s와 같이 %로 시작**
  - 형식 지정자 형식에 맞게 콘솔로 출력할 값이 표시
  - 형식 지정자는 출력 값의 목록과 순서대로 서로 일치해야 함



- printf() 형식 지정자 종류

형식 지정자	출력 양식
%d, %i, %lld	• 정수 출력, %lld는 long long 자료형 출력에 사용 • %d, %i는 10진수 정수 출력에 사용(동일 기능)
%f, %lf	• 실수 출력
%c	• 문자 출력 (character: 글자 하나)
%s	• 문자열 출력 (string: 여러 문자)
%e, %E	• 지수 형식 출력(1234.567는 %e -> 1.234567e+03, %E: 1.234567E+03)
%g, %G	• 실수를 소수점(%f) 또는 지수(%e, %E) 형식 중 짧은 표현을 자동으로 선택

출력 함수 printf() 다양한 옵션 지정자

- 옵션 지정자

문자	기본(생략)	의미	사용 예
-	좌측 정렬	• 숫자는 지정된 폭에서 좌측 정렬 • '-'생략되면, 기본 우측 정렬	• %-10d: 좌측 정렬 • %10d: 우측 정렬
+	부호 표시	• 숫자 값에 부호 +, - 를 표시 • '+' 생략되면, 음수에만 '-' 표시됨	printf("%+d, %+d\n", 10, -10); -> +10, -10 출력
0	0으로 채움	• 우측 정렬인 경우, 숫자 앞에 0으로 채움	printf("%03d", 5); -> 005 출력
#	0, 0x 추가	• 8진수(%#o): 숫자 앞에 0 자동 추가 • 16진수(%#x): 숫자 앞에 0x가 자동 추가	printf("%#o, %#x\n", 10, 10); -> 012, 0xa 출력

형식 지정자의 출력 폭 지정

- **출력 값의 가로 길이인 폭(width) 지정이 가능**
  - %10d와 같이 %와 d 사이에 폭 길이인 10을 지정
  - 기본 오른쪽 정렬
    - ➢ 출력 폭을 잘못 지정하여 출력 내용보다 좁으면 폭을 무시하고 원래대로 출력

- **형식지정자 %8d는 10진수를 8자리 폭에 출력**한다. **지정한 출력 폭이 출력할 내용보다 넓으면 정렬은 기본적으로 오른쪽**이다. 다음은 정수 7629를 %8d로 출력한 모습이다.


```
printf("%8d\n", 7629);
```

- 형식지정자 %10f는 소수점을 포함한 전체 폭은 10이고 소수점 이하 자릿수는 기본으로 6자릿수로 출력한다.


```
printf("%10f\n", 32.369);
```

- 형식지정자가 %10.3f이면 소수점을 포함하여 전체 폭은 10, 그 중에서 3은 소수점 이하 자리수로 출력하며, 다음은 정수 32.369를 %10.3f로 출력한 모습으로, 소수점도 전체 폭 10에 해당한다.


```
printf("%10.3f\n", 32.369);
```

- 형식지정자 %10s로 문자열 "hello"를 출력해도 오른쪽 정렬로 다음과 같이 출력된다.


```
printf("%10s\n", "Hello");
```

printf에서의 다양한 형식 지정자

- 다양한 형식 지정자: 02printfbasic.c

```c
#include <stdio.h>

int main(void)
{
    double width = 3.424, height = 2.718;
    int shape = 3; //삼각형 또는 사각형

    printf("가로: %f, 세로: %f\n", width, height);
    printf("%d각형 %s: %8.2f\n", shape, "면적", (width * height) / 2);
    printf("%d각형 %s: %10.4f\n", shape + 1, "면적", width * height);

    return 0;
}
```

실행 결과

```
가로: 3.424000, 세로: 2.718000
3각형 면적:     4.65
4각형 면적:     9.3064
```

정수의 8진수 16진수 출력

- **정수를 각각 8진수(%o)와 16진수(%x)로 출력**
  - 020과 0x1a처럼 0과 0x 등을 표기하려면
    - #을 중간에 삽입해 %#o와 %#x을 사용
    - 0%o 또는 0x%x

<03printoctheX.c>

```c
#include <stdio.h>

int main(void)
{
    printf("%3o %3d %3x\n", 10, 10, 10); //10을 8, 10, 16진수로 각각 출력
    printf("%#3o %3d %#3x\n", 12, 12, 12);
    printf("%3o %3i %3X\n", 14, 14, 14);

    return 0;
}
```

%i는 %d와 동일

실행 결과

```
 12  10   a
014  12 0xc
 16  14   E
```

상세 출력 폭과 정렬 방법

- **정렬**
  - 기본 정렬: 오른쪽
  - 왼쪽 정렬: 폭 앞에 minus 기호('-')를 붙여 %-8d로 지정

- **부동소수에서 소수점 이하 자릿수를 지정**
  - "%[전체폭].[소수점이하폭]f" 와 같이 표시
  - 전체폭 생략 가능: %.3f 등

- **형식지정자 %8d는 10진수를 8자리 폭에 출력**한다. **지정한 출력 폭이 출력할 내용보다 넓으면 정렬은 기본적으로 오른쪽**이다. 다음은 정수 7629를 %8d로 출력한 모습이다.


```
printf("%8d\n", 7629);
```

- **출력 폭을 지정하며 정렬을 왼쪽**으로 지정하려면 %-8d처럼 폭 앞에 -를 삽입한다.


```
printf("%-8d\n", 7629);
```

- 형식지정자가 %10.3f이면 소수점을 포함하여 전체 폭은 10, 그 중에서 3은 소수점 이하 자리수로 출력하며, 다음은 정수 32.369를 %10.3f로 출력한 모습으로, 소수점도 전체 폭 10에 해당한다.


```
printf("%10.3f\n", 32.369);
```

- 또한 %10f는 전체 폭은 10이고 소수점 이하 자릿수는 기본으로 6자릿수로 출력한다.


```
printf("%10f\n", 32.369);
```

printf()에서 정수와 실수의 다양한 출력

- 05printfnote.c

```c
#include <stdio.h>

int main(void)
{
    printf("%d * %4d = %#5o\n", 2, 2, 2 * 2);
    printf("%d * %04d = %#5o\n", 2, 3, 2 * 3);
    printf("%d * %+04d = %-#5x\n", 2, 4, 2 * 4); // +: 양수에 + 기호, -: 왼쪽 정렬
    printf("%d * %+-4d = %#5X\n", 2, 5, 2 * 5); // 16진수 문자를 대문자로 출력

    printf("%15.3f\n", 123456.789);
    printf("%e\n", 123456.789);
    printf("%g\n", 12.34e-5); // 숫자 크기에 따라 %f, %e로 출력
    printf("%G\n", 12.34e-6);

    return 0;
}
```

실행 결과

```
2 *    2 =    04
2 * 0003 =    06
2 * +004 = 0x8
2 * +5   =   0XA
      123456.789
1.234568e+05
0.0001234
1.234E-05
```

문자열 출력에서의 출력폭 지정

- **문자열 출력(%s): %[-][전체폭].[출력할 문자수]s**
  - 형식지정자 %10.3s는 전체폭이 10, 출력할 문자 수는 3개이므로 Hel까지 출력되며, 우측정렬이 기본이다.


```
printf("%10.3s\n", "Hello!");
```

  - 만약 좌측정렬을 하려면 %-10.3s를 하면 된다.


```
printf("%-10.3s\n", "Hello!");
```

  - 형식지정자에서 주의할 것은 지정한 전체폭이 출력할 문자열의 수보다 작으면 무시하고, 원래 문자열의 폭만큼 모두 출력한다는 것이다.


```
printf("%4s\n", "Hello!");
```

  - 또한 형식지정자의 전체폭과 정밀도는 형식지정자에 *를 이용한 후, 그에 대응하는 정수를 목록 값으로 지정할 수 있다. 다음은 *이 5으로 대체되어 %10.5s로 문자열 "Hello"를 출력한다.


```
printf("%10.*s\n", 5, "Hello!");
```

LAB 정수와 실수, 문자와 문자열의 출력

- 개인의 성별과 이름, 나이, 성적 등 개인 정보를 출력하는 프로그램

Lab 4-1 lab1basicoutput.c 난이도: ★

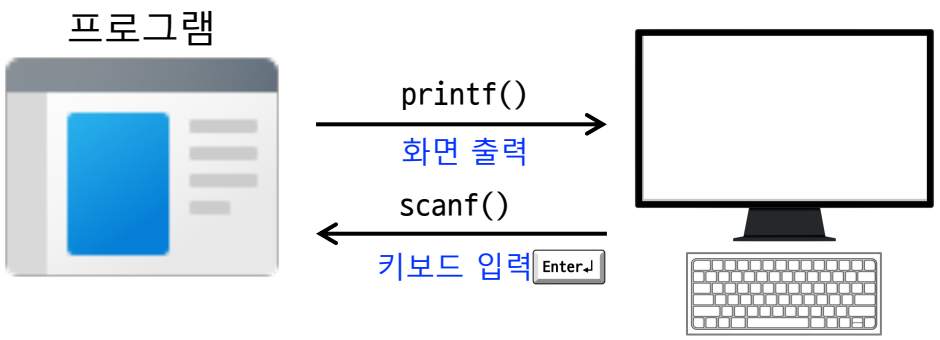
```c
#include <stdio.h>

int main(void)
{
    int age = 20;
    double gpa = 3.88;
    char gender = 'M';
    float weight = 62.489F;

    printf("성별: %c\n", gender);
    printf("이름: \n", "안 병훈");
    printf("나이: %d\n", age);
    printf("몸무게: %.2f\n", weight);
    printf("평균평점(GPA): \n", gpa);

    return 0;
}
```

정답

```c
printf("이름: %s\n", "안 병훈");
printf("평균평점(GPA): %.3f\n", gpa);
```

화면 입력 함수: scanf()

- **scanf() 함수**
  - 키보드로부터 입력된 데이터를 지정된 형식으로 변환하여 변수에 저장하는 함수
  - %d, %c, %f, %llf 등의 형식 지정자를 사용
  - 반드시 변수 앞에 주소를 의미하는 &(주소 연산자)을 붙여 '&변수이름'으로 사용
    - ➢ 변수의 주소를 전달하면, 해당 변수에 값이 저장됨
  - Enter키를 누르기 전까지 실행을 멈춰 사용자의 입력을 기다림

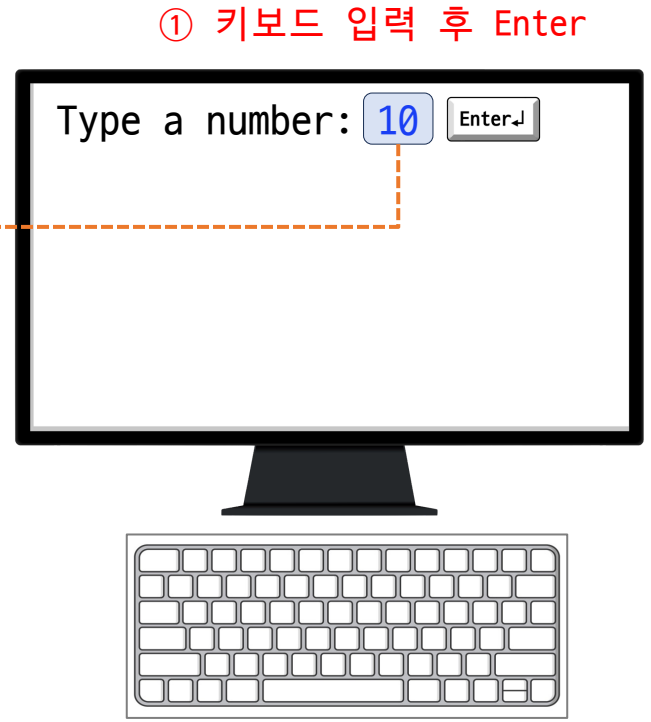



```c
int point = 0;
float value = 0.0;
double data = 0.0;

scanf("%d %f %llf", &point, &value, &data);
```

형식 문자열 입력 변수 목록

간단한 scanf() 예제 (실습)

```c
#include <stdio.h>

int main()
{
    int num = 0;
    printf("Type a number: ");
    scanf("%d", &num);

    printf("num: %d\n", num);
    return 0;
}
```



실행 결과

```
Type a number: 10
num: 10
```

scanf() 활용: 실습

- 06scanf.c

Visual Studio에서는 scanf() 사용시 추가해야 됨

```c
//#define _CRT_SECURE_NO_WARNINGS //scanf() 오류를 방지하기 위한 상수 정의

#include <stdio.h>

int main(void)
{
    int month = 0;
    printf("1년은 몇 달? ");
    scanf("%d", &month);
    printf("1년은 %d달\n\n", month);

    int snum, credit;
    printf("당신의 학번과 신청 학점은? ");
    scanf("%d %d", &snum, &credit);
    printf("학번: %d 신청학점: %d\n", snum, credit);

    return 0;
}
```

입력된 값은 공백을 기준으로 분리됨

실행 결과

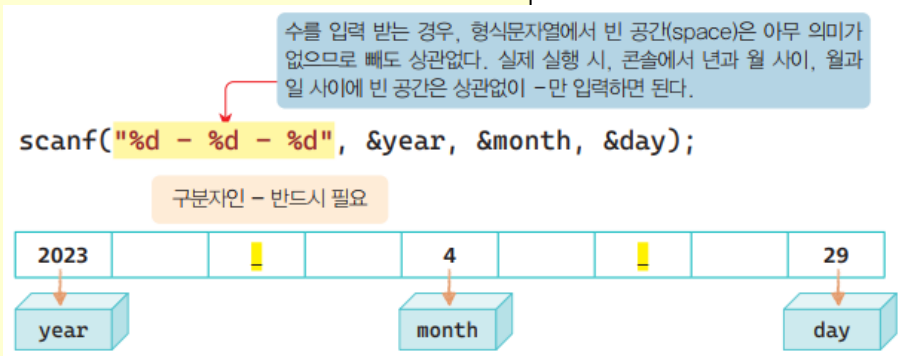
```
1년은 몇 달? 12 Enter↵
1년은 12달

당신의 학번과 신청 학점은? 2025 21 Enter↵
학번: 2025 신청학점: 21
```

공백으로 구분

여러 값 입력에서 구분자 문자 사용 (실습)

- **년, 월, 일을 2025-4-3과 같이 중간에 '-'를 넣어 입력 받음: 구분자 추가**
  - 함수 scanf("%d - %d - %d", …) 처럼 형식문자열에 입력 형식을 명시

```c
#define _CRT_SECURE_NO_WARNINGS //scanf() 오류를 방지하기 위한 상수 정의

#include <stdio.h>

int main(void)
{
    int year, month, day;
    printf("당신의 생년월일은? ");
    scanf("%d - %d - %d", &year, &month, &day);
    printf("생년월일: %d %d %d\n", year, month, day);

    return 0;
}
```



실행 결과 #1

구분자('-')를 기준으로 값이 저장

```
당신의 생년월일은? 2005-4-3 Enter↵
생년월일: 2005 4 3
```

실행 결과 #2 (잘못된 입력)

구분자('-')를 입력해야 됨

```
당신의 생년월일은? 2005 4 3 Enter↵
생년월일: 2005 1 44023808
```

실수와 다양한 자료형의 입력: 08floatscan.c (실습)

- **제어문자 %f와 %lf, %c**
  - printf()에서 실수의 출력을 위한 형식 지정자로 %f와 %lf를 모두 사용
- **입력 scanf()**
  - 저장될 자료형이 float이면 %f
  - double이면 %lf로 구분해 사용

<08floatscan.c>

```c
#include <stdio.h>

int main(void)
{
    float mile = 0;
    printf("1 킬로미터(km)는 몇 마일(mile)? "); //0.621 mile
    scanf("%f", &mile);
    printf("80 킬로미터: %.2f 마일\n\n", mile * 80);

    double liter = 0;
    printf("1 갤론(gallon)은 몇 리터(liter)? "); //3.785 liter
    scanf("%lf", &liter);
    printf("18 갤론: %.2f 리터\n", liter * 18);

    return 0;
}
```

실행 결과

```
1 킬로미터(km)는 몇 마일(mile)? 0.621 Enter↵
80 킬로미터: 49.68 마일

1 갤론(gallon)은 몇 리터(liter)? 3.785 Enter↵
18 갤론: 68.13 리터
```

scanf() 다양한 형식 지정자

- scanf()의 형식 지정자

형식 지정자	화면 입력 값의 형태	입력 변수 인자 유형
%d	• 10진수 정수로 인식	int num; scanf("%d", &num);
%i	• 10진수로 인식 • 입력 값에 0이 붙으면 8진수로 인식, 0x가 붙으면 16진수로 인식	int num; scanf("%d", &num);
%u	• unsigned int로 인식	unsinged int num; scanf("%u", &num);
%o	• 8진수로 인식	int num; scanf("%o", &num);
%x, %X	• 16진수로 인식	int num; scanf("%x", &num);
%f	• float형 실수로 인식	float value; scanf("%f", &value);
%lf	• double 형 실수로 인식	double value; scanf("%lf", &value);
%e, %E	• 지수 형태의 float형 변수로 인식 (double형 지수: %le)	float value; scanf("%e", &value);
%c	• 문자로 인식 (char 변수에 값 저장)	char ch; scanf("%c", &ch);
%s	• 문자열(string)로 인식	char str[10]; scanf("%s", str);
%p	• 주소(address)값으로 인식 (unsigned int 변수에 값 저장)	unsigned int ptr; scanf("%p", &ptr);

scanf()로 16진수를 입력 받아 8, 10, 16진수로 출력

- 09scanfhex.c

```c
#include <stdio.h>

int main(void)
{
    int hex;
    printf("십진수 정수(1A 등)를 입력하세요 >> ");
    scanf("%x", &hex);
    printf("%o %d %x\n\n", hex, hex, hex);

    printf("십진수 정수(0리딩 표시방식으로 0x1a 등)를 입력하세요 >> ");
    scanf("%i", &hex);
    printf("%#o %d %#x\n\n", hex, hex, hex);

    return 0;
}
```

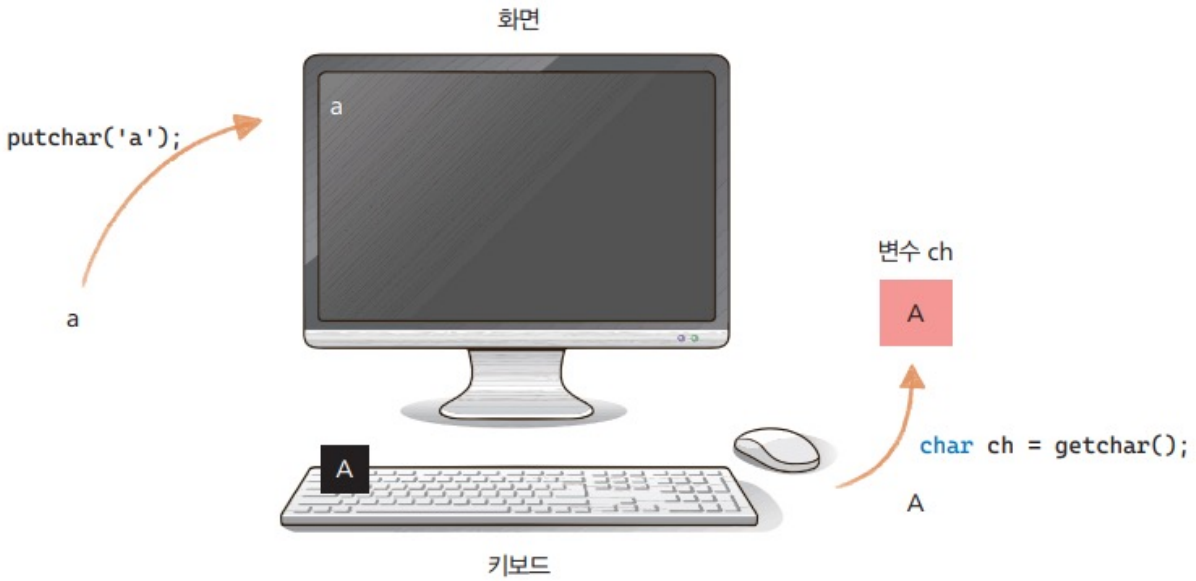
%i
- 입력 값이 03과 같이 0으로 시작하는 수는 8진수로 인식
- 0x1B와 같이 0x로 시작하는 수는 16진수로 인식

실행 결과

```
십진수 정수(1A 등)를 입력하세요 >> 1b
33 27 1b

십진수 정수(0리딩 표시방식으로 0x1a 등)를 입력하세요 >> 0x1B
033 27 0x1b
```

문자 입력과 출력 함수 getchar()와 putchar()

- **함수 getchar()**
  - 문자 하나를 화면에서 입력 받는 함수
- **함수 putchar()**
  - 문자 하나를 화면 출력하는 함수
  - 헤더파일 stdio.h 가 필요




```c
#include <stdio.h>

int main(void)
{
    char a = '\0' ;

    puts("getchar 문자 하나 입력 >> ");
    a = getchar();

    putchar(a);
    return 0;
}
```

```
getchar 문자 하나 입력 >>
a
a%
```

문자 입력과 출력 함수 getchar()와 putchar()

- 10inputgrade.c

```c
#include <stdio.h>

int main(void)
{
    char abc, plus;

    printf("C 프로그래밍 언어의 원하는 학점(A+, A0처럼)을 입력 >> ");
    //문자 두 개를 연이어서 입력 받음
    abc = getchar();
    scanf("%c", &plus);

    printf("학점 %c%c\n", abc, plus);

    return 0;
}
```

실행 결과

```
C 프로그래밍 언어의 원하는 학점(A+, A0처럼)을 입력 >>  A0
학점 A0
```

Lab. 8진수 정수를 입력 받아 8진수와 10진수, 16진수로 출력

- **8진수 입력을 위한 형식지정자 %o**
  - 출력은 두 줄로 각각 8진수와 10진수, 16진수로 출력
  - 출력 두번째 줄은 8진수와 16진수는 0이나 0X를 표시

Lab 4-2 | lab2octinput.c | 난이도: ★★

```
01  #define _CRT_SECURE_NO_WARNINGS //scanf() 오류를 방지하기 위한 상수 정의
02
03  #include <stdio.h>
04
05  int main(void)
06  {
07      int oct;
08      printf("8진수 정수 입력 >> ");
09      scanf("  ", &oct);
10
11      //입력 받은 정수를 각각 8, 10, 16진수로 출력
12      printf("%o %d %x\n", oct, oct, oct);
13      printf("                ", oct, oct, oct);
14
15      return 0;
16  }
17
```

정답

```
09  scanf("%o", &oct);
13  printf("%#o %i %#X\n", oct, oct, oct);
```

Questions?