2장. C 프로그램 기본

박 종 혁 교수
서울과학기술대학교 컴퓨터공학과
UCS Lab
Tel: 970-6702
Email: jhpark1@seoultech.ac.kr

목차

- C 프로그램의 구성 요소
  - 주석
  - main 함수
  - 출력
- C 언어의 입력과 출력
  - 변수
  - printf 함수
  - scanf 함수

예제 2-1 : 첫 번째 C 프로그램

```c
// ex02_01.c
#include <stdio.h>  // 입출력 라이브러리를 사용하기 위한 준비

int main(void)   // 진입점 함수
{
    printf("첫 번째 C 프로그램입니다.\n"); // 콘솔 출력

    return 0;    // 프로그램의 종료 코드 리턴
}
```

주석

실행결과

```
첫 번째 C 프로그램입니다.
```

주석(Comment)

❖주석을 다는 방법
- /*과 */을 이용한 여러 줄 주석
- //을 이용한 한 줄 주석

❖주석의 용도
- 프로그램에 대한 설명

```
#include <stdio.h>    // 입출력 라이브러리를 사용하기 위한 준비
```

- 프로그램 전체에 대한 대략적인 정보를 제공

```
/*
    프로그램명: ex02_01
    설명: 간단한 출력을 수행하는 C 프로그램
    작성 일시: 2019/1/1
    작성자: 천정아
*/
```

- 주석 처리(comment out)

```
printf("첫 번째 C 프로그램입니다.\n");
//printf("주석 처리된 문장입니다.\n");  // 주석 처리
```

주석 예제

- C 스타일 주석

```
/* a comment */

/*
 * A comment can be written in this fashion
 * to set it off from the surrounding code.
 */

/**************************
 * If you wish, you can   *
 * put comments in a box. *
 **************************/
```

- C++ 스타일 주석

```
// This is a comment in C++
```

어휘 원소, 연산자, C 시스템

❖**구문**
- 올바른 프로그램을 만들 수 있게 하는 규칙

❖**컴파일러**
- C 프로그램이 구문에 맞는지 검사
- 오류가 있다면, 오류 메시지 출력
- 오류가 없다면, 목적 코드 생성

❖**컴파일 과정**
- C 프로그램 → 토큰으로 분리 → 토큰을 목적 코드로 변환
- 토큰 종류 : 키워드, 식별자, 상수, 문자열 상수, 연산자, 구두점

문자와 어휘 원소

❖프로그램에서 사용할 수 있는 문자

- 소문자 : a b c ... z
- 대문자 : A B C ... Z
- 숫자 : 0 1 2 3 4 5 6 7 8 9
- 특수문자 : + - * / = ( ) [ ] < > ' "
- ! @ # $ % & _ | . , ; : ?
- 여백문자 : 공백, 개행, 탭

❖컴파일러는 이러한 문자들을 구문 단위인 토큰으로 모은다

어휘 분석

❖/* Read in two integers and print their sum. */
- 주석문 : /*부터 */까지는 공백으로 대치

❖#include<stdio.h>
- 전처리 지시자 : 전처리기가 처리

❖int main(void){

    int a, b, sum;
- 키워드 : int, void
- 식별자 : main, a, b, sum
- 연산자 : ( )
- 구두점 : “{”, “,”, “;”

inta, b, sum; -> (X)

int absum -->absum을 하나의 식별자

키워드

❖키워드

- C 언어에서 고유한 의미를 가지는 토큰
- 예약된 단어

❖C 키워드

auto	do	goto	signed	unsigned
break	double	if	sizeof	void
case	else	int	static	volatile
char	enum	long	struct	while
const	extern	register	switch	
continue	float	return	typedef	
default	for	short	union	

식별자

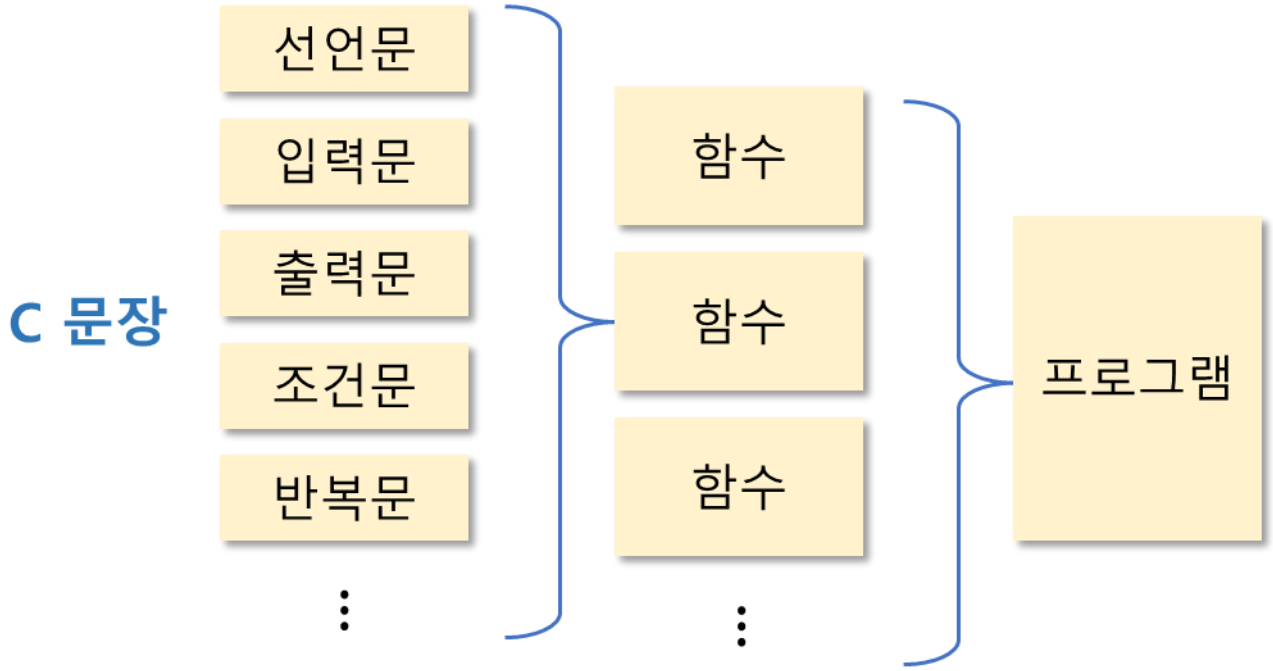
- 식별자는 문자, 숫자, 그리고 특수문자인 밑줄문자(_)로 구성된 토큰으로, 문자 또는 밑줄문자로 시작해야 함

- C 시스템은 소문자와 대문자를 구별함

- 식별자의 선택은 의미를 생각하여 함

문장(statement)

- C 프로그램을 이루는 기본 단위
- 각 문장은 세미콜론(;)으로 끝난다
- ~문: 선언문, 입력문, 출력문, 조건문, 반복문 등



콘솔 프로그램과 main 함수

- 콘솔 프로그램에는 반드시 main 함수가 필요하다

main 함수로만 구성된 프로그램

```
int main(void)
{
    return 0;
}
```

main 함수

여러 개의 함수로 구성된 프로그램

```
void print_line(void)
{
    printf("------\n");
}

int add(int a, int b)
{
    return a + b;
}

int main(void)
{
    return 0;
}
```

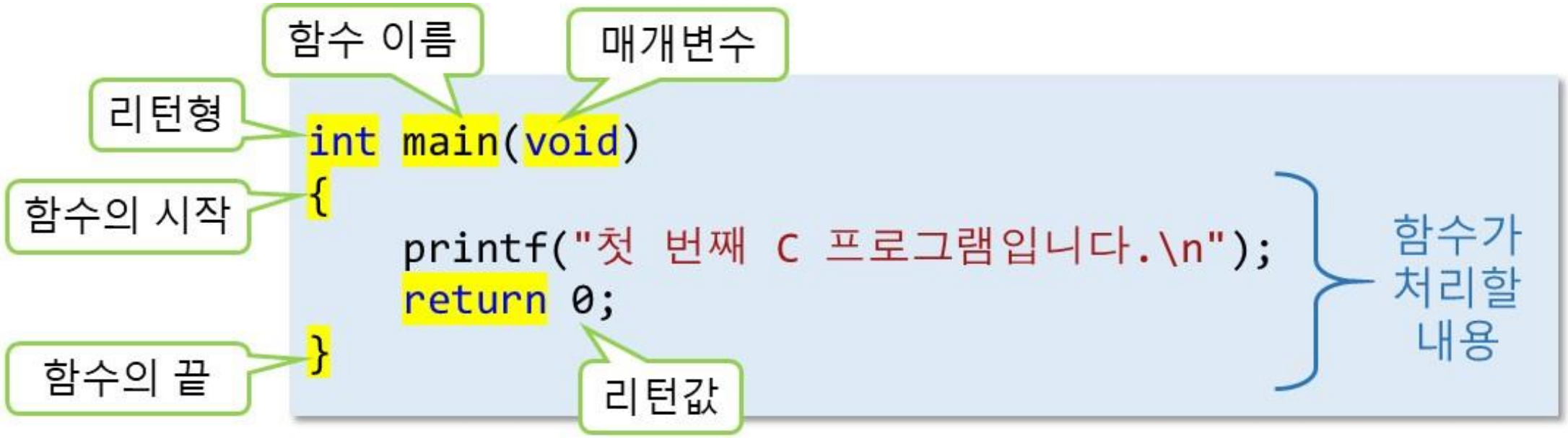
함수

함수

main 함수

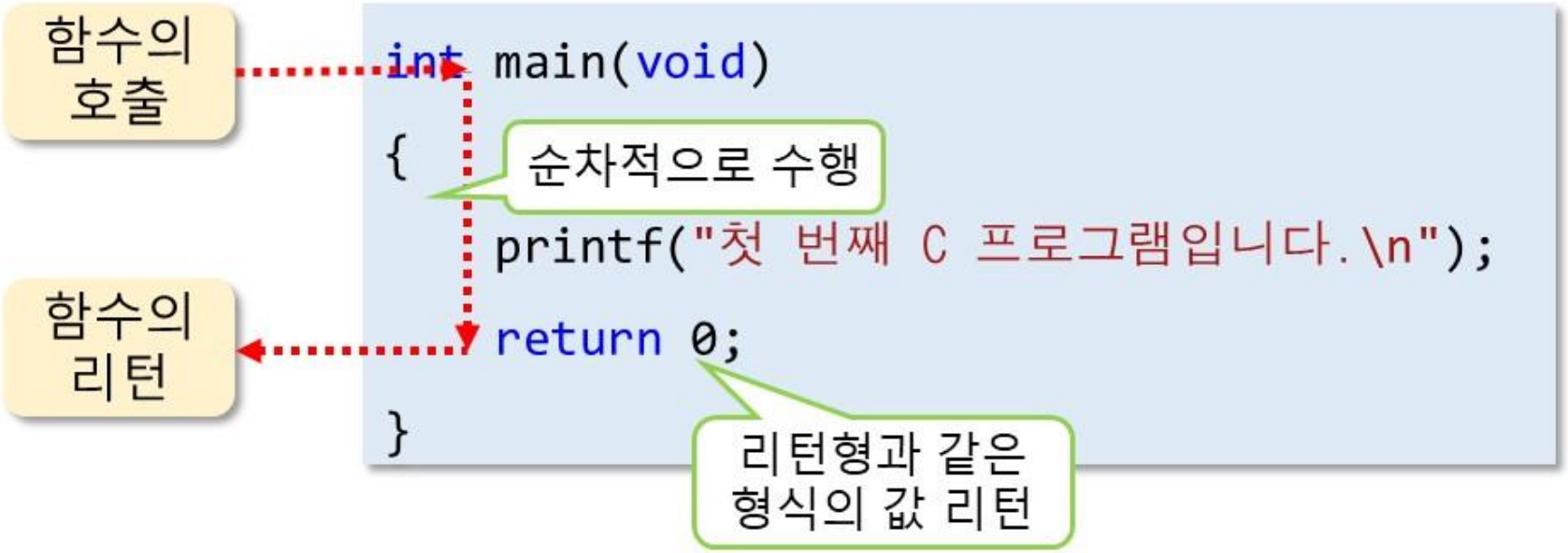
함수의 구성 요소

- 함수를 만들 때는 리턴형, 함수 이름, 매개변수가 필요하다



진입점(entry-point) 함수

- C 프로그램이 처음 시작될 때 호출되는 함수
- main 함수는 콘솔 프로그램의 진입점 함수이다



- C 프로그램에는 main 함수가 반드시 필요하다

main 함수의 리턴 값

❖프로그램의 **종료 코드(exit code)**를 리턴한다

- main 함수의 리턴 값은 운영체제로 전달
- 0이면, 정상 종료이고 0이 아니면 비정상 종료

```
int main()
{
    ...

    return 0;
}
```

정상 종료

```
int main()
{
    if (/*메모리 할당 실패*/)
        return 1;
    ...
}
```

비정상 종료

❖main 함수의 return문은 생략할 수 있다

- 프로그램 종료 시 0 리턴

들여쓰기

❖알아보기 쉽도록 한 줄에 한 문장씩 작성한다

❖**블록(block)** : { }로 묶인 문장들

```
int main()
{
    printf("Good\n");
    return 0;
}
```

들여 쓰기를 하면 알아보기 쉽다.

```
int main() {
printf("Not Good\n"
);
return 0; }
```

들여 쓰기를 안 하면 알아보기 어렵다.

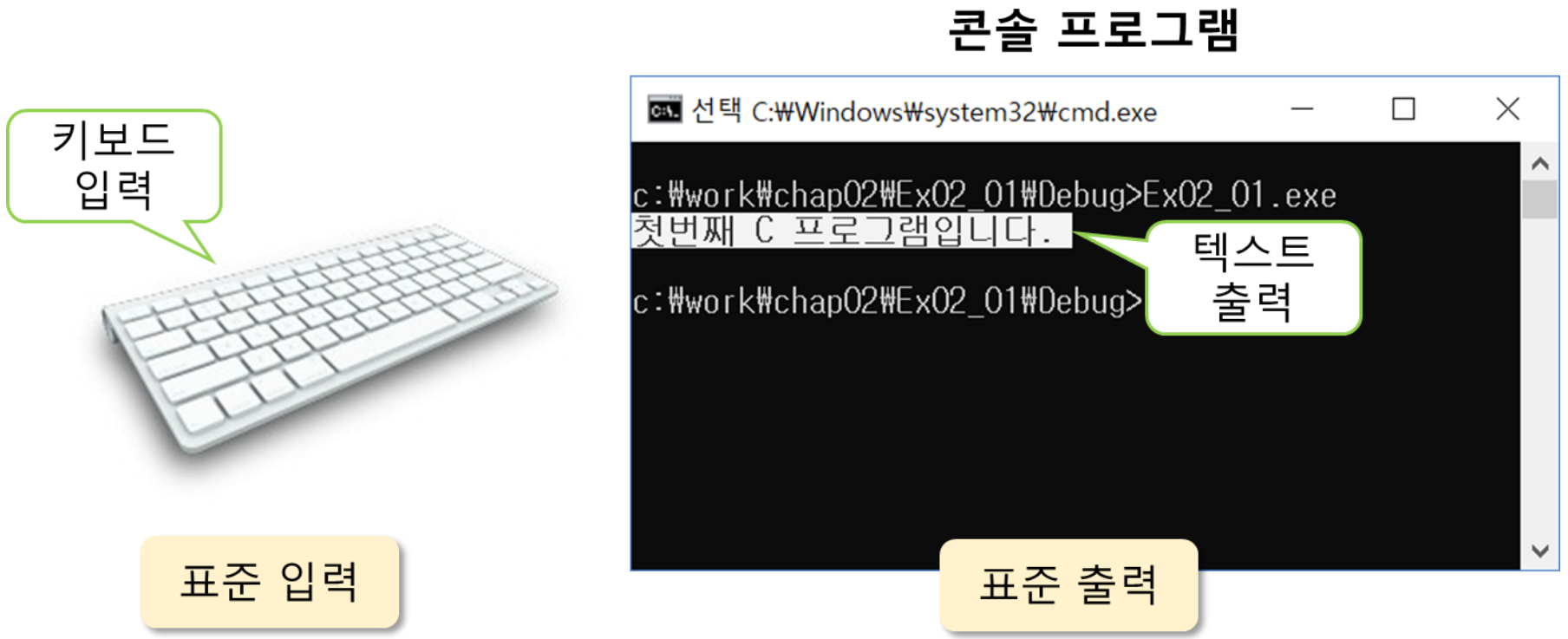
❖Visual Studio의 들여쓰기 단축키

- Ctrl+K, Ctrl+D 또는 블록 선택 후 Ctrl+K, Ctrl+F

입출력

❖콘솔 프로그램

- 콘솔(명령 프롬프트)에서 실행되는 프로그램
- **키보드**로부터 입력을 받아서 처리 결과를 콘솔에 **텍스트**로 출력

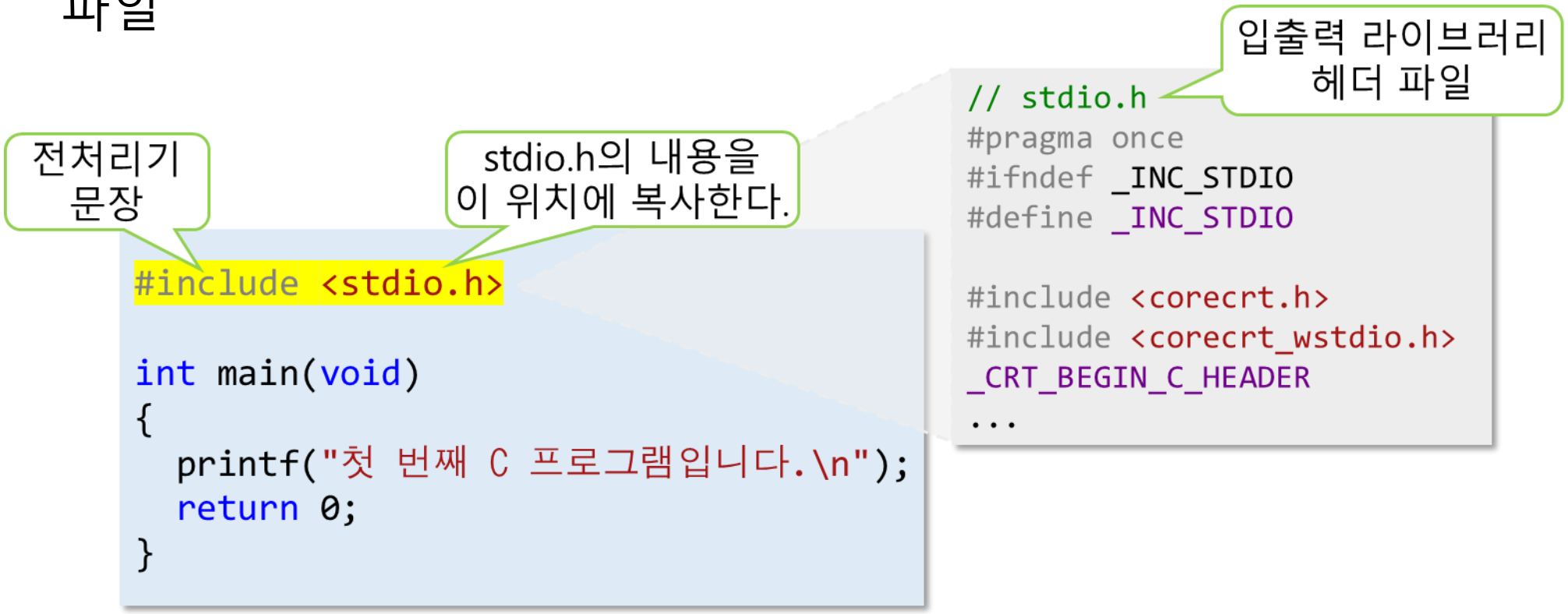


입출력 라이브러리를 사용하기 위한 준비

❖<stdio.h>를 포함한다

❖헤더 파일(.h)

- 라이브러리 함수의 이름, 리턴형, 매개변수에 대한 정보를 담고 있는 파일



콘솔 출력

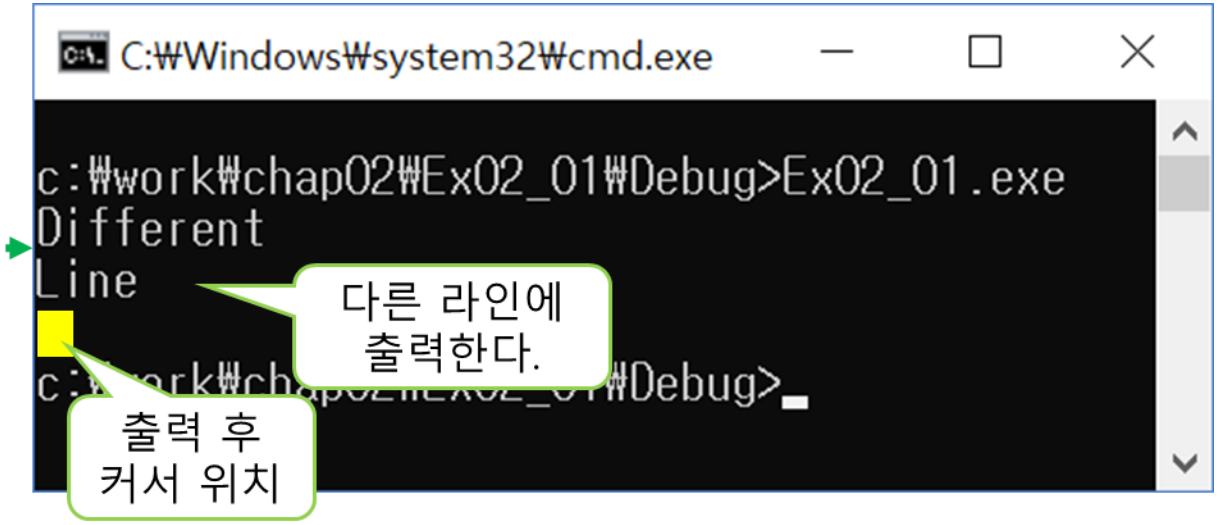
- 콘솔에 텍스트를 출력하려면 printf 함수를 이용한다
- 출력할 내용을 " "로 묶어서 printf 함수의 ( ) 안에 써준다

```
printf("첫 번째 C 프로그램입니다.\n"); // 출력하고 커서를 다음 줄로 이동한다.
```

```
int main()
{
    printf("Different\n");
    printf("Line\n");
}
```

줄 바꿈 문자를 사용하는 경우

실행



#define 과 #include의 사용

❖# : 전처리기 지시자 (Preprocessing Directive)

- **전처리기(Preprocessor)**
  - 컴파일 전에 전처리 지시자로 먼저 정의 되어있는 것들을 프로세서에 등록하는 것
- **전처리 지시자(Preprocessor Directives)**
  - #include와 같이 앞에 #이 붙는 명령어들

❖예)

#include"my_file.h"

- my_file.h 파일의 사본 포함
- C에서 제공하는 표준 헤더파일 : stdio.h, string,h, math.h, ... <xxx.h>

#define LIMIT 100

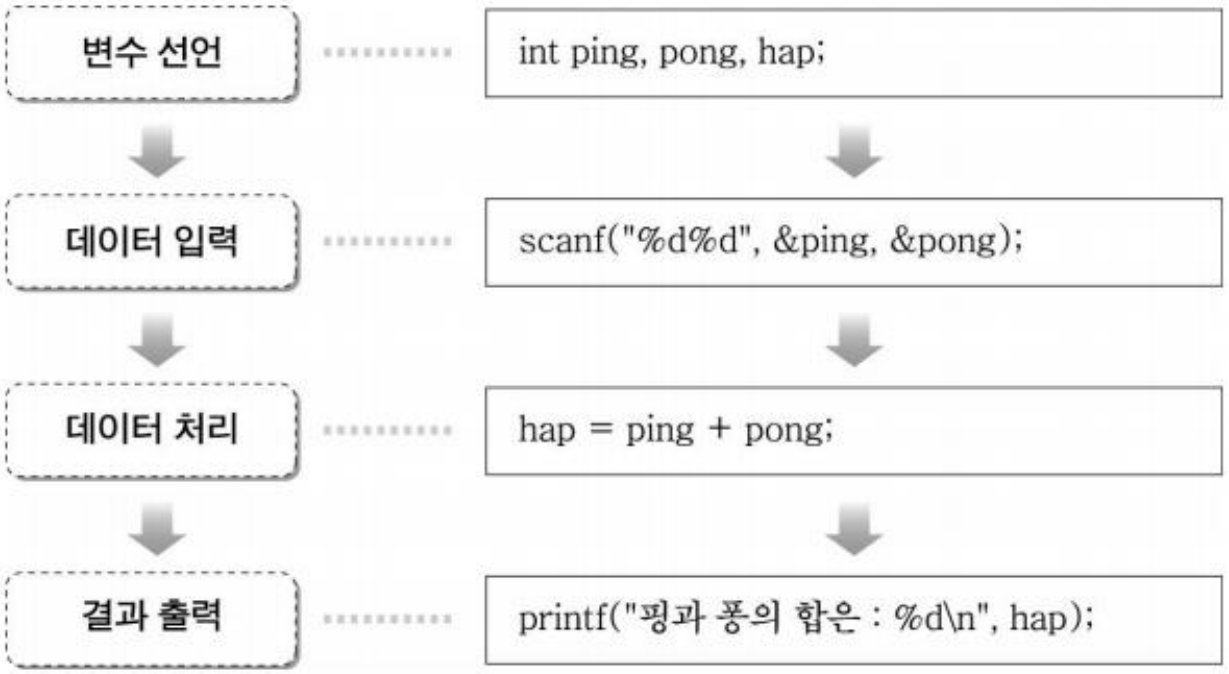
#define PI 3.14159

#define C 299792.458

프로그램 작성 순서

❖프로그램을 작성하는 순서

- 데이터를 입력하기 전에 반드시 입력할 데이터를 저장할 기억공간이 있어야 한다 (변수선언이 입력문 전에 있어야 한다)
- **일반적인 프로그램의 작성 순서**

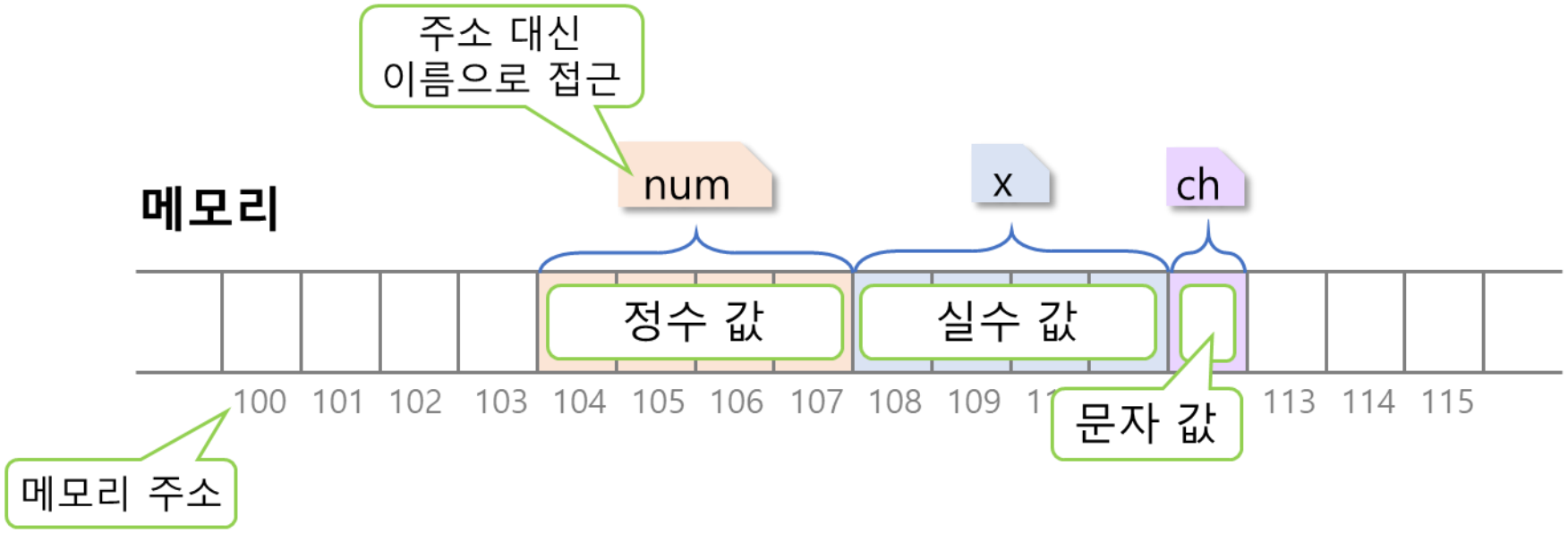


변수

- 어떤 값을 저장하기 위한 공간
- 변수를 사용하려면 **변수명**과 **데이터형(data type)**이 필요하다

```
int num;   // 정수형 변수 선언
float x;   // 실수형 변수 선언
char ch;   // 문자형 변수 선언
```

변수는 선언 후 사용해야 한다.



변수의 선언 및 사용

❖변수의 선언

- 변수명은 영문자와 숫자, 밑줄 기호(_)를 사용해서 만든다
- 첫 글자로는 반드시 영문자나 밑줄 기호가 와야 한다

형식 데이터형 변수명;

사용예

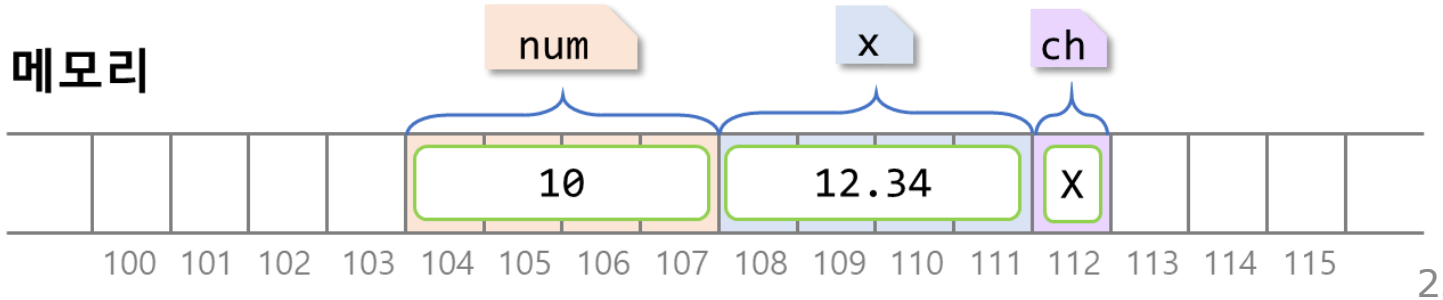
```
int num;
float x;
char ch;
```

❖변수의 사용

- 변수에 값을 대입하려면 =을 이용한다

```
num = 10;
x = 12.34;
ch = 'X';
```

변수와 같은 데이터 형의 값을 대입한다.



입력과 출력: printf() 와 scanf()

❖printf() : 화면 출력
- printf("서식지정문자열", "변수");

❖서식지정 문자열
- 일반문자열, 변환문자열(%), 확장문자열(₩)

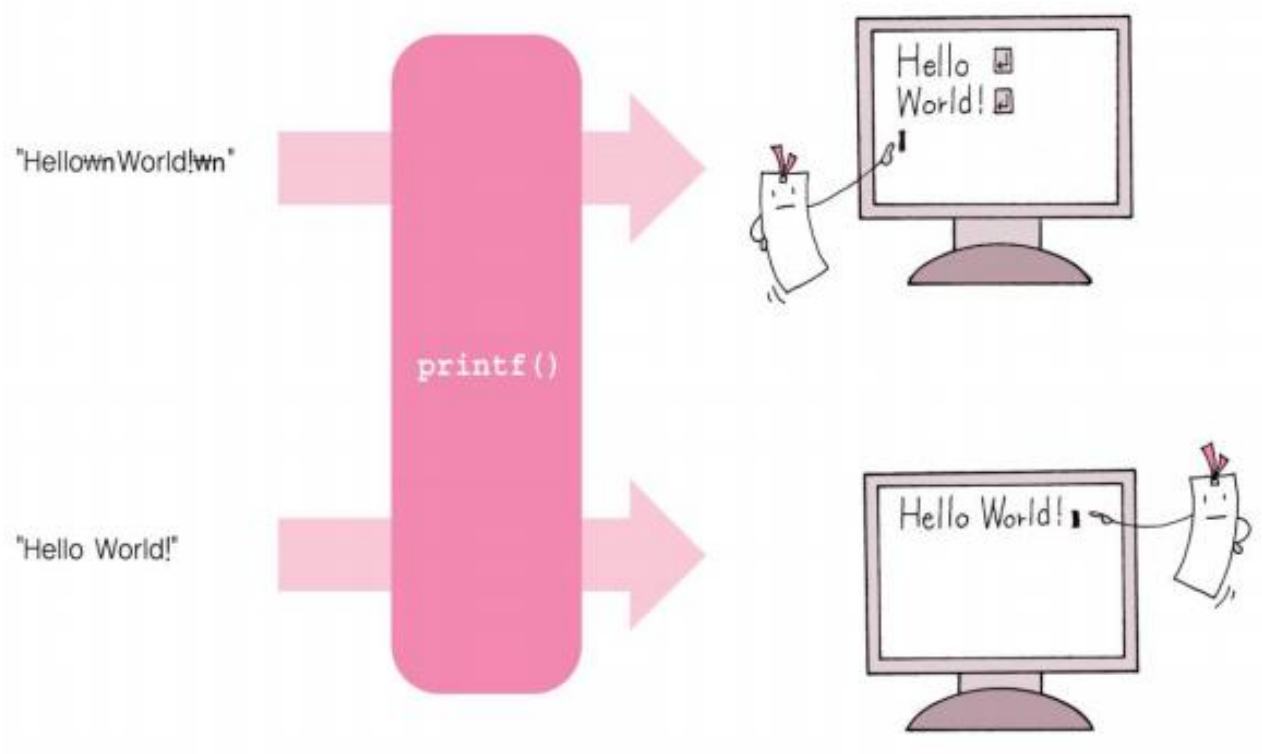
❖printf()의 변환문자열
- printf("%변환문자","변수"); ”변수“를 변환형식에 맞추어 화면 출력

❖scanf() : 키보드 입력

❖scanf()의 변환문자열

❖scanf(“%변환문자”, &변수); 변환문자 형식으로 입력 받아들임

입력과 출력: printf() 와 scanf()

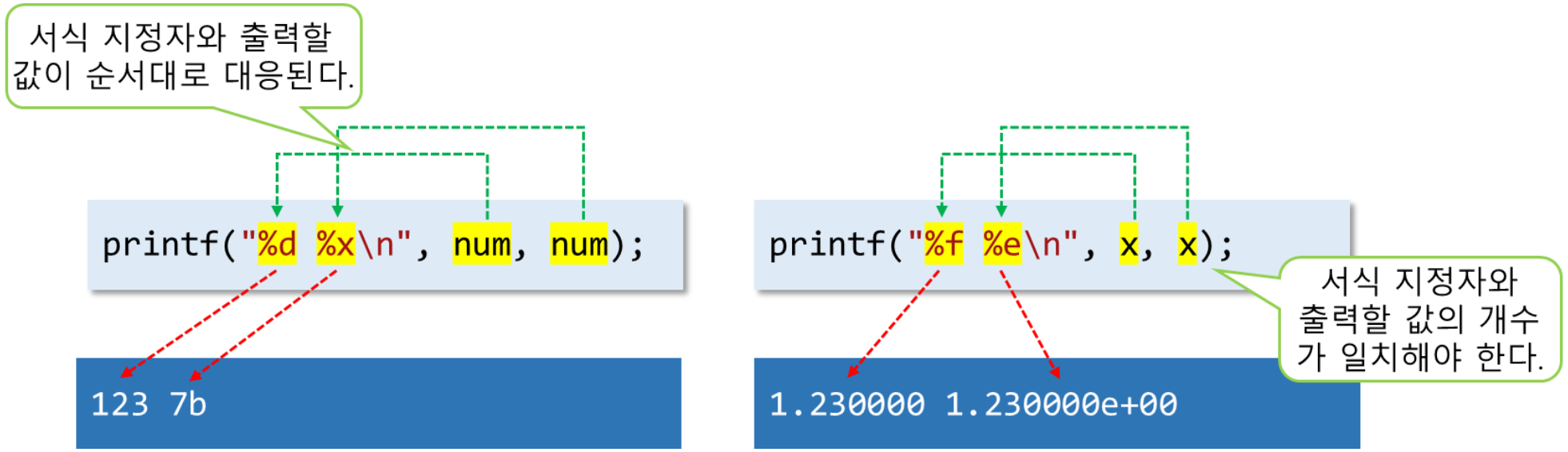


printf 함수의 형식 문자열 (1/3)

서식 지정자	의미	사용 예	실행 결과
%d	정수를 10진수로 출력	int num = 123; printf("%d", num);	123
%x	정수를 16진수로 출력 (0~6, a~f 이용)	int num = 123; printf("%x", num);	7b
%X	정수를 16진수로 출력 (0~9, A~F 이용)	int num = 123; printf("%X", num);	7B
%f, %F	실수를 부동소수점 표기 방식으로 출력	float x = 1.23; printf("%f", x);	1.230000
%e, %E	실수를 지수 표기 방식으로 출력	float x = 1.23; printf("%e", x);	1.230000e+00
%c	문자 출력	char ch = 'A'; printf("%c", ch);	A
%s	문자열 출력	char name[20] = "abc"; printf("%s", name);	abc

printf 함수의 형식 문자열 (2/3)

- 서식 지정자의 개수와 출력할 값의 개수가 일치해야 한다



- 16진수 정수 출력

```
printf("%x\n", num);     // 7b 출력
printf("%X\n", num);     // 7B 출력
printf("%#x\n", num);    // 0x7b 출력
printf("%#X\n", num);    // 0X7B 출력
```

예제 2-2 : 형식 문자열을 이용해서 출력하기

```c
#include <stdio.h>

int main()
{
    int num;        // 정수형 변수 선언
    float x;        // 실수형 변수 선언
    char ch;        // 문자형 변수 선언

    num = 123;      // 정수형 변수에 값 대입
    x = 1.23;       // 실수형 변수에 값 대입
    ch = 'A';       // 문자형 변수에 값 대입

    printf("%d %x\n", num, num);        // 10진수, 16진수 출력
    printf("%f %e\n", x, x);            // 부동소수점, 지수 출력
    printf("%c\n", ch);                 // 문자 출력

    printf("%x\n", num);        // 7b 출력
    printf("%X\n", num);        // 7B 출력
    printf("%#x\n", num);       // 0x7b 출력
    printf("%#X\n", num);       // 0X7B 출력

    return 0;
}
```

형식 문자열

형식 문자열 안에 서식 지정자를 여러 개 지정할 수 있다.

#을 지정하면 0x나 0X를 함께 출력

실행결과

```
123 7b
1.230000 1.230000e+00
A
7b
7B
0x7b
0X7B
```

10진수 123은 16진수 7b에 해당

printf 함수의 형식 문자열 (3/3)

❖문자 폭 지정

```
printf("%08d\n", num);      // 빈칸 대신 문자 폭의 남는 부분에 0 출력
```

❖정밀도 지정

- 실수의 정밀도
  - 소수점 이하 자릿수

```
printf("%f\n", x);       // 폭과 정밀도를 지정하지 않는 경우
printf("%.2f\n", x);     // 정밀도를 지정하는 경우 (소수점 이하 2자리)
printf("%8.2f\n", x);    // 폭과 정밀도를 지정하는 경우
```

- 정수의 정밀도
  - 출력할 숫자의 자릿수

```
printf("%8.4d\n", 123);      // ␣␣␣␣0123 출력 (␣은 빈칸)
```

printf() 옵션

❖printf()의 옵션 지정

- %필드 폭자리수변환문자
- %d ->123
- %5d->_ _123
- %10d->_ _ _ _ _ _ _123
- %2d ->123 (지정폭이 작아도 필요한 폭은 확보)
- %f ->654.321000(표준폭으로 출력)
- %12f ->_ _ 654.321000 (소수점 넣어 12자리로출력, 이하는 표준폭으로 출력)
- %9.2f ->_ _ _ 654.32 (소수점 넣어 9자리로출력, 이하는 2자리로 출력)

예제 2-3 : 문자 폭 지정하기

```
03  int main()
04  {
05      int num = 12345;
06
07      printf("%d\n", num);
08      printf("%d\n", num * 10);
09      printf("%d\n", num * 100);
10      printf("%d\n", num * 1000);
11
12      printf("%8d\n", num);              // 8문자 폭에 맞춰서 출력
13      printf("%8d\n", num * 10);
14      printf("%8d\n", num * 100);
15      printf("%8d\n", num * 1000);
16      printf("%8d\n", num * 100000);   // 문자 폭보다 큰 수 출력
17
18      printf("%08d\n", num);             // 빈칸 대신 문자 폭의 남는 부분에 0 출력
```

%d는 폭을 지정하지 않았으므로 왼쪽에서부터 출력

%8d는 폭을 지정하였으므로 8칸에 대하여 오른쪽으로 맞춰서 출력

실행결과

```
12345
123450
1234500
12345000
   12345
  123450
 1234500
12345000
1234500000
00012345
```

왼쪽에 빈칸 3개 출력

왼쪽에 빈칸 2개 출력

왼쪽에 빈칸 1개 출력

폭보다 큰 값은 폭을 넘어서 출력

왼쪽에 0을 3개 출력

예제 2-4 : 정밀도 지정하기

```c
#include <stdio.h>

int main()
{
    float x;

    x = 12.34567;           // x에 값 저장

    printf("%f\n", x);      // 폭과 정밀도를 지정하지 않는 경우
    printf("%.2f\n", x);    // 정밀도를 지정하는 경우 (소수점 이하 2자리)
    printf("%8.2f\n", x);   // 폭과 정밀도를 지정하는 경우

    return 0;
}
```

실행결과

```
12.345670
12.35
   12.35
```

12.345670 — 디폴트로 소수점 이하 6자리 출력

12.35 — 소수점 이하 2자리 출력

12.35 — 전체 8문자 폭에 소수점 이하 2자리 출력

scanf 함수의 형식 문자열 (1/3)

서식 지정자	의미	사용 예
%d	정수를 10진수로 입력	`int num;` `scanf("%d", &num);`
%x	정수를 16진수로 입력	`int num;` `scanf("%x", &num);`
%i	정수를 10진수, 8진수, 16진수로 입력 (012는 8진수, 0x12는 16진수)	`int num;` `scanf("%i", &num);`
%f	float형 실수 입력	`float x;` `scanf("%f", &x);`
%lf	double형 실수 입력	`double y;` `scanf("%lf", &y);`
%c	문자 입력	`char ch;` `scanf("%c", &ch);`
%s	문자열 입력	`char name[20];` `scanf("%s", name);`

예제 2-5 : 입력받은 10진수 정수를 16진수로 변환해서 출력하기

```c
#include <stdio.h>

int main()
{
    int num;            // 정수형 변수 선언

    printf("정수? ");    // 정수를 입력을 하도록 사용자에게 알려주기 위한 출력문
    scanf("%d", &num);  // num에 10진수로 정수 입력

    printf("10진수 %d는 16진수 %x입니다.\n", num, num);
                  10진수 정수출력        16진수 정수출력
    return 0;
}
```

실행결과

```
정수? 10        10진수 정수 입력
10진수 10는 16진수 a입니다.
```

scanf 함수의 형식 문자열 (2/3)

❖문자 배열에 입력받을 때는 &를 지정하지 않는다

```
char name[20];      // 문자 20개를 연속으로 저장하는 변수
scanf("%s", name);  // name에는 &가 필요 없다.
```

❖서식 지정자를 여러 개 사용할 수도 있다

- 서식 지정자와 입력받을 변수의 개수가 같아야 한다

```
scanf("%s %d %c", name, &age, &gender);
```

3개 3개

❖실수형 변수 입력

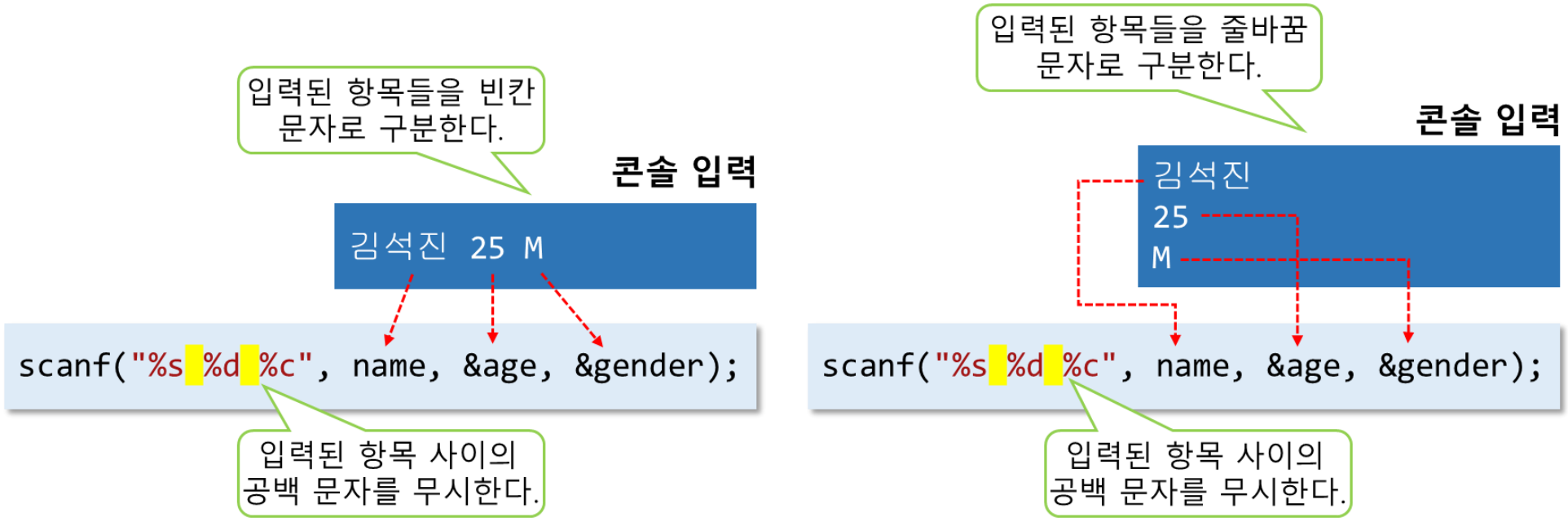
- %f : float 입력
- %lf : double 입력

```
float x;
double y;
scanf("%f", &x);    // float형 변수에 실수 값 입력
scanf("%lf", &y);   // double형 변수에 실수 값 입력
```

scanf 함수의 형식 문자열 (3/3)

❖형식 문자열의 공백 문자

- 이전 입력 이후의 공백을 모두 무시하고 다음 입력을 읽어 오게 한다



예제 2-6 : 여러 개의 서식 지정자 사용하기

```
03  int main()
04  {
05      char name[20];
06      int age;
07      char gender;
08
09      printf("이름, 나이, 성별(M/F) 순으로 입력하세요.\n");
10      scanf("%s %d %c", name, &age, &gender);     // 3개의 변수 입력
11      printf("이름: %s\n", name);
12      printf("나이: %d\n", age);
13      printf("성별: %c\n", gender);
14
15      return 0;
16  }
```

name은 문자 배열이므로 &가 필요 없다.

실행결과

```
이름, 나이, 성별(M/F) 순으로 입력하세요.
김석진 25 M
이름: 김석진
나이: 25
성별: M
```

세 가지 항목을 빈칸으로 구분해서 입력

제어의 흐름

❖if 문

- 일반적인 형태 : if (조건식) { 문장1 }
- 조건식이 참(true)이면 (0이 아니면) 문장1 실행, 단문이면 {} 생략

```
a=1
if (b==3) a=5; /* == : '--와 같다' 연산자) */
  printf("%d", a);
```

  - b가 3이면 a=5
  - b가 3이 아니면 문장(a=5) 실행 안함, printf() 문 실행 1 출력

❖if-else 문

- 일반적인 형태 : if (조건식) { 문장1 }
  else { 문장2 }
- 조건식이 참(true)이면 (0이 아니면) 문장1 실행 그렇지 않으면 문장2 실행

제어의 흐름 – 예시

- 예)

```
if(cnt==0){
        a=2;
        b=3;
        c=5;
}
else {
        a=-1;
        b=-2;
        c=-3;
}
printf("%d", a+b+c);
```

➢ cnt 가 0값을 가지면 10 출력, 그렇지 않으면 -6 출력

반복문 – while

❖while 루프
- 일반적인 형태 : while (조건식) { 문장 }

```
#include<stdio.h>
int main(void){
        int i=1, sum=0;
        while (i<=5) {
                sum+=i;
                ++i;
        }
        printf("sum= %d ₩n", sum);
        return 0;
}
```

참고

++i, i++; 증가

--i, i--; 감소

i=i+1; i=i-1;

반복문 – for

❖ for 루프
- 일반적인 형태 : for (조건식) { 문장 }

```
#include <stdio.h>
int main(void)
{
    int sum=0,i;
    for (i=1;i<=5;++i){
        sum+=i;
    }
    printf("sum= %d \n",sum);
    return 0;
}
```

질의 및 응답

참고문헌

- 천정아,『Core C Programming』, 연두에디션(2019)
- C가 보이는 그림책, ANK Co., Ltd. , 성안당 (2018)
- Greg Perry, Dean Miller『어서와 C언어는 처음이지』, 천인국 옮김, 인피니티북스(2015)
- KELLEY (역 : 김명호 외),『A Book on C』, 홍릉과학출판사 (2003)
- 윤성우,『 열혈 C 프로그래밍』, 오렌지미디어
- 천인국,『 쉽게 풀어쓴 C언어 Express』, 생능출판사
- 서현우,『 뇌를 자극하는 C 프로그래밍』, 한빛미디어
- 강성수,『 쾌도난마 C프로그래밍』, 북스홀릭
- 고응남,『 C프로그래밍 기초와 응용실습』, 정익사