Clickseo.com

C Programming

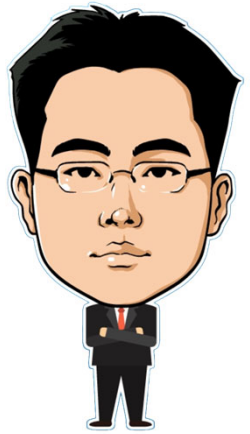
C 언어 개요

(C Language Overview)

Seo, Doo-Ok

Clickseo.com

clickseo@gmail.com



목 차

백문이불여일타(百聞而不如一打)

- C 언어 개요
- 데이터 표현
- 표준 입출력

C 언어 개요

백문이불여일타(百聞而不如一打)

- **C 언어 개요**
  - C 언어의 역사 및 특징
  - C 프로그램 개발 단계
- 데이터 표현
- 표준 입출력

C 언어의 역사 및 특징 (1/4)

- **C 언어의 역사:** B 언어와 C 언어

[Thompson (left) with Dennis Ritchie]

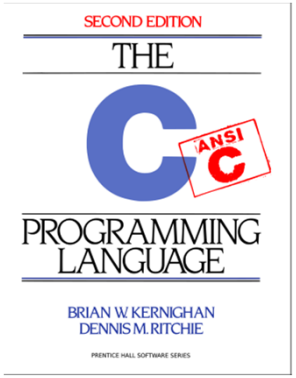
  - **ALGOL**(ALGOrithmic Language)
    - 1960년, **ALGOL 60** 제정
      - 1958년, ALGOL 58 (원래는 IAL)
      - 1973년, ALGOL 68 개정
  - **BCPL**(Basic Combined Programming Language)
    - 1966년, 캠브리지 대학교의 **마틴 리처드**(Martin Richard)가 설계
    - **절차적 명령형, 구조적 컴퓨터 프로그래밍 언어**
      - 자료형을 지원하지 않고 자료 객체만 지원
  - **B 프로그래밍 언어**
    - 1969년, AT&T Bell 연구소의 **켄 톰슨**(Ken Thomson)
  - **C 프로그래밍 언어**
    - 1972년, AT&T Bell 연구소의 **데니스 리치**(Dennis Ritchie)
    - **UNIX 운영체제에서 사용하기 위해 개발된 프로그래밍 언어**
      - BCPL과 B언어를 결합, 자료형 지원

C 언어의 역사 및 특징 (2/4)

● C 언어의 역사: 표준화

❍ 전통적인 C 언어의 표준

- 1989년, **(C89)** 미국 국립 표준화 협회(ANSI)에서 표준으로 지정: **ANSI C, 표준 C**
- **1990년, ISO/IEC 9899:1990(C90) 국제 표준화 기구(ISO)가 ANSI 표준 채택**
- 1995년, 일종의 부록 형태로 C90의 확장 발표 : Normative Addendum 1**(NA1)**
  - 유럽을 중심으로 터져 나온 'C90 이 지나치게 미국적' 이라는 불만을 잠재우기 위해...
- C90과 NA1을 합해 **C95** 라고 부르기도 한다.



- **1999년, ISO/IEC 9899:1999(C99) 발표**
  - 다양한 변화(예: C++와 유니코드의 발전 등)를 수용하기 위해...
  - **inline** 함수, 혼합된 선언 및 코드 등
  - 여러가지 새로운 자료형: **_Bool, long long int, complex** 등
  - C++ 에서와 같이 단일 행 주석 지원: **//**
  - 새로운 라이브러리 함수와 헤더 파일: **snpritf** 와 **<stdbool.h>** 등

C 언어의 역사 및 특징 (3/4)

C 언어의 역사: 표준화

새로운 C 언어의 표준

- 2007년, **(C1X)** C 표준화 위원회에서 **새로운 기능 채택을 제안**한 가이드 라인 발표
- **2011년 12월, ISO/IEC 9899:2011(C11) 새로운 C 프로그래밍 언어 표준 발표**
  - **C 언어와 라이브러리에 수 많은 기능 추가와 C++ 와의 호환성 향상**
  - 멀티 스레딩 지원과 gets 함수 제거 등

- **2018년 06월, ISO/IEC 9899:2018(C17) C11의 결함에 대한 기술적 수정 사항**
  - **C11 (표준 ISO/IEC 9899: 2011)을 대체:** C11의 결함에 대한 기술적 수정 사항
    - » **새로운 언어의 특징을 도입하지 않는다.**
  - C17 지원 컴파일러 : GCC 8.1.0, LLVM Clang 7.0.0

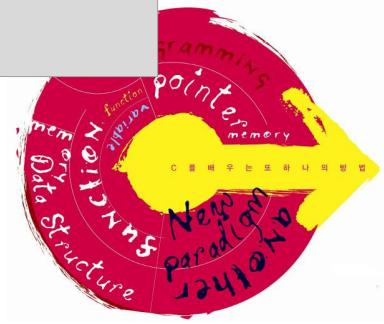
- **C23(C2X), ISO/IEC 9899:2023(C23) C 표준 개정판(2024년에 출판될 것으로 예상)**
  - **C 언의 "Original Principles"에 새로운 원칙을 추가한다.**
  - GCC 9.0과 Clang 9.0에서는 이 표준을 지원하는 **-std=c2x** 옵션이 있다.

C 언어의 역사 및 특징 (4/4)

- **C 언어의 주요 특징**
  - 이식성이 좋다.
  - 유연성
    - 고급 언어와 저급 언어의 특징을 동시에 가진 언어이다.
    - 기계 중심의 언어: Assembly
    - 사용자 중심의 언어: C/C++, Java, FORTRAN, COBOL, PASCAL 등
  - 구조적 프로그래밍을 지원하는 함수 언어이다.
  - 풍부한 내장 함수 라이브러리를 제공한다.
  - 프로그래밍 구조가 간결하고 명료하다.
  - 실행 파일의 크기가 작고 빠른 성능을 자랑한다.
  - 분할 컴파일 기능을 지원한다.

C 언어 개요

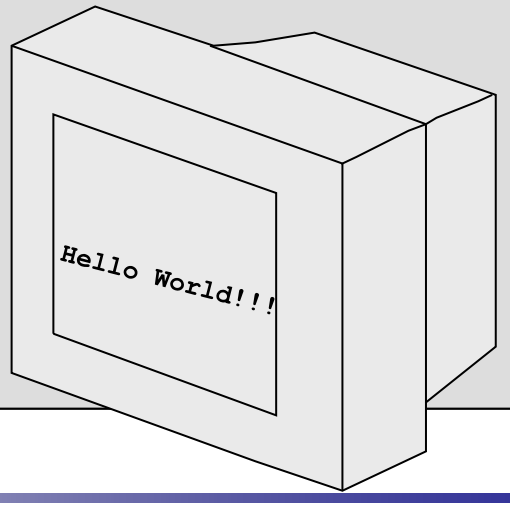
C 프로그램 개발 단계



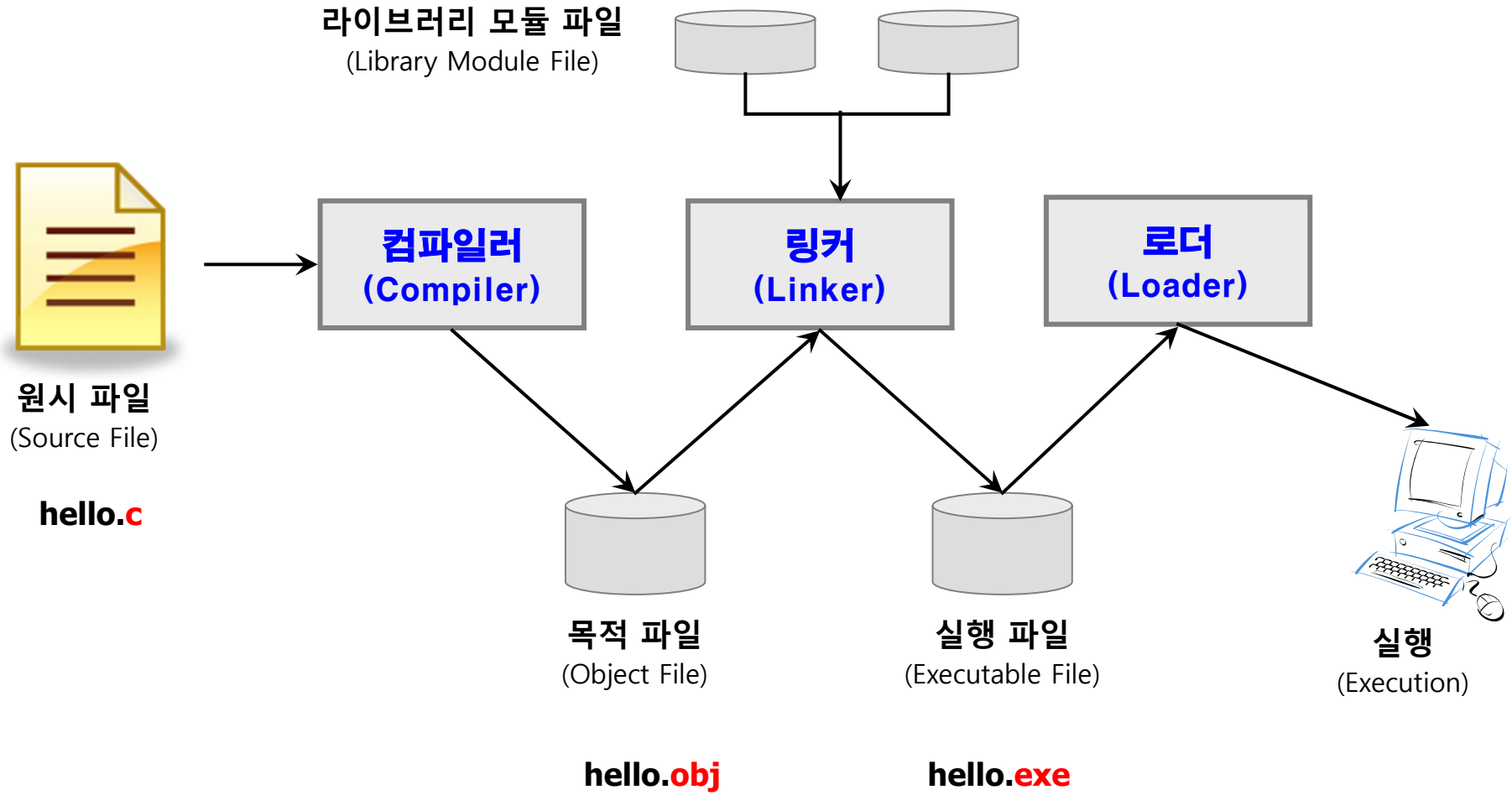
C 프로그램 개발 단계 (1/3)

- C 프로그램 구조

```
/*
   프로그램: Hello World
   작 성 자 : 서두옥(Clickseo)
   작 성 일 : 0000년 00월 00일
*/

// 헤더파일: 전처리기(preprocessor)
#include <stdio.h>

// main 함수
int main(void)
{
          printf("Hello World!!!\n");
          return 0;   // 함수 종료(반환값: 0)
}
```



C 프로그램 개발 단계 (2/3)



C 프로그램 개발 단계 (3/3)

● 통합 개발 환경

❍ IDE(Integrated Development Environment)

- **코딩, 컴파일, 디버그, 배포 등** 프로그램 개발에 관련된 모든 작업을 하나의 프로그램 안에서 처리하는 환경을 제공하는 소프트웨어
  - 종래의 소프트웨어 개발에서는 **텍스트 편집기, 컴파일러, 디버거 등**을 따로 사용

❍ 대표적인 통합 개발 환경

- **Visual Studio:** visualstudio.microsoft.com
- **Visual Studio Code:** code.visualstudio.com
  - **C/C++:** C/C++ for Visual Studio Code
- **Code::Blocks:** codeblocks.org -- **The free C/C++ and Fortran IDE.**
- **Dev-C++:** bloodshed.net -- **Open Source C/C++ IDE for Windows**
- **Eclipse CDT:** github.com/eclipse-cdt -- **C/C++ Development Tooling**
- **Xcode:** developer.apple.com/xcode/

데이터 표현

- C 언어 개요
- **데이터 표현**
  - 식별자
  - 변수와 상수
  - 자료형
- 표준 입출력

백문이불여일타(百聞而不如一打)

식별자 (1/4)

- **C 언어의 구문과 어휘**

문 자	종 류
영문 소문자	a, b, c, d, …, x, y, z
영문 대문자	A, B, C, D, …, X, Y, Z
숫 자	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
특수문자	+, =, _, -, (,), *, &, %, $, #, !, |, <, >, ., [,], ^, ~, ,, ;, :, ", `, …
whitespace	**blank, new line, tab**

식별자 (2/4)

- **식별자(identifier)**
  - 프로그래머가 임의로 정의하여 사용하는 변수, 함수, 상수 등에 부여한 명칭

  - 식별자에 사용될 수 있는 문자
    - 영문 소문자(lowercase letters) : **a, b, c, ... , z**
    - 영문 대문자(uppercase letters) : **A, B, C, ... , Z**
    - 숫자(digits) : **0, 1, 2, 3, 4, 5, 6, 7, 8, 9**
    - '_' 문자

  - 식별자를 작성하는 규칙
    - **첫 번째 문자: 영문 대소문자 또는 '_'**
      - **숫자는 첫 번째 문자로 사용 불가능(두 번째 문자부터 가능)**
    - **사용 불가능: 예약어(keyword)와 특수 문자**
    - 변수의 이름은 최대 63자까지 가능하다.
      - 변수 이름의 길이가 63자를 초과하면 63번째 문자까지 인식한다.
    - **C 언어에서 식별자는 대소문자를 정확히 구분한다.**

식별자 (3/4)

식별자: 예제

식별자 작성 예

- **SUM, Sum, sum** : 서로 다른 식별자로 인식**(C 언어는 대소문자를 구분)**
- **i5Clickseo** : 두 번째 문자로 숫자 사용
- **name_of_variable, _Clickseo**
- **getNumber, printSum, clickSeo** // 낙타체 스타일
- **CustomerName, ClickSeo** // 파스칼 스타일
- **iNumber, iClickseo** // 헝가리안 스타일

잘못 작성된 예

- **5iClickseo : 숫자는 첫 번째 문자로 사용불가**
- **Clickseo#35 : 특수문자는 사용 불가**
- **Click-seo : - (hyphen) 사용 불가**

식별자 (4/4)

- **예약어(Keyword)**
  - C 언어에서 특별한 의미로 사용되는 단어
    - 프로그램 내에서 재 정의되거나 다른 용도로 사용되어서는 안 된다.

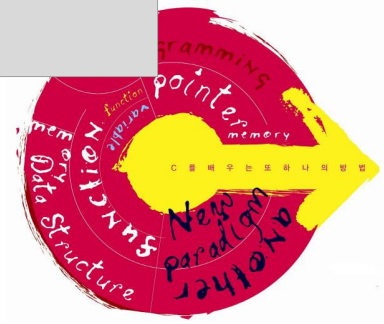
  - **C89/C90** : 32개 예약어

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	

- **C99** : 추가된 5가지 예약어
  - `_Bool`, `_Complex`, `_Imaginary`, `inline`, `restrict`
- **C11** : 추가된 7가지 예약어
  - `_Alignas`, `_Alignof`, `_Atomic`, `_Generic`, `_Noreturn`, `_Static_assert`, `_Thread_local`

데이터 표현

변수와 상수



변수와 상수 (1/7)

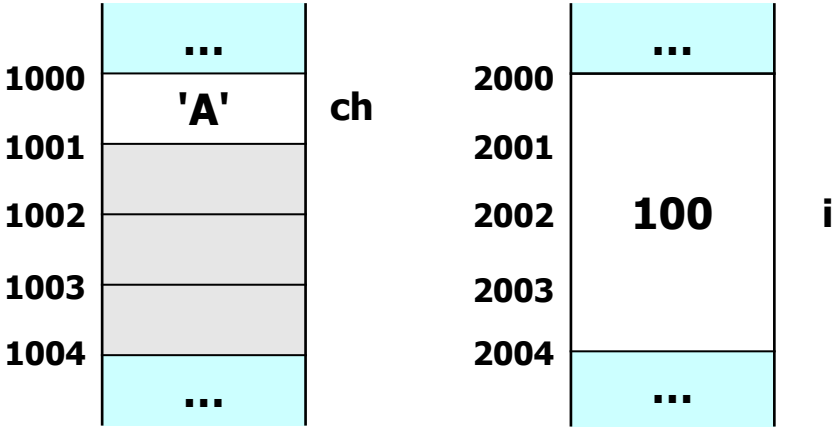
● 변수(Variable)

❍ 프로그램에서 사용되는 자료를 저장하기 위한 공간

- **할당 받은 메모리의 주소 대신 부르는 이름**
  - 사용자가 변수 이름을 만들어 저장하기 위한 자료의 형태를 지정하면, 컴파일러는 컴퓨터 메인 메모리의 주소 값과 연결시켜준다.
- **프로그램 실행 중에 값을 변경하는 것이 가능하다.**
- **사용되기 이전에 선언되어야 한다.**

```
char    ch;
int     i;
```

프로그램



메모리

변수와 상수 (2/7)

- **변수:** 선언 및 초기화

```c
#include <stdio.h>
int main(void)
{
        // 변수 선언
        int     a;
        char    ch;

        // 데이터 저장
        a = 10;
        ch = 'A';

        return 0;
}
```

```c
// 변수는 선언과 초기화를 동시에 할 수 있다.
#include <stdio.h>
int main(void)
{
        int     a = 10;
        char    ch = 'A';

        return 0;
}
```

변수와 상수 (3/7)

- **상수**(constant)
  - 프로그램 수행 중에 값을 변경할 수 없는 데이터

  - 리터럴 상수(literal constant)
    - 정수형 상수(integer constant)
    - 실수형 상수(real constant)
    - 문자 상수(character constant)
    - 문자열 상수(string constant)

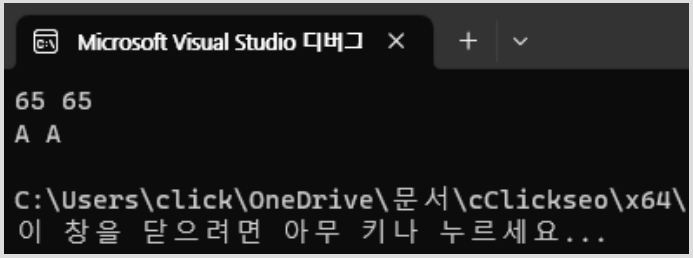
  - 기호 상수(symbolic constant)
    - 예약어 **const** 를 이용하는 방식

  - 매크로 상수(macro constant)
    - **전처리**의 매크로를 이용하는 방식

변수와 상수 (4/7)

- **문자 상수**(character constant)
  - 영문자나 숫자, 특수 기호 등을 숫자로 표현하기 위해 사용
    - 작은 따옴표(' ')를 이용
    - "내부적으로는 문자형 상수도 숫자로 취급된다."

```c
#include <stdio.h>
int main(void)
{
        char    ch1 = 'A';
        char    ch2 = 65;

        printf("%d %d\n", ch1, ch2);
        printf("%c %c\n", ch1, ch2);

        return 0;
}
```



변수와 상수 (5/7)

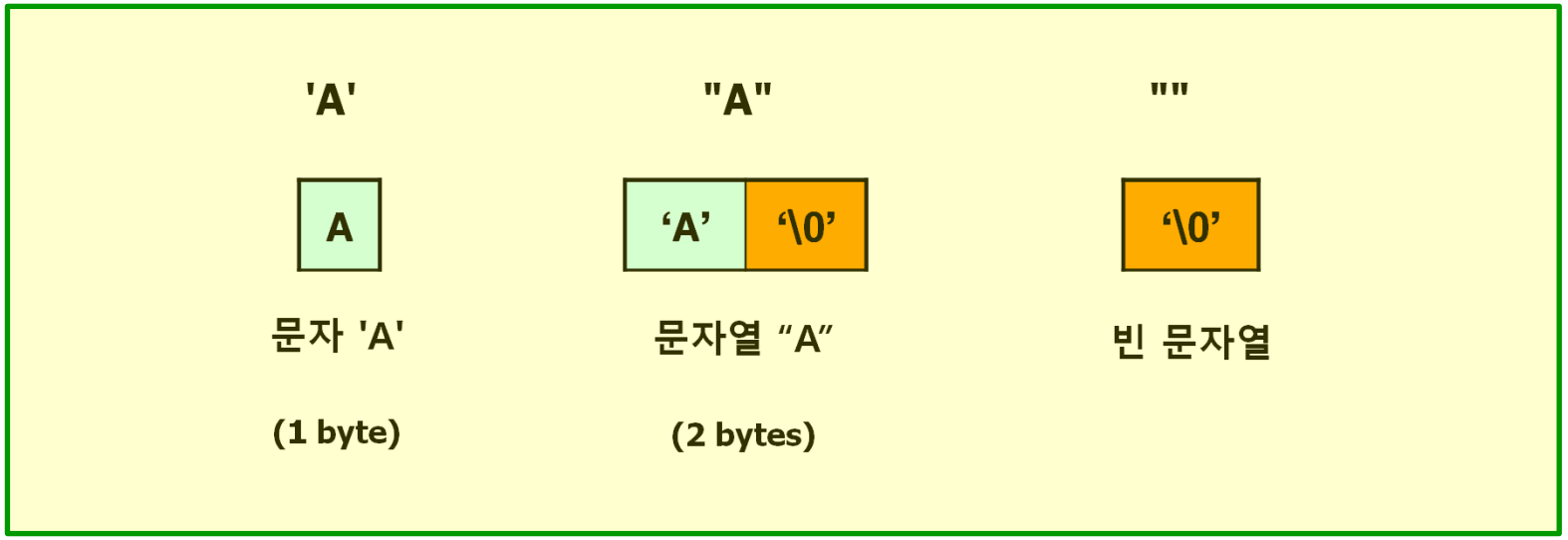
● 문자 상수: 확장 문자열

❍ 확장 문자열: 이스케이프(escape) 문자

확장 문자열	ASCII 코드	의 미	
\0	0	NULL	널 문자(Null Character)
\a	7	BEL	경고음(alarm) 소리 발생(alert)
\b	8	BS	커서를 뒤로 한 칸 이동한 효과(BackSpace)
\t	9	HT	Tab 키를 누른 만큼 커서 이동(Tab : 보통 4열)
\n	10	LF	커서를 다음 행으로 이동(LF, Line Feed / new line)
\v	11	VT	수직 탭(vertical tab)
\f	12	FF	인쇄 시 프린터의 종이를 한 장 넘김(FF, Form Feed)
\r	13	CR	커서를 줄의 처음으로 이동(CR, Carriage Return)
\"	34		큰 따옴표(double quote) 출력
\'	39		작은 따옴표(single quote) 출력
\\	92		역 슬래시(literal backslash) 출력

변수와 상수 (6/7)

- **문자열 상수**(string constant)
  - 큰 따옴표 내에 하나 이상의 문자를 묶어서 표현하는 문자열
    - 문자열 상수는 문자열 끝에 **널 문자('\0')** 가 자동으로 삽입되는 문자 상수



변수와 상수 (7/7)

● 기호 상수(symbolic constant)

❍ 예약어 const 를 이용하는 방식

```
const int MAX = 100;
```

- (주의할 점)
  1) 자료형 한정자가 제일 먼저 나타나야 한다.
  2) 초기화가 반드시 되어야 한다.
  3) 상수로 선언되었으므로 그 값이 바뀔 수 없다.

● 매크로 상수(macro constant)

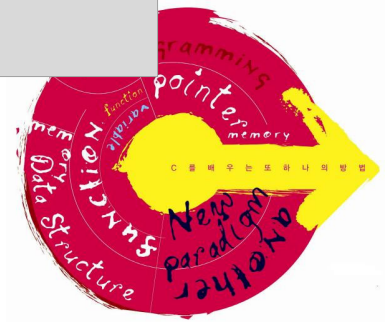
❍ 전처리의 매크로를 이용하는 방식

- 전 처리기 명령어 **define** 을 사용

```
#define MAXSIZE 1024
```

데이터 표현

자료형



자료형 (1/12)

논리형	_Bool // since C99
문자형	char, unsigned char
정수형	short int, unsigned short int int, unsigned int long int, unsigned long int long long int, unsigned long long int // since C99
실수형	float, double, long double
형 없음	void

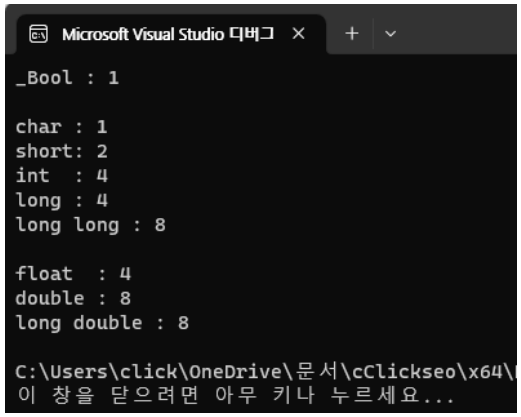
자료형 (2/12)

자료형		크기(byte)	데이터 표현 범위
char	(signed) char	1 (8bits)	$-2^7 \sim 2^7 - 1$: -128 ~ 127
	unsigned char		$0 \sim 2^8 - 1$: 0~255
int	(signed) short int	2 (16bits)	$-2^{15} \sim 2^{15}-1$: -32,768 ~ 32,767
	unsigned short int		$0 \sim 2^{16}-1$: 0 ~ 65,535
	(signed) int	4 (32bits)	$-2^{31} \sim 2^{31}-1$: -2,147,483,648 ~ 2,147,483,647
	unsigned int		$0 \sim 2^{32}-1$: 0 ~ 4,294,967,295
long	(signed) long int	4 (32bits)	$-2^{31} \sim 2^{31} - 1$: -2,147,483,648 ~ 2,147,483,647
	unsigned long int		$0 \sim 2^{32} - 1$: 0 ~ 4,294,967,295
long long	(signed) long long int	8 (64bits)	$-2^{63} \sim 2^{63} - 1$: -9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
	unsigned long long int		$0 \sim 2^{64} - 1$: 0 ~ 18,446,744,073,709,551,615
float	float	4 (32bits)	$-10^{128} \sim 10^{127}$: 소수 6자리 표현
double	double	8 (64bits)	$-10^{128} \sim 10^{127}$: 소수 15자리 표현
	long double	8 (64bits) 또는 그 이상	차이를 많이 보임 : double의 정밀도와 같거나 크다.

자료형 (3/12)

예제 1-1: 다양한 자료형 Visual Studio Community 2022 (64bits)

```
#include <stdio.h>
int main(void)
{
        // warning C4477: 'printf' : 서식 문자열 '%d'에 'int' 형식의 인수가 필요하지만
        //                           variadic 인수 1의 형식이 'size_t'입니다.
        // message : 서식 문자열에서 '%zd'을(를) 사용하는 것이 좋습니다.
        // printf("_Bool : %d \n\n", sizeof(_Bool) );
        printf("_Bool : %zd \n\n", sizeof(_Bool) );

        printf("char : %zd \n", sizeof(char) );
        printf("short: %zd \n", sizeof(short) );
        printf("int  : %zd \n", sizeof(int) );
        printf("long : %zd \n", sizeof(long) );
        printf("long long : %zd \n\n", sizeof(long long) );

        printf("float  : %zd \n", sizeof(float) );
        printf("double : %zd \n", sizeof(double) );
        printf("long double : %zd \n", sizeof(long double) );

        return 0;
}
```



자료형 (4/12)

예제 1-2: 다양한 자료형 GNU/Linux GCC (64bits)

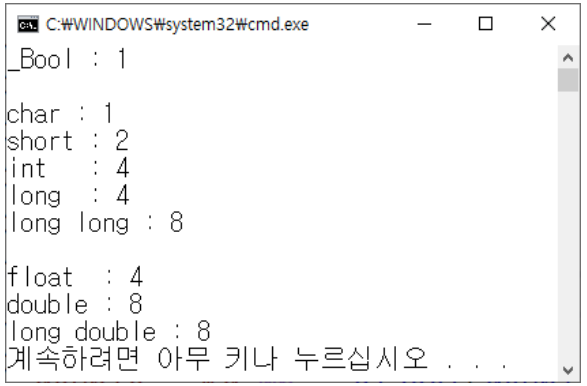
```
#include <stdio.h>
int main(void)
{
        // warning: format '%d' expects argument of type 'int',
        // but argument 2 has type 'long unsigned int'
        // printf("char: %d \n", sizeof(char));
        printf("char   : %ld \n", sizeof(char));
        printf("short  : %ld \n", sizeof(short));
        printf("int    : %ld \n", sizeof(int));
        printf("long   : %ld \n", sizeof(long));
        printf("long long : %ld \n", sizeof(long long));

        printf("float  : %ld \n", sizeof(float));
        printf("double : %ld \n", sizeof(double));
        printf("long double : %ld \n", sizeof(long double));

        return 0;
}
```

```
clickseo@clickseo-VirtualBox:~/gcc$ gcc dataType.c
clickseo@clickseo-VirtualBox:~/gcc$ ./a.out
char   : 1
short  : 2
int    : 4
long   : 8
long long : 8

float  : 4
double : 8
long double : 16
clickseo@clickseo-VirtualBox:~/gcc$
```

자료형 (5/12)

예제 1-3: 다양한 자료형 Visual Studio Community 2019 (32bits)

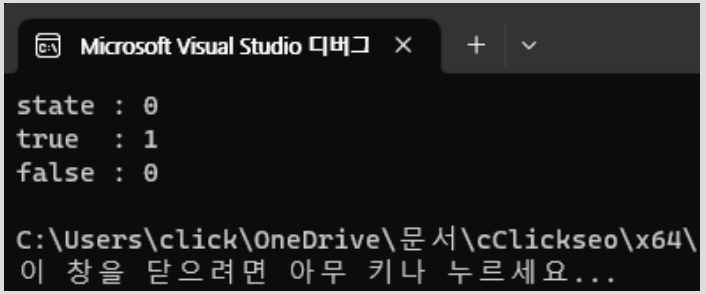
```c
#include <stdio.h>
int main(void)
{
        printf("_Bool : %d \n\n", sizeof(_Bool));

        printf("char : %d \n", sizeof(char));
        printf("short: %d \n", sizeof(short));
        printf("int  : %d \n", sizeof(int));
        printf("long : %d \n", sizeof(long));
        printf("long long : %d \n\n", sizeof(long long));

        printf("float  : %d \n", sizeof(float));
        printf("double : %d \n", sizeof(double));
        printf("long double : %d \n", sizeof(long double));

        return 0;
}
```



자료형 (6/12)

- **논리형(Boolean Type): _Bool**
  - **C99 에서 추가된 논리 유형의 자료형**
    - **<stdbool.h> 헤더 파일** : **C99** 에서 추가된 헤더 파일
      - **_Bool** 의 별칭으로 **bool** 을 정의
      - **true** 및 **false** 에 대한 매크로 제공

```c
#include <stdio.h>
#include <stdbool.h>                    // bool, true, false

int main(void)
{
        _Bool    state = true;          // bool    state = true;

        state = false;
        printf("state : %d \n", state);

        printf("true  : %d \n", true);
        printf("false : %d \n", false);

        return 0;
}
```

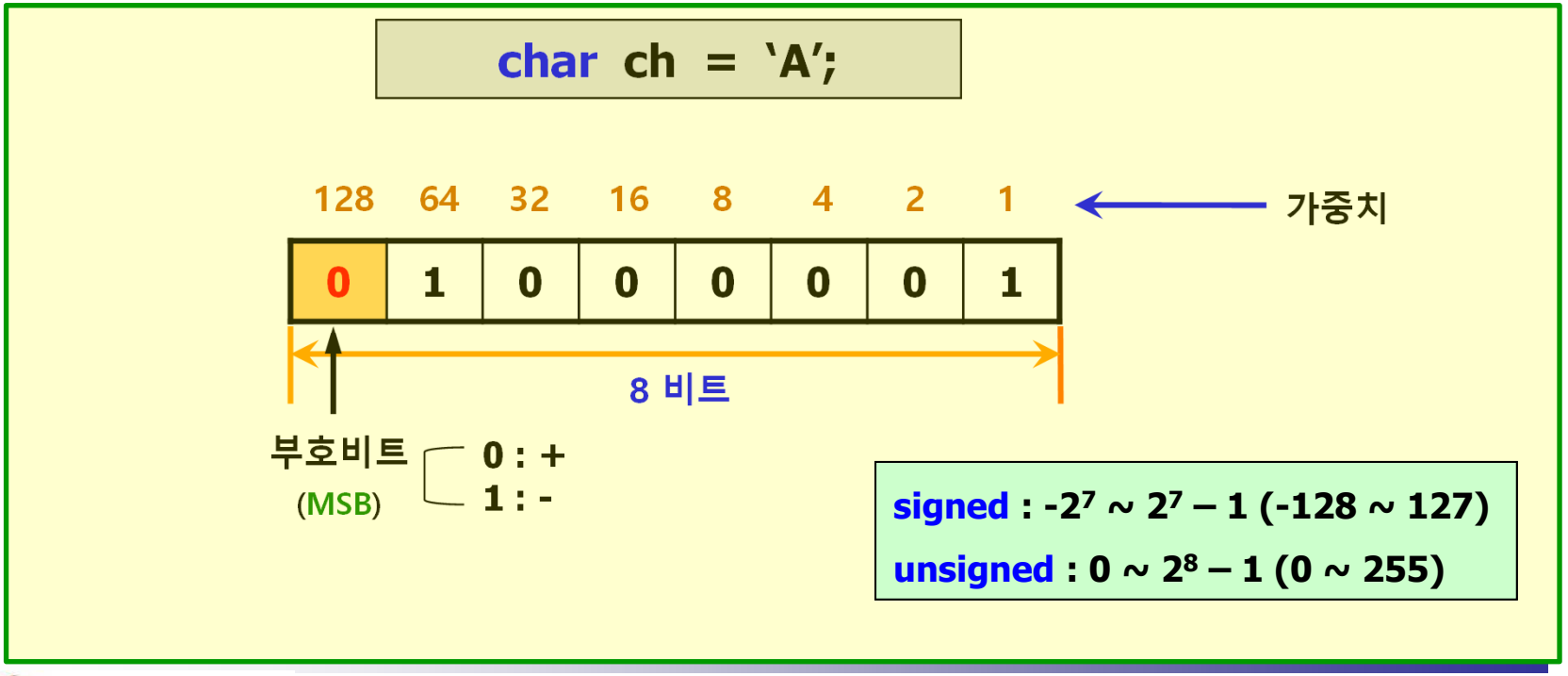


자료형 (7/12)

● 문자형(Character Type)

❍ 컴퓨터에서 문자는 정수 값을 부여하여 처리

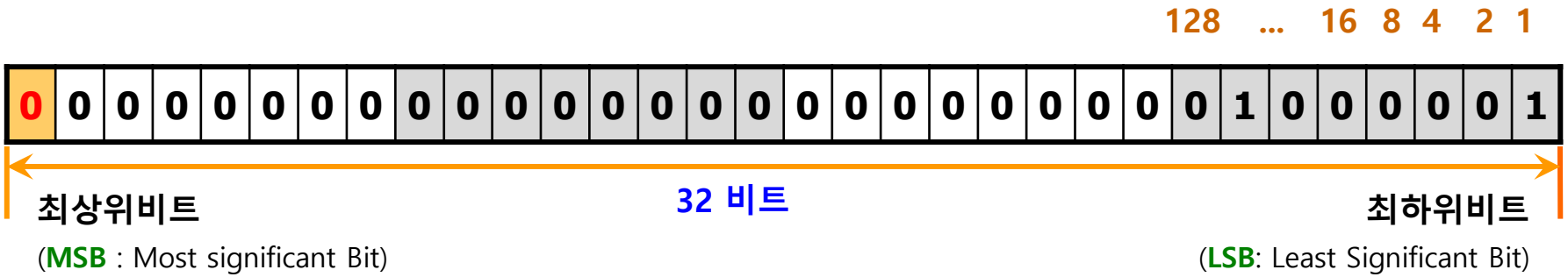
- **ASCII**(American Standard Code for Information Interchange)
  - 컴퓨터는 문자 상수를 자동적으로 내부에서 ASCII 코드 값으로 변환한다.
  - 문자도 기본적인 산술연산이 가능하다.



자료형 (8/12)

- **정수형**(Integer Type)
  - 정수, 즉 분수(소수점 이하) 부분을 가지지 않는 수

```
int i = 65;
```



정수 표현 범위 : 4bytes(32bits)

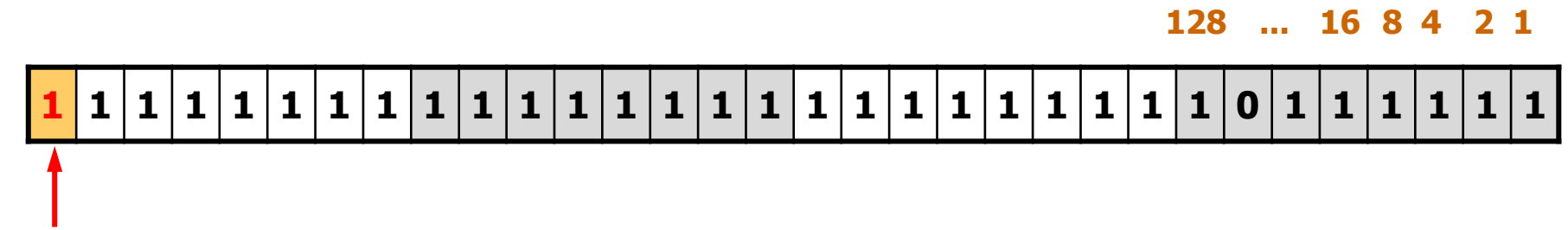
int 로 선언 시 : -2,147,483,648 ~ 2,147,483,647 (-2^{31} ~ $2^{31}-1$)

unsigned 로 선언 시 : 0 ~ 4,294,967,295 (0 ~ $2^{32}-1$)

자료형 (9/12)

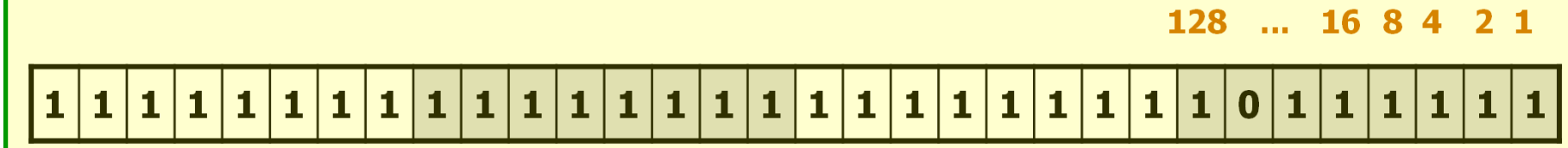
- **정수형:** 음수 표현
  - 음수 표현: 2의 보수로 표현

```
int i = -65;
```



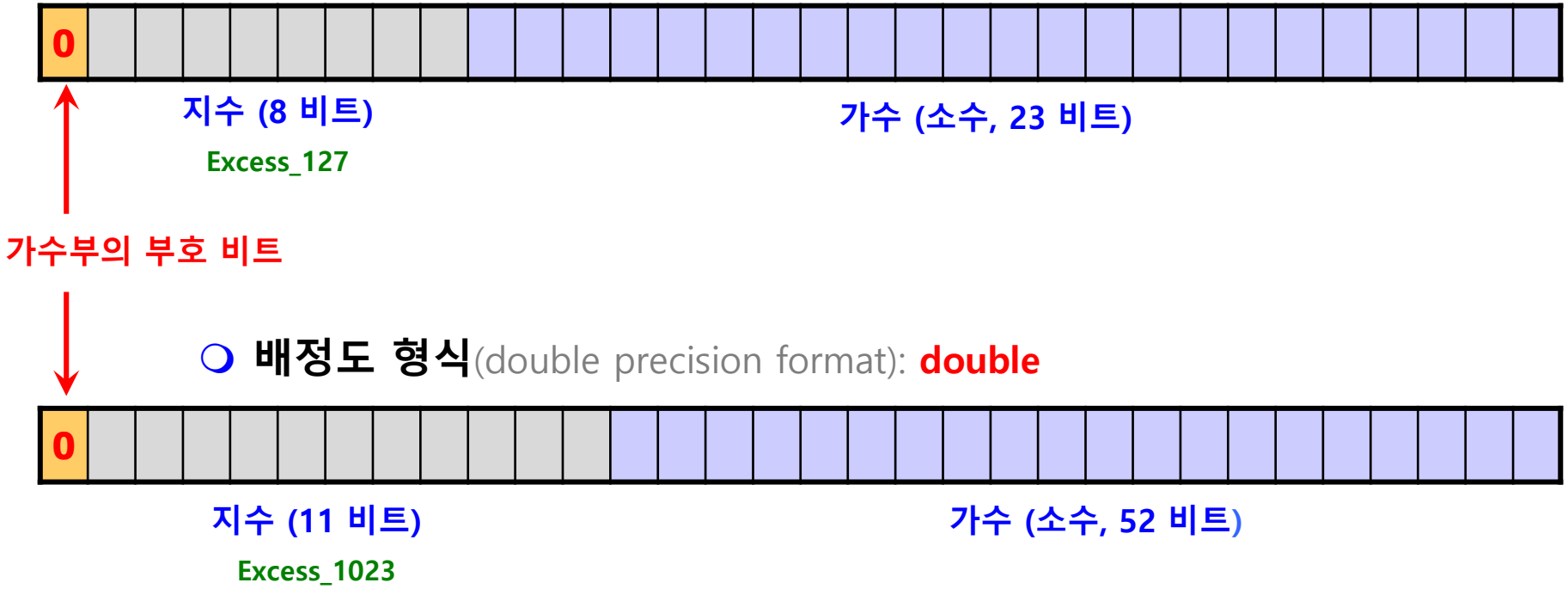
부호비트(MSB)가 1 이면 음수(2의 보수로 표현)

```
unsigned int i = 4294967231;
```



자료형 (10/12)

- **실수형**(Real Type)
  - **실수:** 소수점 이하의 값을 가진다(정수 부분과 소수 부분).
  - **단정도 형식**(single precision format): **float**

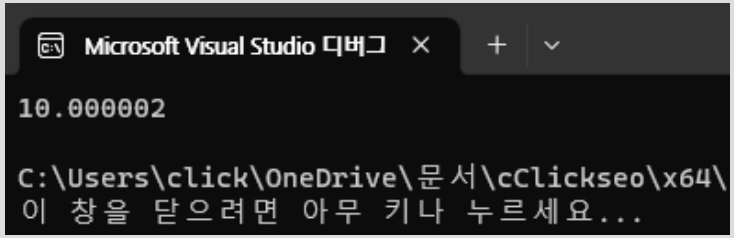


  - **배정도 형식**(double precision format): **double**

자료형 (11/12)

- **실수형:** 부동 소수점 오차
  - **부동 소수점 오차**
    - **컴퓨터는 우리가 표현하고자 하는 실수의 값을 정확하게 표현하는 것이 아니라, 아주 가까운, 전혀 문제가 없을 만큼의 근사치를 통해서 실수를 표현한다.**
      - 부동소수점 오차에 대한 문제는 컴퓨터의 문제이지, C 언어의 문제는 아니다.
      - 따라서, 다른 프로그래밍 언어들도 기본적으로 지니고 있는 문제이다.

```c
#include <stdio.h>
int main(void)
{
        int     i;
        float   f = 0.0f;

        for(i=0;  i<100;  i++)
                f += 0.1f;

        printf("%f \n", f);

        return 0;
}
```



자료형 (12/12)

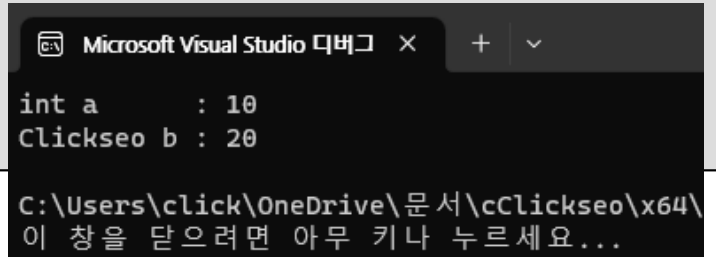
● typedef 지정자(Typedef Specifier)

❍ 식별자를 별칭 형식으로 선언

- 주로 복잡한 유형 이름을 대체하는데 사용되는 방법이다.

```c
#include <stdio.h>

typedef int Clickseo;

int main(void)
{
        int     a = 10;
        Clickseo  b = 20;

        printf("int a      : %d\n", a);
        printf("Clickseo b : %d\n", b);

        return 0;
}
```

```c
// stdio.h 파일에서 typedef 선언 적용 예
#include <stdio.h>

typedef unsigned int size_t;
```



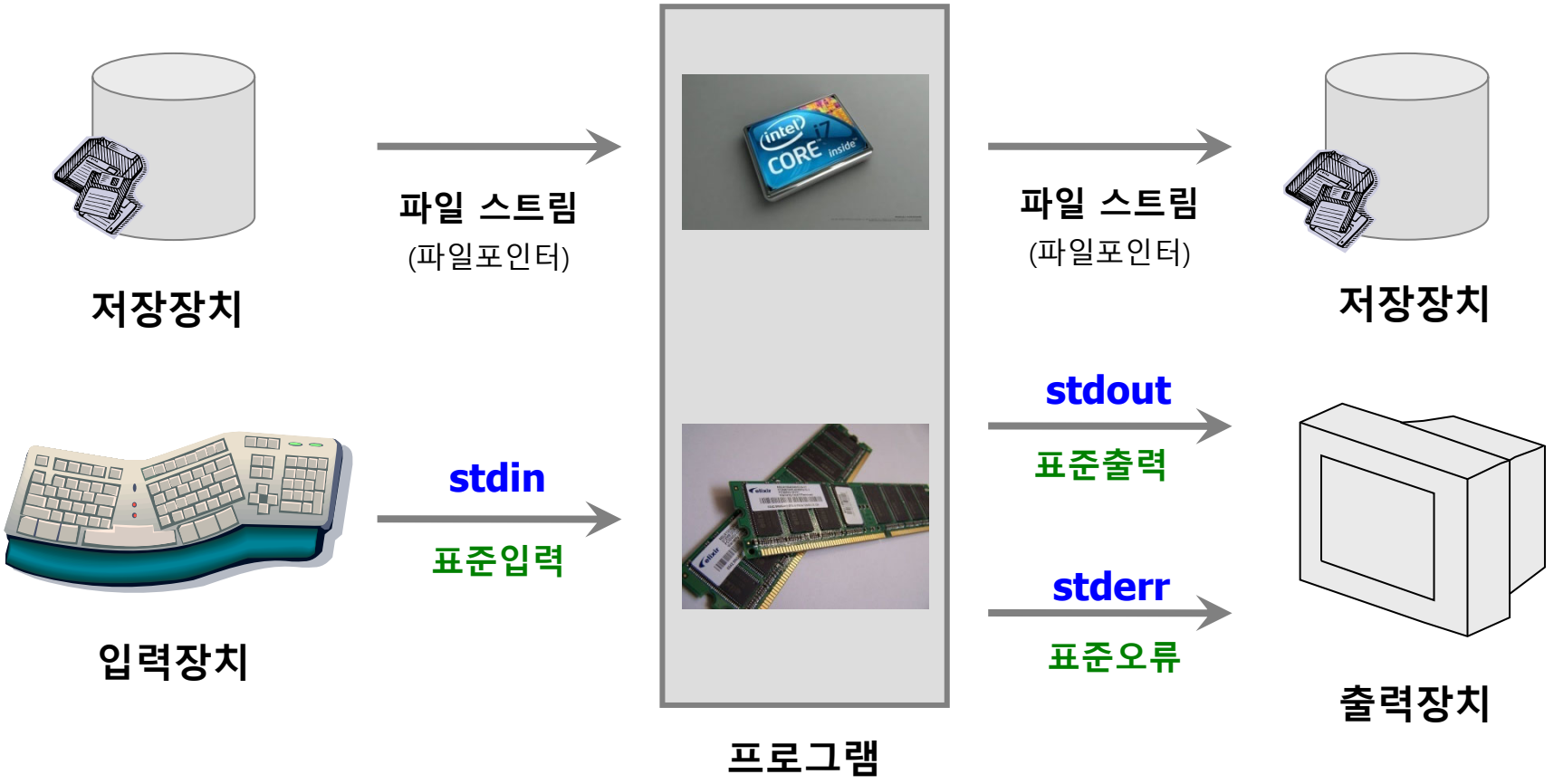
표준 입출력

백문이불여일타(百聞而不如一打)

- C 언어 개요
- 데이터 표현
- **표준 입출력**
  - 문자 입출력
  - 형식화된 입출력

표준 입출력 (1/2)

- **스트림(stream):** 표준 입출력과 파일 입출력



표준 입출력 (2/2)

● 스트림(Stream)

❍ 데이터의 논리적 흐름

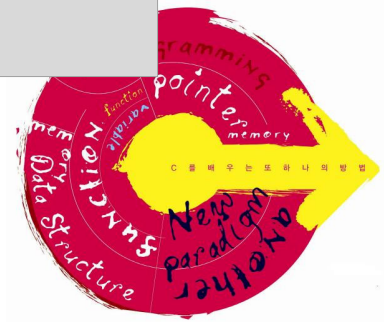
- 개발자와 하드웨어 장치 사이에 존재하는 추상적 계층
- 하드웨어 장치 파일과 연결되어 데이터 전송을 중재

❍ 표준 입출력: <stdio.h>

표준 입출력	기능	장치
stdin	표준 입력	키보드
stdout	표준 출력	모니터
stderr	표준 오류	모니터
stdprn	표준 프린터	프린터(LPT1)
stdaux	표준 보조 입출력	직렬 포트(COM1)

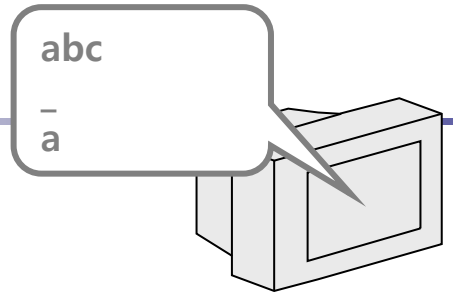
표준 입출력

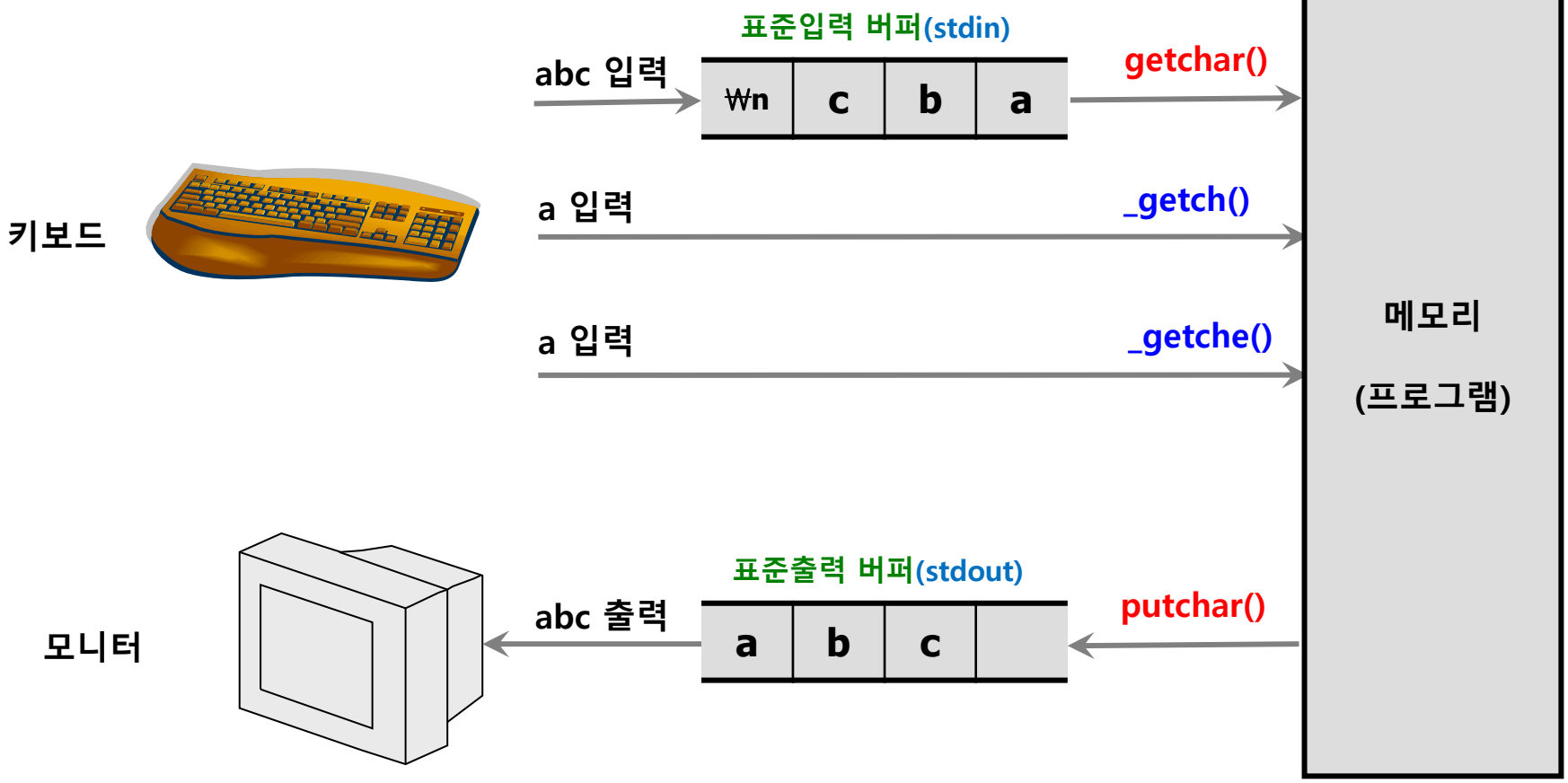
문자 입출력



문자 입출력 (1/4)

- 문자 입출력 함수





문자 입출력 (2/4)

- **표준 입출력 함수:** getchar, putchar
  - 형식화되지 않은 입출력: 문자 입출력

함수의 선언	기 능	헤더 파일
int getchar(void);	stdin 에서 다음 한 문자를 읽어 들인다. getc(stdin) 와 동일	stdio.h
int putchar(int ch);	stdout 에 한 문자를 출력한다. putc(ch, stdout) 와 동일	
int getc(FILE *stream);	지정된 입력 stream 으로부터 다음 문자를 읽어 들인다.	
int putc(int ch, FILE *stream);	문자 ch 를 지정된 출력 stream 에 출력한다.	
int ungetc(int ch, FILE *stream);	지정된 stream 에 문자 ch 를 되돌려 놓는다.	

문자 입출력 (3/4)

● 비 표준 입출력 함수: _getch, _getche, _putch

함수의 선언	기 능	헤더 파일
int _getch(void);	한 문자를 읽어 들인다(입력한 문자는 화면에 표시되지 않는다).	conio.h
int _putch(int ch);	한 문자를 출력('\n'을 CR/LF의 조합으로 변환하지 않는다).	
int _getche(void);	한 문자를 읽어 들인다(입력한 문자는 화면에 출력).	
int _kbhit(void);	키의 입력 여부 조사(키의 입력이 있을 경우 0이 아닌 수를 반환)	

비 표준 입력 함수 : 직접 문자 입력 함수
- getchar 함수보다 빠른 입력 속도

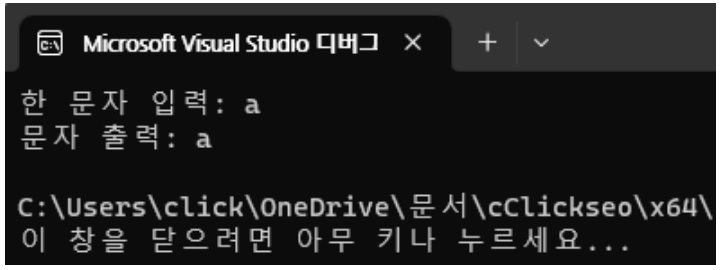
// 헤더 파일 : conio.h
콘솔 입출력 함수를 제공하는 헤더 파일
MS-DOS 시절부터 사용되었다.
지금은 C 언어 표준도 아니고 POSIX 함수도 아니다.
GNU/Linux 와 macOS 에서는 사용할 수 없다.

```
clickseo@clickseo-VirtualBox:~/gcc$ cat _conio.c
#include <stdio.h>
#include <conio.h>

int main(void)
{
        char    ch1, ch2;

        ch1 = _getch();
        getchar();
        ch2 = _getche();

        putchar(ch1);
        putchar(ch2);

        return 0;
}
clickseo@clickseo-VirtualBox:~/gcc$ gcc _conio.c
_conio.c:2:10: fatal error: conio.h: 그런 파일이나 디렉터리가 없습니다
 #include <conio.h>
          ^~~~~~~~~
compilation terminated.
clickseo@clickseo-VirtualBox:~/gcc$
```

문자 입출력 (4/4)

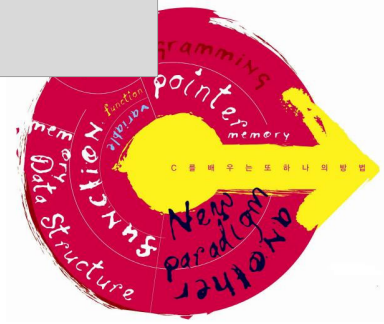
예제 1-4: 한 문자 입출력 -- getchar / putchar

```
#include <stdio.h>
int main(void)
{
        char    ch;

        printf("한 문자 입력: ");
        ch = getchar();

        printf("문자 출력: ");
        putchar( ch );
        putchar('\n');

        return 0;
}
```



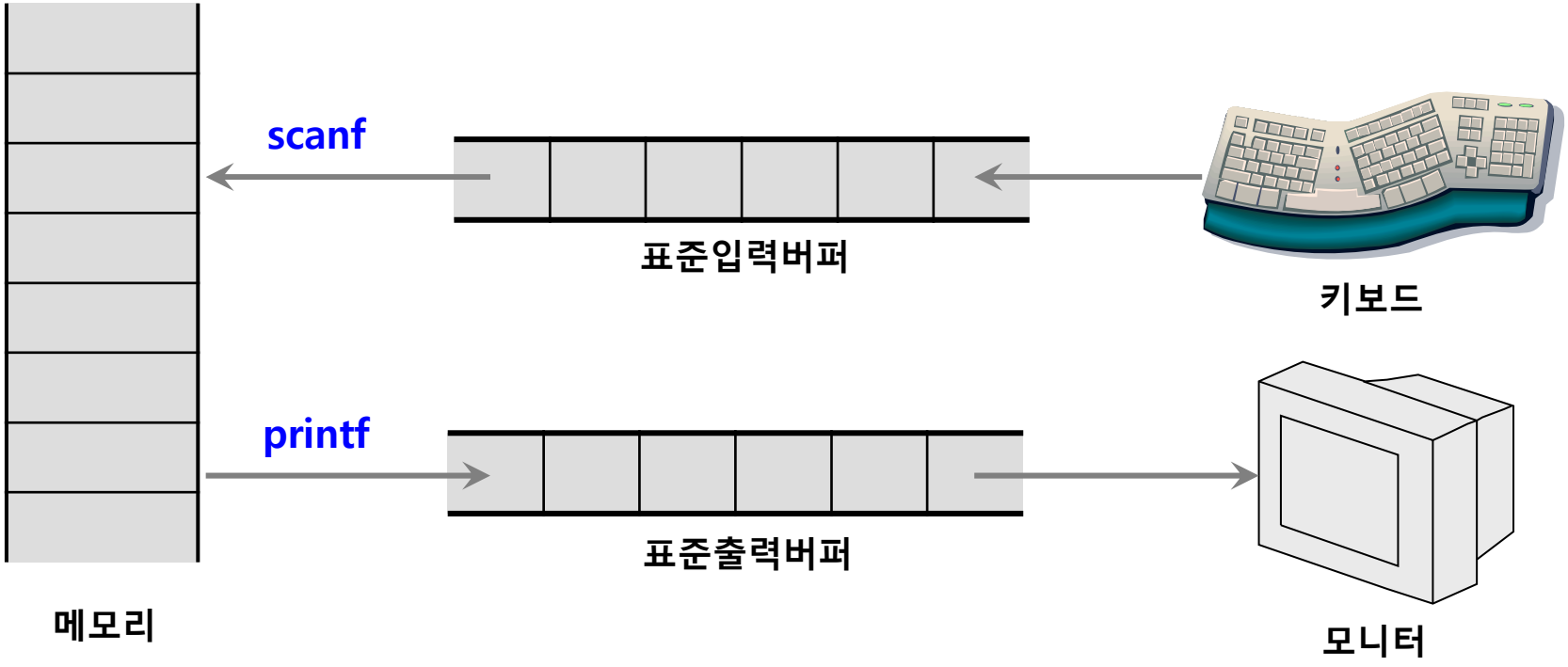
표준 입출력

형식화된 입출력



형식화된 입출력 (1/16)

- **표준 입출력 함수**
  - 표준 입출력
    - **표준 입력**(standard input): **키보드**
    - **표준 출력**(standard output): **모니터**



형식화된 입출력 (2/16)

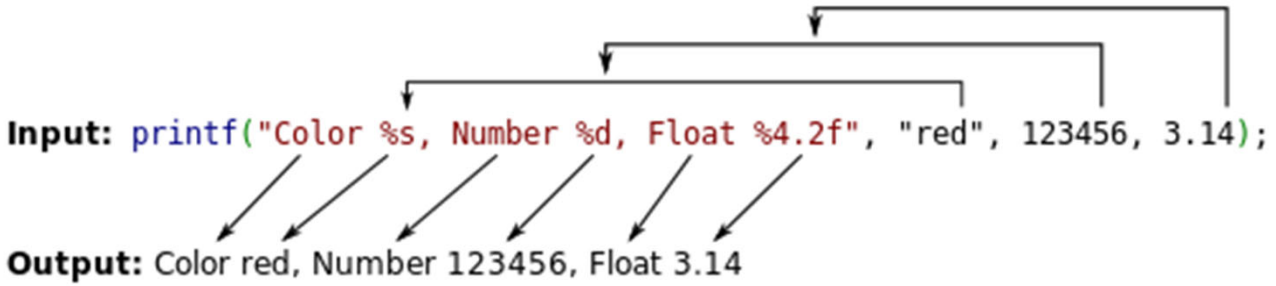
- **표준 입출력 함수:** 콘솔 입출력
  - 형식화된 입출력: **printf 와 scanf 계열 함수**

함수 원형 및 기능		헤더 파일
~~int printf(const char *control string, ...);~~	// until C99	stdio.h
문자열을 주어진 서식에 맞춰 표준 출력(stdout) 버퍼로 출력 한다.		
int printf(const char *restrict format, ...);	// since C99	
int printf_s(const char *restrict format, ...);	// since C11	
~~int scanf(const char *control string, ...);~~	// until C99	
표준 입력(stdin) 버퍼로부터 형식화된 바이트 문자를 읽어 들인다.		
int scanf(const char *restrict format, ...);	// since C99	
int scanf_s(const char *restrict format, ...);	// since C11	

형식화된 입출력 (3/16)

- **형식화된 출력: printf 계열 함수**
  - **필드 명세**
    - **% 부호와 변환 코드는 필수**이며, 나머지는 생략 가능

% <플래그> <출력할 최소 자릿수> <정확도 변환 지정자> <크기> 변환코드



  - 일반적으로 출력은 오른쪽으로 맞춰진다.
  - 맞춤을 반대로 하기 위해서는 플래그를 마이너스 기호(-)로 지정한다.

형식화된 입출력 (4/16)

● 형식화된 출력: printf 계열 함수 서식 지정자

서식 지정자	의 미	데이터형
%c	인자를 **char** 형의 한 문자로 출력	문자형
%d 또는 %i	인자를 **부호 있는 10진수**로 출력	정수형
%u	인자를 **부호 없는 10진수**로 출력(unsigned)	정수형
%o	인자를 **부호 없는 8진수**로 출력	정수형
%x, %X	인자를 **부호 없는 16진수**로 출력	정수형
%f	인자를 **float** 나 **double** 형의 실수로 출력	부동 소수형
%e, %E	인자를 **과학 기술 계산용 표기법**으로 출력	부동 소수형
%g, %G	%e 와 %f 중 더 짧은 표현 선택	부동 소수형
%s	**문자열** 출력	문자열 포인터
%p	**포인터의 주소**를 출력	포인터
%%	**%부호**를 그대로 출력	

형식화된 입출력 (5/16)

- **형식화된 출력:** printf 계열 함수
  - 출력을 위한 양식

크기	변환 코드	자료형	사용 예
none	c	char	%c
h	d / i	short int	%hd / %hi
h	u	unsigned short int	%hu
none	d / i	int	%d / %i
none	u	unsigned int	%u
l / L	d / i	long int	%ld / %li
l / L	u	unsigned long int	%lu
ll / LL	d / i	long long int	%lld / %lli
ll / LL	u	unsigned long long int	%llu
none	f	float	%f / %F
l	f	double	%lf / %LF
L	f	long double	%LF / %LF

형식화된 입출력 (6/16)

- **형식화된 출력:** printf 계열 함수의 부동소수점 출력
  - %e, %E : 부동소수점 표현 방식에 의한 출력

$3.1245e+2 \rightarrow 3.1245 \times 10^{+2}$

$2.45e-4 \rightarrow 2.45 \times 10^{-4}$

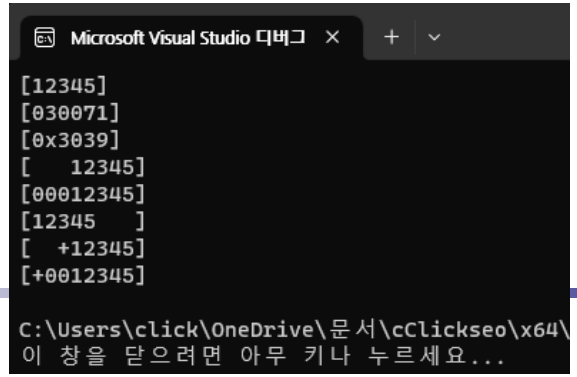
  - %g, %G
    1. **%f 형태 출력:** 표현하고자 하는 실수의 값이 **소수점 이하 6자리인 경우**
    2. **%e 형태 출력:** 이 범위를 넘길 경우 %e 의 형태로 출력

```
#include <stdio.h>
int main(void)
{
        printf("%g \n", 0.00123);          // 0.00123 출력
        printf("%G \n", 0.000123);         // 0.000123 출력
        printf("%g \n", 0.0000123);        // 1.23e-05 출력
        printf("%G \n", 0.00000123);       // 1.23E-06 출력
        return 0;
}
```

형식화된 입출력 (7/16)

예제 1-5: printf 계열 함수의 필드 명세 -- 정수형 데이터

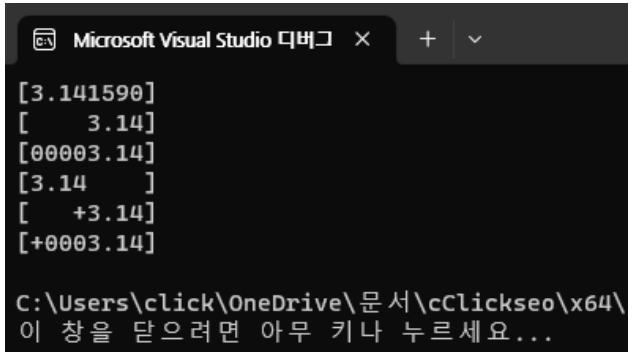
```
#include <stdio.h>
int main(void)
{
        int     a = 12345;

        printf("[%d]\n", a);          // 정수 출력: 10진수
        printf("[%#o]\n", a);         // 정수 출력: 8진수
        printf("[%#x]\n", a);         // 정수 출력: 16진수

        printf("[%8d]\n", a);         // 출력할 최소 자릿 수: 8
        printf("[%08d]\n", a);        // 0 플래그: 빈 칸을 공백으로 채움

        printf("[%-8d]\n", a);        // - : 왼쪽 정렬
        printf("[%+8d]\n", a);        // + : 부호(+) 출력
        printf("[%+08d]\n", a);

        return 0;
}
```



형식화된 입출력 (8/16)

예제 1-6: printf 계열 함수의 필드 명세 -- 실수형 데이터

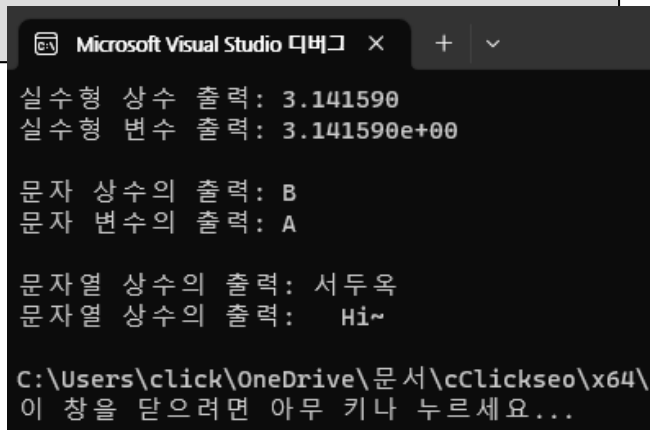
```c
#include <stdio.h>
int main(void)
{
        double  d = 3.14159;

        printf("[%f]\n", d);  // 실수 출력

        printf("[%8.2f]\n", d);       // 출력할 최소 자릿 수(8)와 소수점 2번째 자리까지
        printf("[%08.2f]\n", d);

        printf("[%-8.2f]\n", d);      // - : 왼쪽 정렬
        printf("[%+8.2f]\n", d);      // + : 부호(+) 출력

        printf("[%+08.2f]\n", d);

        return 0;
}
```



형식화된 입출력 (9/16)

예제 1-7: 다양한 데이터 출력

```
#include <stdio.h>
int main(void)
{
        char            ch = 'A';
        const double    d = 3.14159;

        printf("실수형 상수 출력: %lf\n", 3.14159 );    // long double
        printf("실수형 변수 출력: %le\n\n", d );        // 지수형 소수 형태 출력

        printf("문자 상수의 출력: %c\n", 'B' );
        printf("문자 변수의 출력: %c\n\n", ch );

        printf("문자열 상수의 출력: %s\n", "서두옥" );
        printf("문자열 상수의 출력: %5.3s\n", "Hi~ Clickseo" );

        return 0;
}
```



형식화된 입출력 (10/16)

● 형식화된 입력: scanf 계열 함수

❍ 필드 명세

- % 부호로 시작하며, 정확히 하나의 필드(변수)를 기술한다.

```
% <플래그> <입력 받을 최대 자리 수> <크기> 변환코드
```

  - 문자 형식 코드(%c)를 제외하고, scanf 계열 함수는 모든 여백은 건너뛴다.
    - » 즉, 입력의 모든 공백, 탭, 줄 바꿈을 무시한다.

- 입력 변환은 다음과 같은 상황이 발생할 때까지 입력 문자를 처리한다.
  - 파일의 끝(EOF, end of file)에 도달했을 때
  - 부적절한 문자를 만났을 때
  - 읽은 문자의 수가 명시된 최대 필드 길이에 도달했을 때

- 주소 연산자(Address Operator) : & 부호
  - 모든 필드 명세에 대하여 대응되는 변수가 주소 리스트에 존재한다.

형식화된 입출력 (11/16)

● 형식화된 입력: scanf 계열 함수

❍ scanf 계열 함수는 문자열 입력에 적당한 함수가 아니다.

- 문자열에 포함된 공백 문자로 인해 긴 문장을 입력 받기에 상당히 불편
- 단어 단위의 입력에는 **scanf 계열 함수**가 적합할지 모르겠으나,
  문자열의 입력은 **gets 계열 함수**를 사용하는 편이 훨씬 효율적이다.

❍ scanf 계열 함수의 필터링 기능

- **입력 받을 대상 문자를 선별**(Filtering)**하기 위해 사용**
  - 원하지 않는 문자의 입력을 막을 수 있는 유용한 기능

```
%[A-Z]          // 대문자 A-Z 사이의 문자만 받아들인다.
%[A-Za-z0-9]    // 영문자 및 숫자만 받아들인다.
%[^0-9]         // 숫자를 제외한 모든 문자를 받아들인다.
```

형식화된 입출력 (12/16)

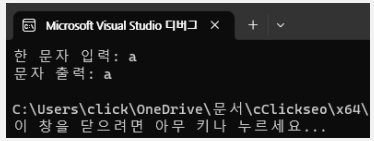
- **형식화된 입력:** scanf 계열 함수

서식 지정자	의 미	자료형
%c	입력 데이터를 하나의 **문자 상수**로 읽어 들임	문자형
%d, %i	입력 데이터를 **10진수**로 읽어 들임	정수형
%u	**부호 없는 10진수** 정수를 읽어 들임	정수형
%o	**부호 없는 8진수** 정수로 읽어 들임	정수형
%x	**부호 없는 16진수** 정수로 읽어 들임	정수형
%f, %e, %g	입력 데이터를 **실수**로 읽어 들임(**float**)	실수형
%lf, %LF	입력 데이터를 **실수**로 읽어 들임(**double**, **long double**)	실수형
%p	포인터를 읽어 들임	포인터
%s	입력 데이터를 문자열로 읽어 들임	문자 포인터
%n	읽어 들인 문자 수를 저장	정수 포인터
%[]	입력 대상 문자의 필터링(scanset의 지정)	

형식화된 입출력 (13/16)

예제 1-8: 한 문자 입출력 -- scanf Visual Studio Community

```
#include <stdio.h>
int main(void)
{
        char    ch;

        printf("한 문자 입력: ");
        scanf("%c", &ch);

        printf("문자 출력: %c \n", ch);

        return 0;
}
```



// scanf 함수 사용 시 Visual Studio 에러 메시지

error C4996: 'scanf': This function or variable may be unsafe. Consider using scanf_s instead.

To disable deprecation, use _CRT_SECURE_NO_WARNINGS. See online help for details.

---

1) 첫 번째 해결방법 : #define _CRT_SECURE_NO_WARNINGS // until Visual Studio 2017

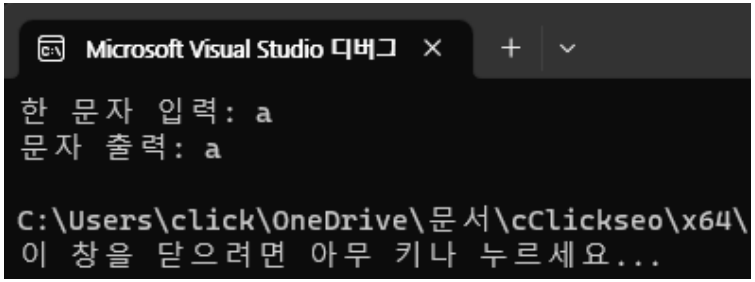
2) 두 번째 해결방법 : #pragma warning(disable:4996)

3) 세 번째 해결방법 : scanf_s 함수 사용

형식화된 입출력 (14/16)

예제 1-9: 한 문자 입출력 -- scanf_s Visual Studio Community

```
#include <stdio.h>
int main(void)
{
        char    ch;

        printf("한 문자 입력: ");
        scanf_s("%c", &ch, (int)sizeof(ch));      // scanf("%c", &ch);

        printf("문자 출력: %c \n", ch);

        return 0;
}
```




```
// scanf_s 함수 사용 시 Visual Studio 2019 경고 메시지
warning C4473: 'scanf_s': 서식 문자열에 대한 인수가 충분하게 전달되지 않았습니다.
-------------------------------------------------------------------------------------------------------
* 해결방법 scanf_s("%c",  &ch, sizeof(ch) );          // 인자로 버퍼의 크기 지정
```

```
// scanf_s 함수 사용 시 Visual Studio 2022 경고 메시지
warning C4477: 'scanf_s' : 서식 문자열 '%c'에 'unsigned int' 형식의 인수가 필요하지만 variadic 인수 2의 형식이 'size_t'입니다.
-------------------------------------------------------------------------------------------------------
* 해결방법 scanf_s("%c",  &ch, (int)sizeof(ch) );      // 인자로 버퍼의 크기 지정
```

형식화된 입출력 (15/16)

예제 1-9: 한 문자 입출력 -- scanf_s GNU/Linux GCC

```
clickseo@clickseo-VirtualBox: ~/gcc
파일(F)  편집(E)  보기(V)  검색(S)  터미널(T)  도움말(H)
clickseo@clickseo-VirtualBox:~/gcc$ cat scanf_s.c
#include <stdio.h>

int main(void)
{
        char    ch;

        printf("한 문자 입력 : ");
        scanf_s("%c", &ch);

        printf("문자 출력 : %c \n", ch);

        return 0;
}
clickseo@clickseo-VirtualBox:~/gcc$ gcc scanf_s.c
scanf_s.c: In function ‘main’:
scanf_s.c:8:2: warning: implicit declaration of function ‘scanf_s’; did you mea
n ‘scanf’? [-Wimplicit-function-declaration]
  scanf_s("%c", &ch);
  ^~~~~~~
  scanf
/tmp/ccjFqAcr.o: In function `main':
scanf_s.c:(.text+0x3c): undefined reference to `scanf_s'
collect2: error: ld returned 1 exit status
clickseo@clickseo-VirtualBox:~/gcc$
```

// GNU/Linux 환경에서 GCC 는 scanf_s 함수를 지원하지 않는다.

warning: implicit declaration of function 'scanf_s';

did you mean 'scanf'?

[-Wimlicit-function-declaration]

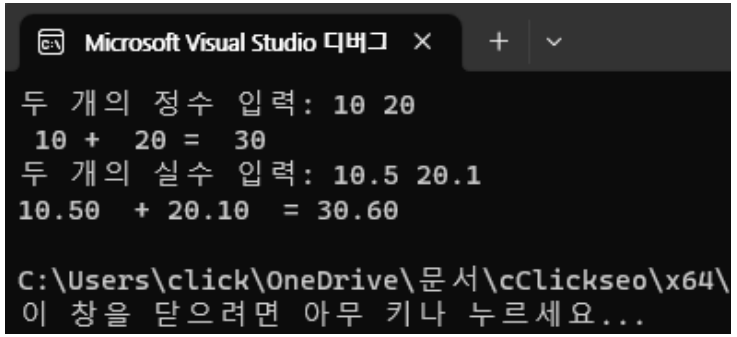
---

undefined reference to 'scanf_s'

형식화된 입출력 (16/16)

예제 1-10: 두 개의 정수와 실수 입력과 계산 그리고 출력

```c
#include <stdio.h>
int main(void)
{
        // 두 개의 정수 입출력과 계산
        int     a, b;

        printf("두 개의 정수 입력: ");
        scanf_s("%d  %d", &a, &b);   // scanf("%d  %d",  &a,  &b);
        printf("%3d + %3d = %3d\n", a, b, a + b);

        // 두 갱의 실수 입출력과 계산
        double  c, d;

        printf("두 개의 실수 입력: ");
        scanf_s("%lf %lf", &c, &d);  // scanf("%lf %lf", &c, &d);
        printf("%.2f  + %.2f  = %.2f \n", c, d, c + d);

        return 0;
}
```



참고문헌

[1] 서현우, "혼자 공부하는 C 언어 : 1:1 과외 하듯 배우는 프로그래밍 자습서", 한빛미디어, 2023.

[2] Paul Deitel, Harvey Deitel, "C How to Program", Global Edition, 8/E, Pearson, 2016.

[3] Kamran Amini, 박지윤 번역, "전문가를 위한 C : 동시성, OOP부터 최신 C, 고급 기능까지!", 한빛미디어, 2022.

[4] 서두옥, "(열혈강의) 또 하나의 C : 프로그래밍은 셀프입니다", 프리렉, 2012.

[5] Behrouz A. Forouzan, Richard F. Gilberg, 김진 외 7인 공역, "구조적 프로그래밍 기법을 위한 C", 도서출판 인터비젼, 2004.

[6] Brian W. Kernighan, Dennis M. Ritchie, 김석환 외 2인 공역, "The C Programming Language", 2/E, 대영사, 2004.

[7] "C reference", cppreference.com, 2023 of viewing the site, https://en.cppreference.com/w/c.