기초 연산

01 변수

02 자료형

03 변수와 메모리

04 자료형과 메모리

05 형변환

06 연산자

07 입출력

<변수 (variable)>

[변수]
- 데이터를 저장하기 위해 프로그램에 의해 이름을 할당받은 메모리 공간
- 프로그램이 실행되는 동안 언제든지 저장된 값이 변경될 수 있는 공간
  (↔ 상수(constant))

[변수 생성 규칙]
- 변수의 이름은 영문자(대소문자), 숫자, 언더스코어(_)로만 구성된다.
- 변수의 이름은 숫자로 시작될 수 없다.
- 변수의 이름 사이에는 공백을 포함할 수 없다
- 변수의 이름으로 C언어에서 미리 정의된 키워드는 사용할 수 없다.

<자료형 (Data Type)>

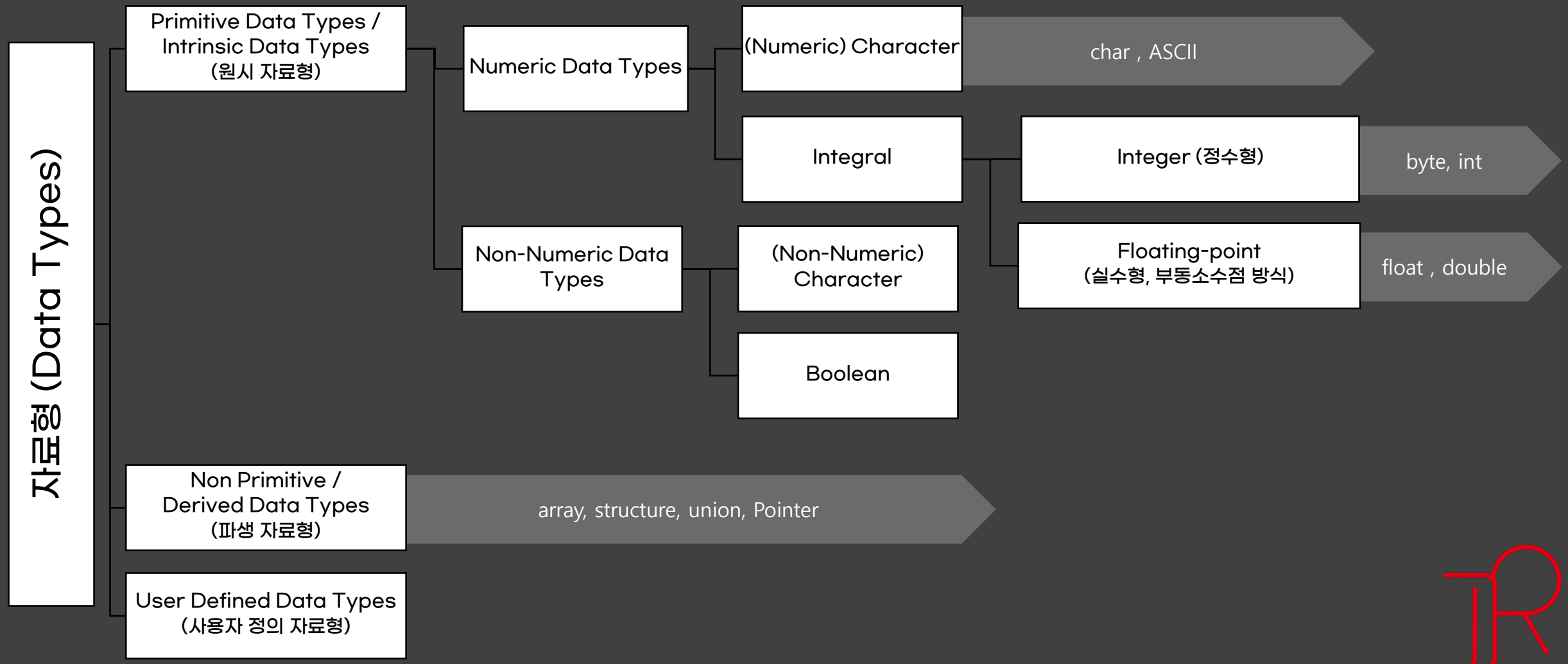
자료형 : 여러 종류의 데이터를 식별하는 분류

[C언어에서 주로 사용되는 자료형]

- **정수 자료형 (int)**
  - 정수 자료형은 말그대로 정수 자료형 long과 short는 정수 표현 범위를 결정한다.
- **실수 자료형 (float, double)**
  - 실수 자료형은 실수를 나타내는 자료형으로 소수를 가진다는 특징이 있다.
  - long과 short는 소수점 아래 자릿수를 이야기한다
  - float는 소수점 6번째 자리까지 만 나타내고 double은 더 많은 자릿수를 나타낸다. 따라서 더 정확한 실수 연산을 할 때에는 double을 사용한다.
- **문자 자료형 (char)**
  - char 문자 자료형은 우리가 문자라고 이야기 하지만 유니코드와 아스키코드와 같이 문자를 숫자 데이터로 정리해 놓은 자료형이라 볼 수 있다.
- **논리 자료형 (Boolean)**
  - False(0)와 True(1)로 이루어진 자료형
  - Boolean 자료형이 실제로 존재하진 않는다.

<자료형 (Data Type) - 추가 자료>

[자료형의 종류]



<변수와 메모리>

[변수 선언 방법]

[자료형] [변수명]

[메모리]

- 메모리는 데이터를 저장할 수 있는 수많은 메모리 셀의 집합
- 컴퓨터는 메모리에 데이터를 저장하고 CPU 를 사용해 연산
- CPU 는 필요한 데이터를 메모리에서 가져와 연산하고
  그 결과를 다시 메모리에 저장하는 방식으로 동작

[변수와 메모리 구조]

- 변수를 선언하게 되면 변수는 메모리의 공간을 할당받는다.
- 이때 변수의 메모리 공간의 위치를 메모리 주소(address)라 하고
- 변수는 기본적으로 메모리의 주소(address)를 기억하는 역할을 한다.
- 변수를 참조하게 되면 변수의 메모리 주소로 가서 메모리 공간에 저장되어 있는 데이터를 참조

<자료형과 메모리>

[자료형과 메모리]
정수형, 실수형, 문자형 등 다양한 자료형이 존재
But 컴퓨터는 오직 0과 1 이진수로 이루어져 있다.
컴퓨터는 각 자료형을 이진수로 변환한다

[비트(bit)와 바이트(byte)]
컴퓨터는 모든 데이터를 2진수로 표현
비트(bit) : 컴퓨터가 데이터를 처리하기 위해 사용하는 데이터의 최소 단위 (0과 1)
바이트(byte) : 비트가 8개 모여 구성되며, 한 문자를 표현할 수 있는 최소 단위

<자료형과 메모리>

[자료형에 따른 메모리 크기]

- 부호 유무
  - signed : 부호가 있다
  - unsigned : 부호가 없다

- 메모리 크기
  - short : 2 byte
  - long : 4 byte
  - long long : 8 byte

자료형	메모리 크기	값의 범위
char	1 Bytes	-128 ~ 127
(signed) short	2 Bytes	-32,768 ~ 32,767
Int	4 Bytes	-2,147,483,648 ~ 2,147,483,647
(signed) long	4 Bytes	-2,147,483,648 ~ 2,147,483,647
(signed) long long	8 Byte	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
unsinged char	1 Bytes	0 ~ 255
unsigned short	2 Bytes	0 ~ 65,535
unsigned int	4 Bytes	0 ~ 4,294,967,295
unsigned long	4 Bytes	0 ~ 4,294,967,295
float	4 Bytes	1.2E-38 ~ 3.4E38
double	8 Bytes	2.2E-308 ~ 1.8E308
void	0 Bytes	값이 없다

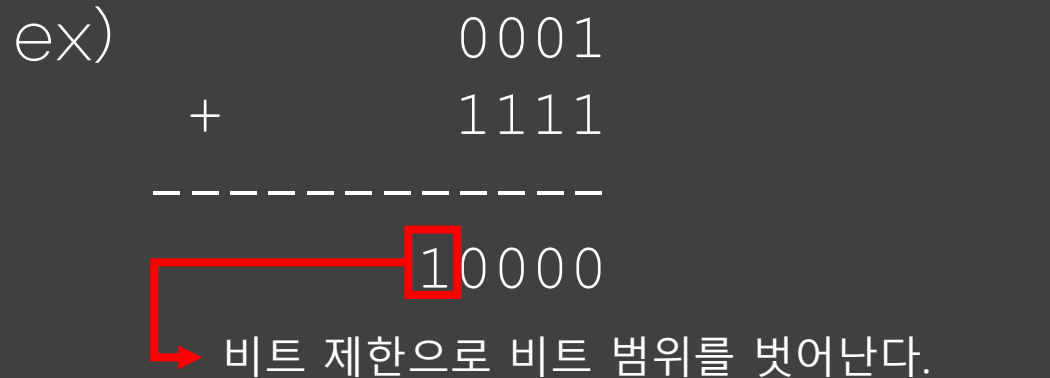
<자료형과 메모리>

[부호가 있는 (signed) 자료형 - 2의 보수]

- (-3) + 3 = 0과 같이 절댓값이 동일하고

  부호가 서로 다른 두수의 합이 0이 되기 위해서 2의 보수를 사용한다
- 2의 보수를 나타 내는 법 : 어떤 수를 비트 반전(NOT 연산자로 0을 1로,

  1을 0으로 반전 시키는 것)을 한 뒤 1을 더하면 어떤 수에 대한 2의 보수이다.
- 원리 비트 제한으로 비트 범위를 벗어나는 것을 이용

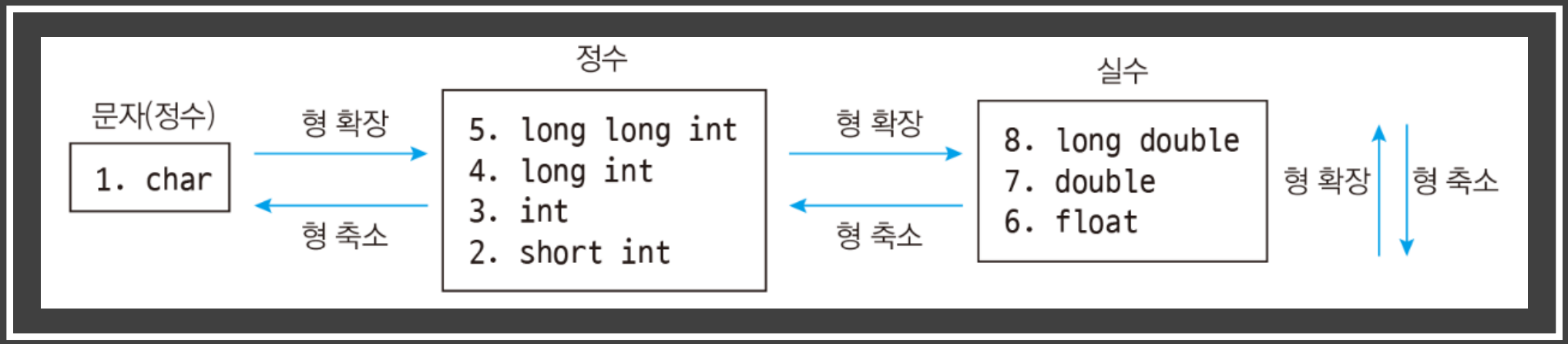


<형 변환 (Type Casting)>

[형 변환]

형 변환 : 데이터의 자료형을 다른 자료형으로 변경하는 것
자료형 마다 주어진 메모리의 크기가 다르고
이진수로 저장하는 방식이 다르기 때문에 중요하다

[자료형의 메모리 크기 - 형 확장과 축소]



<형 변환 (Type Casting)>

[명시적 형변환 (Explicit Casting)]

사용자가 원하는 시점에 형변환을 하는 것
변수나 값앞에 괄호 ( ) 를 사용하여 변환할 자료형을 명시

```c
#include <stdio.h>

int main()
{
    int a = 5;
    int b = 2;
    float c = (float)a / b;
    printf("%f",c);
}
```

```c
#include <stdio.h>

void main()
{
  int num;
  float Fnum = 123.953

  num = Fnum;
  printf("%d\n", num);
  Fnum = num;
  printf("%f\n", Fnum);

  printf("%f\n", Fnum + num);
  printf("%d\n", Fnum + num);
}
```

[암시절 형변환 (Implicit Casting)]

자료형의 우선순위에 따라
데이터의 자료형이 자동적으로 바뀌는 형 변환
자료형의 메모리의 크기가 클수록 우선순위가 높다

<연산자 (operator)>

[연산자 종류]

구분	연산자
대입 연산자	=
산술 연산자	+, -, *, /, &, ++, --
관계 연산자	<, >, <=, >=, ==, !=
논리 연산자	&&, ||, !
할당 연산자	+=, -=, *=, /=, %= 등...
삼항 연산자	?
비트 연산자	&, |, ~, ^, <<, >>

<연산자 (operator)>

[대입 연산자 (Assignment Operator)]

- [변수 식별자] = [식]
- [Variable] = [Expression]

```
int a = 1, b = 0;

b = a++;     // b = 1, a = 2
b = a--;     // b = 1, a = 0

b = ++a;     // b = 2, a = 2;
b = --a;     // b = 0, a = 0;
```

[산술 연산자 (Arithmetic Operator)]

연산자	기능	문법	
+	더하기	a = 5 + 2;	5와 2를 더해 a에 넣는다
-	빼기	a = 5 - 2;	5에서 2를 빼 a에 넣는다.
*	곱하기	a = 5 * 2	5와 2를 곱해 a에 넣는다.
/	나누기	a = 5 / 2	5를 2로 나눈 몫을 a에 넣는다.
%	나머지	a = 5 % 2	5를 2로 나눈 나머지를 a에 넣는다.
++	1을 증가시킨다.	a++	a = a +1
--	1을 감소시킨다.	a--	a = a - 1

매개변수의 자료형과 반환 자료형은
자료형 형변환 참고

단항 연산자 (Unary Operator) (++, --)

- 전위형 (++a)
- 후위형 (a++)

<연산자 (operator)>

[할당 연산자 (Compound Assignment Operator)]

연산자	문법	의미
+=	a += 1	a = a + 1
-=	a -= 2	a = a - 2
*=	a *= 3	a = a * 3
/=	a /= 4	a = a / 4
%=	a %= 5	a = a % 5

<연산자 (operator)>

[관계 연산자 (Relational Operator)]

- 매개변수의 자료형과 상관없이 반환 자료형은 항상 "0과1"

연산자	기능
<	왼쪽이 오른쪽보다 작으면 1, 크거나 같으면 0
>	왼쪽이 오른쪽보다 크면 1, 작거나 같으면 0
<=	왼쪽이 오른쪽보다 작거나 같으면 1, 크면 0
>=	왼쪽이 오른쪽보다 크거나 같으면 1, 작으면 0
==	두 값이 같으면 1, 다르면 0
!=	두 값이 다르면 1, 같으면 0

[삼항 연산자 (Ternary Operator)]

연산자	설명	문법
? :	조건부 연산자. 조건식이 참이면 : 앞의 값을 반환, 거짓이면 : 뒤의 값을 반환	x ? a : b;

<연산자 (operator)>

[논리 연산자 (Logical Operator)]

- 매개변수의 자료형과 반환 자료형은 항상 “0과1”

연산자	논리	기능
&&	AND	양쪽이 모두 참이면 참
||	OR	양쪽 중 하나이상이 참이면 참
!	NOT	참이면 거짓, 거짓이면 참

```
int a = 3, b = 1, c = 0;

c = (a > b) && (b == 1);     // c = 1
c = (a > b) && (b < 1);      // c = 0
c = (a > b) || (b == 1);     // c = 1
c = (a > b) || (b < 1);      // c = 1
c = !(a > b);                // c = 0
c = !(b < 0);                // c = 1

c = 100 && -10;     // c = 1
c = -20 || -10;     // c = 1
c = !123;           // c = 0
```

* C언어에서의 Boolean은 '0'이 아닌 모든 값을 '참 (True)'으로 처리하고 있다

<연산자 (operator)>

[비트 연산자 (Bit Operator)]

연산자	기능
&	비트단위 AND 연산
|	비트단위 OR 연산
~	비트단위 NOT 연산
^	비트단위 XOR 연산
<<	왼쪽으로 비트 이동
>>	오른쪽으로 비트 이동

```
a = 1010 0101, b = 1001 1001일때

c = a & b;     // c = 1000 0001
c = a | b;     // c = 1011 1101
c = ~a;        // c = 0101 1010
c = a^b;       // c = 0011 1100

a = 0011 1010; 일때

c = a << 1;     // c = 0111 0100
c = a << 2;     // c = 1110 1000

c = a >> 1;     // c = 0001 1101
c = a >> 2;     // c = 0000 1110

int a = 12;

c = a << 1;     // c = 24 (= a * 2)
c = a << 3;     // c = 96 (= a * 8)

c = a >> 1;     // c = 6 (= a / 2)
c = a >> 2;     // c = 3 (= a / 4)
```

<연산자 (operator)>

[연산자 우선 순위]

기호 1	연산 유형	associativity
`[` `]` `(` `)` `.` `->` `++` `--` (후위)	식	왼쪽에서 오른쪽
`sizeof` `&` `*` `+` `-` `~` `!` `++` `--` (전위)	단항	오른쪽에서 왼쪽
형식 캐스팅	단항	오른쪽에서 왼쪽
`*` `/` `%`	곱하기	왼쪽에서 오른쪽
`+` `-`	더하기	왼쪽에서 오른쪽
`<<` `>>`	비트 시프트	왼쪽에서 오른쪽
`<` `>` `<=` `>=`	관계	왼쪽에서 오른쪽
`==` `!=`	같음	왼쪽에서 오른쪽
`&`	비트 AND	왼쪽에서 오른쪽
`^`	비트 제외 OR	왼쪽에서 오른쪽
`|`	비트 포함 OR	왼쪽에서 오른쪽
`&&`	논리 AND	왼쪽에서 오른쪽
`||`	논리 OR	왼쪽에서 오른쪽
`?` `:`	조건식	오른쪽에서 왼쪽
`=` `*=` `/=` `%=` `+=` `-=` `<<=` `>>=` `&=` `^=` `|=`	단순 및 복합 할당 2	오른쪽에서 왼쪽
`,`	순차적 계산	왼쪽에서 오른쪽

<입출력>

[C언어 기본 구조]

```
#include <stdio.h>
int main() {
	return 0;
}
```

```
#include <stdio.h>
int main() {
	printf(“Hello, World!”);
	return 0;
}
```

[stdio.h]

<stdio.h>은

Standard Input/Output library

(표준입출력 라이브러리)의 약어

stdin - standard input 키보드

stdout - standard output 모니터

즉, 별도로 지정하지 않으면 입력은 키보드

출력은 모니터로 하란 의미

<입출력>

[printf() 함수]

C언어 표준 출력 함수

printf() 함수의 f는 formatted의 약자로 서식화된 출력을 지원한다는 의미

출력할 데이터를 어떤 서식에 맞춰 출력할지 서식 지정자(format specifier)를 통해 직접 지정
서식문자열 내에 특수문자와 서식문자 포함

```
int printf(const char * restrict format, ...);

ex) printf("%d", n);
    printf("Hello World!")
```

<입출력>

[출력시 문자열 특수 문자]

특수문자	의미
₩₩	화면에 역슬래시(₩) 출력
₩'	작은 따옴표의 출력
₩"	큰 따옴표의 출력
₩?	물음표의 출력
₩a	경고음
₩b	백스페이스
₩f	Form feed
₩n	return
₩r	carriage return
₩t	수평 탭
₩v	수직 탭

<입출력>

[scanf() 함수]

C언어 표준 입력 함수

scanf() 함수의 f는 formatted의 약자로 서식화된 출력을 지원한다는 의미

입력할 데이터를 어떤 서식에 맞춰 출력할지 서식 지정자(format specifier)를 통해 직접 지정
서식문자열 내에 특수문자와 서식문자 포함

```
int scanf(const char * restrict format, ...);

ex) scanf("%d", %n);
```

<입출력>

[서식 문자]

종류		문자	자료형	비고
정수	10진수	%hi	signed short	부호있는 2바이트 크기의 정수
		%hu	unsigned short	부호없는 2바이트 크기의 정수
		%d	signed int	운영체제마다 다름/보통 4바이트
		%u	unsigned int	운영체제마다 다름/보통 4바이트
		%li	signed long	부호있는 4바이트 크기의 정수
		%lu	unsigned long	부호없는 4바이트 크기의 정수
		%lli	signed long long	부호있는 8바이트 크기의 정수
		%llu	unsigned long long	부호있는 8바이트 크기의 정수
	8진수	%o	unsigned int	8진수 정수 출력
	16진수	%x	unsigned int	16진수 정수 소문자 출력
		%X	unsigned int	16진수 정수 대문자 출력
실수		%f	float	4바이트 크기의 10진수 실수
		%lf	double	8바이트 크기의 10진수 실수
		%llf	long double	12바이트 크기의 10진수 실수
		%e	float , double	부동소수점 e(소문자) 방식 실수
		%E	float , double	부동소수점 E(대문자) 방식 실수
		%g	float , double	경우에 따라 %f와 %e
		%G	float , double	경우에 따라 %f와 %e
문자		%c	char	문자 출력
문자열		%s	char*,char[]	문자열 출력
포인터		%p	void*	포인터 주소값 출력

[특용 문자]

특용문자	출력	설명
\\	\	특수문자 표현할 때 사용되는 문자 \ 출력
\"	"	문자열상수 표현할 때 사용되는 문자 " 출력
\'	'	문자상수 표현할 때 사용되는 문자 ' 출력
%%	%	서식지정 표현할 때 사용되는 문자 % 출력

<입출력>

[특수 문자]

특수문자	동작	설명
\n	new line	다음줄 첫칼럼위치로 커서이동
\t	tab	탭위치로 커서이동
\b	back	뒤로 한칼럼 커서이동
\r	carriage return	현재라인의 첫칼럼으로 커서이동
\v	vertical tab	수직 탭효과, 프린터에서 여러라인 이동
\f	form feed	프린팅시 현재 종이밀어내고, 다음 장의 첫라인으로

THANK YOU!