

---

# C 프로그래밍 및 실습

## 2. 변수와 자료형

세종대학교

---

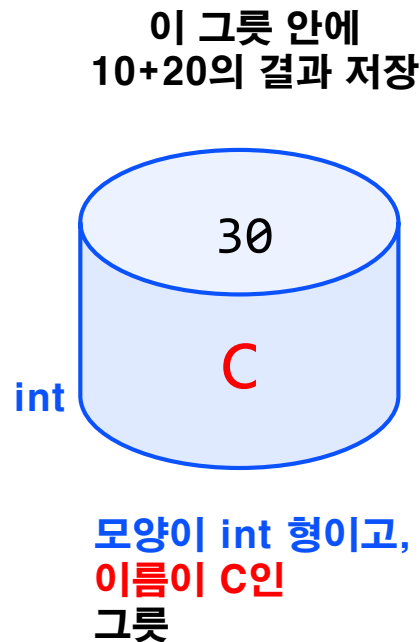
# 목차

---

- 1) 변수와 자료형 개요
- 2) 변수 선언과 사용
- 3) 정수 자료형
- 4) 부동소수 자료형
- 5) 문자 자료형
- 6) 자료형 변환

# 1) 변수와 자료형 개요

- 변수: 값을 담을 그릇
- 자료형: 그릇의 모양



자료형

```
int main()  
{  
    int c ;
```

변수

```
    c=10+20;
```

```
    printf("c=10+20 출력:");  
    printf("%d", c);
```

```
    return 0;
```

```
}
```

# 1) 변수와 자료형 개요

---

- **변수**

- 값을 저장하기 위한 기억 장소
- 사용하기 전에 반드시 선언
  - ✓ `int c;`

- **자료형**

- 자료 값의 형태
- 컴퓨터 내부에서 값이 저장되고 처리되는 방식을 결정짓는 매우 중요한 요소

- **상수**

- 변하지 않는 수로 변수와 대비되는 개념
- 10, 20, 30 과 같은 특정 값

## 2) 변수 선언과 사용

- 변수 선언

```
int    c    ;
```

- 자료형을 앞에 명시한 후 사용할 변수 이름을 적음
- 변수 선언도 하나의 문장이므로 세미콜론을 붙여야 함

- 예)

```
int num; ⇒ int 형 변수 num 선언
char ch; ⇒ char 형 변수 ch 선언
float x; ⇒ float 형 변수 x 선언
double y; ⇒ double 형 변수 y 선언
```

- ✓ `int`, `char`, `float`, `double` : 자료형 이름, 미리 정해진 단어
- ✓ `num`, `ch`, `x`, `y` : 변수이름, 프로그래머가 지은 단어

## 2) 변수 선언과 사용

---

- 다양한 변수 선언의 형태

```
int a;  
double b;  
int c;  
double d;
```

① 가능

```
int a,b;  
double b,d;
```

② 가능

```
int a, float b;  
float d, int c;  
(X)
```

③ 불가능

## 2) 변수 선언과 사용

### ■ 변수 값 저장

- 선언된 변수에 값을 저장하기 위해서는 대입연산자 '=' 사용
- 왼쪽 변수에 오른쪽의 값을 대입(저장)하라는 의미

`c = 10 + 20 ;`

`c ← 10 + 20`

✓ 수학에서 사용되는 등호(=)와는 다른 의미

- 변수에 새로운 값을 대입하면 이전 값은 사라짐

`age = 20;`

`age = 21;      // 이전에 저장한 20은 사라짐`

## 2) 변수 선언과 사용

- 변수 값 참조

- 변수 이름 사용

```
printf("%d", c);
```

✓ 여기서 c는 변수에 저장된 값을 의미

- 변수의 위치에 따라 의미 다름

- 대입 연산자 왼쪽: 저장 공간 자체
- 대입 연산자 오른쪽: 저장된 값

```
a = b; → 변수 a에 변수 b의 값 대입  
b = a; → 변수 b에 변수 a의 값 대입
```

```
a = a + 10; → 이런 문장도 가능
```



## 2) 변수 선언과 사용

- [예제 2.1] 변수 선언과 사용

```
int main()
```

```
{
```

```
    int id;
```

```
    int credits;
```

```
    id = 20160120; // 변수 id에 값 대입(저장)
```

```
    credits = 18;
```

```
    printf("학번 : %d\n", id); // 변수 id 값 출력(참조)
```

```
    printf("신청학점 : %d\n", credits);
```

```
    return 0;
```

```
}
```

실행결과

학번 : 20160120

신청학점 : 18

## 2) 변수 선언과 사용

### ■ 변수 초기화

- 변수를 선언만 하고 값을 대입하지 않으면 쓰레기 값(garbage value)이 저장되어 있음
- 선언과 동시에 변수 값 지정 (변수 초기화)

```
int num = 123;
```

- 여러 변수 동시 초기화 가능, 일부 변수만 초기화 가능

```
int a, b, c;  
a = 123;  
b = 456;
```



```
int a = 123, b = 456, c;
```

## 2) 변수 선언과 사용

- [예제 2.2] 변수 초기화

```
int main()
{
    int math = 99;      // int형 변수 math 선언 후 99로 초기화
    int korean = 90;
    [ 빈 칸 ]

    //더하기 기호인 +를 사용하여 총합을 변수 total에 저장
    int total = math + korean + science;

    printf("수학 : %d\n", math);    // 변수 값 출력
    printf("국어 : %d\n", korean);
    printf("과학 : %d\n", science);
    [ 빈 칸 ]

    return 0;
}
```

실행결과

수학	: 99
국어	: 90
과학	: 94
총점	: 283

## 2) 변수 선언과 사용

---

- 키워드

- C 언어에서 특별한 의미를 가지도록 미리 정해 놓은 단어
- 예) char, int, double 등 기본 자료형, 이외에도 많음

- 식별자

- 변수처럼 프로그래머가 지어서 사용하는 이름
- 식별자로 사용할 수 없는 이름의 예
  - ✓ num-01    num.a    → 밑줄이 아닌 특수 문자는 사용할 수 없음
  - ✓ **3**card       **9**99       → 첫 문자에 숫자 사용할 수 없음
  - ✓ **int**           **char**       → 키워드는 사용할 수 없음

# 목차

---

- 1) 변수와 자료형 개요
- 2) 변수 선언과 사용
- 3) 정수 자료형
- 4) 부동소수 자료형
- 5) 문자 자료형
- 6) 자료형 변환

### 3) 정수 자료형

- 정수 자료형 종류

- `int` : 정수를 나타내는 가장 기본적인 자료형
- `short, long, long long` : 정수를 나타내지만 자료형의 크기가 다름
  - ✓  $\text{short} \leq \text{int} \leq \text{long} \leq \text{long long}$
- 같은 자료형이라도 시스템마다 크기가 다를 수 있음
  - ✓ 자료형의 크기는 `sizeof` 연산자를 이용하여 확인

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

- 자료형의 크기는 표현할 수 있는 수의 범위 결정
  - ✓ 예) `int`의 크기는 보통 4 바이트 (32 비트)로 총  $2^{32}$ 개의 수 표현 가능  
반은 음수, 나머지 반은 0과 양수를 표현하여  
-  $2^{31} \sim 2^{31} - 1$  사이의 정수를 나타냄

### 3) 정수 자료형

---

- **signed 와 unsigned**
  - signed : 음수와 양수 모두 표현
  - unsigned : 0과 양수만 표현 (음수 표현 불가)
- int, short 등의 앞에 부호 여부 명시해주면 됨
  - ✓ 예) unsigned int, signed short
  - ✓ 명시하지 않으면 기본적으로 signed
  - ✓ 즉, int = signed int

### 3) 정수 자료형

- 정수 자료형의 크기 및 표현할 수 있는 값의 범위
  - VS 2017 기준

부호	자료형	메모리 크기	값의 범위
있음	short	2 bytes	$-2^{15} \sim 2^{15} - 1$
	int	4 bytes	$-2^{31} \sim 2^{31} - 1$
	long	4 bytes	$-2^{31} \sim 2^{31} - 1$
	long long	8 bytes	$-2^{63} \sim 2^{63} - 1$
없음	unsigned short	2 bytes	$0 \sim 65,535$
	unsigned int	4 bytes	$0 \sim 4,294,967,295$
	unsigned long	4 bytes	$0 \sim 4,294,967,295$
	unsigned long long	8 bytes	$0 \sim 18,446,744,073,709,551,615$



### 3) 정수 자료형

- 다양한 정수 자료형 사용하기

실행결과

저장값 : 32000 -2140000000  
저장값 : 65000 4280000000

```
int main()
{
    short sVar = 32000;           //-32768 ~ 32767
    int iVar = -2140000000;       // 약 -21억 ~ 21억 정도

    unsigned short usVar = 65000; // 0 ~ 65535
    unsigned int uiVar = 4280000000; // 0 ~ 42억 정도

    printf("저장값 : %d %d\n", sVar, iVar);
    printf("저장값 : %u %u\n", usVar, uiVar);

    return 0;
}
```

unsigned 값을 출력할 경우 %u 사용

### 3) 정수 자료형

---

- [예제 2.3]

- 이전 프로그램에서 다음과 같이 각 자료형이 나타낼 수 있는 최댓값 또는 최솟값으로 초기화하여 출력해보자.

```
short sVar = -32768;
```

⇒ 최솟값

```
int iVar = 2147483647;
```

⇒ 최댓값

```
unsigned short usVar = 65535;
```

⇒ 최댓값

```
unsigned int uiVar = 4294967295;
```

⇒ 최댓값

결과는?

### 3) 정수 자료형

---

- [예제 2.4]

- 이전 프로그램에서 다음과 같이 각 자료형이 나타낼 수 있는 수의 범위를 벗어난 값으로 초기화하여 출력해보자.

```
short sVar = 72000;  
int iVar = 2150000000;  
  
unsigned short usVar = -1000;  
unsigned int uiVar = 4294967300;
```

결과는?

### 3) 정수 자료형

- [예제 2.5]

- 이전 프로그램에서 다음과 같이 각 자료형이 나타낼 수 있는 최댓값 보다 1 큰 수 또는 최솟값보다 1 작은 수로 초기화하여 출력해보자.

```
short sVar = -32768-1;           ⇒ 최솟값 - 1
```

```
int iVar = 2147483647+1;         ⇒ 최댓값 + 1
```

```
unsigned short usVar = 0-1;      ⇒ 최솟값 - 1
```

```
unsigned int uiVar = 4294967295+1; ⇒ 최댓값 + 1
```

결과는?

# 목차

---

- 1) 변수와 자료형 개요
- 2) 변수 선언과 사용
- 3) 정수 자료형
- 4) **부동소수 자료형**
- 5) 문자 자료형
- 6) 자료형 변환

## 4) 부동소수 자료형

---

- 부동소수(floating point)형 종류
  - 3.14, 3.26567과 같이 실수를 표현하는 자료형
  - 자료형 키워드: float, double, long double
  - 부동소수형 출력: printf의 서식 지정자 '%f' 사용

```
float x = 3.14;
```

```
double y = 3.141592;
```

```
printf("x: %f\n", x); ⇨ 부동소수형 출력 시 %f 사용
```

```
printf("y: %f\n", y); ⇨ 부동소수형 출력 시 %f 사용
```

## 4) 부동소수 자료형

- 부동소수형 표현 방식

- 0.000023의 다른 표현 →  $2.3 \times 10^{-5}$
- 컴퓨터에서는 후자의 방식으로 표현
- 컴퓨터에서 정수 3 과 부동소수 3.0 은 전혀 다름

```
double x = 3.0;
```

```
printf("x: %f\n", x); → 부동소수형으로 출력 (정상)
```

```
printf("x: %d\n", x); → 정수형으로 출력 (잘못된 결과)
```

실행결과

```
x : 3.000000
```

```
x : 0
```

## 4) 부동소수 자료형

---

- 부동소수형의 크기

- $\text{float} \leq \text{double} \leq \text{long double}$
- 실제 크기는 시스템 종류에 따라 다를 수 있음  
✓ VS2017 기준

자료형	메모리 크기	값의 범위
float	4 bytes	유효 자릿수 약 7개, 최대 지수 약 $10^{38}$
double	8 bytes	유효 자릿수 약 16개, 최대 지수 약 $10^{308}$
long double	8 bytes	유효 자릿수 약 16개, 최대 지수 약 $10^{308}$



## 4) 부동소수 자료형

- 부동소수형의 크기
  - 유효 자릿수 확인

```
float x = 12345678901234567890.0;    ⇒ 20자리 수
double y = 12345678901234567890.0;   ⇒ 20자리 수

printf("x : %f\n", x);                ⇒ float 형 변수 출력
printf("y : %f\n", y);                ⇒ double 형 변수 출력
```

실행결과

```
x=12345679395506094080.000000
y=12345678901234567168.000000
```

# 목차

---

- 1) 변수와 자료형 개요
- 2) 변수 선언과 사용
- 3) 정수 자료형
- 4) 부동소수 자료형
- 5) 문자 자료형
- 6) 자료형 변환

## 5) 문자 자료형

### ■ 문자형

- char, signed char, unsigned char
  - ✓ 문자형 자료형의 크기는 모두 1바이트
- 문자는 작은 따옴표 ' ' 를 사용하여 표현
- 출력: printf의 서식 지정자 '%c'

char ch = 'z';                    ⇨ 문자 'z'로 초기화

printf("ch: %c\n", ch);    ⇨ 문자형 출력 시 %c 사용

실행결과

ch: z

## 5) 문자 자료형

### ■ 문자형의 실체

- 특정 문자에 해당하는 정수값을 지정: 아스키(ASCII) 코드
  - ✓ 예: 영어 대문자 'A'의 아스키 코드 값은 65
- 문자형은 본질적으로 정수형과 동일

```
char c1 = 'A';
```

⇒ 문자 'A'로 초기화

```
char c2 = 65;
```

⇒ 정수 65로 초기화

```
printf("c1: %c %d\n", c1, c1);
```

⇒ c1의 값

```
printf("c2: %c %d\n", c2, c2);
```

⇒ c2의 값

실행결과

```
c1: A 65
```

```
c2: A 65
```

## 5) 문자 자료형

### ■ 아스키 코드 표: 외울 필요 없음

아스키 코드표									
10진수	16진수	8진수	2진수	ASCII	10진수	16진수	8진수	2진수	ASCII
0	0x00	000	00000000	NULL	32	0x20	040	01000000	SP
1	0x01	001	00000001	SOH	33	0x21	041	01000001	!
2	0x02	002	00000010	STX	34	0x22	042	01000010	"
3	0x03	003	00000011	ETX	35	0x23	043	01000011	#
4	0x04	004	00000100	EOT	36	0x24	044	01000100	\$
5	0x05	005	00000101	ENQ	37	0x25	045	01000101	%
6	0x06	006	00000110	ACK	38	0x26	046	01000110	&
7	0x07	007	00000111	BEL	39	0x27	047	01000111	'
8	0x08	010	00010000	BS	40	0x28	050	01010000	(
9	0x09	011	00010001	HT	41	0x29	051	01010001	)
10	0x0A	012	00010010	LF	42	0x2A	052	01010010	*
11	0x0B	013	00010011	VT	43	0x2B	053	01010011	+
12	0x0C	014	00011000	FF	44	0x2C	054	01011000	,
13	0x0D	015	00011001	CR	45	0x2D	055	01011001	-
14	0x0E	016	00011010	SO	46	0x2E	056	01011010	.
15	0x0F	017	00011011	SI	47	0x2F	057	01011011	/
16	0x10	020	00100000	DLE	48	0x30	060	01100000	0
17	0x11	021	00100001	DC1	49	0x31	061	01100001	1
18	0x12	022	00100010	SC2	50	0x32	062	01100010	2
19	0x13	023	00100011	SC3	51	0x33	063	01100011	3
20	0x14	024	00101000	SC4	52	0x34	064	01101000	4
21	0x15	025	00101001	NAK	53	0x35	065	01101001	5
22	0x16	026	00101010	SYN	54	0x36	066	01101010	6
23	0x17	027	00101011	ETB	55	0x37	067	01101011	7
24	0x18	030	00110000	CAN	56	0x38	070	01110000	8
25	0x19	031	00110001	EM	57	0x39	071	01110001	9
26	0x1A	032	00110010	SUB	58	0x3A	072	01110010	:
27	0x1B	033	00110011	ESC	59	0x3B	073	01110011	;
28	0x1C	034	00111000	FS	60	0x3C	074	01111000	<
29	0x1D	035	00111001	GS	61	0x3D	075	01111001	=
30	0x1E	036	00111010	RS	62	0x3E	076	01111010	>
31	0x1F	037	00111011	US	63	0x3F	077	01111011	?

아스키 코드표									
10진수	16진수	8진수	2진수	ASCII	10진수	16진수	8진수	2진수	ASCII
64	0x40	100	10000000	@	96	0x60	140	11000000	`
65	0x41	101	10000001	A	97	0x61	141	11000001	a
66	0x42	102	10000010	B	98	0x62	142	11000010	b
67	0x43	103	10000011	C	99	0x63	143	11000011	c
68	0x44	104	10000100	D	100	0x64	144	11000100	d
69	0x45	105	10000101	E	101	0x65	145	11000101	e
70	0x46	106	10000110	F	102	0x66	146	11000110	f
71	0x47	107	10000111	G	103	0x67	147	11000111	g
72	0x48	110	10010000	H	104	0x68	150	11010000	h
73	0x49	111	10010001	I	105	0x69	151	11010001	i
74	0x4A	112	10010010	J	106	0x6A	152	11010010	j
75	0x4B	113	10010011	K	107	0x6B	153	11010011	k
76	0x4C	114	10011000	L	108	0x6C	154	11011000	l
77	0x4D	115	10011001	M	109	0x6D	155	11011001	m
78	0x4E	116	10011010	N	110	0x6E	156	11011010	n
79	0x4F	117	10011011	O	111	0x6F	157	11011011	o
80	0x50	120	10100000	P	112	0x70	160	11100000	p
81	0x51	121	10100001	Q	113	0x71	161	11100001	q
82	0x52	122	10100010	R	114	0x72	162	11100010	r
83	0x53	123	10100011	S	115	0x73	163	11100011	s
84	0x54	124	10101000	T	116	0x74	164	11101000	t
85	0x55	125	10101001	U	117	0x75	165	11101001	u
86	0x56	126	10101010	V	118	0x76	166	11101010	v
87	0x57	127	10101011	W	119	0x77	167	11101011	w
88	0x58	130	10110000	X	120	0x78	170	11110000	x
89	0x59	131	10110001	Y	121	0x79	171	11110001	y
90	0x5A	132	10110010	Z	122	0x7A	172	11110010	z
91	0x5B	133	10110011	[	123	0x7B	173	11110011	{
92	0x5C	134	10111000	\	124	0x7C	174	11111000	
93	0x5D	135	10111001	]	125	0x7D	175	11111001	}
94	0x5E	136	10111010	^	126	0x7E	176	11111010	~
95	0x5F	137	10111011	_	127	0x7F	177	11111011	DEL

## 5) 문자 자료형

- 문자형은 본질적으로 정수(1 바이트)
  - 정수 연산 가능

```
char ch = 'A'+1;    ⇨ 66 즉, 문자 'B' 저장
```

- 부호 없는 자료형 가능: unsigned char

자료형	메모리 크기	값의 범위
char	1 bytes (8 bits)	-128 ~ 127
unsigned char	1 bytes (8 bits)	0 ~ 255

## 5) 문자 자료형

- 문자 '0'과 숫자 0은 다르다
  - 문자 '0'의 아스키 코드 값은 48
    - ✓ 즉, 문자 '0' == 정수 48

```
char c1 = '0' ;           ⇒ 문자 '0'으로 초기화
char c2 = 0 ;             ⇒ 정수 0으로 초기화
```

```
printf("문자: %c %c\n", c1, c2);
printf("정수: %d %d\n", c1, c2);
```

실행결과

```
문자: 0
정수: [ 빈 칸 ]
```

## 5) 문자 자료형

- 특수 문자 (이스케이프 시퀀스)
  - \ 과 다음 문자를 묶어서 하나의 문자로 간주

문자	역할	비고
\n	새로운 줄로 이동	[Enter] 키 효과와 동일
\t	다음 탭으로 이동	[Tab] 키 효과와 동일
\b	앞으로 한 칸 이동	[Back Space] 키 효과와 동일
\r	줄의 맨 앞으로 이동	[Home] 키 효과와 동일
\a	‘뽁’소리를 냄	
\\	역슬래쉬 \	
\'	작은따옴표 '	
\"	큰따옴표 "	



# 목차

---

- 1) 변수와 자료형 개요
- 2) 변수 선언과 사용
- 3) 정수 자료형
- 4) 부동소수 자료형
- 5) 문자 자료형
- 6) **자료형 변환**

## 6) 자료형 변환

- 서로 다른 자료형의 값을 대입하면?

- 정수는 소수 부분을 표현할 수 없어 123.45가 123으로 출력된 점 이외에는 큰 문제없이 동작

```
int a = 123.45 ;    ⇒ 실수 123.45 대입  
double b = 123 ;   ⇒ 정수 123 대입
```

```
printf("a: %d\n", a); ⇒ a의 값 출력  
printf("b: %f\n", b); ⇒ b의 값 출력
```

실행결과

```
a: 123  
b: 123.000000
```

## 6) 자료형 변환

- 자동 형변환

- 정수  $\leftrightarrow$  부동소수
- `int a = 123.45;`

a	$\leftarrow$	123 (정수)	$\leftarrow$	123.45 (부동소수)
형 변환				

- `double b = 123;`

b	$\leftarrow$	123.0 (부동소수)	$\leftarrow$	123 (정수)
형 변환				

- 위 변환 과정은 자동으로 수행: **자동 형변환 (묵시적 형변환)**

## 6) 자료형 변환

- 정보 유실 주의

```
double fnum1 = 13.5;
double fnum2 = 12.5;
int inum1 = fnum1;    ⇒ inum1에는 13 대입
int inum2 = fnum2;    ⇒ inum2에는 12 대입

printf("fnum1+fnum2 = %f \n", fnum1+fnum2);
printf("inum1+inum2 = %d \n", inum1+inum2);
```

실행결과

```
fnum1+fnum2 = 26.000000
inum1+inum2 = 25
```

✓ 왜 위와 같은 결과가 나오는 지 생각해보자.

## 6) 자료형 변환

- 명시적 형변환

- printf의 서식 지정자에 따라 형변환이 자동으로 발생하지 않음

```
printf("12.3: %d\n", 12.3);  
printf("123: %f\n", 123);
```

실행결과

```
12.3: -1717986918  
123: 0.000000
```

- 명시적 형변환 필요

```
printf("12.3: %d\n", (int) 12.3);  
printf("123: %f\n", (double) 123);
```

실행결과

```
12.3: 12  
123: 123.000000
```