



C프로그래밍

Lecture 05. 상수와 기본 자료형

동덕여자대학교
데이터사이언스 전공
권 범

목차

- ❖ 01. C언어가 제공하는 기본 자료형의 이해
- ❖ 02. 문자의 표현 방식과 문자를 위한 자료형
- ❖ 03. 상수에 대한 이해
- ❖ 04. 자료형의 변환
- ❖ 05. 연습 문제

01. C언어가 제공하는 기본 자료형의 이해

- 02. 문자의 표현 방식과 문자를 위한 자료형
- 03. 상수에 대한 이해
- 04. 자료형의 변환
- 05. 연습 문제

01. C언어가 제공하는 기본 자료형의 이해

❖ 변수를 이용한 데이터의 저장

“ 변수 ” (Variable)

값을 저장할 수 있는 메모리 공간에 붙여진 이름
혹은 메모리 공간 자체를 의미합니다.

변수라는 것을 선언하면 메모리 공간이 할당되고,
할당된 메모리 공간에 이름이 붙습니다.

예를 들어

- ✓ 정수를 저장할 메모리 공간 4바이트를 할당하고 나서
그 메모리 공간에 num이라는 이름을 붙여준다면,
num이 바로 변수가 되는 것입니다.

- vary
(동사) 다르다 (달라지다)
- variable
(형용사) 변동이 심한
(명사) 변수

01. C언어가 제공하는 기본 자료형의 이해

❖ 선언할 변수의 특징을 나타내기 위한 키워드: **자료형(Data Type)** (1/2)

이름 이외에 메모리 공간 할당에 있어서
필요한 두 가지 정보

① 정수를 저장할 것인지? 실수를 저장할 것인지?

- ✓ 값을 저장하는 방식이 정수인지, 실수인지에 따라서 달라지기 때문에 용도를 결정해야 합니다.

② 얼마나 큰 수를 저장할 것인지?

- ✓ 큰 수를 표현하기 위해서는 많은 수의 바이트(Byte)가 필요합니다.

01. C언어가 제공하는 기본 자료형의 이해

❖ 선언할 변수의 특징을 나타내기 위한 키워드: **자료형(Data Type)** (2/2)

"아! 제가 **정수**를 저장할 건데요. **크기는 4바이트**로 하려고 합니다.
그리고 변수의 **이름은 num**으로 할게요."

요청의 예

int num;

C언어에서의 예

자료형의 수는 데이터 표현 방법의 수를 뜻합니다.
즉, C언어가 제공하는 자료형의 수가 10개라면
C언어가 제공하는 기본적인 데이터 표현 방식의 수는 10개라는 뜻입니다.

01. C언어가 제공하는 기본 자료형의 이해

❖ 기본 자료형의 종류와 데이터의 표현 범위 (1/3)

“ 기본 자료형 ” (Primitive Data Types)

C언어는 여러 가지 형태의 자료형을 기본적으로 제공하고 있습니다. 이를 기본 자료형이라고 합니다.

01. C언어가 제공하는 기본 자료형의 이해

❖ 기본 자료형의 종류와 데이터의 표현 범위 (2/3)

	자료형	크기	값의 표현범위
정수형	char	1바이트	-128이상 +127이하
	short	2바이트	-32,768이상 +32,767이하
	int	4바이트	-2,147,483,648이상 +2,147,483,647이하
	long	4바이트	-2,147,483,648이상 +2,147,483,647이하
	long long	8바이트	-9,223,372,036,854,775,808이상 +9,223,372,036,854,775,807이하
실수형	float	4바이트	$\pm 3.4 \times 10^{-37}$ 이상 $\pm 3.4 \times 10^{+38}$ 이하
	double	8바이트	$\pm 1.7 \times 10^{-307}$ 이상 $\pm 1.7 \times 10^{+308}$ 이하
	long double	8바이트 이상	double 이상의 표현범위

- ✓ 자료형의 바이트 수가 커지면 표현할 수 있는 데이터의 범위도 커지는 것을 알 수 있습니다.
- ✓ long double은 시스템 및 컴파일러에 따라서 할당되는 메모리의 크기나 표현 가능한 데이터의 범위가 유동적인 자료형입니다.

- short는 short int를 줄인 표현입니다.
- long은 long int를 줄인 표현입니다.

크게 정수형과 실수형으로 나뉩니다.

정수형에도 실수형에도
둘 이상의 기본 자료형이 존재합니다.

표현하고자 하는 값의 크기에 따라서
자료형을 적절히 선택합니다.

01. C언어가 제공하는 기본 자료형의 이해

❖ 기본 자료형의 종류와 데이터의 표현 범위 (3/3)

왜 이리도 많은 수의 자료형을 제공하고 있는 것일까요?

이유1: 데이터의 표현 방식이 다르기 때문입니다.

- ✓ 자료형의 종류를 크게 두 가지(정수형, 실수형)으로 나눠 놓은 이유는 컴퓨터가 내부적으로 정수형 데이터와 실수형 데이터를 처리하는 방식이 다르기 때문입니다.

이유2: 메모리 공간을 적절히 사용하기 위함입니다.

- ✓ 예를 들어, 3.14라는 실수를 메모리 공간에 담길 원한다고 가정해 봅시다. 이러한 경우 4바이트 float형 변수만을 가지고 충분히 표현이 가능한데, 8바이트 double형 변수를 이용하게 되면 메모리 공간을 낭비하게 됩니다.
- ✓ 3.4×10^{38} 보다 큰 수를 담길 원한다면 double형 변수를 사용해야 합니다. 그렇지 않으면 데이터의 손실이 발생하게 됩니다.

01. C언어가 제공하는 기본 자료형의 이해

❖ 연산자 sizeof를 이용한 바이트 크기 확인 (1/2)

sizeof 연산자는 단항 연산자로서
피연산자의 메모리 크기를 반환합니다.

피연산자로는 변수, 상수, 자료형의 이름 정도가 올 수 있습니다.

```
int main(void)
{
    int num = 10;
    // 변수 num과 자료형 int의 크기를 계산하고,
    // 계산 결과값으로 sz1, sz2를 초기화
    int sz1 = sizeof(num); // sizeof num
    int sz2 = sizeof(int);
    ...
}
```

- ✓ sizeof 연산자의 피연산자로 자료형 이름이 올 경우에는 소괄호()를 반드시 적어야 합니다.
- ✓ 그 외의 경우에는 소괄호()는 선택적입니다.
- ✓ 하지만 모든 피연산자를 대상으로 소괄호()를 감싸주는 것이 일반적입니다!

01. C언어가 제공하는 기본 자료형의 이해

❖ 연산자 sizeof를 이용한 바이트 크기 확인 (2/2)

```
1  /* sizeof.c */
2  #include <stdio.h>
3  int main(void)
4  {
5      char ch = 9;
6      int inum = 1052;
7      double dnum = 3.1415;
8      printf("변수 ch의 크기: %d\n", sizeof(ch));
9      printf("변수 inum의 크기: %d\n", sizeof(inum));
10     printf("변수 dnum의 크기: %d\n", sizeof(dnum));
11
12     printf("char의 크기: %d\n", sizeof(char));
13     printf("int의 크기: %d\n", sizeof(int));
14     printf("long의 크기: %d\n", sizeof(long));
15     printf("long long의 크기: %d\n", sizeof(long long));
16     printf("float의 크기: %d\n", sizeof(float));
17     printf("double의 크기: %d\n", sizeof(double));
18
19     return 0;
20 }
```

변수 ch의 크기: 1
변수 inum의 크기: 4
변수 dnum의 크기: 8
char의 크기: 1
int의 크기: 4
long의 크기: 4
long long의 크기: 8
float의 크기: 4
double의 크기: 8

01. C언어가 제공하는 기본 자료형의 이해

❖ 정수의 표현 및 처리를 위한 일반적 자료형 선택 (1/2)

일반적인 선택은 int입니다.

- ✓ CPU가 연산하기에 가장 적합한 데이터의 크기가 int형의 크기로 결정됩니다.
- ✓ 연산이 동반이 되면 int형으로 형 변환이 되어서 연산이 진행됩니다.
- ✓ 따라서 연산을 동반하는 변수의 선언을 위해서는 int로 선언하는 것이 적합합니다.


char형, short형 변수는 불필요한가요?

- ✓ 연산을 수반하지 않으면서(최소한의 연산만 요구가 되면서) 많은 수의 데이터를 저장해야 한다면, 그리고 그 데이터의 크기가 char 또는 short로 충분히 표현 가능하다면, char 또는 short로 데이터를 표현 및 저장하는 것이 적절합니다.

01. C언어가 제공하는 기본 자료형의 이해

❖ 정수의 표현 및 처리를 위한 일반적 자료형 선택 (2/2)

```
1  /* char_short_add.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      char num1 = 1, num2 = 2, result1 = 0;
7      short num3 = 300, num4 = 400, result2 = 0;
8
9      printf("size of num1 & num2: %d, %d\n", sizeof(num1), sizeof(num2));
10     printf("size of num3 & num4: %d, %d\n", sizeof(num3), sizeof(num4));
11     printf("size of char add: %d\n", sizeof(num1 + num2));
12     printf("size of short add: %d\n", sizeof(num3 + num4));
13
14     result1 = num1 + num2;
15     result2 = num3 + num4;
16     printf("size of result1 & result2: %d, %d\n", sizeof(result1), sizeof(result2));
17     return 0;
18 }
```



size of num1 & num2: 1, 1
size of num3 & num4: 2, 2
size of char add: 4
size of short add: 4
size of result1 & result2: 1, 2

01. C언어가 제공하는 기본 자료형의 이해

❖ 실수(Real Number)의 표현 및 처리를 위한 일반적 자료형 선택 (1/2)

실수 자료형의 선택 기준은 정밀도입니다.

- ✓ 실수의 표현 범위는 float, double 둘 다 충분히 넓습니다.
- ✓ 그러나 8바이트 크기의 double이 float보다 더 정밀하게 실수를 표현합니다.

일반적인 선택은 double입니다.

- ✓ 컴퓨팅 환경의 발전으로, double형 데이터의 표현 및 연산에 대한 부담이 줄어들었습니다.
- ✓ float형 데이터의 정밀도는 부족한 경우가 많습니다.

실수 자료형	소수점 이하 정밀도	바이트 수
float	6자리	4
double	15자리	8
long double	18자리	12

01. C언어가 제공하는 기본 자료형의 이해

❖ 실수(Real Number)의 표현 및 처리를 위한 일반적 자료형 선택 (2/2)

```
1  /* circle_area.c */
2  #define _CRT_SECURE_NO_WARNINGS
3  #include <stdio.h>
4
5  int main(void)
6  {
7      double rad;
8      double area;
9
10     printf("원의 반지름 입력: ");
11     scanf("%lf", &rad);
12
13     area = 3.1415 * rad * rad;
14     printf("원의 넓이: %f\n", area);
15     return 0;
16 }
```

원의 반지름 입력: 2.4
원의 넓이: 18.095040

double형 변수의 출력 서식 문자: %f
double형 변수의 입력 서식 문자: %lf

**%f를 사용해서 double형으로 데이터를 입력받는 경우,
전혀 엉뚱한 값이 들어가게 됩니다.**

01. C언어가 제공하는 기본 자료형의 이해

❖ unsigned를 붙여서 0과 양의 정수만 표현

정수 자료형	크기	값의 표현범위
char	1바이트	-128이상 +127이하
unsigned char		0이상 (128 + 127)이하
short	2바이트	-32,768이상 +32,767이하
unsigned short		0이상 (32,768 + 32,767)이하
int	4바이트	-2,147,483,648이상 +2,147,483,647이하
unsigned int		0이상 (2,147,483,648 + 2,147,483,647)이하
long	4바이트	-2,147,483,648이상 +2,147,483,647이하
unsigned long		0이상 (2,147,483,648 + 2,147,483,647)이하
long long	8바이트	-9,223,372,036,854,775,808이상 +9,223,372,036,854,775,807이하
unsigned long long		0이상 (9,223,372,036,854,775,808 + 9,223,372,036,854,775,807)이하

- MSB(Most Significant Bit):
가장 왼쪽에 위치한 비트로, 부호를 나타내는 비트

- ✓ 정수 자료형의 이름 앞에는 unsigned 선언을 붙일 수 있습니다.
- ✓ unsigned가 붙으면, MSB도 데이터의 크기를 표현하는데 사용이 됩니다.
- ✓ 따라서 표현하는 값의 범위가 양의 정수로 제한이 되며 양의 정수로 두 배 늘어납니다.

02. 문자의 표현 방식과 문자를 위한 자료형

- 01. C언어가 제공하는 기본 자료형의 이해
- 03. 상수에 대한 이해
- 04. 자료형의 변환
- 05. 연습 문제

02. 문자의 표현 방식과 문자를 위한 자료형

❖ 문자의 표현을 위한 약속! 아스키(ASCII) 코드 (1/4)

컴퓨터는 문자를 표현 및 저장하지 못합니다.

따라서 문자 표현을 목적으로
각 문자에 고유한 숫자를 지정합니다.

인간이 입력하는 문자는 해당 문자의 숫자로
변환되어 컴퓨터에 저장 및 인식이 됩니다.

컴퓨터에 저장된 숫자는 문자로
변환되어 인간의 눈에 보여지게 됩니다.

미국 표준 협회에 의해서 제정된 ASCII 코드
(American Standard Code for Information Interchange)

ASCII 코드	ASCII 코드 값
A	65
B	66
C	67
a	97
~	126

02. 문자의 표현 방식과 문자를 위한 자료형

❖ 문자의 표현을 위한 약속! 아스키(ASCII) 코드 (2/4)

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

Source: www.LookupTables.com

02. 문자의 표현 방식과 문자를 위한 자료형

❖ 문자의 표현을 위한 약속! 아스키(ASCII) 코드 (3/4)

```
int main(void)
{
    char ch1 = 'A';
    char ch2 = 'C';
    ...
}
```

```
int main(void)
{
    char ch1 = 65;
    char ch2 = 67;
    ...
}
```

- ✓ C 소스 코드상에서 문자는 작은 따옴표(")로 묶어서 표현합니다.
- ✓ 컴파일 시 각 문자는 해당 아스키 코드 값으로 변환됩니다.
- ✓ 따라서 실제로 컴퓨터에게 전달되는 데이터는 문자가 아닌 숫자입니다.


문자를 char형 변수에 저장하는 이유

- ✓ 모든 아스키 코드는 1바이트로도 충분히 표현 가능합니다.
- ✓ 문자는 덧셈, 뺄셈과 같은 연산을 동반하지 않습니다. 단지 표현에 사용될 뿐입니다.
- ✓ 따라서 1바이트 크기인 char형 변수가 문자를 저장하기 최적의 자료형입니다.
- ✓ 문자는 int형 변수에도 저장이 가능합니다.

02. 문자의 표현 방식과 문자를 위한 자료형

❖ 문자의 표현을 위한 약속! 아스키(ASCII) 코드 (4/4)

```
1  /* how_char.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      char ch1 = 'A', ch2 = 65;
7      char ch3 = 'Z', ch4 = 90;
8
9      printf("%c %d\n", ch1, ch1);
10     printf("%c %d\n", ch2, ch2);
11     printf("%c %d\n", ch3, ch3);
12     printf("%c %d\n", ch4, ch4);
13     return 0;
14 }
```



```
A 65
A 65
Z 90
Z 90
```

서식 문자 %c를 통해서
해당 숫자의 아스키 코드를 출력할 수 있습니다.

03. 상수에 대한 이해

- 01. C언어가 제공하는 기본 자료형의 이해
- 02. 문자의 표현 방식과 문자를 위한 자료형
- 04. 자료형의 변환
- 05. 연습 문제

03. 상수에 대한 이해

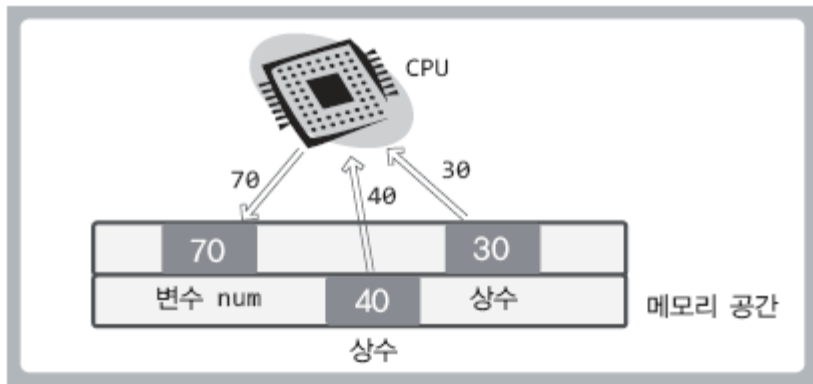
❖ 이름을 지니지 않는 리터럴(Literal) 상수

```
int main(void)
{
    int num = 30 + 40;
    ...
}
```

단계 1 정수 30과 40이 메모리 공간에 상수의 형태로 저장됩니다.

단계 2 두 상수를 기반으로 덧셈이 진행됩니다.

단계 3 덧셈의 결과로 얻어진 정수 70이 변수 num에 저장됩니다.



- ✓ 연산을 위해서는 30, 40과 같이 소스 코드상에 표현되는 숫자도 메모리 공간에 저장되어 있어야 합니다.
- ✓ 이렇게 저장되는 값은 이름이 존재하지 않으니 변경이 불가능한 상수입니다.
- ✓ 변수와 달리 이름이 없는 상수를 가리켜 리터럴 상수라 합니다.

03. 상수에 대한 이해

❖ 리터럴 상수의 자료형

```
1  /* literal_constant_size.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      printf("literal int size: %d\n", sizeof(7));
7      printf("literal double size: %d\n", sizeof(7.14));
8      printf("literal char size: %d\n", sizeof('A'));
9      return 0;
10 }
```

literal int size: 4
literal double size: 8
literal char size: 4

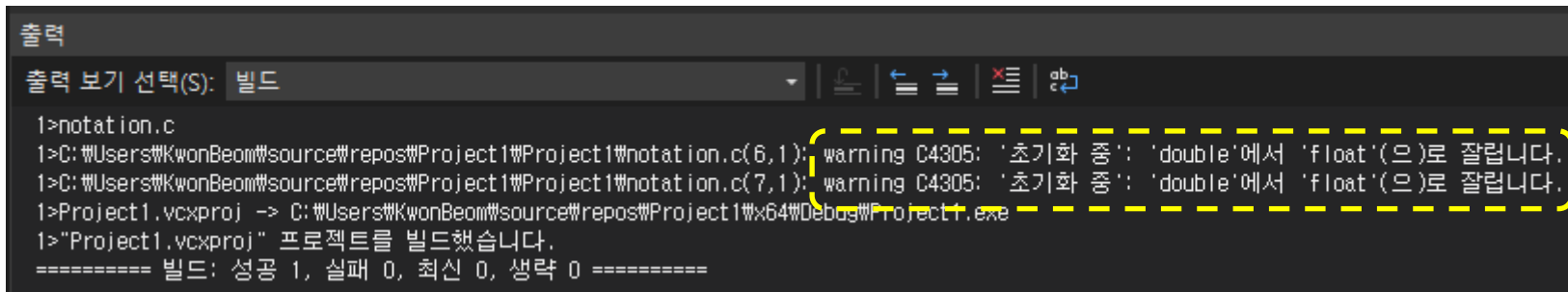
**리터럴 상수도 자료형이 결정되어야
메모리 공간에 저장될 수 있습니다.**

- ✓ 정수는 기본적으로 int형으로 표현됩니다.
- ✓ 실수는 기본적으로 double형으로 표현됩니다.
- ✓ 문자는 기본적으로 char형으로 표현됩니다.

03. 상수에 대한 이해

❖ 접미사를 이용한 다양한 상수의 표현 (1/3)

```
1 int main(void)
2 {
3     float num1 = 5.789;
4     float num2 = 3.24 + 5.12;
5     return 0;
6 }
```



```
출력
출력 보기 선택(S): 빌드
1>notation.c
1>C:\Users\KwonBeom\source\repos\Project1\Project1\notation.c(6,1): warning C4305: '초기화 중': 'double'에서 'float'(으)로 잘립니다.
1>C:\Users\KwonBeom\source\repos\Project1\Project1\notation.c(7,1): warning C4305: '초기화 중': 'double'에서 'float'(으)로 잘립니다.
1>Project1.vcxproj -> C:\Users\KwonBeom\source\repos\Project1\x64\Debug\Project1.exe
1>"Project1.vcxproj" 프로젝트를 빌드했습니다.
===== 빌드: 성공 1, 실패 0, 최신 0, 생략 0 =====
```

실수는 double형 상수로 인식이 되어
데이터 손실에 대한 경고 메시지(warning C4305)가 발생합니다.

5.789라는 8바이트짜리 double형 상수를 4바이트짜리
float형 변수 num1에 대입하다 보면 데이터가 손실될 수도 있다는 경고입니다.

03. 상수에 대한 이해

❖ 접미사를 이용한 다양한 상수의 표현 (2/3)

접미사	자료형	사용의 예
U	unsigned int	unsigned int n = 1025U
L	long	long n = 2467L
UL	unsigned long	unsigned long n = 3456UL
LL	long long	long long n = 5768LL
ULL	unsigned long long	unsigned long long n = 8979ULL

정수형 상수의 표현을 위한 접미사

접미사	자료형	사용의 예
F	float	float f = 3.15F
L	long double	long double f = 5.789L

실수형 상수의 표현을 위한 접미사

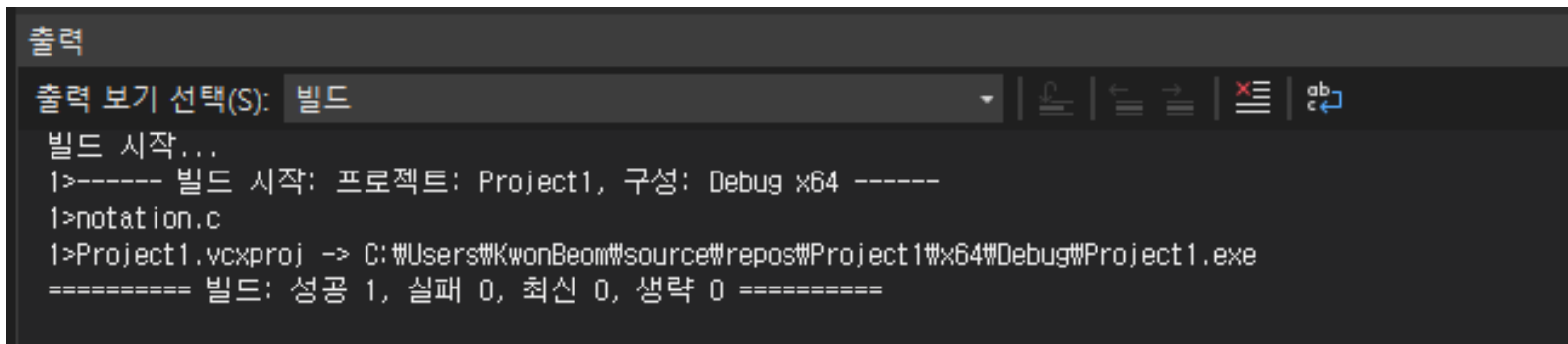
접미사를 통해서 상수의 자료형을 변경할 수 있습니다.

03. 상수에 대한 이해

❖ 접미사를 이용한 다양한 상수의 표현 (3/3)

```
1 int main(void)
2 {
3     float num1 = 5.789f;
4     float num2 = 3.24F + 5.12F;
5     return 0;
6 }
```

소문자 f 대신에 대문자 F를 사용하여도 됩니다.



The screenshot shows the 'Output' window in Visual Studio with the 'Build' tab selected. The output text is as follows:

```
출력
출력 보기 선택(S): 빌드
빌드 시작...
1>----- 빌드 시작: 프로젝트: Project1, 구성: Debug x64 -----
1>notation.c
1>Project1.vcxproj -> C:\Users\KwonBeom\source\repos\Project1\x64\Debug\Project1.exe
===== 빌드: 성공 1, 실패 0, 최신 0, 생략 0 =====
```

경고 메시지(warning C4305)가 발생하지 않습니다.

03. 상수에 대한 이해

❖ 이름을 지니는 심볼릭 상수: const 상수 (1/2)

“ 심볼릭 상수 ” (Symbolic Constants)

변수와 마찬가지로 이름을 지니는 상수입니다.

심볼릭 상수를 표현하는 방법에는 두 가지가 있습니다.

- ① const 키워드를 이용하는 방식
- ② 매크로를 이용하는 방식

const 키워드를 이용해서 상수를 만드는 방법

✓ 변수 선언 시 const라는 키워드를 앞에 붙여 주기만 하면 됩니다.

03. 상수에 대한 이해

❖ 이름을 지니는 심볼릭 상수: const 상수 (2/2)

```
int main(void)
{
    const int MAX = 100;           // MAX는 상수! 따라서 값의 변경 불가능!
    const double PI = 3.1415;      // PI는 상수! 따라서 값의 변경 불가능!
    ...
}
```

```
int main(void)
{
    const int MAX;                 // 쓰레기 값으로 초기화 되어버림
    MAX = 100;                     // 값의 변경 불가능! 따라서 컴파일 에러 발생!
    ...
}
```

상수의 이름은 모두 대문자로 표시하고,
둘 이상의 단어를 연결할 때에는 MY_AGE와 같이
언더바를 이용해서 두 단어를 구분하는 것이 관례입니다.

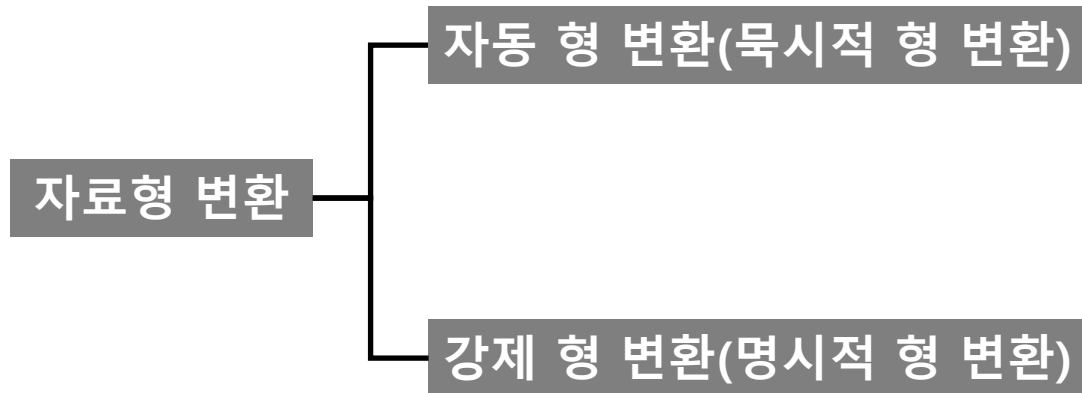
04. 자료형의 변환

- 01. C언어가 제공하는 기본 자료형의 이해
- 02. 문자의 표현 방식과 문자를 위한 자료형
- 03. 상수에 대한 이해
- 05. 연습 문제

04. 자료형의 변환

❖ 자료형의 변환이란?

- int형 데이터가 float형으로 변환되거나,
float형 데이터가 double형으로 변환되는 등의 일들을 말합니다.
- 자료형 변환은 크게 두 가지로 나뉩니다.



04. 자료형의 변환

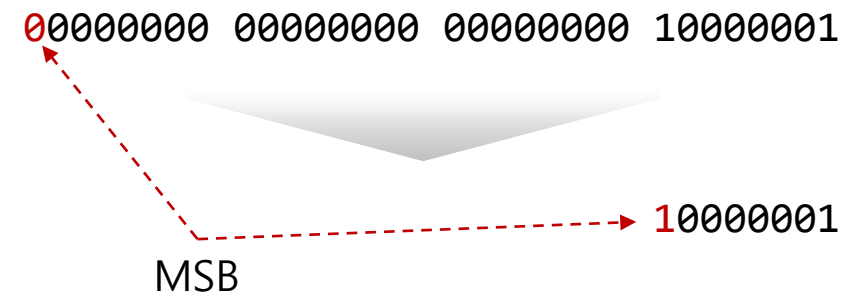
❖ 대입 연산 과정에서 발생하는 자동 형 변환

```
1  /* auto_conversion.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      double num1 = 245; // int형 정수 245를 double형으로 자동 형 변환
7      int num2 = 3.1415; // double형 실수 3.1415를 int형으로 자동 형 변환
8      int num3 = 129;
9      char ch = num3; // int형 변수 num3에 저장된 값이 char형으로 자동 형 변환
10
11     printf("정수 245를 실수로: %f\n", num1);
12     printf("실수 3.1415를 정수로: %d\n", num2);
13     printf("큰 정수 129를 작은 정수로: %d\n", ch);
14     return 0;
15 }
```

정수 245를 실수로: 245.000000
실수 3.1415를 정수로: 3
큰 정수 129를 작은 정수로: -127

- ✓ 정수 245는 245.0의 비트열로 재구성되어 변수 num1에 저장됩니다.
- ✓ 실수 3.1415는 int형 데이터 3의 비트열로 재구성되어 변수 num2에 저장됩니다.

- ✓ 4바이트 변수 num3에 저장된 4바이트 데이터 중 상위 3바이트가 손실되어 변수 ch에 저장됩니다.



04. 자료형의 변환

❖ 자동 형 변환의 방식 정리

형 변환의 방식에 대한 유형별 정리

- ① 정수를 실수로 형 변환: 3은 3.0으로 5는 5.0으로 데이터 손실 없이 자동 형 변환됩니다.
- ② 실수를 정수로 형 변환: 소수점 이하의 값이 소멸됩니다.
- ③ 큰 정수를 작은 정수로 형 변환: 작은 정수의 크기에 맞춰서 상위 바이트가 소멸됩니다.

값의 표현 범위가 넓은 데이터로의 형 변환은 문제가 되지 않습니다.
하지만 그 반대의 경우는 문제가 발생합니다.

04. 자료형의 변환

❖ 정수의 승격에 의한 자동 형 변환

- ✓ 일반적으로 CPU가 처리하기에 가장 적합한 크기의 정수 자료형을 int로 정의합니다.
- ✓ 따라서 int형 연산의 속도가 다른 자료형의 연산 속도에 비해서 동일하거나 더 빠릅니다.

따라서 다음과 같은 방식의
형 변환이 발생합니다.

```
int main(void)
{
    short num1 = 15, num2 = 25;
    short num3 = num1 + num2; // num1과 num2가 int형으로 형 변환
    ...
}
```

**short형 데이터가 연산을 위해서 int형 데이터로 승격되었습니다.
이를 가리켜 정수의 승격(Integral Promotion)이라 합니다.**

- ✓ char, short와 같은 정수형 데이터 연산은 일단 int형 데이터로 자동 형 변환된 다음에 이뤄집니다.
- ✓ int형 데이터 사이의 연산을 진행할 경우에는 형 변환이 발생하지 않습니다.
- ✓ 즉, 형 변환 과정을 거치지 않기 때문에 int형 연산이 다른 정수형 연산에 비해서 빠릅니다.

04. 자료형의 변환

❖ 피연산자의 자료형 불일치로 발생하는 자동 형 변환

```
int main(void)
{
    double num1 = 5.15 + 19;
    ...
}
```

- ✓ 두 피연산자의 자료형은 일치해야 합니다.
- ✓ 일치하지 않으면 일치하기 위해서 자동으로 형 변환이 발생합니다.

- ✓ 아래의 자동 형 변환 규칙을 근거로 int형 데이터 19가 double형 데이터 19.0으로 형 변환이 되어 덧셈이 진행됩니다.

산술 연산에서의 자동 형 변환 규칙



- ✓ 바이트 크기가 큰 자료형이 우선시 됩니다.
- ✓ 정수형보다 실수형을 우선시 합니다.
- ✓ 이는 데이터의 손실을 최소화하기 위한 기준입니다.

04. 자료형의 변환

❖ 명시적 형 변환: 강제로 일으키는 형 변환 (1/3)

```
1  /* conversion_division1.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int num1 = 3, num2 = 4;
7      double divResult;
8      [divResult = num1 / num2;]
9
10     printf("나눗셈 결과: %f\n", divResult);
11     return 0;
12 }
```

✓ num1과 num2가 정수이기 때문에
몫만 반환이 되는 정수형 나눗셈이
진행됩니다.

나눗셈 결과: 0.000000

04. 자료형의 변환

❖ 명시적 형 변환: 강제로 일으키는 형 변환 (2/3)

```
1  /* conversion_division2.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int num1 = 3, num2 = 4;
7      double divResult;
8      [divResult = (double)num1 / num2;]
9
10     printf("나눗셈 결과: %f\n", divResult);
11     return 0;
12 }
```

나눗셈 결과: 0.750000

(type)은 type형으로의 형 변환을 의미합니다.

- ✓ num1이 double형으로 명시적 형 변환이 발생합니다.
- ✓ num1과 num2의 / 연산 과정에서 산술적 자동 형 변환이 발생합니다.
- ✓ 그 결과 실수형 나눗셈이 진행되어 divResult에는 0.75가 저장됩니다.

04. 자료형의 변환

❖ 명시적 형 변환: 강제로 일으키는 형 변환 (3/3)

권장하는 소스 코드 작성 스타일

```
int main(void)
{
    int num1 = 3;
    double num2 = 2.5 * num1;
    ...
}
```

```
int main(void)
{
    int num1 = 3;
    double num2 = 2.5 * (double)num1;
    ...
}
```

자동 형 변환이 발생하는 위치에
명시적 형 변환 표시를 해서
형 변환이 발생함을 알리는 것이 좋습니다!

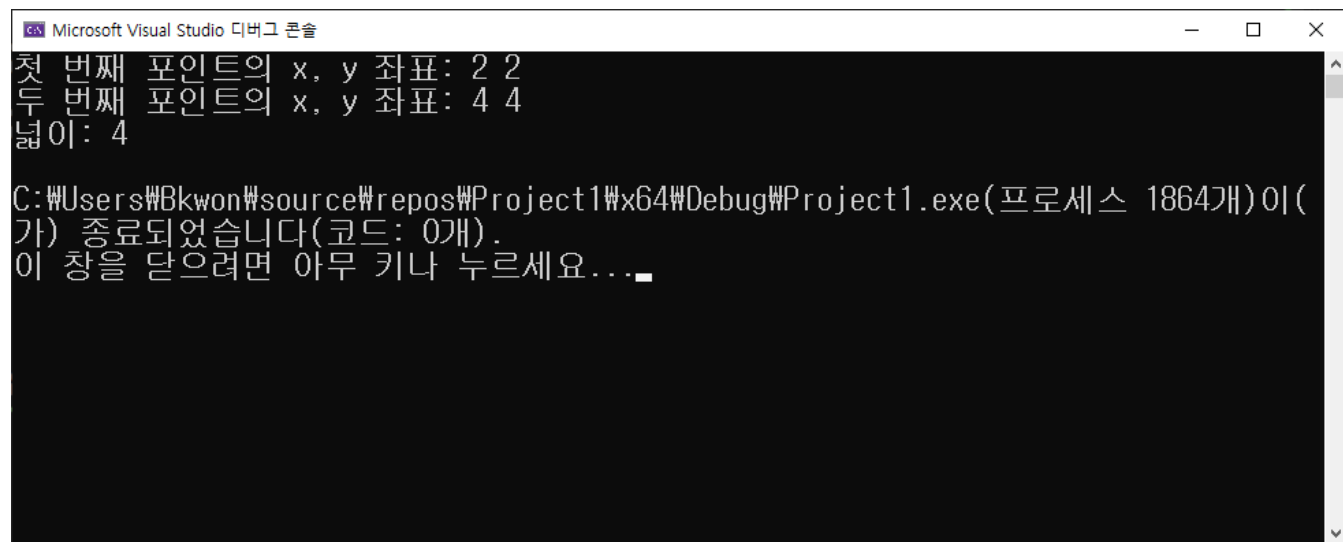
05. 연습 문제

- 01. C언어가 제공하는 기본 자료형의 이해
- 02. 문자의 표현 방식과 문자를 위한 자료형
- 03. 상수에 대한 이해
- 04. 자료형의 변환

05. 연습 문제

❖ 연습 문제 1.

- 사용자로부터 직사각형의 위치 정보를 입력받아서, 넓이를 계산하는 프로그램을 작성해 보세요.
입력해야 할 직사각형의 위치 정보는 두 개의 x, y 좌표(총 4개의 정수)가 되어야 할 것입니다.
단, 조건이 있습니다. 첫 번째로 입력되는 포인트 정보는 직사각형의 좌 하단 좌표이고, 두 번째로 입력되는 포인트 정보는 직사각형의 우 상단 좌표입니다. 또한 좌 하단 좌표의 x, y 값은 우 상단 좌표의 x, y 값보다 작다고 가정합니다. 아래는 실행의 예입니다.



```
Microsoft Visual Studio 디버그 콘솔
첫 번째 포인트의 x, y 좌표: 2 2
두 번째 포인트의 x, y 좌표: 4 4
넓이: 4
C:\Users\Bkwon\source\repos\Project1\bin\x64\Debug\Project1.exe(프로세스 1864개)이(
가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요....
```


05. 연습 문제

❖ 연습 문제 1. 정답 및 해설

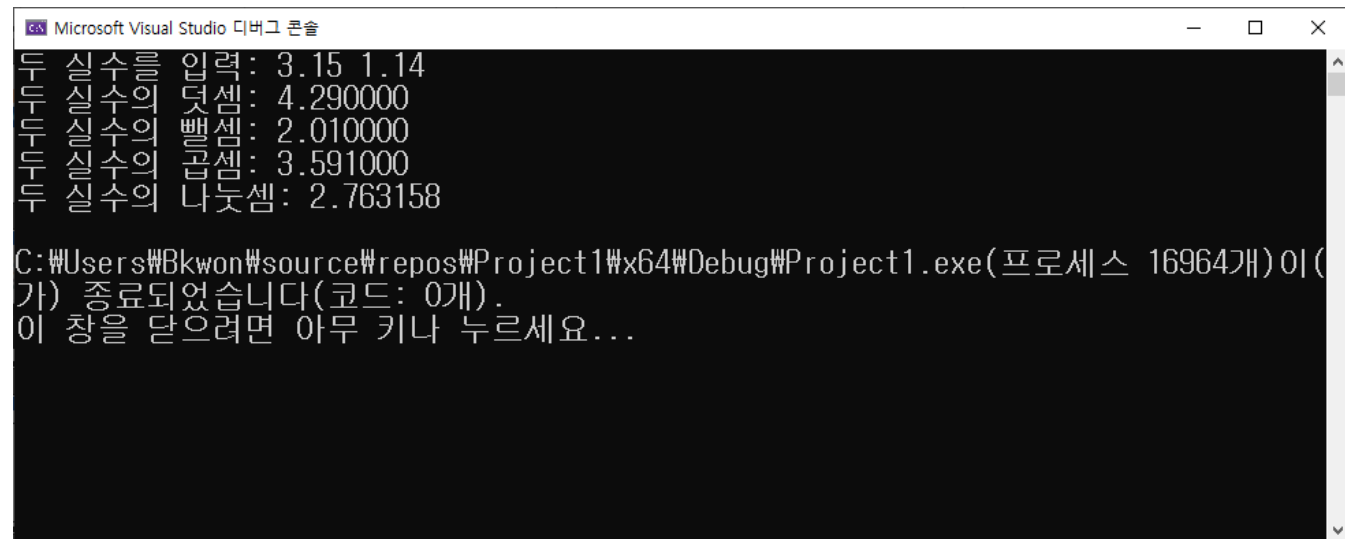
```
1  /* example1.c */
2  #define _CRT_SECURE_NO_WARNINGS
3  #include <stdio.h>
4
5  int main(void)
6  {
7      int x1, y1;
8      int x2, y2;
9      int area;
10
11     printf("첫 번째 포인트의 x, y 좌표: ");
12     scanf("%d %d", &x1, &y1);
13     printf("두 번째 포인트의 x, y 좌표: ");
14     scanf("%d %d", &x2, &y2);
15
16     area = (x2 - x1) * (y2 - y1);
17     printf("넓이: %d\n", area);
18     return 0;
19 }
```

첫 번째 포인트의 x, y 좌표: 2 2
두 번째 포인트의 x, y 좌표: 4 4
넓이: 4

05. 연습 문제

❖ 연습 문제 2.

- 사용자로부터 두 개의 double형 실수를 입력받습니다. 그리고 두 수의 덧셈, 뺄셈, 곱셈, 나눗셈의 결과를 출력하는 계산기 프로그램을 작성해 보세요. 아래는 실행의 예입니다.



```
Microsoft Visual Studio 디버그 콘솔
두 실수를 입력: 3.15 1.14
두 실수의 덧셈: 4.290000
두 실수의 뺄셈: 2.010000
두 실수의 곱셈: 3.591000
두 실수의 나눗셈: 2.763158
C:\Users\Bkwon\source\repos\Project1\Debug\Project1.exe(프로세스 16964개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

05. 연습 문제

❖ 연습 문제 2. 정답 및 해설

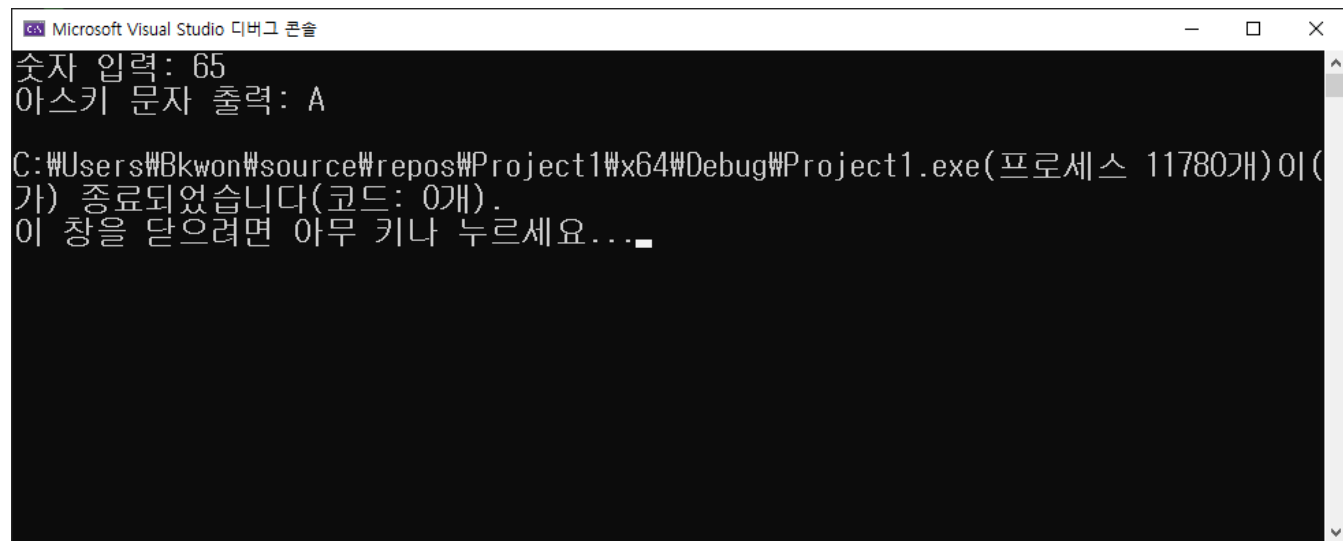
```
1  /* example2.c */
2  #define _CRT_SECURE_NO_WARNINGS
3  #include <stdio.h>
4
5  int main(void)
6  {
7      double d1, d2;
8
9      printf("두 실수를 입력: ");
10     scanf("%lf %lf", &d1, &d2);
11
12     printf("두 실수의 덧셈: %f\n", d1 + d2);
13     printf("두 실수의 뺄셈: %f\n", d1 - d2);
14     printf("두 실수의 곱셈: %f\n", d1 * d2);
15     printf("두 실수의 나눗셈: %f\n", d1 / d2);
16     return 0;
17 }
```

두 실수를 입력: 3.15 1.14
두 실수의 덧셈: 4.290000
두 실수의 뺄셈: 2.010000
두 실수의 곱셈: 3.591000
두 실수의 나눗셈: 2.763158

05. 연습 문제

❖ 연습 문제 3.

- 다음과 같은 프로그램을 작성해 보세요. 사용자로부터 아스키 (ASCII) 코드 범위 내의 값을 하나 입력받습니다. 그리고 이에 해당하는 아스키 코드 문자를 출력합니다. 예를 들어 사용자가 숫자 65를 입력하면 문자 A를 출력해 줍니다. 아래는 실행의 예입니다.



```
Microsoft Visual Studio 디버그 콘솔
숫자 입력: 65
아스키 문자 출력: A

C:\Users\Bkwon\source\repos\Project1\x64\Debug\Project1.exe(프로세스 11780개)이(
가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요....
```

05. 연습 문제

❖ 연습 문제 3. 정답 및 해설

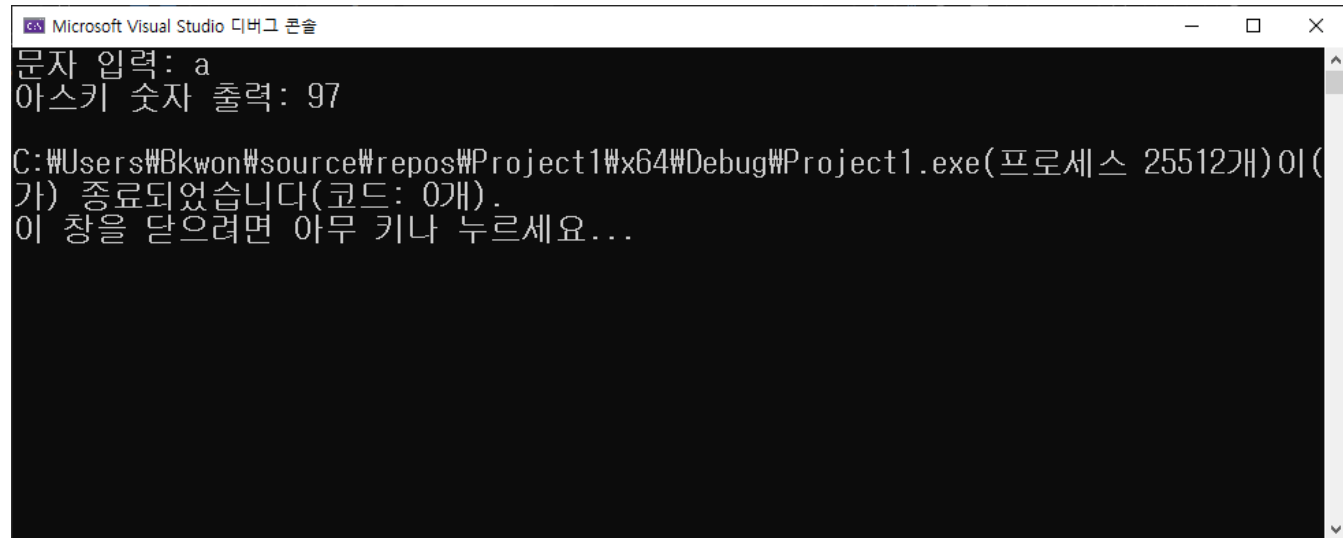
```
1  /* example3.c */
2  #define _CRT_SECURE_NO_WARNINGS
3  #include <stdio.h>
4
5  int main(void)
6  {
7      int asc;
8
9      printf("숫자 입력: ");
10     scanf("%d", &asc);
11
12     printf("아스키 문자 출력: %c\n", asc);
13     return 0;
14 }
```

숫자 입력: 65
아스키 문자 출력: A

05. 연습 문제

❖ 연습 문제 4.

- 다음과 같은 프로그램을 작성해 보세요. 사용자로부터 문자를 하나 입력받습니다. 그리고 이에 해당하는 아스키 코드 숫자를 출력합니다. 예를 들어 사용자가 문자 a를 입력하면 숫자 97를 출력해 줍니다. 아래는 실행의 예입니다.



```
Microsoft Visual Studio 디버그 콘솔
문자 입력: a
아스키 숫자 출력: 97

C:\Users\Bkwon\source\repos\Project1\x64\Debug\Project1.exe(프로세스 25512개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

05. 연습 문제

❖ 연습 문제 4. 정답 및 해설

```
1  /* example4.c */
2  #define _CRT_SECURE_NO_WARNINGS
3  #include <stdio.h>
4
5  int main(void)
6  {
7      char asc;
8
9      printf("문자 입력: ");
10     scanf("%c", &asc);
11
12     printf("아스키 숫자 출력: %d\n", asc);
13     return 0;
14 }
```

문자 입력: a
아스키 숫자 출력: 97

- ❖ 01. C언어가 제공하는 기본 자료형의 이해
- ❖ 02. 문자의 표현 방식과 문자를 위한 자료형
- ❖ 03. 상수에 대한 이해
- ❖ 04. 자료형의 변환
- ❖ 05. 연습 문제

THANK YOU!

Q & A

- Name: 권범
- Office: 동덕여자대학교 인문관 B821호
- Phone: 02-940-4752
- E-mail: bkwon@dongduk.ac.kr