


프로그래밍을 통한 논리적사유연습





제 5 장.

기본 자료형과 형변환

키워드(keyword)

- 고유한 의미를 갖는 예약된 단어(reserved words)
- 예약어라고 부르기도 함

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(identifier)

- 키워드 외에 프로그램에서 사용자가 필요에 따라 이름을 만들어 사용하는 단어
- 식별자 작성 규칙
 - 영문 대/소문자(A~Z, a~z), 숫자(0~9), 밑줄(_)을 포함한 63개의 문자로만 구성
 - 첫 글자로 숫자를 사용할 수 없음
 - 대소문자를 구별하며, 키워드는 사용할 수 없음
 - 올바르게 사용된 식별자와 잘못 사용된 식별자의 예
 - 올바른 예: data7, _data, _score, iValue, m_7Name, SEOUL
 - 잘못된 예: 1data, -data, int, _@mail, #count, double

상수(constant)

- 프로그램의 실행 시작부터 끝날 때까지 값이 변하지 않는 자료
- 문자 1개는 ' '로 표시하고, 문자열은 " "로 표시
- 기호 상수는 전처리문인 #define을 이용해 상수명과 값을 지정한 후 사용
 - 컴파일 이전에 상수명(PI) 단어를 찾아 모두 값(3.141592)로 단순 치환하는 개념
- 리터럴 상수(Literal constants)
- 정의된 상수(Defined constants) 또는 매크로 상수(Macro constants)
`#define PI 3.141592`
- 메모리 상수(Memory constants)
`const double PI = 3.141592;`

상수 종류	예
정수형	50, 100, -120
문자열	"Hello, World"
기호	#define PI 3.141592
실수형	3.141592, 2.88e-4
문자	'b', 'z', '6'

변수(variable)

- 변수

- 값이 계속 변환될 수 있는 값
- 프로그램에서 임시로 자료 값을 저장할 수 있는 저장 장소
- 변수에 값을 저장할 수 있고 이 값은 계속 변경 가능
- 변수 선언을 해야 사용할 수 있음

- 변수 선언의 예

```
int age;  
자료형 변수명;
```



① 자료형 : 사용하는 변수 종류

- int : 정수형 변수 선언할 때 사용

② 변수명 : 실제로 정수값이 저장되는 곳의 이름

- age : 변수명

변수(variable)

- 변수의 초기화
 - 선언된 변수에 처음으로 값을 저장하는 것

종류	사용 예
선언과 동시에 초기화하는 방법	<code>int age = 20;</code>
선언 후에 초기화하는 방법	<code>int age; age = 20;</code>

- 초기값이 저장되기 전 age에는 알 수 없는 무의미한 값(dummy) 저장
- 변수값 변경 방법 : 초기값 저장 후 직접 값 지정

```
int age = 20;  
age = 22;
```

변수(variable)

- 변수 동시 선언
 - 동일한 자료형의 변수를 여러 개 사용해야 할 경우, 동시에 선언

구분	사용 예
변수 2개를 따로 선언할 경우	<code>int a = 10;</code> <code>int b = 20;</code>
변수 2개를 동시에 선언할 경우	<code>int a = 10, b = 20;</code>

변수(variable) - 실습1

- 변수를 선언하고 초기화하는 예제

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int num1 = 10, num2;
06
07     printf("num1 초기값 : %d\n", num1);
08     printf("num2 초기값 : %d\n", num2);
09
10     num1 = 20;    // num1 값에 새로운 정수값 저장
11     num2 = 30;    // num2 값에 새로운 정수값 저장
12
13     printf("num1 새로운 값 : %d\n", num1);
14     printf("num2 새로운 값 : %d\n", num2);
15     return 0;
16 }
```

코드	설명
C4700	초기화되지 않은 'num2' 지역 변수를 사용했습니다.

변수(variable) - 실습2

- 다음 소스코드를 실행하면 나타나는 오류를 수정하십시오.

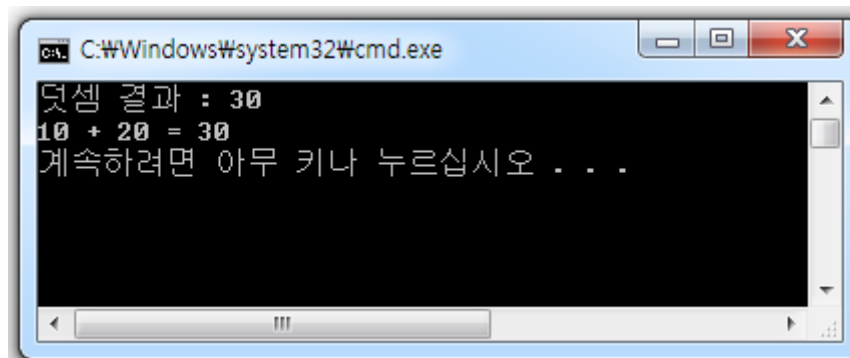
```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int num1 = 10;
06     const int num2 = 50;
07
08     printf("num1 초기값 : %d\n", num1);
09     printf("num2 초기값 : %d\n", num2);
10
11     num1 = 20;    // num1 값에 새로운 정수값 저장
12     num2 = 30;    // num2 값에 새로운 정수값 저장
13
14     printf("num1 새로운 값 : %d\n", num1);
15     printf("num2 새로운 값 : %d\n", num2);
16     return 0;
17 }
```

코드	설명
E0137	식이 수정할 수 있는 lvalue여야 합니다.
C3892	'num2': const인 변수에 할당할 수 없습니다.

변수(variable) - 실습3

- 다음 소스코드를 실행하십시오.

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int num1 = 10, num2 = 20;
06     int result = num1 + num2;
07
08     printf("덧셈 결과 : %d\n", result);
09     printf("%d + %d = %d\n", num1, num2, result);
10
11     return 0;
12 }
```

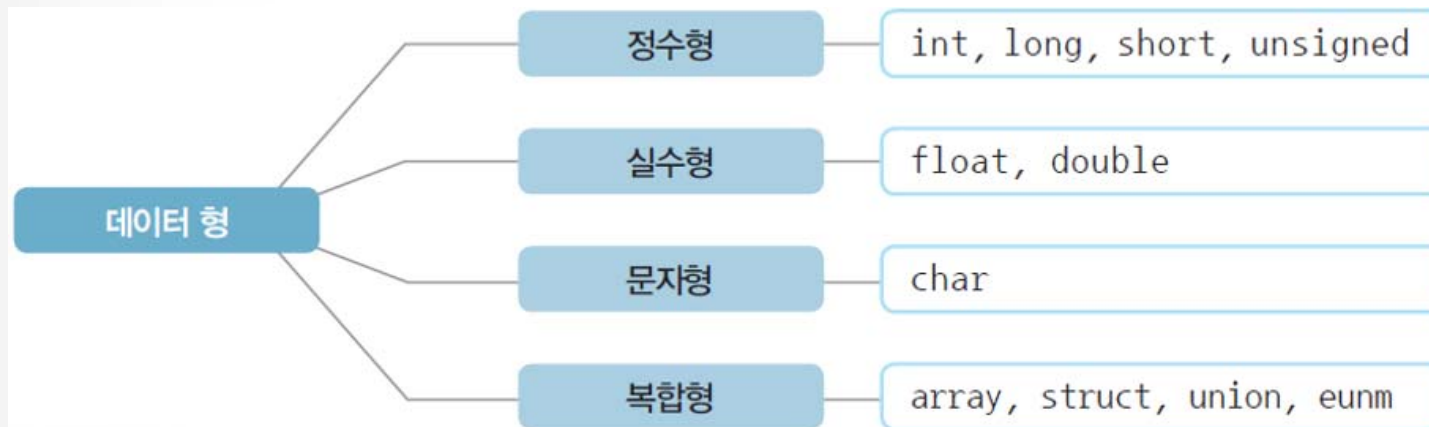


The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The output of the program is displayed as follows:

```
덧셈 결과 : 30
10 + 20 = 30
계속하려면 아무 키나 누르십시오 . . .
```

기본 자료형

- 자료형
 - 프로그램에서 선언된 변수들이 기억 공간에서 어떻게 저장되고 처리되어야 할지, 컴파일러에 알려줌
 - 정수형, 실수형, 문자형 등의 자료형, 사용자 정의 자료형



정수형

- 정수
 - 소수점이 없는 숫자
 - 정수형 상수에는 양수, 0, 음수가 있으며 양수, 음수에 따라 +, - 부호 사용 가능

정수형	바이트 수	허용 범위
(signed) short (int)	2	$-2^{15} \sim 2^{15} - 1$
unsigned short (int)	2	$0 \sim 2^{16} - 1$
(signed) int	4	$-2^{31} \sim 2^{31} - 1$
unsigned int	4	$0 \sim 2^{32} - 1$
long (int)	4	$-2^{31} \sim 2^{31} - 1$
unsigned long (int)	4	$0 \sim 2^{32} - 1$

-32,768 ~ 32,767

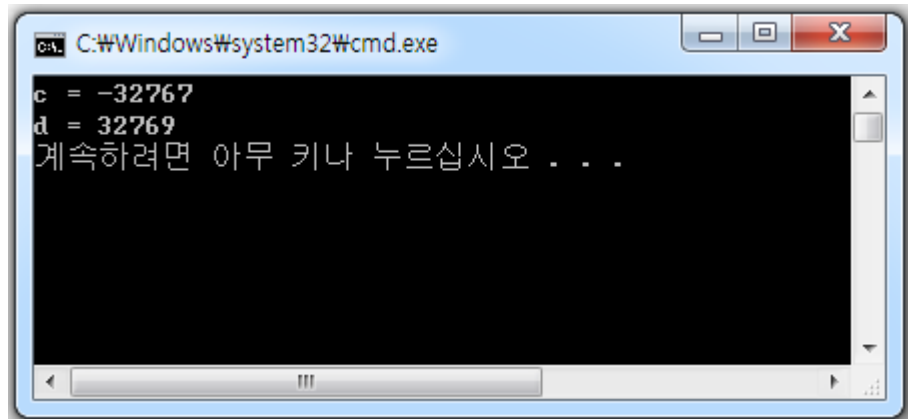
0 ~ 65,535

- 오버플로우(overflow)
 - 데이터의 허용 범위를 넘는 값을 변수에 저장할 때, 사용자가 의도한 값이 아닌 전혀 다른 값이 저장되는 것

정수형

- 정수형의 허용 범위와 오버플로우 예제

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      short a=32767, b=2, c;
06      unsigned short d;
07
08      c = a + b;
09      d = a + b;
10
11      printf("c = %d\n", c);
12      printf("d = %d\n", d);
13
14      return 0;
15  }
```

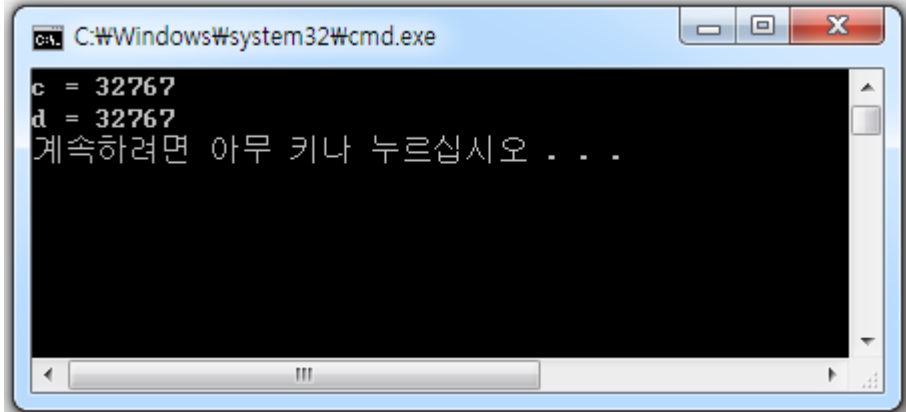


```
C:\Windows\system32\cmd.exe
c = -32767
d = 32769
계속하려면 아무 키나 누르십시오 . . .
```


정수형

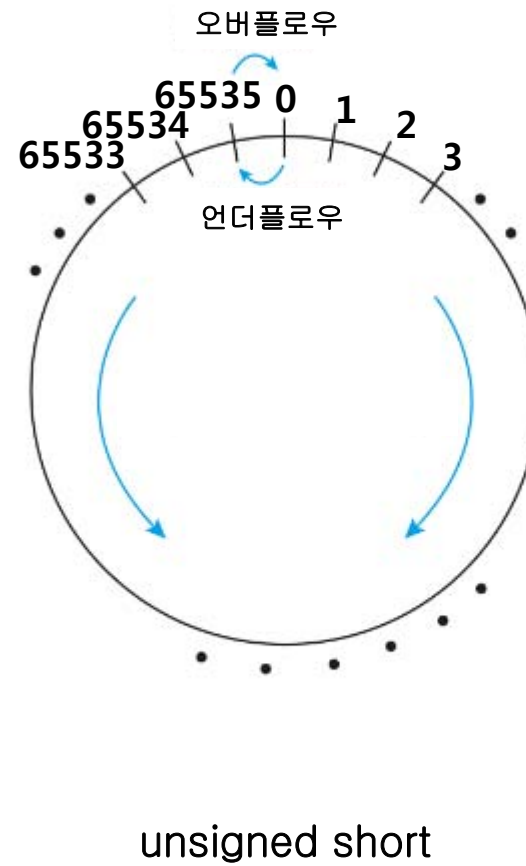
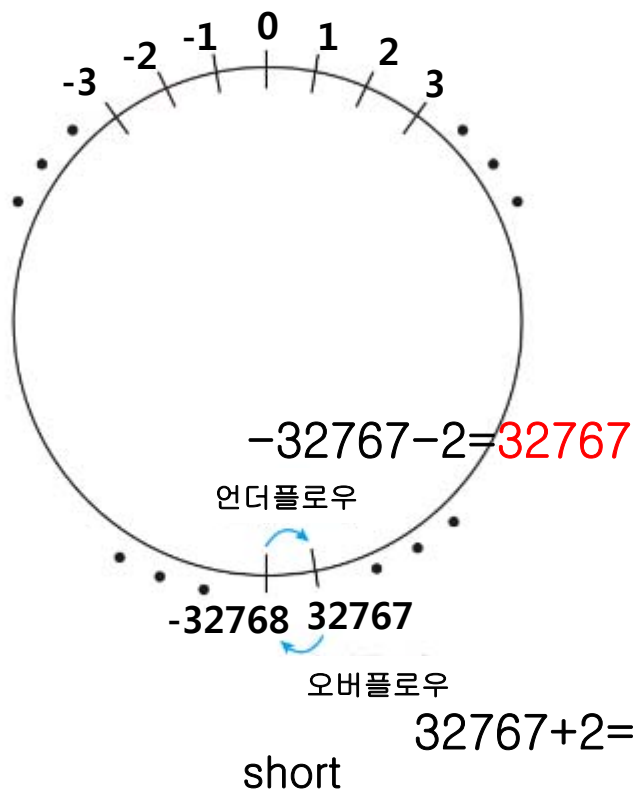
- 정수형의 허용 범위와 언더플로우 예제

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      short a=-32767, b=-2, c;
06      unsigned short d;
07
08      c = a + b;
09      d = a + b;
10
11      printf("c = %d\n", c);
12      printf("d = %d\n", d);
13
14      return 0;
15  }
```




```
C:\Windows\system32\cmd.exe
c = 32767
d = 32767
계속하려면 아무 키나 누르십시오 . . .
```

정수형



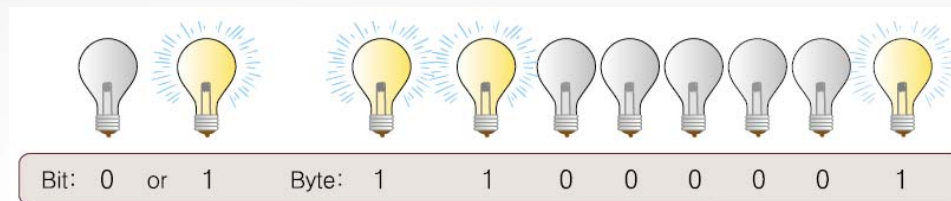


정수형

- 
- 유튜브는 왜 싸이의 강남스타일 뮤직 비디오 조회수를 21억 이상 셀 수 없었던 걸까?
 - <http://newspeppermint.com/2014/12/16/binary-bug/>

데이터 표현 단위

- 비트(bit)와 바이트(byte)



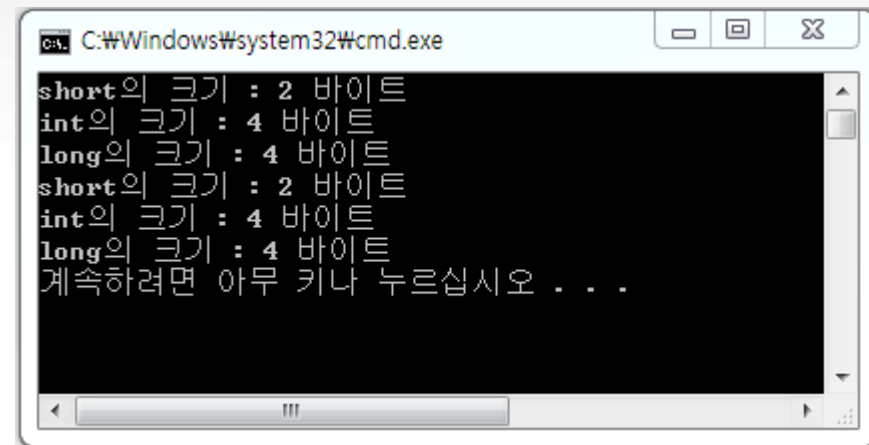
- 데이터 표현 단위

비트(bit)	0, 1
바이트(byte)	1byte=8bit
킬로바이트(KB: KiloByte)	1KB=1024byte= 2^{10} byte
메가바이트(MB: MegaByte)	1MB=1024Kbyte= 2^{20} byte
기가바이트(GB: GigaByte)	1GB=1024Mbyte= 2^{30} byte
테라바이트(TB: TeraByte)	1TB=1024Gbyte= 2^{40} byte
페타바이트(PB: PetaByte)	1PB=1024Tbyte= 2^{50} byte

자료형 크기

- sizeof 표현식
- sizeof(표현식)
- sizeof(자료형)

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      short x = 1;
6      int y=2;
7      long z=3;
8
9      printf("short의 크기 : %d 바이트\n", sizeof x); //sizeof 표현식
10     printf("int의 크기 : %d 바이트\n", sizeof(y)); //sizeof(표현식)
11     printf("long의 크기 : %d 바이트\n", sizeof(z));
12     printf("short의 크기 : %d 바이트\n", sizeof(short)); //sizeof(자료형)
13     printf("int의 크기 : %d 바이트\n", sizeof(int));
14     printf("long의 크기 : %d 바이트\n", sizeof(long));
15
16     return 0;
17 }
18
```



```
C:\Windows\system32\cmd.exe
short의 크기 : 2 바이트
int의 크기 : 4 바이트
long의 크기 : 4 바이트
short의 크기 : 2 바이트
int의 크기 : 4 바이트
long의 크기 : 4 바이트
계속하려면 아무 키나 누르십시오 . . .
```

실수형

- 실수형
 - 소수점이나 지수가 있는 수
 - float, double, long double이 있음

종류	바이트 수	선언 예
float	4	float a = 3.14f
double	8	double a = 5.6847
long double	8	long double a = 5.68424L

- float a = 3.14;

warning C4305: '초기화 중': 'double'에서 'float'(으)로 잘립니다.

실수형

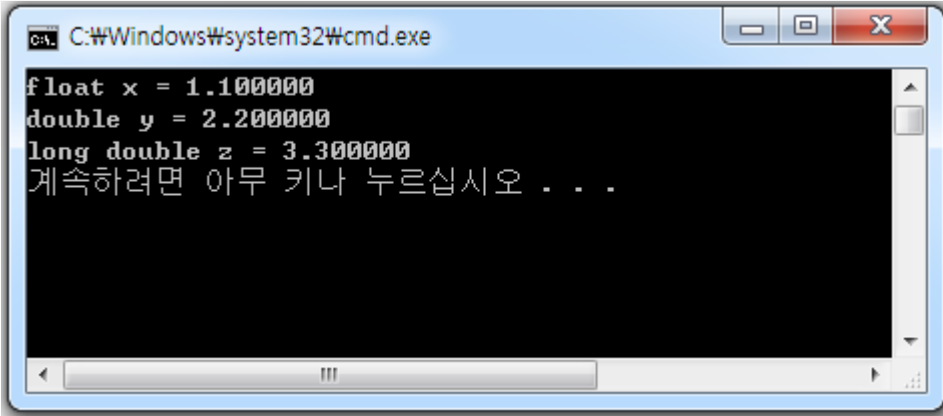
종류	바이트 수	범위	소수점 이하 정밀도	비고
float	4	$-3.4 \times 10^{38} \sim 3.4 \times 10^{38}$	6	단정밀도
double	8	$-1.79 \times 10^{308} \sim 1.79 \times 10^{308}$	15	배정밀도
long double	8	$-1.79 \times 10^{308} \sim 1.79 \times 10^{308}$	18	배정밀도

- 실수형
 - 실수를 출력할 때의 서식문자(변환기호) : `print()`문
 - float인 경우 : `%f`
 - double인 경우 : `%f`
 - long double인 경우 : `%lf`

실수형

- 실수형 변수를 선언하고 초기화하는 예제

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     float x = 1.1f;
06     double y = 2.2;
07     long double z = 3.3L;
08
09     printf("float x = %f \n", x);
10     printf("double y = %f \n", y);
11     printf("long double z = %Lf \n", z);
12
13     return 0;
14 }
```



```
C:\Windows\system32\cmd.exe
float x = 1.100000
double y = 2.200000
long double z = 3.300000
계속하려면 아무 키나 누르십시오 . . .
```

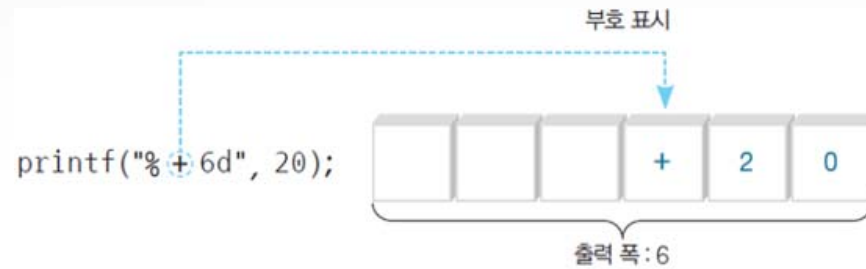
실수형

- 실수는 출력되는 값의 폭을 조정할 수 있고 출력되는 방향도 조정 가능
- `printf("%d", 20);`
- `printf("%6d", 20);`
 - `%6d`에서 6은 출력폭이 6칸이라는 것을 의미
 - 값은 기본적으로 오른쪽 정렬



실수형

- `printf("%+6d", 20);`
 - `%+6d`에서 `+`는 출력시 부호와 값이 함께 출력



- `printf("%-6d", 20);`
 - `%-6d`에서 `-`는 출력되는 값을 왼쪽 정렬



실수형

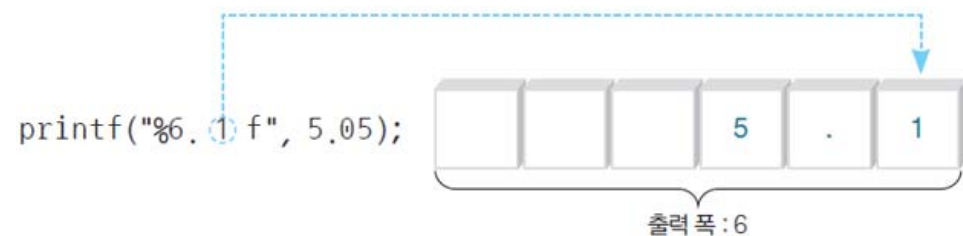
- 실수형을 출력하는 경우, 출력 폭은 정수형과 작성하는 방법이 동일
- 소수점의 자리수 지정 가능

- `printf("%f", 5.06);`



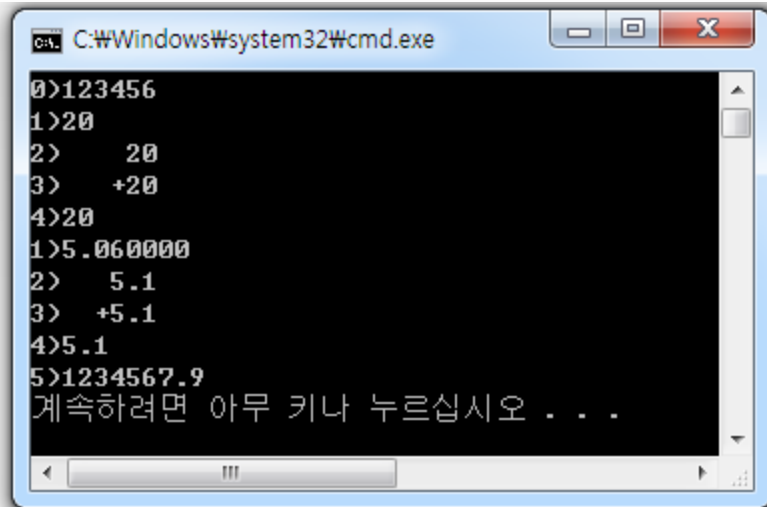
- `printf("%6.1f", 5.06);`

- %6.1f에서 6은 전체 출력 폭을 의미
- 1은 소수점의 자리수를 한자리로 한다는 의미
- 소수점 자리에 맞게 숫자는 반올림됨



실수형

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      int a = 20;
06      float b = 5.06f;
07      float c = 1234567.89f;
08
09      printf("0)%d \n", 123456);
10      printf("1)%d \n", a);
11      printf("2)%6d \n", a);
12      printf("3)%+6d \n", a);
13      printf("4)%-6d \n", a);
14
15      printf("1)%f \n", b);
16      printf("2)%6.1f \n", b);
17      printf("3)%+6.1f \n", b);
18      printf("4)%-6.1f \n", b);
19      printf("5)%6.1f \n", c);
20
21      return 0;
22  }
```



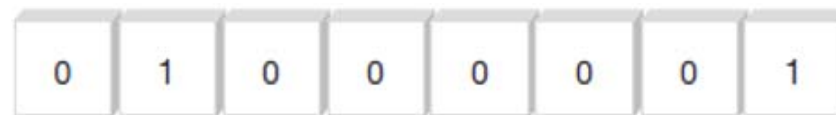
```
C:\Windows\system32\cmd.exe
0>123456
1>20
2> 20
3> +20
4>20
1>5.060000
2> 5.1
3> +5.1
4>5.1
5>1234567.9
계속하려면 아무 키나 누르십시오 . . .
```


문자형

- 문자형

```
char a = 'A' ;
```

- char : 문자형을 다루는 자료형
- 문자 1개를 작은따옴표(' ') 사이에 넣어서 사용하는 값
- 작은따옴표 내에 있는 문자를 데이터 1개로 취급함
- 사용되는 문자는 8비트(1바이트)로 처리됨
- 내부적인 문자 코드(아스키(ASCII) 코드)에 상응하는 숫자로 바뀌어 기억됨(총 128개의 문자)
- 0 ~ 127 사이의 부호 없는(unsigned) 정수에 문자를 정의한다.



2^6

+

2^0

=

65

문자형

- C 언어에서 다루는 문자
 - 아스키 코드 표에는 숫자로 표현되어 있음

구분	종류	의미
일반문자	A ~ Z	대문자를 나타낸다.
	a ~ z	소문자를 나타낸다.
	0 ~ 9	숫자 문자를 나타낸다.
	\$.),(특수 기호를 나타낸다.
특수문자	\a	벨소리를 낸다.
	\b	왼쪽으로 한 문자를 지운다.
	\f	프린터에서 한 페이지를 이동한다.
	\n	새로운 줄로 바꾼다.
	\r	커서를 1열로 이동한다.
	\t	한 탭을 이동한다.
	\',\'	\',\'를 출력시킨다.
	\0	Null 문자를 처리한다.

문자형

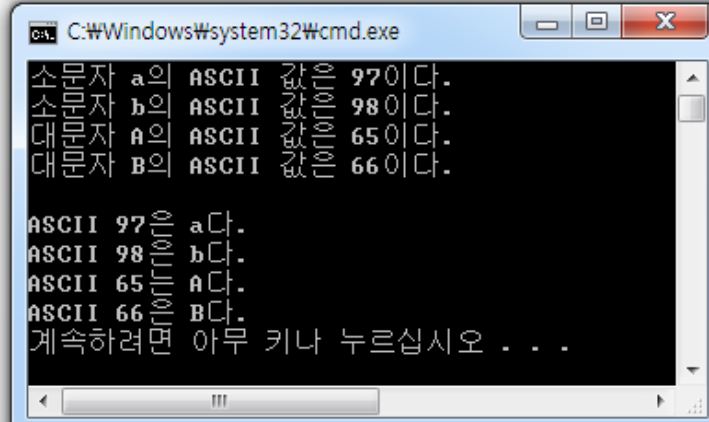
- 아스키 코드 (ASCII)
 - American Standard Code for Information Interchange
 - 숫자와 문자를 연결시켜주는 프로토콜
 - Dec: 정수
 - Hex: 16진수
 - Ex) A : 65 정수값(Dec)
a : 97 정수값(Dec)

Dec	Hex	문자	Dec	Hex	문자	Dec	Hex	문자	Dec	Hex	문자
0	0x00	NUL	32	0x20	SP	64	0x40	@	96	0x60	`
1	0x01	SOH	33	0x21	!	65	0x41	A	97	0x61	a
2	0x02	STX	34	0x22	"	66	0x42	B	98	0x62	b
3	0x03	ETX	35	0x23	#	67	0x43	C	99	0x63	c
4	0x04	EOT	36	0x24	\$	68	0x44	D	100	0x64	d
5	0x05	ENQ	37	0x25	%	69	0x45	E	101	0x65	e
6	0x06	ACK	38	0x26	&	70	0x46	F	102	0x66	f
7	0x07	BEL	39	0x27	'	71	0x47	G	103	0x67	g
8	0x08	BS	40	0x28	(72	0x48	H	104	0x68	h
9	0x09	HT	41	0x29)	73	0x49	I	105	0x69	i
10	0x0a	LF	42	0x2a	*	74	0x4a	J	106	0x6a	j
11	0x0b	VT	43	0x2b	+	75	0x4b	K	107	0x6b	k
12	0x0c	FF	44	0x2c	,	76	0x4c	L	108	0x6c	l
13	0x0d	CR	45	0x2d	-	77	0x4d	M	109	0x6d	m
14	0x0e	SO	46	0x2e	.	78	0x4e	N	110	0x6e	n
15	0x0f	SI	47	0x2f	/	79	0x4f	O	111	0x6f	o
16	0x10	DLE	48	0x30	0	80	0x50	P	112	0x70	p
17	0x11	DC1	49	0x31	1	81	0x51	Q	113	0x71	q
18	0x12	DC2	50	0x32	2	82	0x52	R	114	0x72	r
19	0x13	DC3	51	0x33	3	83	0x53	S	115	0x73	s
20	0x14	DC4	52	0x34	4	84	0x54	T	116	0x74	t
21	0x15	NAK	53	0x35	5	85	0x55	U	117	0x75	u
22	0x16	SYN	54	0x36	6	86	0x56	V	118	0x76	v
23	0x17	ETB	55	0x37	7	87	0x57	W	119	0x77	w
24	0x18	CAN	56	0x38	8	88	0x58	X	120	0x78	x
25	0x19	EM	57	0x39	9	89	0x59	Y	121	0x79	y
26	0x1a	SUB	58	0x3a	:	90	0x5a	Z	122	0x7a	z
27	0x1b	ESC	59	0x3b	;	91	0x5b	[123	0x7b	{
28	0x1c	FS	60	0x3c	<	92	0x5c	\	124	0x7c	
29	0x1d	GS	61	0x3d	=	93	0x5d]	125	0x7d	}
30	0x1e	RS	62	0x3e	>	94	0x5e	^	126	0x7e	~
31	0x1f	US	63	0x3f	?	95	0x5f	_	127	0x7f	DEL

문자형

- 알파벳 문자와 아스키 코드값에 관한 예제

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      printf("소문자 a의 ASCII 값은 %d이다. \n", 'a');
06      printf("소문자 b의 ASCII 값은 %d이다. \n", 'b');
07      printf("대문자 A의 ASCII 값은 %d이다. \n", 'A');
08      printf("대문자 B의 ASCII 값은 %d이다. \n\n", 'B');
09
10      printf("ASCII 97은 %c다. \n", 97);
11      printf("ASCII 98은 %c다. \n", 98);
12      printf("ASCII 65는 %c다. \n", 65);
13      printf("ASCII 66은 %c다. \n", 66);
14
15      return 0;
16  }
```



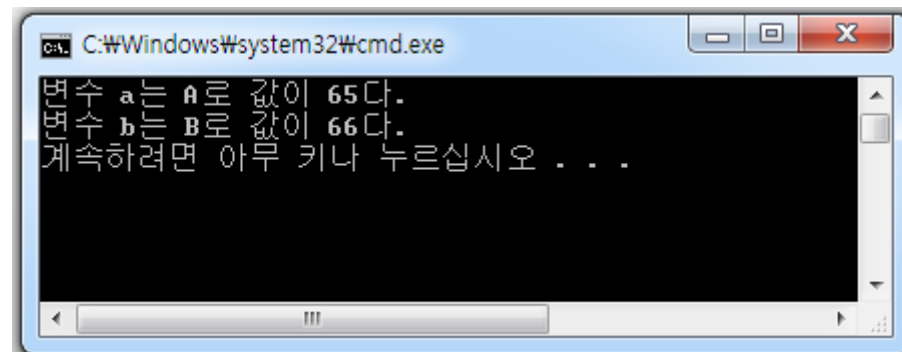
```
C:\Windows\system32\cmd.exe
소문자 a의 ASCII 값은 97이다.
소문자 b의 ASCII 값은 98이다.
대문자 A의 ASCII 값은 65이다.
대문자 B의 ASCII 값은 66이다.

ASCII 97은 a다.
ASCII 98은 b다.
ASCII 65는 A다.
ASCII 66은 B다.
계속하려면 아무 키나 누르십시오 . . .
```

문자형

- 다음 프로그램을 실행해보시오.

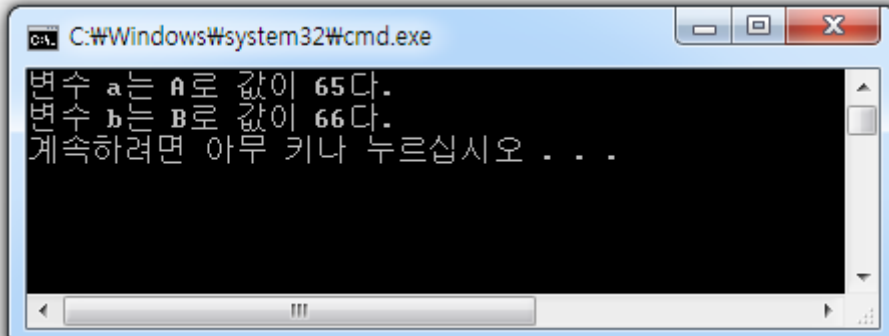
```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      char a = 'A';
06      char b = 'B';
07
08      printf("변수 a는 %c로 값이 %d다. \n", a, a);
09      printf("변수 b는 %c로 값이 %d다. \n", b, b);
10
11      return 0;
12  }
```



문자형

- 다음 프로그램을 실행해보시오.

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      char a = 65;
06      char b = 66;
07
08      printf("변수 a는 %c로 값이 %d다. \n", a, a);
09      printf("변수 b는 %c로 값이 %d다. \n", b, b);
10
11      return 0;
12  }
```



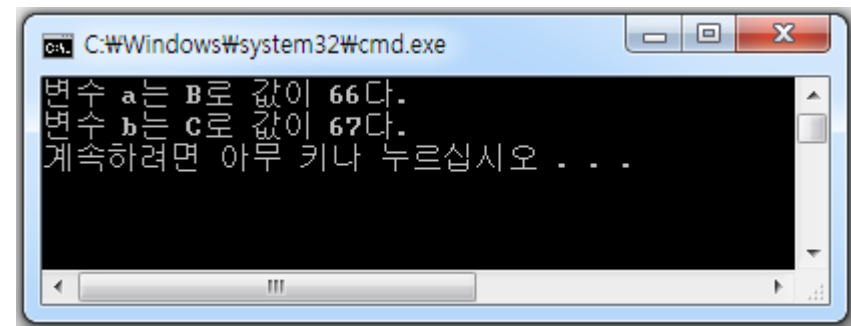
C:\Windows\system32\cmd.exe

```
변수 a는 A로 값이 65다.
변수 b는 B로 값이 66다.
계속하려면 아무 키나 누르십시오 . . .
```


문자형

- 다음 프로그램을 실행해보시오.

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      char a = 'A' + 1;
06      char b = 'B' + 1;
07
08      printf("변수 a는 %c로 값이 %d다.\n", a, a);
09      printf("변수 b는 %c로 값이 %d다.\n", b, b);
10
11      return 0;
12  }
```



형 변환

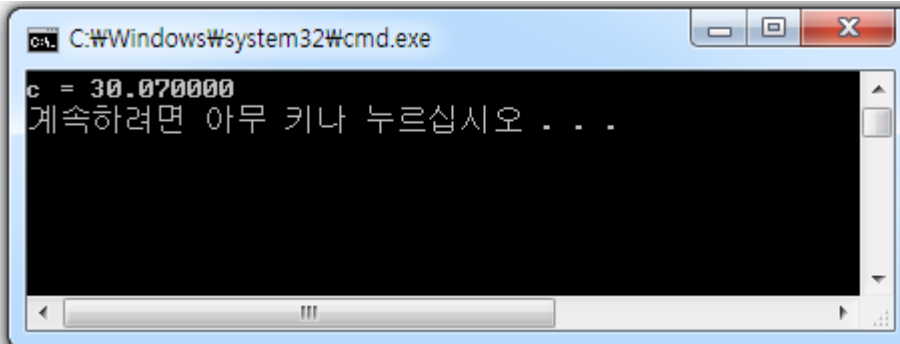
- 형 변환
 - 하나의 자료형을 다른 자료형으로 변환시켜 자료형을 같게 하는 것
 - 변수 2개로 연산을 수행할 경우에 사용함
- 묵시적 형 변환
 - 컴파일러가 자동으로 변환
 - 데이터의 값을 잃지 않는 쪽으로 형변환이 이루어짐

`char → short → int → unsigned → long → float → double`

- 정수를 실수로 형 변환하는 경우
- 실수를 정수로 형 변환하는 경우

형 변환

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int a = 10;
06     float b = 20.07f, c;
07
08     c = a + b;
09
10     printf("c = %f \n", c);
11
12     return 0;
13 }
```



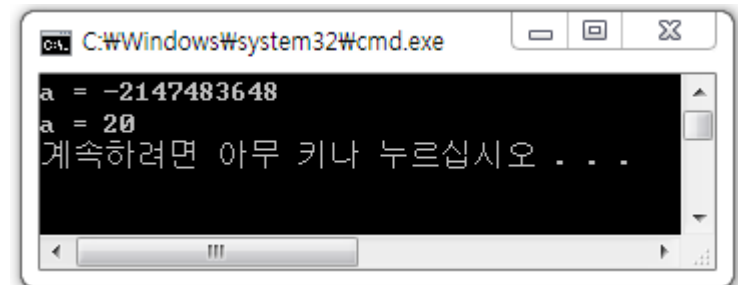
A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Windows\system32\cmd.exe'. The command prompt displays the output of the C program: 'c = 30.070000' followed by a Korean message '계속하려면 아무 키나 누르십시오 . . .'. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

형 변환

- 명시적 형 변환
 - 사용자가 강제로 형변환 시키는 것
 - 캐스팅 연산자를 사용
 - 연산식 앞에 형변환연산자()를 붙이고 ()안에는 변환시키려는 자료형

예) float형의 변수를 int형으로 강제 변환

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      float a = 20.07f;
06
07      printf("a = %d \n", a);
08      printf("a = %d \n", (int)a);
09
10      return 0;
11  }
```

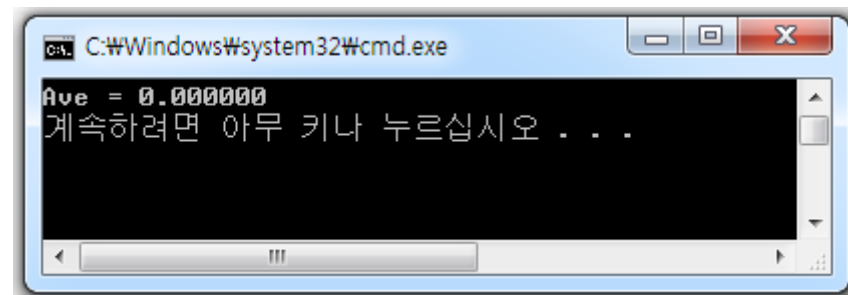


```
C:\Windows\system32\cmd.exe
a = -2147483648
a = 20
계속하려면 아무 키나 누르십시오 . . .
```

형 변환

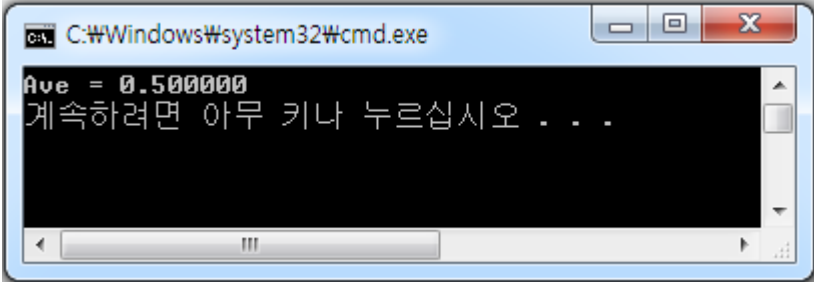
```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int x = 1, y = 2;
06     float Average;
07
08     Average = x / y;
09
10     printf("Ave = %f \n", Average);
11
12     return 0;
13 }
```

정수/정수=정수



형 변환

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int x = 1, y = 2;
06     float Average;
07
08     Average = (float)x / y;
09     // Average = x / (float)y;
10     // Average = (float)x / (float)y;
11
12     printf("Ave = %f \n", Average);
13
14     return 0;
15 }
```



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window shows the output of the C program: "Ave = 0.500000" followed by the Korean text "계속하려면 아무 키나 누르십시오 . . .".

형 변환

- 묵시적 형변환과 명시적 형변환

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int a = 10, b = 20, c;
06     float d = 5.6, e = 8.5, f;
07
08     c = d + e;
09     printf("c = d + e 연산결과 : %.2f\n", (float)c);
10
11     f = d + e;
12     printf("f = d + e 연산결과 : %.2f\n", f);
13
14     f = d + (int)e;
15     printf("f = d + (int)e 연산결과 : %.2f\n", f);
16
17     return 0;
18 }
```