C 프로그래밍 및 실습

[2주차 1강]

변수와 자료형

학습 내용

2.1 변수와 자료형 개요
2.2 변수 선언과 사용
2.3 정수 자료형
2.4 부동소수 자료형
2.5 문자 자료형
2.6 자료형 변환

학습 목표

2.1 변수와 자료형이란 무엇인지 이해한다.
2.2 변수를 선언하고 사용할 수 있다.
2.3 정수 자료형에 대해 이해한다.
2.4 부동소수 자료형에 대해 이해한다.
2.5 문자 자료형에 대해 이해한다.
2.6 자료형 변환에 대해 이해한다.

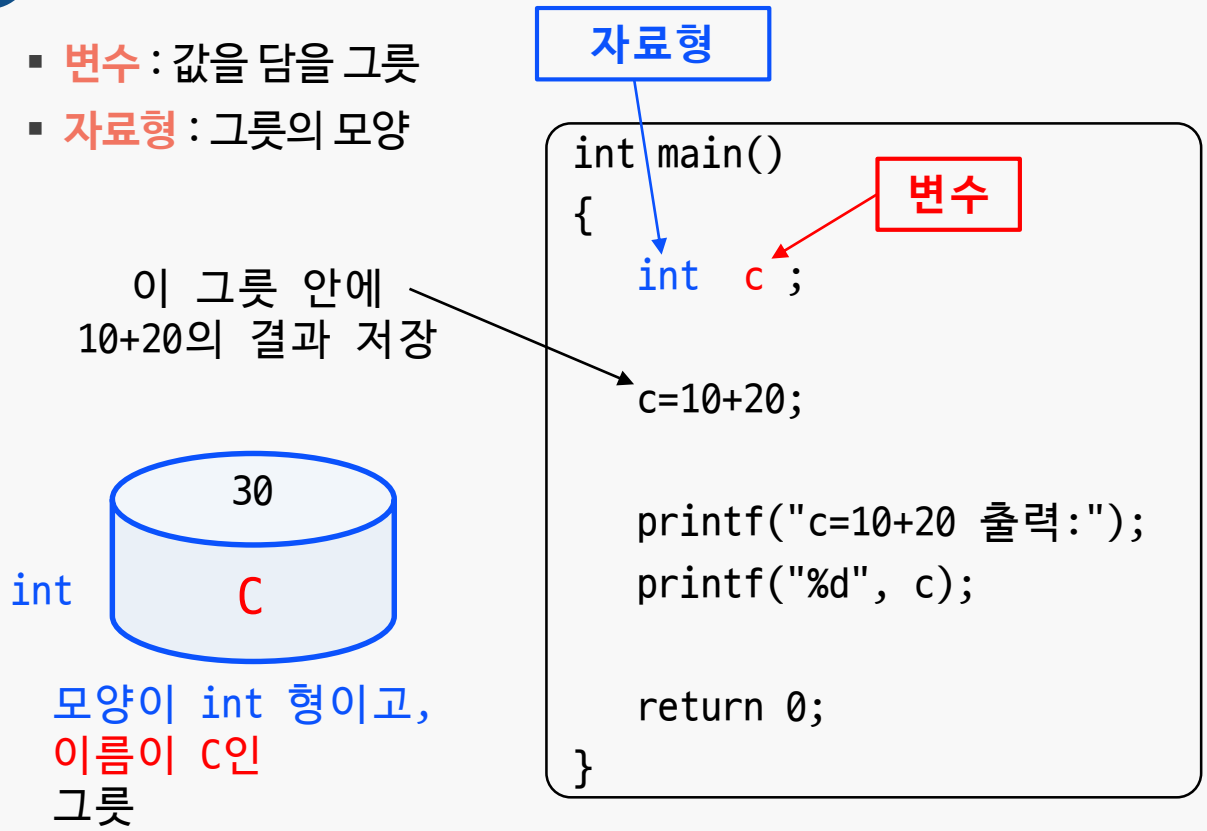
C 프로그래밍 및 실습

2.1 변수와 자료형 개요
2.2 변수 선언과 사용
2.3 정수 자료형
2.4 부동소수 자료형
2.5 문자 자료형
2.6 자료형 변환

세종대학교

변수와 자료형

- **변수** : 값을 담을 그릇
- **자료형** : 그릇의 모양



변수, 자료형, 상수

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

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

세종대학교

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

세종대학교

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
```

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
```

세종대학교

[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
```

세종대학교

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;
```

세종대학교

[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
```

키워드와 식별자

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
3card	999	▪ 첫 문자에 숫자 사용할 수 없음
int	char	▪ 키워드는 사용할 수 없음

C 프로그래밍 및 실습

2.1 변수와 자료형 개요
2.2 변수 선언과 사용
2.3 정수 자료형
2.4 부동소수 자료형
2.5 문자 자료형
2.6 자료형 변환

세종대학교

정수 자료형 종류

- **int** : 정수를 나타내는 가장 기본적인 자료형
- **short, long, long long** : 정수를 나타내지만 자료형의 크기가 다름

short ≤ int ≤ long ≤ 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) int의 크기는 보통 4 바이트 (32 비트)로 총 2^{32}개의 수 표현 가능
    반은 음수, 나머지 반은 0과 양수를 표현하여
    -2^{31} ~ $2^{31} - 1$ 사이의 정수를 나타냄

signed와 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

세종대학교

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 ~ 65,535
	unsigned int	4 bytes	0 ~ 4,294,967,295
	unsigned long	4 bytes	0 ~ 4,294,967,295
	unsigned long long	8 bytes	0 ~ 18,446,744,073,709,551,615

세종대학교

다양한 정수 자료형 사용하기

```
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);
                    unsigned 값을 출력할 경우 %u 사용
    return 0;
}
```

실행결과

```
저장값 : 32000 -2140000000
저장값 : 65000 4280000000
```

실습 하기

[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?
```

실습 하기

[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?
```

실습 하기

[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?
```

C 프로그래밍 및 실습

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
```

세종대학교

부동소수형 표현 방식

- 0.000023의 다른 표현 ➔ 2.3×10^{-5}
- 컴퓨터 내부에서는 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
```

세종대학교

부동소수형의 크기

float ≤ double ≤ 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}

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=123456793955060940800.000000
y=12345678901234567168.000000
```

C 프로그래밍 및 실습

2.1 변수와 자료형 개요
2.2 변수 선언과 사용
2.3 정수 자료형
2.4 부동소수 자료형
2.5 문자 자료형
2.6 자료형 변환

세종대학교

문자형

char

signed char

unsigned char

- 문자형 자료형의 크기는 모두 1바이트
- 문자는 작은 따옴표 '' 를 사용하여 표현
- 출력 : printf의 서식 지정자는 %c

```
char ch = 'z';     ⇨ 문자 'z'로 초기화
printf("ch: %c\n", ch);    ⇨ 문자형 출력 시 %c 사용
```

실행결과

```
ch: z
```

세종대학교

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
```

세종대학교

아스키 코드 표 : 외울 필요 없음

아스키 코드표

10진수	16진수	8진수	2진수	ASCII	10진수	16진수	8진수	2진수	ASCII
0	0x00	000	0000000	NULL	32	0x20	040	0100000	SP
1	0x01	001	0000001	SOH	33	0x21	041	0100001	!
2	0x02	002	0000010	STX	34	0x22	042	0100010	"
3	0x03	003	0000011	ETX	35	0x23	043	0100011	#
4	0x04	004	0000100	EOT	36	0x24	044	0100100	$
5	0x05	005	0000101	ENQ	37	0x25	045	0100101	%
6	0x06	006	0000110	ACK	38	0x26	046	0100110	&
7	0x07	007	0000111	BEL	39	0x27	047	0100111	'
8	0x08	010	0001000	BS	40	0x28	050	0101000	(
9	0x09	011	0001001	HT	41	0x29	051	0101001	)
10	0x0A	012	0001010	LF	42	0x2A	052	0101010	*
11	0x0B	013	0001011	VT	43	0x2B	053	0101011	+
12	0x0C	014	0001100	FF	44	0x2C	054	0101100	'
13	0x0D	015	0001101	CR	45	0x2D	055	0101101	-
14	0x0E	016	0001110	SO	46	0x2E	056	0101110	.
15	0x0F	017	0001111	SI	47	0x2F	057	0101111	/
16	0x10	020	0010000	DLE	48	0x30	060	0110000	0
17	0x11	021	0010001	DC1	49	0x31	061	0110001	1
18	0x12	022	0010010	SC2	50	0x32	062	0110010	2
19	0x13	023	0010011	SC3	51	0x33	063	0110011	3
20	0x14	024	0010100	SC4	52	0x34	064	0110100	4
21	0x15	025	0010101	NAK	53	0x35	065	0110101	5
22	0x16	026	0010110	SYN	54	0x36	066	0110110	6
23	0x17	027	0010111	ETB	55	0x37	067	0110111	7
24	0x18	030	0011000	CAN	56	0x38	070	0111000	8
25	0x19	031	0011001	EM	57	0x39	071	0111001	9
26	0x1A	032	0011010	SUB	58	0x3A	072	0111010	:
27	0x1B	033	0011011	ESC	59	0x3B	073	0111011	;
28	0x1C	034	0011100	FS	60	0x3C	074	0111100	<
29	0x1D	035	0011101	GS	61	0x3D	075	0111101	=
30	0x1E	036	0011110	RS	62	0x3E	076	0111110	>
31	0x1F	037	0011111	US	63	0x3F	077	0111111	?

아스키 코드표

10진수	16진수	8진수	2진수	ASCII	10진수	16진수	8진수	2진수	ASCII
64	0x40	100	1000000	@	96	0x60	140	1100000	`
65	0x41	101	1000001	A	97	0x61	141	1100001	a
66	0x42	102	1000010	B	98	0x62	142	1100010	b
67	0x43	103	1000011	C	99	0x63	143	1100011	c
68	0x44	104	1000100	D	100	0x64	144	1100100	d
69	0x45	105	1000101	E	101	0x65	145	1100101	e
70	0x46	106	1000110	F	102	0x66	146	1100110	f
71	0x47	107	1000111	G	103	0x67	147	1100111	g
72	0x48	110	1001000	H	104	0x68	150	1101000	h
73	0x49	111	1001001	I	105	0x69	151	1101001	i
74	0x4A	112	1001010	J	106	0x6A	152	1101010	j
75	0x4B	113	1001011	K	107	0x6B	153	1101011	k
76	0x4C	114	1001100	L	108	0x6C	154	1101100	l
77	0x4D	115	1001101	M	109	0x6D	155	1101101	m
78	0x4E	116	1001110	N	110	0x6E	156	1101110	n
79	0x4F	117	1001111	O	111	0x6F	157	1101111	o
80	0x50	120	1010000	P	112	0x70	160	1110000	p
81	0x51	121	1010001	Q	113	0x71	161	1110001	q
82	0x52	122	1010010	R	114	0x72	162	1110010	r
83	0x53	123	1010011	S	115	0x73	163	1110011	s
84	0x54	124	1010100	T	116	0x74	164	1110100	t
85	0x55	125	1010101	U	117	0x75	165	1110101	u
86	0x56	126	1010110	V	118	0x76	166	1110110	v
87	0x57	127	1010111	W	119	0x77	167	1110111	w
88	0x58	130	1011000	X	120	0x78	170	1111000	x
89	0x59	131	1011001	Y	121	0x79	171	1111001	y
90	0x5A	132	1011010	Z	122	0x7A	172	1111010	z
91	0x5B	133	1011011	[	123	0x7B	173	1111011	{
92	0x5C	134	1011100	\	124	0x7C	174	1111100	|
93	0x5D	135	1011101	]	125	0x7D	175	1111101	}
94	0x5E	136	1011110	^	126	0x7E	176	1111110	~
95	0x5F	137	1011111	_	127	0x7F	177	1111111	DEL

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

세종대학교

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 0 으로 초기화

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

실행결과

```
문자: 0
정수:[ 빈 칸 ]
```

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	'삑' 소리를 냄	
\\	역슬래쉬 \	
\'	작은따옴표 '	
\"	큰따옴표 "	

C 프로그래밍 및 실습

세종대학교

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
```

세종대학교

자동 형변환

- 정수 ←→ 부동소수
- int a = 123.45;

```
a   ⇐   123 (정수)   ⇐   123.45 (부동소수)
                  형 변환
```

- double b = 123;

```
b   ⇐   123.0 (부동소수)   ⇐   123 (정수)
                        형 변환
```

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

세종대학교

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

```
fum1+fum2 = 26.000000
inum1+inum2 = 25
```

왜 위와 같은 결과가
나오는 지 생각해보자.

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
```

학습 정리

- **변수**는 값을 저장하기 위한 기억 장소로 사용 전에 반드시 선언해야 하고, **자료형**은 자료 값의 형태를 의미함
- 변수에 값을 저장하기 위해서는 **대입 연산자(=)**를 사용하며, 변수의 위치에 따라 **저장 공간**을 의미하기도 하고, 공간에 **저장된 값**을 의미하기도 함
- **키워드**는 C 언어에서 특별한 의미를 가지도록 미리 정해 놓은 단어이고, **식별자**는 프로그래머가 지어서 사용하는 이름임
- 자료형이 표현할 수 있는 값의 범위는 할당되는 메모리 크기에 의해 결정되고, 자료형의 크기는 **sizeof 연산자**를 이용하여 확인할 수 있음
- 컴퓨터 내부에서 **정수 3**과 **부동소수 3.0**은 전혀 다른 데이터임
- 서로 다른 자료형 사이에는 **형변환**이 필요함