C 프로그래밍 및 실습

2. 변수와 자료형

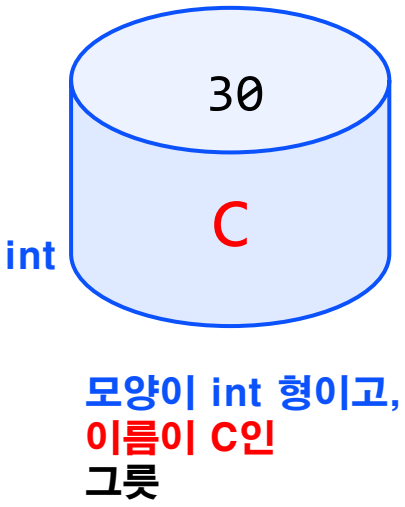
p. 1 수정
p. 5 수정
p. 6 명시한 후
p. 9 의미 다름
p. 12 total 선언
p. 36 수정 했음

세종대학교

목차

1) 변수와 자료형 개요

- 변수: 값을 담을 그릇
- 자료형: 그릇의 모양



자료형

변수

```
int main()
{
    int   c ;

    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

목차

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 snum;
    int credits;

    snum = 20160120; // 변수 snum에 값 대입(저장)
    credits = 18;

    printf("학번 : %d\n", snum); // 변수 snum 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 저장
  - 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;
   int science = 94, total;

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

   printf("수학 : %d\n", math);   // 변수 값 출력
   printf("국어 : %d\n", korean);
   printf("과학 : %d\n", science);
   printf("총점 : %d\n", total);

   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
    - ✓ 3card 999 → 첫 문자에 숫자 사용할 수 없음
    - ✓ int char → 키워드는 사용할 수 없음

목차

3) 정수 자료형

- **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

3) 정수 자료형

- **signed vs. 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 ~ 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

3) 정수 자료형

- **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);

    return 0;
}
```

unsigned 값을 출력할 경우 %u 사용

실행결과

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

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
```

3) 정수 자료형

- **[예제 2.4]**
  - 이전 프로그림에서 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;
```

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
```

목차

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 ➔ 2.3×10^{-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**
  - 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}

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=1234567939550609408 0.000000
y=12345678901234567168.000000
```

목차

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

실행결과

```
문자: A A
정수: 65 65
```

5) 문자 자료형

- 아스키 코드 표: 외울 필요 없음

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	&#127;	DEL

Source: www.asciitable.com

5) 문자 자료형

- **문자 '0'과 숫자 0의 구별**
  - 문자 '0'의 아스키 코드 값은 48
    - ✓ 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: 48 0
```

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	'삑'소리를 냄	
\\	역슬래쉬 \	
\'	작은 따옴표 '	
\"	큰 따옴표 "	

목차

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 ←→ 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)**

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("fum1+fum2 = %f \n", fnum1+fnum2);
printf("inum1+inum2 = %d \n", inum1+inum2);
```

실행결과

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

6) 자료형 변환

- **명시적 형변환**
  - printf의 서식 지정자에 따라 형변환이 자동으로 발생하지 않음

```
printf("123.45: %d\n", 123.45);
printf("123: %f\n", 123);
```

실행결과

```
123.45: -858993459
123: 0.000000
```

  - 명시적 형변환 필요

```
printf("123.45: %d\n", (int) 123.45);
printf("123: %f\n", (double) 123);
```

실행결과

```
123.45: 123
123: 123.000000
```