3장. 기본 자료형

3.1 선언, 수식, 배정

■ 선언(Declaration)

모든 변수는 사용전에 선언.

-목적 : 메모리 공간 확보, 올바른 연산 수행

```
#include      <stdio.h>
int main(void)
{
int     a,b,c;                  /*declaration*/
float   x, y=3.3, z=-7.7;       /*declaration with initializations*/

printf("Input two integer: ");      /*function call*/
scanf("%d%d",  &b, &c);             /*function call*/
a=b+c;                              /*assignment statement*/
x=y+z;                              /*assignment statement*/
....
```

■ 수식(Expression)

- meaningful combinations of constants, variables, operators, and function calls
- A constant, variable, or function call by itself can also be considered an expression

```
a+b                  /* 변수의 연산 */
sqrt(7.333)          /* 함수 호출 */
5.0 * x -tan(9.0/x)  /* 함수호출과 변수연산 */
```

■ 배정(Assignment)

- When followed by a semicolon, an expression becomes a statement, or moreexplicitly, an expression statement

```
= : 배정연산자
i=7     /* 배정 수식 */
i=7;    /* 문장 */

3.777;   ?
a+b;     ?
```

Assigment expression vs. mathematical expression

```
x+2=0    /* wrong */
x=x+1;
```

3.2 기본 자료형

Fundmental data types		
char	signed char	unsigned char
short	int	long
unsigned short	unsigned	unsigned long
float	double	long double

declaration ::= *type identifier* {, *identifier*}0+;

3.3 문자 및 char 자료형

character constant :	'a'	'A'	'0'	'&'
corresponding values :	97	65	48	38

■ 확장 문자열

확장 문자열	의 미	ASCII 코드 (16진수)
\a	경보음	07
\n	개행	0A
\t	탭코드	09
\xhh	16진수 hh	hh
\0	문자코드 0	00

단일 문자코드로 사용 : '\t' '\n'
문자열 안에서의 확장 문자열 : "\tabcde\tXYZ\n" (그냥 쓴다)
(예)

```
printf("%c", 'a');
printf("\"abc\"");      /* "abc" is printed */
printf(" 'abc' ");      /* 'abc' is printed */
```

(예) 문자 -> 정수 취급, 정수 -> 문자 취급

```
char   c='a';
printf("%c", c);        /* a is printed */
printf("%d", c);        /* 97 is printed */
printf("%c%c%c", c, c+1, c+2);   /* abc is printed */

char   c;
int    i;
for(i='a'; i<='z'; ++i)
```

```
            printf("%c", i);        /* abc ... z is printed */
        for(c=65; c<=90; ++c)
            printf("%c", c);        /* ABC ... Z is printed */
        for(c='0'; c<='9'; ++c)
            printf("%d", c);        /* 48 49 ... 57  is printed */
```

- signed char : -128 ~127, - unsigned char : 0 ~ 255

3.4 int 자료형

- int 형
  (32bit) : -2^{31} ~ $2^{31}-1$ (-21억 ~ 21억)
  (16bit) : -2^{16} ~ $2^{16}-1$ (-32천 ~ 32천)

```
#define     BIG   2000000000   /* 2billion */
int main(void)
{
        int     a, b = BIG, c=BIG;
        a=b+c;          /* out of range */
        .....
```

==> a <- 부정확한 값 (integer overflow)

3.5 정수적 형 short, long, unsigned

- short 형
  -2^{16} ~ $2^{16}-1$ (-32천 ~ 32천)
- long 형
  (32bit) : -2^{31} ~ $2^{31}-1$ (-21억 ~ 21억)
- unsigned 형
  (32bit) : 0 ~ $2^{32}-1$ (0 ~ 42억)
  (16bit) : 0 ~ $2^{16}-1$ (0 ~ 65천)

3.6 부동형

■ 부동형 : float, double, long double

Suffix	Type	Example
f or F	float	3.7F
l or L	long double	3.7L

Any unsuffixed floating constant is of type double

(예) 부동형 상수의 예

```
3.14159
314.159e-2F  /* of type float */
0e0          /* equivalent to 0.0 */
1.           /* equivalent to 1.0, but harder to read */
```

(예) 부동형 상수가 아닌 예

```
3.14,159     /* comma not allowed */
314159       /* decimal point or exponential part needed */
.e0          /* integer part(0) or frractional part(.77) is needed */
```

- float : 4 bytes, 10^{38} ~ 10^{-38} ,유효숫자 6자리
- double : 8 bytes, 10^{308} ~ 10^{-308} ,유효숫자 15자리

3.7 typedef의 사용

■ typedef : 식별자를 특정한 형과 연관

```
typedef     char              uppercase;
typedef     int               INCHES, FEET;
typedef     unsigned long     size_t;
- 변수, 함수 선언시 사용
uppercase   u;
INCHES      length, width;
```

3.8 sizeof 연산자 (Not function)

■ sizeof : 객체 저장시 메모리 할당 byte 수 알기 위해 사용

sizeof(*object*)

sizeof(char)=1
sizeof(short) ≤ sizeof(int) ≤ sizeof(long)
sizeof(signed) = sizeof(unsigned) = sizeof(int)
sizeof(float) ≤ sizeof(double) ≤ sizeof(long double)

3.9 getchar()와 putchar()의 사용

■ getchar() : 키보드에서 문자 읽는 매크로
■ putchar() : 화면에 문자 출력하는 매크로

(예)

```
#include      <stdio.h>
int main(void)
{
        int c;
        while ( (c=getchar() ) != EOF ) {
                putchar(c);
                putchar(c);
        }
        return 0;
}
```

▶ #include <stdio.h>
#define EOF (-1)

(예) 소문자 -> 대문자

```
#include      <stdio.h>
int main(void)
{
        int c;
        while ( (c=getchar() ) != EOF ) {
                if(c>= 'a' && c <= 'z')
                        putchar(c + 'A' - 'a');
                else
                        putchar(c);
        }
        return 0;
}
```

3.10 수학 함수

■ 수학 Library에서 제공

sqrt() pow() exp() log() sin() cos() tan()

(예)

```
#include      <math.h>
#include      <stdio.h>
int main(void)
{
        double        x;
        printf("/n%s/n%s/n%s/n/n",
                "The square root of x and x raised",
                "to the x power will be computed.",
                "____");
        while(1){                              /* do it forever  */
                printf("Input x:  ");
                scanf("%lf", &x);
                if(x>=0.0)
                printf("\n%15s%22.15e\n%15s%22.l5e\n%15s%22.l5e\n\n,
                "x =",x,
                "sqrt(x) = " , sqrt(x),
                "pow(x,x) = ", pow(x, x);
        else
                printf("\n Sorry, your number must be nonnegative. \n\n");
        }
        return 0;
}
```

===>

```
        The square root of x and x raised
        to the x power will be computed.
        _ _ _

        Input x:  2
                        x=2.000000000000000e+000
                sqrt(x)=1.4144213562373095e+000
              pow(x, x)=4.000000000000000e+000
        Input x:
```

■ abs() --> int, stalib.h, fabs() --> double, math.h

3.11 변환과 캐스트

■ 일반적 자동 변환 - 수식에서 제일 큰 형으로 변환한다.

```
char c;      short s;       int i;
unsigned u;    unsigned long ul;
float f;      double d;      long double ld;
```

수식	형
u * 7 - i	unsigned
f * 7 - i	float
7 * s * ul	unsigned long
ld + c	long double
u - ul	unsigned long
u - l	system-dependent

수식	형
c - s / i	int
u * 2.0 - i	double
c + 3	int
c + 5.0	double
d + s	double
2 * i / l	long

■ 캐스트 - 명시적 변환

```
(double) i
(long) ('A' + 1.0)
x = (float) ((int) y + 1)
(double) (x = 77)
(double) x = 77          /* equivalent to ((double) x) = 77, error */
(float) i + 3 <==> ((float) i) + 3
```