시스템 프로그래밍 기초

Introduction to System Programming

ICT융합학부 조용우

3. The Fundamental Data Types

선언(Declaration)

- 모든 변수는 사용전에 선언.
- 목적 : 메모리 공간 확보, 올바른 연산 수행
- 올바른 연산자의 선택

```
int a, b, c;
a = b + c; /* 정수 + 연산 */
float d, e, f;
d = e + f; /* 실수 + 연산 */
```

선언 예제

```
#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 */
    x = y + z;                      /* assignment */
    ...
```

수식(Expression)

- 상수, 변수, 연산자, 함수 호출 등의 의미있는 결합
- 상수, 변수, 함수 호출은 그 자체가 수식임

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

배정(Assignment)

- 문장: 수식 뒤에 세미콜론이 오면, 수식은 문장이 됨
  - `i = 7  /* 배정 수식 */`
  - `i = 7; /* 문장 */`
  - `3.777; /* 문법은 문제 없으나 용도가 없음 */`
  - `a + b; /* 문법은 문제 없으나 용도가 없음 */`
- 배정 수식 vs. 수학 등식
  - `x + 2 = 0  /* 틀림 */`
  - `x = x + 1; /* 배정 수식 */`

기본 자료형

Fundamental types grouped by functionality			
Integral types	**char**	**signed char**	**unsigned char**
	short (signed short int)	**int** (signed int)	**long** (signed long int)
	unsigned short (unsigned short int)	**unsigned** (unsigned int)	**unsigned long** (unsigned long int)
Floating types	**float**	**double**	**long double**
Arithmetic types	Integral types + Floating types		

declaration ::= type identifier { , identifier }$_{0+}$;

문자 및 char 자료형

Some character constants and their corresponding integer values					
Character constants	'a'	'b'	'c'	…	'z'
Corresponding values	97	98	99	…	112
Character constants	'A'	'B'	'C'	…	'Z'
Corresponding values	65	66	67	…	90
Character constants	'0'	'1'	'2'	…	'9'
Corresponding values	48	49	50	…	57
Character constants	'&'	'*'	'+'		
Corresponding values	38	42	43		

ASCII Table (1972)

b_4	b_3	b_2	b_1	Row ↓ / Column →	0 (000)	1 (001)	2 (010)	3 (011)	4 (100)	5 (101)	6 (110)	7 (111)
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

3.3 문자 및 char 자료형

Name of character	Written in C	Integer value
alert	\a	7
backslash	\\	92
backspace	\b	8
carriage return	\r	13
double quote	\"	34
formfeed	\f	12
horizontal tab	\t	9
newline	\n	10
null character	\0	0
single quote	\'	39
vertical tab	\v	11
question mark	\?	63

확장문자열

- 단일 문자코드로 사용 : `' '`, `'\n'`
- 문자열 안에서의 확장 문자열 : `"\tabcde\tXYZ\n"` (그냥 쓴다)
- (예)

```
→ printf("%c", '\a'); /* causes the bell to ring */
→ printf("\"abc\"");  /* "abc" is printed */
→ 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 */
}
```

int 자료형

- int 형
  - (32bit): -2^{31} ~ $2^{31}-1$ (-21억 ~ 21억)
  - (16bit): -2^{16} ~ $2^{16}-1$ (-3만2천 ~ 3만2천)

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

- a: 부정확한 값 (integer overflow)
  - 프로그램은 계속 실행되지만, 논리적으로 부정확한 값이 배정됨
  - 따라서 프로그래머는 변수별로 적절한 범위를 항상 고려해야함

정수적형(Integral Type): short, long, unsigned

- **short** 형
  - → (16bit): 2^{15} ~ 2^{15}-1 (-3만2천 ~ 3만2천)
- **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 ~ 6만5천)

부동형

- 부동형 : float, double, long double

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

- 기본적으로 접미사가 없는 부동형은 double형으로 간주
- float (4 bytes): 10^{38} ~ 10^{-38} ,유효숫자 6자리
- double (8 bytes): 10^{308} ~ 10^{-308} ,유효숫자 15자리

(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 fractional part(.77) is needed */`

typedef의 사용

- typedef : 식별자를 특정한 형과 연관
  - `typedef  char  uppercase;`
  - `typedef  int  INCHES, FEET;`
  - `typedef  unsigned long  size_t;  /* found in stddef.h */`
- 변수, 함수 선언시 사용
  - `uppercase  u;`
  - `INCHES  length, width;`
  - 긴 선언문을 축약해 쓸 수 있음
  - 목적에 따라 형 이름을 사용
  - 사용하는 컴퓨터에 따라 형의 메모리 할당 byte 수가 달라질 경우, 이식하기 쉽게 만듦

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)

getchar()와 putchar()의 사용

- <stdio.h>에 정의된 매크로
- getchar()
  - → 키보드에서 문자 읽는 매크로
- putchar()
  - → 화면에 문자 출력하는 매크로

(예) double_out.c (EOF는 ctrl-d)

```
#include <stdio.h>

int main(void)
{
    int   c;
    while ((c = getchar()) != EOF) {
        putchar(c);
        putchar(c);
    }
    return 0;
}
```

(예) capitalize.c

```
#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;
}
```

수학 함수

- C에는 내장 수학함수가 없다.
- 수학 Library에서 제공 **<math.h>**
- **sqrt() pow() exp() log() sin() cos() tan()**
- 절대값 함수
  - **abs()**
    - **int, <stdlib.h>**
  - **fabs()**
    - **double, <math.h>**

(예) power_square.c

```
#include <math.h>
#include <stdio.h>

int main(void)
{
    double  x;
    printf("/n%s",
        "The following will be computed:\n"
        "\n"
        "    The square root of x\n"
        "    x raised to the power x\n"
        "\n");
```

(예) power_square.c

```
    while (1) {
        printf("Input x: ");
        if (scanf("%lf", &x) != 1)
            break;
        if (x >= 0.0)
            printf("\n%15s%22.15e\n%15s%22.15e\n%15s%22.15e\n\n",
                "x =", x,
                "sqrt(x) = ", sqrt(x),
                "pow(x, x) = ", pow(x, x));
        else {
            printf("\nSorry, your number must be nonnegative.\n");
            break;
        }
    }
    printf("\nBye!\n\n");
    return 0;
}
```

실행 결과

```
The following will be computed:

    The square root of x
    x raised to the power x

Input x: 2

             x = 2.000000000000000e+000
      sqrt(x) = 1.414213562373095e+000
    pow(x, x) = 4.000000000000000e+000

Input x:
```

변환과 캐스트

- 정수적 승격 (the integral promotions)
  - signed 혹은 unsigne의 char, short, 또는 열거형을 int나 unsigned int형이 사용될 수 있는 수식에서 대신 사용 가능
  - 이 경우, 이들 형들 중 모든 값을 int로 표현할 수 있으면 int로 변환되고, 그렇지 않으면 unsigned int로 변환됨
  -
```
char c = 'A';
printf("%d\n", c); /* c는 승격이 일어나 int형이 됨 (65출력) */
```

변화과 캐스트

- 일반적 자동 변환 (the usual arithmetic conversions)
  - → 수식에서 제일 큰 형으로 변환한다.
  - → long double, double, float
  - → unsigned long, long, unsigned, int의 순서
  - → long과 unsigned의 경우, 모든 값을 long으로 표현할 수 있으면 long으로 변환되고, 그렇지 않으면 unsigned long으로 변환됨

일반적 자동 변화 예제

- `char c; short s; int i; long l; unsigned u;`
  `unsigned long ul; float f; double d; long double ld;`
- `c - s / i` (int)
- `u * 2.0 - i` (double)
- `c + 3` (int)
- `c + 5.0` (double)
- `d + s` (double)
- `2 * i / l` (long)

일반적 자동 변화 예제

- `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*)

캐스트(casts) - 명시적 변환

- 캐스트 예
  - → (double) i
  - → (long) ('A' + 1.0)
  - → f = (float) ((int) d + 1)
  - → d = (double) i / 3
  - → (double) (x = 77)
- 틀린 예
  - → (double) x = 77 /* equivalent to ((double) x) = 77, error */
- (float) i + 3 <==> ((float) i) + 3
  - → (float)가 + 보다 우선순위 높음

Homework

- Exercise #7, 8, 10, 17