2장. 어휘원소, 연산자와 C 시스템

2.1 문자와 어휘 원소

■ 프로그램에 사용되는 문자

lowercase letters : a b c .... z
uppercase letters : A B C .... Z
digits : 0 1 2 ... 9
other characters : + - * / = ( ) [ ] ....
white space : blank, new line, ...

■ 예제 프로그램

```
/* Read in two integer and print their sum */
#include      <stdio.h>

int     main(void)
{
int     a, b, sum;

printf("Input two integer:  ");
scanf("%d%d", &a, &b);
sum=a + b ;
prtintf(" %d + %d = %d\n", a, b, sum);
return 0;
}
```

▶ /* Read in two integer and print their sum */
주석문, 컴파일러는 공백으로 간주

▶ #include <stdio.h>
전처리기

▶ int main(void)
{
int a, b, sum;
main : 식별자, (): 연산자, {, , ,; : 분리자(punctuaor)
int : 키워드, a, b, sum : 식별자

inta, b, sum; -> (X), int absum -->absum을 하나의 식별자,

▶ printf("Input two integer: ");

scanf("%d%d", &a, &b);

printf, scanf : 식별자, (): 함수임을 알림

"Input two integer: " : " 문자열 상수 "

& : 주소연산자, & a, & b (O), &a,&b (O), &a &b (X), a&, &b (X)

▶ sum=a + b ;

=, + : 연산자, sum=a+b; (O), sum = a + b ; (O),

s u m = a + b ;(X)

● The compiler either ignores white space or uses it to separate elements of the language

2.2 구문 법칙

■ C의 구문 : Backus-Naur Form(BNF) 규칙 시스템

2.3 주석

■ 주석 : 한계자(delimiter) /* 과 */ 사이의 문자열

프로그램의 문서화(documentation)를 위한 도구

▶ 주석문 사용 예

```
/* a comment */, /**** another comment *****/, /***********/
/*
 * A comment  can be written in this fashion
 * to set it off from the surrounding code
 */
/*****************************************************************
 **** A comment  can be written in this fashion  ****
 **** to set it off from the surrounding code      ****
 *****************************************************************/
```

▶ C++ 에서의 간단한 주석

// This is a comment in C++

//

```
// This is a comment in C++

/* //This is a comment in C++ */
```

2.4 키워드 (Keyword)

■ 키워드 : C 언어에서 고유한 의미를 가지는 토큰(문법단위)으로 예약된 단어

■ C 언어에서 사용되는 키워드

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	

2.5 식별자 (Identifier)

■ 식별자 : 문자, 숫자, 밑줄문자(_)로 구성된 토큰

식별자의 첫 번째 문자 : 문자, 밑줄문자(_)

■ 식별자의 예

```
k
_id
iamanidentifier2
so_am_i
```

■ 식별자의 틀린 예

```
not#me      /* special character # not allowed */
101_south   /* must not start with a digit */
-plus       /* do not mistake - for _ */
```

■ C 표준 라이브러리에 정의된 식별자 : printf, scanf, ...

Ⓟ 의미가 쉽게 연상되는 식별자(변수) 사용할것.

Ⓟ 밑줄문자(__)로 시작되는 식별자는 가급적 사용하지 말것.

2.6 상 수 (Integer, Constant)

■ C의 상수

수치 상수, 문자 상수, 문자열 상수

■ 수치 상수

: 8진수, 10진수, 16진수, 지수, long 상수, 부호없는 정수

수치 상수	표 기 법	예
8진수	0을 맨 앞에 붙인다	011, 0345
10진수	상기 이외의 상수치	6800, 8089
16진수	0x 또는 0X를 맨 앞에 붙인다	0xab, 0x2BCD
지수	e 또는 E를 붙인다	5e2(=500), 6E3(=6000)
소수점	소수점을 사용한다	1.34, 25.89

2.7-1 문자 상수 (Character Constant)

■ 문자 상수 : 한쌍의 작은 따옴표 (‘ ’)에 묶인 문자

'A'

c='A'; /* 변수 c에 ‘A’ 문자에 해당하는 문자코드인 65가 대입 */

c=65;

2.7-2 문자열 상수 (String Constant)

■ 문자열 상수 : 한쌍의 큰 따옴표 (“ ”)에 묶인 문자열

ASCII 코드 채용 (PC,..)

■ 문자열 상수의 예

“a string of text"

"" /* the null string */

" " /* a string of blank */

" a = b + c ; " /* nothing is executed */

" /* this is not a comment */ "

" a string with double quotes \" within "

" a single backslash \\ is in this string "

"abc" "def" == "abcdef" (ANSI C)

■ 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)

2.8 연산자와 구두점

■ Operators can be used to separate identifiers (token)

■ 산술 연산자

연산자	설 명	사용 예
+	덧셈	a=b+c
-	뺄셈	a=b-c
*	곱셈	a=b*c
/	나눗셈	a=b/c
%	나머지	a=b%c

■ a+b /* a+b */

a_b /* 3-character indentifier */

■ printf("%d", a); /* 형식 제어 문자 */

a=b%7; /* 나머지 연산자 */

2.9 연산자의 우선순위와 결합법칙

■ 우선 순위 : (* /), (+-)

1+2*3 == 1+(2*3) --> 7

(1+2)*3 --> 7

■ 결합 법칙 : 좌에서 우

1+2-3+4-5 == (((1+2)-3)+4)-5

■ 연산자 우선 순위

(1) (), ++(postfix), --(postfix)	: L->R
(2) +(unary) -(unary) ++(prefix) --(prefix)	: R->L
(3) * / %	: L->R
(4) + -	: L->R
(4) = += -= *= /= etc	: R->L

2.10 증가 연산자와 감소 연산자

■ 증가/감소 연산자

연산자	설 명	사용 예
++	1 더하기	++a 또는 a++
--	1 빼기	--a 또는 a--

▶ ++과 --는 단항 연산자 (-, +)와 같은 우선 순위

■ 증가/감소 연산자 틀린 사용

```
777++          /* constant can not be incremented */
++(a*b-1)      /* ordinary expression not be incremented */
```

■ 예

```
int     a, b, c = 0 ;
a=++c;          /* c=c+1; a=c;  (prefix, 전위형(선) 증가 연산자) a=1,c=1,b=0*/
b=c++;          /*b=c; c=c+1;  (postfix, 후위형(후) 증가 연산자) a=1,c=2,b=1*/
printf("%d %d %d\n", a, b, ++c);     /* 1 1 3 is printed */
```

2.11 배정 연산자

■ 배정 연산자 : 변수의 값을 변경(배정)

= += -= *= /= %= >>= <<= &= ^= |=

우선 순위 : 최하위 결합성 : R->L

■ 예

b=2;

c=3;

a= b+c;

<===> a= (b=2) + (c=3);

a=b=c=0; k +=2 <==> k=k+2

■ 예

int i=1, j=2, k=3, m=4;

Expression	Equi. Expression	Equi. Expression	Value
i+=j+k	i+=(j+k)	i=(i+(j+k))	6
j*=k=m+5	j*=(k=(m+5))	j=(j*(k=(m+5)))	18