시스템 프로그래밍 기초

Introduction to System Programming

ICT융합학부 조용우

2. Lexical Elements, Operators, and the C System

- Characters and Lexical Elements

문자와 어휘 원소

- 구문
  - → 올바른 프로그램을 만들 수 있게 하는 규칙
- 컴파일러
  - → C 프로그램이 구문에 맞는지 검사
  - → 오류가 있다면, 오류 메시지 출력
  - → 오류가 없다면, 목적 코드 생성
- 컴파일 과정
  - → C 프로그램 → 토큰으로 분리 → 토큰을 목적 코드로 변환

ANSI C 토큰 종류

- 키워드(keywords)
- 식별자(identifiers)
- 상수(constants)
- 문자열 상수(string constants)
- 연산자(operators)
- 구두점(puctuators)

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

Dissection

```
/* 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
- "{", ",", ";": 구두점
- int: 키워드
- a, b, sum: 식별자
  - inta, b, sum; → (X) 공백문자가 필요
  - int absum; → absum을 하나의 식별자로 간주

Dissection

```
printf("Input two integer: ");
scanf("%d%d", &a, &b);
```

- printf, scanf: 식별자, ()가 따라와서 함수임을 알림
- "Input two integer: ": 문자열 상수
- &: 주소연산자
  - & a, & b (O), &a,&b (O)
  - &a &b (X), a&, &b (X) &는 오른쪽에 operand가 와야함

```
sum = a + b ;
```

- =, +: 연산자
  - sum=a+b; (O), sum = a + b ; (O)
  - s u m = a + b ;(X)

구문 법칙

- C의 구문 : Backus-Naur Form(BNF) 규칙 시스템
  - 1960년 ALGOL60을 위해 처음 사용
  - 고급언어를 기술하는 표준 형식

예) *digit* ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

  - 의미 : 구문 카테고리 digit는 기호 0 또는 1, ..., 또는 9로 다시 쓸 수 있다.

생산 규칙에 사용되는 기호들

- *italics* 구문 카테고리
- ::= "다시 쓰면"의 기호
- | 선택들을 분리
- { }$_{1}$ 괄호 안의 항목 중 하나만 선택
- { }$_{0+}$ 괄호 안의 항목을 영번 이상 반복
- { }$_{1+}$ 괄호 안의 항목을 한번 이상 반복
- { }$_{opt}$ 옵션인 항목

letter_or_digit

- *letter_or_digit* ::= *letter* | *digit*
- *letter* ::= *lowercase_letter* | *uppercase_letter*
- *lowercase_letter* ::= a | b | c | ... | z
- *uppercase_letter* ::= A | B | C | ... | Z
- digit ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

생성 규칙 예제

- *alphanumeric_string* ::= {*letter_or_digit*}$_{0+}$
- *u_alpha_string* ::= *uppercase_letter*{*letter_or_digit*}$_{0+}$
- *u_alpha_string* ::= *uppercase_letter alphanumeric_string*

- *conditional_statement* ::= if (*expression*) *statement*
  {else *statement*}$_{opt}$

주석

- 주석 : 한계자(delimiter) /* 과 */ 사이의 문자열
- 프로그램의 문서화(documentation)를 위한 도구
- 컴파일러는 주석을 하나의 공백 문자로 대치

주석문 사용 예

```
/* a comment */
/*** another comment ***/
/*****/


/*
 * A comment can be written in this fashion
 * to set it off from the surrounding code.
 */


/*****************************
*  If you wish, you can     *
*  put comments in a box.   *
*****************************/
```

C++ 에서의 간단한 주석

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

//
// This is one common way of writing
// a comment in C++ that consists
// of many lines.
//

/*
// This C comment style mimics the
// previous C++ comment style.
*/
```

키워드

- 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	

식별자

- 식별자 : 문자, 숫자, 밑줄문자(_)로 구성된 토큰, 대소문자 구별
- 식별자의 첫 번째 문자 : 문자, 밑줄문자(_)
- *identifier* ::= {*letter*|*underscore*}$_1${*letter*|*underscore*|*digit*}$_{0+}$
- *underscore* ::= _
- 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)를 사용
- 밑줄문자(_)로 시작되는 식별자는 시스템 이름과 충돌될 수 있으므로 가급적 사용하지 말 것.

상수

- 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

상수

- 정수 상수
  - → 0, 17, 234, 0x17
- 실수 상수
  - → 1.0, 3.141592, 23E2
- 문자 상수
  - → 'a', 'b', '+', '₩n'
- 문자열 상수
  - → "hello", "very good"
- 열거 상수
  - → enum에 의해 선언된 상수
- (주의) -49는 상수 수식임

문자 상수 (Character Constants)

- 문자 상수 : 한쌍의 작은 따옴표 (' ')에 묶인 문자
- 'A'
- c='A'; /* 변수 c에 'A' 문자에 해당하는 문자코드인 65가 대입 */
- c=65;

문자열 상수 (String Constants)

- 문자열 상수 : 한 쌍의 큰 따옴표 (" ")에 묶인 문자열
- ASCII 코드

문자열 상수의 예

```
"a string of text"
""                /* the null string */
"   "             /* a string of blanks */
" 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)


/* "this is not a string" */
"and
neither is this"
```

확장 문자열

확장 문자열	의미	ASCII 코드 (16진수)
\a	경보음	07
\n	개행	0A
\t	탭코드	09
\xhh	16진수	hh hh
\0	문자코드0	00

- 단일 문자코드로 사용 : '\t' '\n'
- 문자열 안에서의 확장 문자열 : "\tabcde\tXYZ\n"

산술 연산자

연산자	설명	사용예
+	덧셈	a = b + c
-	뺄셈	a = b - c
*	곱셈	a = b * c
/	나눗셈	a = b / c
%	나머지	a = b % c

산술 연산자

- 공백없는 이항 연산자

```
a+b  /* this is the expression a plus b */
a_b  /* this is a 3-character identifier */
```

- %의 용도 (변환명세, 나머지 연산자)

```
printf("%d", a); /* 형식 제어 문자 */
a = b % 7; /* 나머지 연산자 */
```

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 순위와 결합법칙

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

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 */

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);
```

- 다중배정 (R->L)

```
a = b = c = 0;
a = (b = (c = 0));
```

배정 연산자 예제

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

```
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
```

C 시스템

- C 시스템
  - → C 언어, 전처리기, 컴파일러, 라이브러리, 편집기 등으로 구성
- 전처리기
  - → #으로 시작하는 행을 전처리지시자라고 함
  - → #include <filename>
    #include "filename"
    #define PI 3.141592
- 표준 라이브러리
  - → 프로그램에 유용한 함수들로 C 시스템이 제공함
  - → printf(), scanf(), 등
  - → 사용자가 알아서 해당 헤더파일을 포함시켜야 함

Homework

- Exercises #11, 13, 17