A Book on C

2. 어휘 원소, 연산자, C 시스템

서울시립대학교 기계정보공학과
성민영

2020.08

/* 이 강의노트는 숭실대 김영호 교수님의 “A Book on C” 자료를 바탕으로 제작되었습니다 */

어휘 원소, 연산자, C 시스템

- 어휘 원소(**lexical elements**), 연산자(**operator**), **C** 시스템
- 구문 (**syntax**)
  - 올바른 프로그램을 만들 수 있게 하는 규칙
- 컴파일러 (**compiler**)
  - C 프로그램이 구문에 맞는지 검사
  - 오류가 있다면, 오류 메시지 출력
  - 오류가 없다면, 목적 코드(object code) 생성
- 컴파일 과정
  - C 프로그램 → 토큰(token)으로 분리 → 토큰을 목적 코드로 변환
  - 토큰 종류 :
    - 키워드 (keyword),
    - 식별자 (identifier),
    - 상수 (constant),
    - 문자열 상수 (string),
    - 연산자 (operator),
    - 구두점 (punctuation mark)

문자와 어휘 원소

- 프로그램에서 사용할 수 있는 문자
  - 소문자 : a b c ... z
  - 대문자 : A B C ... Z
  - 숫자 : 0 1 2 3 4 5 6 7 8 9
  - 특수문자 : + - * / = ( ) [ ] < > ' " !

    @ # $ % & _ | . , ; : ?
  - 여백문자 : *공백 (space)*, *개행 (new line)*, *탭 (tab)*

- 컴파일러는 이러한 문자들을 구문 단위인 토큰으로 모은다.

어휘 분석

- **sum.c 프로그램**

```
/* Read in two integers 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;
  printf("%d + %d = %d\n", a, b, sum);
  return 0;
}
```

키워드

식별자

연산자

구둣점

문자열 상수

상수

어휘 분석

- **/* Read in two integers and print their sum. */**
  - 주석문 (comments): /*부터 */까지는 공백으로 대치
- **#include <stdio.h>**
  - 전처리 지시자 (preprocessor directive): 전처리기가 처리
- **int main(void)**

```
{
    int   a, b, sum;
```

  - 키워드 : int, void
  - 식별자 : main, a, b, sum
  - 연산자 : ( )
  - 구두점 : “{”, “,”, “;”

어휘 분석

- **"Input two integers : "**
  - 문자열 상수 : 큰 따옴표로 둘러싸인 문자들

- **return 0**
  - 키워드 : return
  - 상수 : 0

구문 규칙

- **BNF (Backus-Naur Form)** 으로 기술

  예) *digit* ::= 0|1|2|3|4|5|6|7|8|9

  - 생산 규칙 (production rule)
  - 의미 : “구문 범주 *digit*는 기호 0 또는 1, ..., 또는 9로 다시 쓸 수 있다”

- 생산 규칙에 사용되는 기호들
  - *italics* 구문 범주 (syntactic category)
  - ::= "다시 쓰면"의 기호
  - | 선택들을 분리
  - $\{\}_1$ 괄호 안의 항목 중 하나만 선택
  - $\{\}_{0+}$ 괄호 안의 항목을 0번 이상 반복
  - $\{\}_{1+}$ 괄호 안의 항목을 1번 이상 반복
  - $\{\}_{opt}$ 옵션 항목들

생산 규칙 예제

- ***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
- *alphanumeric_string* ::= {*letter_or_digit*}$_{0+}$
  - “3”, “ab777c”, “ ” ← 알파뉴메릭 문자열
- *u_alpha_string* ::= *uppercase_letter* {*letter_or_digit*}$_{0+}$
- *conditional_statement* ::= if (*expression*) *statement*

  {else *statement*}$_{opt}$

주석

- **주석 (comments)**
  - /*과 */ 사이에 있는 임의의 문자열
  - 주석은 토큰이 아님
  - 컴파일러는 주석을 하나의 공백 문자로 대치
  - 문서화 (documentation) 도구로 사용함 (프로그램 설명, 정확성 증명 등)

- **C++ 주석**
  - 줄 단위 주석
    - // 다음부터 그 행 끝까지가 주석임
  - C 스타일의 주석도 사용

주석 예제

- **C** 스타일 주석

```
/* a 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++** 스타일 주석

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

QUIZ

Quiz Time

Let's have some fun!

9. 주석(comment)에 대한 설명 중 잘못된 것은?

① 주석은 프로그램에 대한 설명을 써주는 것이다.
② 주석 처리(comment out)된 코드도 실제로 수행된다.
③ 주석의 시작 부분에는 /*를, 주석의 끝 부분에는 */을 써준다.
④ /*와 */로 된 주석은 중첩해서 사용할 수 없다.

QUIZ

10. 다음은 간단한 C 프로그램이다. 프로그램의 각 줄에 대한 설명 중 잘못된 것은?

```
01: /* Test.c */
02: #include <stdio.h>
03:
04: int main( )
05: {
06:         printf("Good Bye!");
07:         return 0;
08: }
```

① 1번째 줄 : 주석도 생성될 실행프로그램에 포함된다.
② 2번째 줄 : 입출력 라이브러리를 사용하기 위해서 필요한 준비이다.
③ 4번째 줄 : main 함수에서부터 C 프로그램이 시작된다.
④ 6번째 줄 : printf 함수를 이용해서 출력할 수 있다.

Quiz Time

Let's have some fun!

키워드

- 키워드 **(keywords)**
  - C 언어에서 고유한 의미를 가지는 토큰
  - 예약된 단어 (reserved words)

- **C** 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(**identifies**)는 문자, 숫자, 그리고 특수문자인 밑줄문자(_)로 구성된 토큰으로, 문자 또는 밑줄문자로 시작해야 함
- **C** 시스템은 소문자와 대문자를 구별함
- 식별자의 선택은 의미를 생각하여 함

식별자

- 식별자 생성 규칙

  identifier ::= {*letter*|*underscore*}$_1${*letter*|*underscore*|*digit*}$_{0+}$

  underscore ::= _

- 올바른 예제

  K, _id, iamanidentifier2, so_am_I

- 잘못된 예제

  not#me, 101_south, -plus

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

문자열 상수

- 문자열 상수 - 큰따옴표에 의해 묶인 일련의 문자들

- 올바른 예제

```
"a string of text"
""
"    "
"   a = b + c; "
"  /* this is not a comment */  "
" a string with double quotes \" within"
```

- 잘못된 예제

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

연산자와 구두점

- 연산자

  +, - , *, /, %

- 구두점
  - 괄호, 중괄호, 콤마, 세미콜론 등

- 연산자와 구두점은 문맥에 따라 결정됨
  - %

    printf("**%**d", a);

    a = b **%** 7;
  - ( )

    printf**(**"hello"**)**;

    A = **(**23 + 2**)** * 2

우선순위와 결합법칙

- 우선순위(**precedence**)와 결합법칙은 평가 순서(**evaluation order**)를 결정함

- 예

  1 + 2 * 3 ←→ 1 + (2 * 3)

  1 + 2 – 3 ←→ ((1 + 2) – 3)

연산자	결합 법칙
( ) ++ (후위) --(후위)	좌에서 우로
+ (단항) - (단항) ++ (전위) -- (전위)	우에서 좌로
* / %	좌에서 우로
+ -	좌에서 우로
= += -= *= /= etc.	우에서 좌로

증가와 감소 연산자

- 전위 증감 연산자

  ++i, --i /* i = i + 1, i = i - 1 */

- 후위 증감 연산자

  i++, i-- /* i = i + 1, i = i - 1 */

- 증감 연산자 수식의 값

  ++i, --i /* i + 1, i - 1 */

  i++, i-- /* i, i */

연산자	연산자의 기능	결합방향
++num	값을 1 증가 후, 속한 문장의 나머지를 진행(선 증가, 후 연산) 예) val = ++num;	←
num++	속한 문장을 먼저 진행한 후, 값을 1 증가(선 연산, 후 증가) 예) val = num++;	←
--num	값을 1 감소 후, 속한 문장의 나머지를 진행(선 감소, 후 연산) 예) val = --num;	←
num--	속한 문장을 먼저 진행한 후, 값을 1 감소(선 연산, 후 감소) 예) val = num--;	←

증가와 감소 연산자

- 예제 코드

```
int   a, b, c = 0;
a = ++c;
b = c++;
printf("%d %d %d\n", a, b, ++c);
```

```
/* 1 1 3 is printed */
```

배정 연산자

- 다른 언어와는 달리 **C**는 **=**를 연산자로 다룸
  a = (b = 2) + (c = 3);

- 배정 **(assignment)** 연산자
  =, +=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=

(주의) j *= k + 3 은 **j = j * k + 3** 이 아니라, **j = j * (k + 3)** 임

선언 및 초기화			
int i = 1, j = 2, k = 3, m = 4;			
수식	동일한 수식	동일한 수식	결과 값
i += j + k j *= k = m + 5	i += (j + k) j *= (k = (m + 5))	i = (i+ (j + k)) j = (j * (k = (m +5)))	

예제

- **2**의 거듭제곱 계산

```
/* Some powers of 2 are printed. */
#include <stdio.h>
int main(void)
{
    int   i = 0, power = 1;
    while (++i <= 10)
      printf("%-6d", power *= 2);
    printf("\n");
    return 0;
}
```

- 출력

C 시스템

- **C 시스템**
  - C 언어, 전처리기, 컴파일러, 라이브러리, 편집기 등으로 구성

- **전처리기**
  - #으로 시작하는 행을 전처리 지시자라고 함

```
#include <filename>
#include "filename"
#define   PI   3.141592
```

- **표준 라이브러리**
  - 프로그램에 유용한 함수들로 C 시스템이 제공함
  - printf(), scanf(), 등
  - 사용자가 알아서 해당 헤더파일을 포함시켜야함

QUIZ

35. 다음 중 두 연산식의 의미가 다른 것은?

① w = w * 2 + 1; w *= 2 + 1;
② x = x + 2; x += 2;
③ y = y * 3; y * = 3;
④ z = z << 2; z <<= 2;

Quiz Time

Let's have some fun!