comp321 컴파일러

Tiny Language : Lexical Definitions

- tiny language 는 compiler course 에서의 구현을 목표로 간단하게 정의된, 실험용의 programming language 이다.
- 전체적으로는 C programming language 의 small set 으로 구성되므로, C 언어를 이해하면, 어떤 언어인지 쉽게 파악 할 수 있도록 설계 되었다.
- 단, data type 은 integer 와 float 로 제한한다. string 은 print 를 위한 상수로는 사용될 수 있지만, 변수로는 사용할 수 없다.
- 이 문서는 tiny language 의 token 들에 대해서 설명한다.

delimiter

- 어떠한 token 도 space (' '), tab ('\t'), new-line ('\n'), line-feed ('\r') 을 만나면, 해당 token 이 끝나고, 다음 token 이 시작되는 것으로 판정한다.
- 주의: Window 운영 체제에서는 text 문서의 줄 바꿈이 "\r\n" 의 2 개 문자로 표현된다. 즉, '\n'만 처리해서는 안 되고, '\r'도 처리해 주어야 한다.
- 단, string constant 만은 예외적으로 string constant 의 끝이 될 때까지는 space, tab, new-line 을 string constant 의 일부로 판정한다.

decimal constant (10 진수 상수)

- 숫자 0 은 decimal constant 이다.
- 숫자 1, 2, 3, 4, 5, 6, 7, 8, 9 로 시작하고, 그 뒤에 0 개 이상의 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 가 오면, decimal constant 이다.
- decimal constant 의 값은 대응되는 숫자들을 10 진수로 해석한 값이다.
- decimal constant 앞의 '+' 또는 '-' 기호는 별도의 연산자로 취급하고, decimal constant 의 일부로 보지는 않는다.

octal constant (8 진수 상수)

- 숫자 0 으로 시작하고, 뒤에 1 개 이상의 0, 1, 2, 3, 4, 5, 6, 7 이 오면, octal constant 이다.
- octal constant 의 값은 대응되는 숫자들을 8 진수로 해석한 값이다.
- 연달아 있는 숫자들 중에서, 숫자 8 또는 9 를 만나면, octal constant 의 끝으로 보고, 숫자 8 또는 9 이후는 다른 token 으로 해석한다.
- octal constant 앞의 '+' 또는 '-' 기호는 별도의 연산자로 취급하고, octal constant 의 일부로 보지는 않는다.

hexadecimal constant (16 진수 상수)

- 숫자 0 과 연달아 나오는 영어 소문자 'x' 또는 영어 대문자 'X'로 시작하고, 그 뒤에 1 개 이상의 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, a, b, c, d, e, f 가 오면, hexadecimal constant 이다.
- hexadecimal constant 의 값은 대응되는 문자들을 16 진수로 해석한 값이다.

- hexadecimal constant 앞의 '+' 또는 '-' 기호는 별도의 연산자로 취급하고, hexadecimal constant 의 일부로 보지는 않는다.

floating point constants (실수형 상수)

- 10 진수로 해석할 수 있는 숫자들의 뒤에 '.' (마침표)가 붙으면, floating point constant 로 판정한다.
- 10 진수로 해석할 수 있는 숫자들의 뒤에 '.' (마침표)가 붙고, 그 뒤에 10 진수로 해석할 수 있는 숫자들이 따라오면, floating point constant 로 판정한다.
- '.' (마침표) 뒤에 10 진수로 해석할 수 있는 숫자들이 따라오면, floating point constant 로 판정한다.
- 위의 3 가지 경우에 해당하고, 그 뒤에 'e' 또는 'E' 가 붙고, 그 뒤에 10 진수로 해석할 수 있는 숫자들이 따라오면, floating point 로 판정한다. 단, 'e' 또는 'E'와 그 뒤의 10 진수 사이에는 '+' 또는 '-' 가 들어가도 floating point constant 로 판정한다.
- floating point constant 의 값은 C 언어에서 정의한 값으로 해석한다.

string constants (문자열 상수)

- 모든 문자열은 " (double quote) 로 시작하고, " (double quote) 로 끝난다.
- 시작과 끝을 의미하는 double quote 들 사이에는 " (double quote)를 제외한 어떠한 문자든 올 수 있다.
- 문자열의 값, 즉 해당 문자열을 print 했을 때, 출력되는 문자들은 시작과 끝을 의미하는 double quote 들은 제외한, 그 사이의 문자들만을 의미한다.

identifier (변수명)

- 변수 이름으로 사용될 수 있는 identifier 들은 영어 대소문자로 시작하고, 그 뒤에 영어 대소문자 또는 0 에서 9 까지의 숫자가 연속해서 오는 형태이다.
- identifier 의 길이는 1 문자 이상, 255 문자 이하로 제한한다.
- identifier 의 길이는 255 문자까지 가능하지만, 내부적으로는 최초의 7 문자만 identifier 를 구별하는 데 사용하여도 좋다. 즉, sailing, sailingMan, sailingManNeverDies 는 컴파일러 내부에서는 모두 같은 identifier 로 취급하여도 좋다.
- 단, keyword 로 미리 지정된 영어 단어들은 비록 변수명으로 판정할 수 있더라도, 무조건 keyword 로 판정한다.

keyword (키워드)

- 다음 단어들은 keyword 로 미리 정의되고, 비록 identifier 로 판정할 수 있더라도, 항상 keyword 로 판정한다.
- int
- float
- void
- if
- else

- while
- return

comment (주석)

- shell-style comment : 문자 '#' 로 시작하고, '\n'으로 끝나는 경우, 그 사이에 어떠한 문자들이 오더라도, '#' 와 '\n'을 포함한 모든 문자들을 comment 로 취급하고 완전히 무시한다.
- C++-style comment : 문자열 "//" 로 시작하고,"\n'으로 끝나는 경우, 그 사이에 어떠한 문자들이 오더라도, "//" 와 '\n'을 포함한 모든 문자들을 comment 로 취급하고 완전히 무시한다.
- C-style comment : 문자열 "/*" 로 시작하고, 문자열 "*/" 로 끝나는 경우, 그 사이에 어떠한 문자들이 오더라도, "/*" 와 "*/" 을 포함한 모든 문자들을 comment 로 취급하고 완전히 무시한다.

operator (연산자)

- 위의 어떠한 정의에도 속하지 않는 특수 문자 1 글자는 operator 로 취급할 수 있다.
- 단, "||", "&&", "==", "!=", "<=", ">=" 는 2 글자를 묶어서, 1 개의 operator 로 취급한다.
- 컴파일러의 편의를 위해서, '@'와 같은 기호는 실제 operator 로 사용되지는 않지만, scanner 에서는 operator 로 판정할 수도 있다. (이 경우는 parser 에서 syntax error 로 판정하므로, 문제를 일으키지는 않는다.)
- 최종적으로 parser 에서 사용할 operator 들은 "=", "<", ">", "+", "-", "*", "/", "%", "!", ";", "[", "]", "(", ")", "{", "}", ",", "||", "&&", "==", "!=", "<=", ">=" 등이다.

general rule

- 이 문서에서 정의하지 않았거나, 애매하게 정의되었다고 느껴지는 사항은 C programming language 에서의 정의를 사용하여 상식적으로 판단한다.