60200307 우연정

1. Lexical Analysis
 1-1. Token - Delimiter

2. Parsing
 2-1. Syntax Tree

3. Code Generation
 3-1. Semantics

1. Lexical Analysis
 컴파일러는 우리가 문장을 만들면 그것을 이해해서 컴퓨터가 이해할 수 있는 것으로 바꾸는 역할을 한다. 이를 컴파일이라고 하며, 컴파일 과정에서 “Lexical Analysis(“어휘 분석”[네이버 지식백과])는 컴파일 또는 어셈블을 할 때 원시 프로그램의 문자를 입력하여 주사하고, 다음의 처리 루틴에 넘기기 위하여 문자열을 분해하여 해석하는 루틴을 말한다. 어휘 분석기(Scanner)가 원시 프로그램을 읽으면서 차례로 문자를 검토하여 문법적으로 의미 있는 일련의 문자로 분할하는 것을 의미한다.”[네이버 지식백과]

 1-1. Token - Delimiter
 “컴파일러는 소스 프로그램을 번역할 경우 먼저 워드 해석이라 불리는 기호열로서 입력된 소스 프로그램에서 구문 규칙에 기초하여 이름표, 연산자, 딜리미터(구분자) 등의 식별을 하고, 내부데이터의 열로 변환한다. 이 워드 해석을 하여 식별되는 단위를 실러블(syllable)이라 하며, 그 내부데이터 표현을 토큰이라고 한다.”[정보통신용어사전]

 컴파일 과정 중 어휘 분석 단계에서는 토큰을 뜯어낸다.
 토큰은 문법을 구성하는 기본적인 단위이며 글자와는 다르다. 문법은 주어, 형용사, 부사 등 역할이 있는 것을 말한다. 문자(Symbol)는 그것을 구성하는 또 다른 요소이며, 숫자처럼 경우의 수를 만들어낼 수 있는 체계이다. 문자는 무한대의 경우의 수를 만들어 낼 수 있다.
 심볼(“컴퓨터 프로그래밍에서 변수나 함수 등을 구분하기 위해 사람이 읽을 수 있는 문자열 형태로 정의한 식별자”[IT용어사전])은 우리의 생각으로 무엇인가를 만들어낸 것이다.
 문법은 “언어 구문(syntax)을 형식적(formal)으로 정의한 규칙”이다. [컴퓨터인터넷IT용어대사전]

 우리가 만든 계산기 프로그램은 입력받을 때 엔터와 탭, 스페이스를 무시하고 연산자와 피연산자만 받아들여 계산할 수 있어야 한다. 먼저 입력받을 때 토큰을 구분 문자(Delimiter)로 추출해야 한다. 연산자와 피연산자 그리고 엔터나 탭, 스페이스는 구분 문자로 각각 독립적으로 구분되어야 한다. 구분 문자로 토큰을 끊어낸다. 토큰을 구성하는 문법을 만드는 것이다. 우리가 만드는 토큰은 숫자의 집합이다. 우리가 숫자를 만들 때는 1-9까지 한 번 이상 반복. 뒷자리는 0에서 9까지 한 번 이상 반복해야 한다.
 구분 문자는 “텍스트 중의 문자열(string)을 목적에 따라서 특수문자로 구분하는 경우와 프로

그래밍 언어 중의 문법의 일부로서 구분을 표시하는 경우와 데이터 전송에 있어서 텍스트의 개시, 종료를 표시하는 것 등 명확히 용도를 정해둔 경우가 있다." [컴퓨터인터넷IT용어대사전]

"정규 표현(regular expression)은 특정 문자의 집합 또는 문자열을 기호로 바꾸어 놓는 방법으로, 문자열의 집합을 정확하게 표현하기 위해 쓰이는 표현 규칙이다. 언어의 문법 정의나 검색해야 할 문자열의 지정에 쓰인다."[IT용어사전]

2. Parsing

2-1. Syntax Tree

언어라는 것은 소통을 위한 것의 일부이다. 소통은 내 머릿속 생각을 상대방 머릿속에 집어넣는 것이다. 컴퓨터는 기계이므로 컴퓨터와 소통하기 위해서는 기계어를 넣어주면 되는 것이다.

파싱(parsing)은 "컴파일러 프로그래밍언어를 기계어로 번역하는 과정의 한 단계로, 각 문장의 문법적인 구성 또는 구문을 분석하는 과정이다. 즉 원시("컴파일러에 의해 기계어로 번역되기 전의 프로그램 형태"[IT용어사전]) 프로그램에서 나타난 토큰(token)의 열을 받아들여 이를 그 언어의 문법에 맞게 구문 분석 트리(parse tree)로 구성해 내는 일이다."[IT용어사전]

"구문 트리(syntax tree)는 프로그래밍 언어로 작성된 소스 코드의 추상 구문 구조의 트리이다. 이 트리의 각 노드는 소스 코드에서 발생되는 구조를 나타낸다."[위키백과]

3. Code Generation

컴파일 과정 중 코드 생성(Code Generation) 단계에서는 의미(Semantics)를 만들어낸다. 의미는 상대방이 알아들을 수 있는 무언가를 말한다.

3-1. Semantics

우리가 계산식을 입력하면 컴파일러가 우리가 짠 프로그램의 문법적 구조를 끊어낸다. 토큰을 끊어내서 프로그래밍언어를 기계어로 바꿔준다.

코드 생성(Code Generation)은 "컴파일의 단계 중 맨 마지막으로 어휘 분석, 문장 분석, 의미 분석, 최적화 등의 단계를 거쳐 생성된 중간 부호로부터 목적하는 기계의 기계어 부호나 어셈블리 부호를 생성해 내는 것을 말한다."[IT용어사전]

"Semantics란 Syntax에 따른 문자를 나열(문)한 것으로 어떤 의미를 가진 것인가 결정하는 규칙이며, 프로그램이 컴퓨터 상에서 어떤 처리를 해야 하는가를 정하는 것이다. 즉 문장을 문자, 기호의 배열로 간주하지 않고, 그것에 포함되어 있는 내용과 의미를 문제로 하는 것이다. Syntax는 프로그램을 기호의 열로 간주하며, 바른 프로그램은 어떠한 기호열이어야 하는가를 규정하는 데 대해, Semantics는 프로그램을 어떠한 기계어로 번역해야 하는가를 결정하는 규칙으로 기호와 그 의미를 규정하고 있다."[컴퓨터인터넷IT용어대사전]