프로그래밍 언어 1차 과제

20181256 수학, 컴퓨터공학과 김도현

7. LISP began as a pure functional language but gradually acquired more and more imperative features. Why?

1958년 MIT의 McCarthy 가 개발한 LISP 는 초기에 purely functional language 로 개발 되어 ‘pure LISP’ 라고 불렸다. 이 당시 데이터 형은 atom, list 로 나뉘는데 atom 은 immutable 한 특성을 가지며symbol 이나 numeric literal 을 의미한다. list 는 atom 이나 다른 list를 포함하고 유한하다. 그리고 Imperative language 처럼 많은 변수나 할당문도 사용하지 않았다. 1970년대 이후 LISP로부터 파생된 많은 언어들이 나오게 되었다. 이때 이 언어들로 작성된 프로그램들은 서로 잘 맞지 않는 경우가 생겼다. 그래서 이 모든 언어의 특징을 결합한 Common LISP도 나왔다. 이 언어는 pure LISP 기반으로 이루어졌으나 imperative 한 특징처럼 변수를 special로 선언 시 동적 할당도 가능하고 배열, 스트링 같은 많은 데이터 타입, 구조체 등도 가지고 있다. 또한 최근 대부분의 프로그래밍이 imperative 한 특징을 많이 가지고 있어 LISP 를 이용해 더 효율적인 코드 구현을 위해 imperative 특징들을 더 많이 제공해준다.

12. Describe, in your own words, the concept of orthogonality in programming language design.

프로그래밍 언어에서 orthogonality 라는 의미는 상대적으로 작은 primitive construct 들이 상대적으로 작게 조합하여 기능들을 이루어 내는 것이다. 즉 orthogonality 가 높을 수록 간단하면서 최대한 중복되지 않은 기능들을 이용해 조합해야 하고 예외 역시 거의 나타나서는 안된다. 단순한 기능들을 이용해 여러 구조를 만든다는 의미이기 때문에 복잡한 디자인도 간단하게 만들 수 있고 다른 기능들에 영향도 없다. 이를 통해 orthogonality가 높은 언어일 수록 프로그래머들이 언어를 배우거나 읽고 쓰는게 쉬워진다. (c 언어에서 대표적으로 return 과 array에서 return이 array 타입으로 return 못 시키기 때문에 서로 예외가 발생하여 orthogonal 하지 않다.)

14. What are the arguments both for and against the idea of a typeless language?

typeless language 의 장점은 모든 데이터들이 모든 변수에 저장하기 때문에 구문 자체가 간결하고 유연성도 높다.(BCPL 같은 언어에서는 모든 데이터들을 machine word 로 여겼다.) 그리고 유연성이 높아 프로그래밍 측면에서 더 자유롭기 때문에 객체 들을 더 강하게 제어할 수 있어 프로그램의 설계에도 유용하다.

하지만 단점은 type checking이 없다면 프로그램 실행 없이는 중간에 어떤 데이터가 어떤 유형인지, 제대로 이루어졌는지 확인 할 수 없다. 그리고 사람이 코드만 보고서는 어떤 변수인지 찾기에는 typed language보다는 상대적으로 복잡하고 어려울 수 있다. typed language 는 typeless language 에 비해 덜 유연하지만 데이터 구조들이 다른 객체를 받기 전에 미리 올바른 type 인지 체크해서 어떤 변수인지 알 수 있어 안전한 프로그래밍이 가능하다.

누구와(코드를 읽고 쓰는 사람이 얼마나 많은지) 어떤 프로그램을 코드로 구현 할 것인지에 따라 typeless language 가 좋을 수도 typed language 가 좋을 수도 있다.