**ㄴ**

**프로그래밍언어(가)**

**과제2-1**

**이름 : 허예림**

**학번 : 20182667**

**교수님 : 유재우 교수님**

**교재의 2장을 잘 읽고 이해하고 다음의 질문에 간단히 답하기 바랍니다. 59쪽의 간단한 프로그래밍언어의 역사적 연대표, 12장, 15장, 16장등을 참고하기 바랍니다.**

**(1) Algol 계통 (절차중심적) 언어의 특성과 주요 언어의 출현 배경**

절차 지향 프로그래밍은 top-down방식으로 설계된다. 비교적 작은 규모의 작업을 수행하는 함수를 생성하고 인수의 반환값으로 명령을 전달하고 수행한다. 객체 지향 프로그래밍과의 가장 큰 차이점으로는 데이터와 함수를 별개로 취급한다. 특정 지능을 수행하려면 그 일을 해주는 메소드를 직접 호출해야 한다. Algol은 절차적 프로그래밍 언어의 표준적인 예시이다. Algol은 1950년대 미국에서 만들어진 Fortran에 대항하여 유럽을 중심으로 개발된 프로그래밍 언어이다. 알고리즘의 연구개발에 이용하기 위한 목적으로 만들어졌다. Fortran은 1954년 IBM 704에서 과학적인 계산을 하기 위해 시작된 컴퓨터 프로그램 언어이다. Fortran은 Algol과 함께 과학 계산용으로 주로 사용되는 언어이며, 당시 7명의 전문가가 약 2년 반 동안에 걸쳐 완성한 것이 Fortran의 기본을 이루었고, 그 후 ANSI와 ISO에서 표준화하였다.

**(2) 논리 프로그래밍언어의 특성과 주요 언어의 출현 배경**

논리 프로그래밍의 접근 방법은 데이베이스로서 사실과 사실 사이 관계를 나타내는 규칙의 집합을 사용하고 새로운 명제의 타당성을 검사하기 위해, 데이터베이스의 사실과 규칙이 사실이라고 가정하여, 자동 추론 과정을 사용한다. 이런 접근 방법은 자동 정리 증명을 위해 개발된 접근 방법이다. Prolog는 가장 널리 사용되는 논리 프로그래밍 언어이다. 논리 프로그래밍의 기원은 논리 추론을 위한 해 도출 규칙을 개발한 Robinson에게 있다. Prolog는 Edinburgh의 Kowalski로부터 약간의 도움을 받아 Marseille의 Roussel과 Colmeraur에 의해 처음으로 개발되었다. 논리 프로그램은 해답의 특성은 주어지지만 해답을 얻는 과정은 주어지지 않는 것을 의미하며 비절차적이어야 한다. Prolog 문장은 fact, 규칙 또는 목적이다. 대부분의 문장은 기본 명제인 구조와, 산술 표현식이 허용될지라도, 논리 연산자로 구성된다. 해 도출은 Prolog 인터프리터의 주요한 동작이다. 역행을 광범위하게 사용하는 이 과정은 주로 명제들의 패턴 부합을 포함한다. 변수가 포함되었을 때, 변수는 부합을 제공하기 위해 값으로 사례화된다. 이런 사례화 과정을 단일화라고 부른다. 현 상태의 논리형 프로그래밍은 많은 문제점을 갖고 있다. 효율성의 이유와 무한 루프를 회피하기 위해, 프로그래머는 때때로 프로그램에서 제어 흐름 정보를 언급해야 한다. 또는 닫힌 세계 가정 문제와 부정의 문제가 있다. 논리형 프로그래밍은 많은 다른 영역, 주로 관계형 데이터베이스 시스템, 전문가 시스템, 그리고 자연어 처리에 사용되었다.

**(3) 함수 프로그래밍언어의 특성과 주요 언어의 출현 배경**

함수형 프로그래밍은 함수들의 집합/조합으로써 프로그램을 구성한다. 재귀가 기본이고, 괄호로 프로그램 내포구조를 나타낸다. 실행시 함수 동작의 생성이 가능하고 함수의 파라미터 또는 반환에 의해 함수를 전달할 수 있다. 또한, 임시 변수에 의한 반복문 처리보다는 재귀호출에 의한 구현을 선호한다. Scheme은 LISP의 파생어로 1970년대 중반에 MIT에서 발표되었다. Scheme의 특징은 작은 크기, 정적-영역 규칙의 독점적 사용, 일등급 개체로서 함수의 취급이다. LISP의 초기 버전은 이런 모든 기능을 제공하지 않는다. Common LISP는 Scheme을 포함하여 1980년대 초반의 여러 LISP 파생어의 특징을 하나의 단일 언어로 결합하기 위한 노력으로 탄생했다. 언어들의 결합이기 때문에, Scheme은 매우 크고 복잡한 언어이며, 이러한 관점에서 C++와 C#과 유사하다. 그러나 기반이 원본 LISP이므로 구문, 원시 함수, 기본 함수는 LISP에서 가져온 것이다. ML은 LISP보다는 명령형 언어에 더 유사한 구문을 사용하는 정적 영역이고 강 타입 함수형 프로그래밍 언어이다. ML은 타입 추론 시스템, 예외 처리, 다양한 데이터 구조와 추상 데이터 타입을 포함한다. Haskell은 1987년 오리건주 포틀랜드에서 열린 ‘함수형 프로그래밍 언어와 컴퓨터 구조에 관한 총회’에서 있었던 회의에서, 함수형 언어들을 통합 정리해서 훗날 언어 설계의 기반이 될 수 있는 일반적인 순수 함수형 프로그래밍 언어를 만들자는 데에서 시작되었다. 1997년 말까지 다양한 설계안으로 이어져 마침내 Haskell 98이 나왔다. 이는 교육용이나 확장을 덧붙이는 토대로 쓸 수 있는 안정적이고, 작고, 이식성 좋은 언어 표준을 의도한 결과물로서 표준 라이브러리가 포함되어 있다. F#은 객체지향 프로그래밍을 포함하여 함수형과 명령형 프로그래밍을 지원하는 .NET 프로그래밍 언어이다. F#의 핵심은 ML과 Haskell의 후손 언어인 OCaml에 기반하고 있다.

**(4) 객체지향 프로그래밍언어의 특성과 주요 언어의 출현 배경**

객체 지향 프로그래밍은 세 가지 기본적인 개념으로 추상 데이터 타입, 상속, 동적 바인딩에 기초하고 있다. 객체 지향 프로그래밍 언어는 클래스, 메소드, 객체, 그리고 메시지 전달을 갖는 패러다임을 지원한다. 객체 지향 프로그래밍언어의 특성은 추성화, 캡슐화, 상속성, 다형성이 있다. 첫 번째 추상화는 객체들의 공통적인 특징(속성 과 기능)을 뽑아내는 것이다.즉, 우리가 구현하는 객체들이 가진 공통적인 데이터와 기능을 도출해 내는 것을 의미한다. 추상화는 절차지향 프로그래밍에서도 있었다. 대표적인 것이 구조체와 같은 사용자 데이터형이다. 이것은 데이터를 추상화해서 하나의 새로운 데이터 유형을 만드는 것이다. 예를들어 struct와 같은 구조체가 있다. 하나의 새로운 데이터형을 정의하는 데 그 안에는 여러개의 부속데이터를 둘 수 있다. 이 때 그 데이터를 추상화를 통해 정의한 것이다. 두 번째 캡슐화는 데이터 구조와 데이터를 다루는 방법을 결합시켜 묶는 것을 말한다. 특정 객체가 독립적으로 역할을 제대로 수행하기 위해 필요한 데이터와 기능을 하나로 묶어 관리한다. 객체가 맡은 역할을 수행하기 위한 하나의 목적을 위해 데이터와 기능들을 묶는 것이다. 또한, 이러한 특징은 다른 의미로 사용되기도 한다. 데이터를 은닉하고 그 데이터를 접근하는 기능을 노출시키지 않는다는 의미로 사용할 때 캡슐화라는 용어를 사용한다. 즉, 데이터를 기능이라는 캡슐로 보호한다는 것이다. 세 번째 상속성이다. 상속이라는 개념이 없다면 객체지향과 절차지향은 차이점이 없다. 상속이란 특징을 가지면서 객체지향 언어가 절차지향 언어의 비해 문법의 규모가 2배이상 늘어났다. 상속이란 상위개념의 특징을 하위 개념이 물려받는 것을 말한다. 하나의 클래스가 가지고 있는 특징(데이터와 함수)들을 그대로 다른 클래스가 물려주고자 할 때 상속성의 특징을 사용한다. 네 번째는 다형성이다. 다형성은 약간 다른 방법으로 일을 하는 함수를 동일한 이름으로 호출해 주는 것을 말한다. 객체 지향 언어는 15가지 정도가 있다. 최초의 객체 지향 언어로 알려져있으며, 오슬로의 노르웨이 계산 센터의 Kristen Nygaard과 Ole-Johan Dah가 1962년부터 1967년에 걸쳐, SIMULA의 원형이 되는 SIMULA I와 Simula67을 [ALGOL60|ALGOL 60] 의 확장판으로써 설계했다. SIMULA는 당시 시뮬레이션에 사용되었는데, 후에 범용 언어가 되었다. 특히 SmallTalk는 SIMULA의 객체지향개념을 일반화시킨 것이라고 할 수 있다. C++ 역시, 처음엔 C언어로 SIMULA의 클래스 등의 기능을 추가한 형태였다. Objective-C 언어는 C 프로그래밍 언어에 SmallTalk 스타일의 메시지 구문을 추가한 객체 지향 언어이다. 현재, 이 언어는 애플의 매킨토시의 운영 체제인 맥 오에스 텐과 아이폰의 운영 체제인 iOS에서 사용되고 있다. C++는 AT&T 벨 연구소의 비야네 스트롭스트룹이 1983년 발표하여 발전한 프로그래밍 언어이다. C 언어의 문법을 대부분 사용할 수 있으며, 객체지향성이 더해진 C 언어의 확장형이라고 생각할 수도 있다. 원래 이름은 C with Classes였으나, 증가 연산자를 도입하면서 1984년에 C++로 이름이 변경되었다. 자바는 썬 마이크로시스템즈의 James Gosling과 다른 연구원들이 개발한 객체 지향적 프로그래밍 언어이며, 썬 마이크로시스템즈에서 무료로 제공하고 있다. 1991년 Green Project라는 이름으로 시작해 1995년에 발표했다. 처음에는 가전제품 내에 탑재해 동작하는 프로그램을 위해 개발했지만 현재 웹 애플리케이션 개발에 가장 많이 사용하는 언어 가운데 하나이고, 모바일 기기용 소프트웨어 개발에도 널리 사용하고 있다. 현재 버전 8까지 출시했다. Python은 1991년 프로그래머인 Guido van Rossum이 발표한 고급 프로그래밍 언어로, 플랫폼 독립적이며 인터프리터식, 객체지향적, 동적 타이핑 대화형 언어이다. Ruby는 마츠모토 유키히로가 개발한 동적 객체 지향 스크립트 프로그래밍 언어이다. 루비는 순수 객체 지향 언어이다. 루비에서의 정수나 문자열 등을 포함한 데이터 형식 등 모든 것이 객체이다. 기능에는 클래스 정의, 가비지 컬렉션, 강력한 정규 표현식 처리, 다중 스레드, 예외 처리, 반복, 클로저, Mixin, 연산자 오버로드 등이 있다. 구문은 ALGOL계를 계승하면서 가독성을 중시하고 있다. 오랫동안 언어 사양이 명문화되지 않고 마츠모토의 구현 언어 사양에 준하는 것으로 간주되어 왔지만, 2010년 6월 RubySpec으로 언어 사양을 책정 중이다.

**(5) 스크립트 언어의 특성과 주요 언어의 출현 배경**

스크립트 언어란 소스 코드를 컴파일하지 않고도 실행할 수 있는 프로그래밍 언어를 말한다. 스크립트 언어는 소스를 작성한 후 컴파일 과정 없이 바로 실행하여 결과를 확인할 수 있다는 장점이 있는 반면에 번역 과정을 거쳐야 하기 때문에 다소 느리다는 단점이 있다. 스크립트 언어는 주로 시스템, 웹 어플리케이션에 사용된다. 첫 번째 sh는 파일관리와 단순 파일 필터링과 같은 유틸리티 함수들을 수행하는 시스템 부프로그램들에 대한 호출로서 해석되는 명령어들의 작은 모임으로써 시작되었다. 이들 중 가장 강력하고 널리 알려진 것 중의 하나는 ksh이다. Awk는 벨 연구소의 Al Aho, Brian Kernighan, Pegar Weinberger에 의해 개발되었다. Awk는 보고서 생성 언어로 서 시작되었으나 나중에 더 범용 언어가 되었다. Perl은 Larry Wall에 의해 개발되었고 처음에 sh와 awk의 결합으로 생겨났다. JavaScript는 처음으로 Netscape사의 Brendan Eich에 의해 개발되었다. 새로운 많은 특징과 기능들을 추가하면서 버전 1.0으로부터 1.5로 이동하면서 많은 발전 과정을 거쳤다. Python은 네덜란드의 Stiching Mathematisch Centrum에서 Guido van Rossum에 의해 이루어졌다. Python은 Perl과 같은 유형의 응용인 시스템 관리, CGI 프로그래밍 , 그리고 다른 상대적으로 규모가 작은 컴퓨팅 작업에 대해서 사용되고 있다. 개방 소스 시스템이며, 대부분의 공통적인 컴퓨팅 플랫폼에 대해서 유용 가능하다. Ruby는 1990년대 초반에 Yukihiro Matsumoto에 의해 설계되었고 1996년에 발표되었다. Ruby가 만들어진 동기는 Perl과 Python을 사용한 그 설계자의 불만족이였다. Ruby를 특성화하는 주요 특징은 그 언어가 순수 객체 지향 언어라는 것이다. 또한 클래스와 객체들은 모두 동적이다. Lua는 1990년대 초반에 브라질의 Rio de Jeneiro의 Pontifical 대학교의 Roberto Ierusalimschy, Waldemar Celes, Luis Henrique de Figueiredo에 의해 설계되었다. 이 언어는 확장성을 갖는 절차적이고 함수형 언어를 지원하는 스크립트 언어이다. 그 설계에 영향을 준 언어들 가운데는 Scheme, Icon, Python이 있다.