PUSAN NATIONAL UNIVERSITY

REPORT

프로그래밍언어론 과제1

과 목 명 :	프로그래밍언어론
지도교수 :	
학 과 :	전기컴퓨터공학부 정보컴퓨터공학전공
학 번 :	
이 름 :	장 수현
제 출 일 :	2020년 4월 2일

부산대학교

1. 프로그래밍언어의 특징, 사용 목적, 장점, 단점 등을 정리하시오.

1.1 기계어

1.1.1 특징

컴퓨터가 직접 이해할 수 있는 언어로 0과 1의 2진수 형태로 표현된다.

1.1.2 사용 목적

컴퓨터 시대 초창기에는 고급 언어가 없었기 때문에 명령어의 입력을 하는 유일한 방법이었다. 그래서 프로그램을 짜는 데에 기계어가 사용되었다. 그러나 시간이 흐르면서 프로그램 작성에 적합한 고급 언어들이 나왔고, 지금은 고급 언어로부터 변환되어 컴퓨터가 처리할 수 있게 하는 데에 사용된다.

1.1.3 장점

실행 속도가 빠르다.

1.1.4 단점

읽고 이해하기에 매우 어렵기 때문에 프로그램을 작성하는 데에 많은 시간이 소요된다. 또한 이식성이 낮아 다른 기계어 체계를 사용하는 컴퓨터 간 호환이 되지 않는다.

1.1.5 언어의 분류

어셈블리어와 함께 저급 언어로 분류된다.

1.2 어셈블리어

1.2.1 특징

인간의 관점에서는 사용이 불편한 언어이기 때문에 이를 보완하기 위해 나온 언어로, 기계어에 1대 1로 대응된다.

1.2.2 사용 목적

초창기엔 기계어보다 가독성이 좋았기 때문에 프로그램 작성에 사용되었지만 고급 언어들이 등장함에 따라 상대적으로 프로그램 작성에 어려움이 있어 사용률이 감소했다. 이후 최근에 들어 C언어보다 프로그램 최적화를 할 수 있는 유일한 언어라는 장점 때문에 사물인터넷에 쓰이는 초소형 기기들을 프로그래밍 하는데 많이 쓰이고 있다.

1.2.3 장점

실행 속도가 빠르다. 기계어에 비해 가독성이 좋다. 프로그램 최적화에 적합하다.

1.2.4 단점

기계어에 비하면 가독성이 좋지만, 사람에겐 여전히 읽고 이해하기에 매우 어렵기 때문에 프로그램을 작성하는 데에 많은 시간이 소요된다. 또한 기계어와 마찬가지로 CPU가 채택한 ISA에 따라 명령어 규격이 달라 낮은 이식성을 가진다.

1.2.5 언어의 분류

중간 언어라고 불리기도 하지만 기계어와 함께 저급 언어로 분류된다.

1.3 C언어

1.3.1 특징

사람 중심의 고급 언어이지만, 포인터라는 강력한 무기로 메모리까지 직접 제어가 가능해 저급 언어의 특징도 가지고 있다.

1.2.2 사용 목적

Unix를 만들기 위해 등장했던 언어인 만큼 OS를 만들기에 적합한 언어이다. 사실상 컴퓨터의 전 분야에서 사용할 수 있는 효율적인 언어이다. 특히 C언어의 장점으로 인해 임베디드 혹은 시스템 프로그래밍에서 주로 사용되고 있다. 한 때는 모바일 계열에서도 많이 사용되었지만 모바일의 하드웨어 환경이 나아짐에 따라 생산성이 더욱 중요시 되어 모바일에서는 사용률이 감소하고 있다.

1.3.3 장점

C언어로 작성한 프로그램은 속도도 빠르고 실행파일의 크기가 작다. 또한 이식성이 뛰어나다.

1.3.4 단점

수정 사항을 확인하려면 컴파일이 필요하면, 디버깅도 어려워 생산성이 비교적 낮다.

1.3.5 언어의 분류

고급 언어이고, 보다 상세한 기준으로 분류를 한다면 다음과 같다.

- 컴파일러 방식 언어 : 소스 코드를 미리 기계어로 번역해 실행파일을 만들어 실행한다.
- 비관리 언어 : 동적 할당된 메모리의 해제가 수동으로 이뤄져야 한다.
- 정적 타입 언어 : 묵시적 타입 변환이 일어날 때도 있지만 자료형이 고정되어 있다.
- 절차적 언어 : 알고리즘과 로직에 의거해 문제를 해결하도록 작성한다.

1.4 Python

1.4.1 특징

매우 간단하게 표현할 수 있어 프로그램 작성 시간이 빠른 언어로, 교육용으로도 많이 쓰이고 현장에서 실용적으로도 많이 쓰여 최근 언어 사용률 상위권에 머무르고 있다.

1.4.2 사용 목적

실행의 효율성보다는 프로그램 작성의 효율성이 더 좋다. 그래서 단기간에 개발하기 위한 용도로 많이 쓰인다. 쉽기 때문에 교육용으로도 많이 쓰인다.

1.4.3 장점

문법이 비교적 쉬워 초보자들에게 많이 추천된다. 또한 실사용률과 생산성도 높다. 플랫폼에 독립적인 언어이므로 이식성도 좋다.

1.4.4 단점

수정 사항을 확인하려면 컴파일이 필요하면, 디버깅도 어려워 생산성이 비교적 낮다.

1.4.5 언어의 분류

고급 언어이고, 보다 상세한 기준으로 분류를 한다면 다음과 같다.

- 인터프리터 방식 언어 : 소스 코드를 블록 단위로 해석하여 처리한다.
- 관리 언어 : 동적 할당된 메모리의 해제가 가비지 컬렉터에 의해 자동으로 이루어진다.
- 동적 타입 언어 : 실행 시간에 타입이 결정되고, 실행 전까지는 타입을 알 수 없다.

1.5 MATLAB

1.5.1 특징

공학용 시뮬레이션용 언어로 애플리케이션의 성격도 가지고 있다.

1.5.2 사용 목적

공학용으로 개발되어 공학계열에서 주로 사용된다. 혹은 간단하다는 특성에 주목하여 학생들이 프로그램의 개념을 익히기 위한 교육용 언어로도 사용된다.

1.5.3 장점

문법이 쉽고 공학적인 용도로 사용하기에 시간이 단축되고 유용하다. 그리고 인터프리터 방식을 채택하고 있어 디버깅이 쉽다. 또한 다른 언어로 변환하여 실행속도를 높이거나 임베디드 기계에 직접 이식할 수 있다.

1.5.4 단점

유료이고, 많은 명령어 세트를 가지고 있는 인터프리터 언어이기 때문에 느리고 무겁다. 코드 파일 하나에 함수를 하나씩만 만들 수 있기 때문에 번거로운 면도 있다.

1.5.5 언어의 분류

고급 언어이고, 보다 상세한 기준으로 분류를 한다면 다음과 같다.

- 인터프리터 방식 언어 : 소스 코드를 블록 단위로 해석하여 처리한다.
- 객체지향 언어 : 객체지향을 지원하긴 하지만 매우 기초적인 부분만 지원한다.

1.6 PHP

1.6.1 특징

웹 개발에 특화된 언어로 웹 서버에서 작동하는 스크립트 언어이다.

1.6.2 사용 목적

웹 페이지 제작에, 특히 백 엔드에서 서버의 작동을 구현하기 위해 많이 사용된다.

1.6.3 장점

C언어, Java 등의 언어와 문법이 유사하고 직관적으로 코딩할 수 있다. 그리고 Linux, Windows등 다양한 OS 환경에서도 잘 작동된다.

1.6.4 단점

복잡한 웹을 구축하는 데에는 부적합하다. 또한 접근이 쉬워 보안상의 문제도 있다.

1.6.5 언어의 분류

고급 언어이고, 보다 상세한 기준으로 분류를 한다면 다음과 같다.

- JIT 컴파일 방식 언어 : 실제 실행하는 시점에 기계어로 번역하여 실행파일을 생성한다.
- 특수 목적 언어 : 웹 분야에서 뛰어난 생산성을 보인다.

2. Orthogonality가 C언어에서 발생시킬 수 있는 오류의 예들을 찾아보시오.

2.1 언어의 직교성(Orthogonality)

모든 가능한 조합을 사용하여 프로그래밍 언어의 요소를 결합할 수 있도록 허용하는 것을 의미한다. 즉, 상대적으로 적은 수의 기본 키워드들의 조합을 이용하여 나타낼 수 있는 특성이다.

2.2 C언어의 직교성(Orthogonality)

C언어는 직교성이 뛰어난 언어이다. 기본 타입인 int, float, double, char 등과 포인터 연산자 *, 배열 연산자 [], 구조체, 함수를 조합하여 수많은 데이터 타입에 대해 표현이 가능하다.

예를 들어 int a[3][4]; 라고 선언된 문장을 보면, 변수 a의 타입은 [{(int 타입)을 요소로 가지고 길이가 4인 배열}을 요소로 가지고 길이가 3인 배열] 이다.

또 다른 예로, double *(*p)[10]; 의 선언문에서 변수 p의 타입은 [{("double 타입"을 가리키는 포인터)를 요소로 가지고 길이가 10인 배열}을 가리키는 포인터] 이다.

2.3 Orthogonality가 C언어에서 발생시킬 수 있는 오류의 예

C언어는 직교성이 뛰어나지만 잘못 사용하여 오류가 발생할 수 있으니 주의해야한다. 아래는 C언어에 Orthogonality가 잘못 적용된 예시들이다.

2.3.1 함수의 반환 타입으로 구조체, 함수는 올 수 없다.

- struct X(double);

굳이 해석하면, double타입을 인자로, 구조체를 반환타입으로 가짐

- int(short) Y(float);

굳이 해석하면, float타입을 인자로, (short를 인자로, int를 반환하는 함수)를 반환함.

2.3.2 배열의 요소로 함수는 올 수 없다.

- void(int,int) Z[5];

굳이 해석하면, int 2개를 인자로, void를 반환하는 함수를 요소로 가지고 길이가 5인 배열

2.3.3 구조체는 배열 연산자, 포인터 연산자 이외에는 조합이 불가능하다.

- int struct A;

완전히 틀린 문법이다. 굳이 해석할 수도 없다.

- struct B Struct C;

완전히 틀린 문법이다. 굳이 해석할 수도 없다.

- char* struct D;

완전히 틀린 문법이다. 굳이 해석할 수도 없다.