프로그래밍언어론

2018학년도 2학기 출석수업대체시험

핵심체크 및 출제예상문제

범위 : 교재 1 ~ 3장

제1장 프로그래밍 언어의 소개

1. 프로그래밍 언어란?

(1) 프로그래밍 언어

① 프로그래머의 의사를 전달하는 방법이며, 동시에 프로그램을 작성하는 형식

② 컴퓨터가 읽을 수 있고 사람이 읽을 수 있는 형식으로 컴퓨터 계산(행동)을 서술하기 위한 표기 체계

③ 폰 노이만: 컴퓨터는 전선 연결방법으로 특정 작업이 수행되도록 지시되어서는 안 되며 중앙처리장치가 수행해야 할 작업을 일련의 명령코드로 작성해서 컴퓨터 내부의 자료로 저장해야 한다고 주장

(2) 프로그래밍 언어론의 필요성

① 현재 사용하고 있는 언어를 더욱 잘 이해함

② 유용한 프로그래밍 구사능력 증대

③ 프로그래밍 언어를 선택할 수 있는 능력 증대

④ 새로운 프로그래밍 언어를 배우기 쉬어짐

⑤ 새로운 프로그래밍 언어를 설계하기 쉬어짐

(3) 추상화

① 속성들의 특징적인 일부분만을 가지고 주어진 작업이나 객체들을 표현하고 그들의 공통점을 추출하여 표현

② 대상에 따라 분류: 자료추상화, 제어추상화

③ 정보양에 따라 분류: 기본추상화, 구조적 추상화, 단위추상화

(4) 계산 전형

① 명령형 언어(절차언어): 변수 값을 변경시키기 위한 명령문을 순서대로 나열한 것

② 함수 언어(적용형 언어)

- 알려진 값들을 함수들에 적용
- 매개변수에 함수를 적용함

③ 객체 지향 언어: 실세계에 존재하는 모든 유형 및 무형의 대상이 되는 객체를 클래스로 표현

④ 논리형 언어(선언적 언어): 반복이나 선택문 없이 계산의 내용만을 선언하듯 기술(기호논리학)

(5) 언어의 정의

① 구문론

- 언어 구성요소의 외부적인 형태에 관한 것
- 대부분 문맥 무관형 분법으로 정의되고 있다.
- C언어의 if문을 문맥 무관형 문법으로 정의하면 다음과 같음

  <if문>:: = if (<조건>) <문>

② 의미론: 언어의 표면적 구조만을 나타낸 것, 프로그램이 무엇을 어떻게 수행할지 나타내 줌

2. 프로그래밍 언어의 역사

(1) 1950년경: 기억장치를 고려한 최초의 프로그래밍 언어

① FORTRAN: 어셈블리 언어로 프로그래밍을 하였고 하드웨어가 고가였기 때문에 언어 설계에 있어서 무엇보

다도 효율성이 강조됨(수치 계산에 적합하도록 개발된 언어)
③ Cobol: 사무처리와 같은 사무처리의 목적으로 만들었으며 레코드 구조를 도입하고 자료구조 부분과 실행 부분을 분리
④ Algol 60: 연구용 및 실질적은 응용목적의 언어(후속 언어에 많은 영향을 미침)
⑤ LISP: 일반적인 리스트 구조와 함수 응용을 기본으로 함 (기호계산을 위한 함수 언어)
⑥ APL: 다양한 연산자로 배열조작을 간단히 한 함수(제어구조가 없음)

(2) 1960년대: 다양한 프로그래밍 언어 출현
① Algol 68: Algol 60에 중요한 언어 개념을 추구하고 신택스도 공식적으로 정의한 최초의 언어
② PL/I: FORTRAN, COBOL, Algol의 정점만을 언어에 넣음
③ SNOBOL: 최초의 문자열 처리 및 패턴 매칭 기능 언어
④ BASIC: 사용자와의 상호작용을 강조하여 편리성을 도모한 인터프리터 방식(간단한 언어 교육용, 사무처리 및 가정용으로 널리 보급)
⑤ Simula 67: 시뮬레이션 문제에 적합하게 설계된 언어로서 클래스라는 개념을 최초로 도입

(3) 1970년대; 간결성 추상화, 효율성
① Pascal: Algol의 후속언어, 작고 간결하고 효율적이며 교육적인 목적의 구조적 프로그래밍 언어(범용 언어로 개선)
② C: 컴퓨터 구조에 대한 접근을 제공해 주는 개념을 갖는 중급 프로그래밍 언어(UNIX 운영체제 구현 사용)

(4) 1980년대: 통합과 새로운 방향
① Ada: 기존의 명령형 언어의 개념을 종합한 것으로 볼 수 있으며 추상 자료형 구조, 병렬 프로그래밍 구조, 예외 처리 등의 흥미 있는 기능들이 많이 포함되어 있음
② Smalltalk: 본격적인 객체 지향 언어의 효시
③ 객체지향 언어로서 C++, Eiffel이 발표
④ CLOS는 LISP에 객체 지향 개념을 추가한 것
⑤ Modula-2: 추상화, 부분적 동시 처리 개념과 내장형 시스템 프로그래밍을 목적으로 함(모듈화 프로그램 설계 개념을 도입하여 Pascal의 후속 언어로서 발표)
⑥ Scheme: LISP를 교육용에 맞게 변형시킨 것(인공지능에 대한 관심으로 많이 이용)
⑦ Prolog: 논리 언어로 1972년에 제안되었으며 수학적 논리학을 적용한 프로그래밍 언어

(5) 1990년대: Web 프로그래밍
① 응용문제의 해결과 프로그래밍 간편성에 집중한 프로그래밍 언어들의 개발 집중
② Java: 간결하면서도 어느 기계로든 이동되어 실행될 수 있다는 점 때문에 네트워크 언어로서 각광을 받고 있음

3. 프로그래밍 언어의 설계기준

효율성(목적코드의 효율적, 번역의 효율성, 프로그래밍 효율성), 정확성(행위를 예측할 수 있는 정의에 의존). 표현력(언어가 복잡한 과정이나 구조 표현하는데 용이), 일반성, 획일성, 직교성, 컴퓨터 독립성, 안전성, 기존 표기나 규칙과의 일관성, 확장성, 부분성

<1장 출제예상문제>

1. 프로그래밍 언어론의 필요성으로 잘못된 것은?
① 유용한 프로그래밍 구사 능력이 증대된다.
② 현재 사용하고 있는 언어를 더욱 잘 이해한다.
③ 새로운 프로그래밍 언어를 배우기 쉬어진다.
④ 한 가지의 프로그래밍 언어의 능력이 증대 된다.

[해설] 프로그래밍 언어론의 필요성
- 현재 사용하고 있는 언어를 더욱 잘 이해함
- 유용한 프로그래밍 구사능력 증대
- 프로그래밍 언어를 선택할 수 있는 능력 증대
- 새로운 프로그래밍 언어를 배우기 쉬어짐
- 새로운 프로그래밍 언어를 설계하기 쉬어짐

[정답] 1

2. 프로그래밍 언어에서의 추상화를 정보의 양에 따라 분류한 것에 속하지 않는 것은?
① 구조적 추상화 ② 기본적 추상화
③ 자료 추상화 ④ 단위적 추상화

[해설] 정보의 양에 따라 분류
- 기본적 추상화: 가장 국지적인 기계 정보에 관한 추상화
- 구조적 추상화: 프로그램의 구조에 대한 보다 전역적인 정보에 관한 추상화
- 단위 추상화: 단위 프로그램 전체 정보에 관한 추상화

③ 대상에 따라 분류한 것으로 자료추상화(문자열, 변수, 탐색트리), 제어추상화(반복문, 조건문, 프로시저 호출)가 있음

[정답] 3

3. 반복이나 선택문 없이 계산의 내용만을 선언하듯 기술하는 언어는?
① 명령어 언어 ② 논리형 언어
③ 함수형 언어 ④ 객체지향 언어

[해설] 논리형 언어(선언적 언어)
- 반복이나 선택문 없이 계산의 내용만을 선언하듯 기술
- 기호 논리학에 근거를 두고 있음

[정답] 2

4. Smalltalk의 특징이 될 수 없는 것은?
① 캡슐화
② 계승
③ 함수언어
④ 객체 지향 언어

[해설] Smalltalk
객체지향 접근 방식으로 일관된 객체지향 언어의 순수한 모범 케이스

[정답] 3

5. 좋은 언어가 갖추어야 할 요건에 속하는 것은?
① 한 기종에서만 실행시킬 수 있도록 설계되어야 한다.
② 언어의 개념이 될 수 있으면 복잡해야 한다.
③ 구문이 명백해서는 안 된다.
④ 효율이 좋아야 한다.

[해설] 프로그래밍 언어의 설계 기준
효율성, 표현력, 정확성, 일반성, 직교성, 획일성

[정답] 4

6. 다음 중 논리 언어인 것은?
① Prolog
② Java
③ LISP
④ Smalltalk

[해설] Prolog
논리 언어로 1972년에 제안되었으며 수학적 논리학을 적용한 프로그래밍 언어로서 인공지능분야에 많이 쓰임

② Java: 중간 코드 형태로 제공되어 자바가상기계(JVM)위에서 수행됨으로써 이식성이 좋음
③ LISP: 일반적인 리스트 구조와 함수의 응용을 기본으로 하여 인공지능 분야에 폭넓게 사용

[정답] 1

7. 다음 중 컴파일러 방식과 인터프리터 방식의 혼합 형태를 취한 언어인 것은?
① C++
② Pascal
③ Java

④ Scheme

[해설]
① C++: C언어를 확장하여 객체지향 프로그래밍 개념 도입
② Pascal: 작고 간결하고 효율적이며 교육적인 목적의 구조적 프로그래밍 언어로 분리 컴파일 기능, 문자열 조작, 입출력 기능의 효율화 등의 고기능성을 갖춤
④ Scheme: LISP의 한 버전이며, 인공지능에 대한 관심으로 많이 이용

[정답] 3

8. 다음 중 프로그래밍 언어에 대한 설명으로 틀린 것은 ?
① 컴퓨터가 읽을 수 있고 사람이 읽을 수 있는 형식으로 계산을 서술하는 표기체계이다.
② 프로그램을 작성하는 형식이다.
③ 컴퓨터에 프로그래머의 의사를 전달하는 방법이다.
④ 컴퓨터의 구조를 서술하며 이를 통해 컴퓨터 간의 의사소통이 이루어진다.

[해설]
프로그래밍 언어란 프로그래머의 의사를 전달하는 방법이며, 동시에 프로그램을 작성하는 형식이고, 컴퓨터가 읽을 수 있고 사람이 읽을 수 있는 형식으로 컴퓨터 계산(행동)을 서술하기 위한 표기 체계이다. 컴퓨터의 구조를 서술하지는 않는다.

[정답] 4

9. 다음 중 실세계에 존재하는 모든 유형 및 무형의 대상을 객체라는 개념으로 정의하고 이에 특정값을 저장하는 기억장소와 그 기억장소의 값을 변경할 수 있는 연산으로 구성하는 방법을 사용하는 프로그래밍 언어를 무엇이라 하는가 ?
① 객체지향 언어
② 적용형 언어
③ 함수형 언어
④ 명령형 언어

[해설]
객체 지향 언어는 실세계에 존재하는 모든 유형 및 무형의 대상이 되는 객체를 클래스로 표현한다.

[정답] 1

10. Algol의 한계성을 극복한 하나의 사례로서 유닉스 운영체제의 구현에 사용되면서 많은 가능성을 보여주었던 프로그래밍 언어는 무엇인가 ?
① Fortran
② JAVA
③ Ada

④ C

[해설]
C는 컴퓨터 구조에 대한 접근을 제공해 주는 개념을 갖는 중급 프로그래밍 언어로서 UNIX 운영체제 구현에 사용 되었다.

[정답] 4

제2장 프로그래밍 언어의 구조 및 해석

1. 언어구문

(1) 프로그래밍 언어의 어휘구조

① 프로그래밍 언어 알파벳 문자로 구성된 단어 = 어휘토큰

② 구문구조와 별개지이만 밀접한 관련 있음

③ 번역기

- 어휘분석: 입력프로그램의 일련 문자를 토큰으로 구분
- 구문분석: 구문구조 결정

④ 예약어: 미리 정의된 식별자(for, case 등)

- 장점: 오류 검색 시간 단축, 프로그램의 가독성 향상, 컴파일러의 탐색 시간 단축
- 단점: 프로그래밍 언어 확장시 예약어와 식별자의 충돌, 예악어 관리 어려움

(2) 문맥자유 문법과 BNF

1) 구문형식과 문맥자유 문법

① 문맥자유 문법: 모든 생성 규칙에서 정의될 대상이 하나의 비단말기호만으로 구성된 문법

② 문맥의존 문법: 특수한 문맥에 의존하여 대치되는 문법

2) BNF

① 생성규칙들의 집합

- 생성규칙을 적용하여 정의된 언어로 작성가능한 모든 정상적인 프로그램 산출
- 작성된 프로그램이 구문에 맞는 프로그램인지 아닌지를 결정하기 위한 구문 규율로 사용
- 왼쪽: 정의될 대상
- 오른쪽: 그 대상에 대한 정의

② 예: BNF 표기법에 의한 식별자 정의

<identifier> ::=<letter> | <identifier><letter> | <identifier><digit>

<letter> ::=A | B | C ... | X | Y | Z

<digit> ::= 0 | 1 | 2 ... |8 | 9

- 비단말기호: 각진 괄호 '<>'로 묶인 기호, 다시 정의될 대상
- 단말기호: 각진 괄호로 묶이지 않은 기호, 알파벳 문자 집합, 예약어
- 메타기호: '::=', '|', '<', '>'

3) EBNF

① 보다 읽기 쉽고 간결하게 표현할 수 있는 확장된 BNF

② 특수 의미의 메타 기호를 더 사용하여 반복되는 부분이나 선택적인 부분을 간결하게 표현함

③ 표현: <compound-statement> ::=begin <statement> {; <statement>} end

BNF표현: <compound-statement> ::=begin <statement-list> end

<statement-list> ::=<statement> | <statement-list> ; <statement>

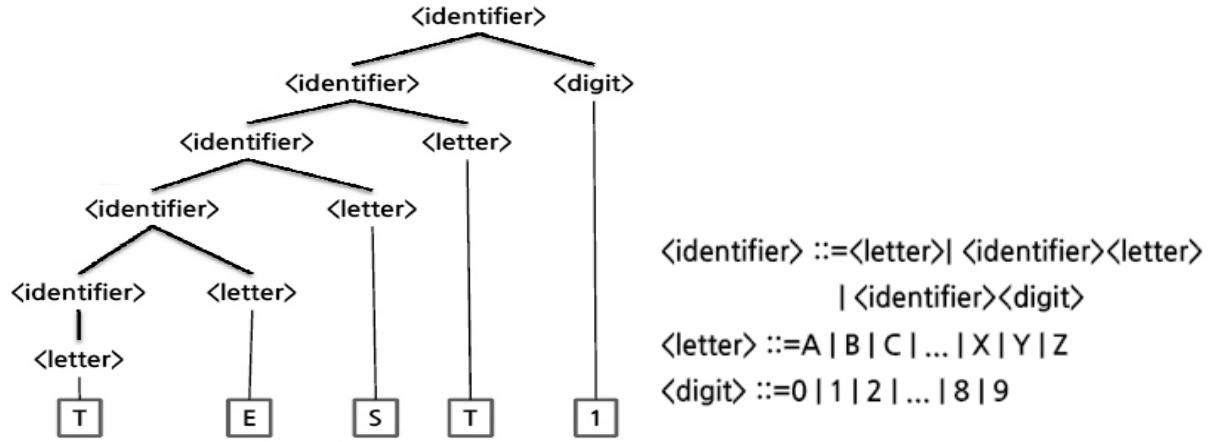
(3) 구문도표

① EBNF 방법 외에 구문에 대한 형식 정의하는 또 다른 방법
② 순서도와 비슷하며 EBNF 선언과 바로 대응시킬 수 있음
③ 비단말 기호: 사각형, 단말기호: 원 또는 타원, 정의관계: 지시선

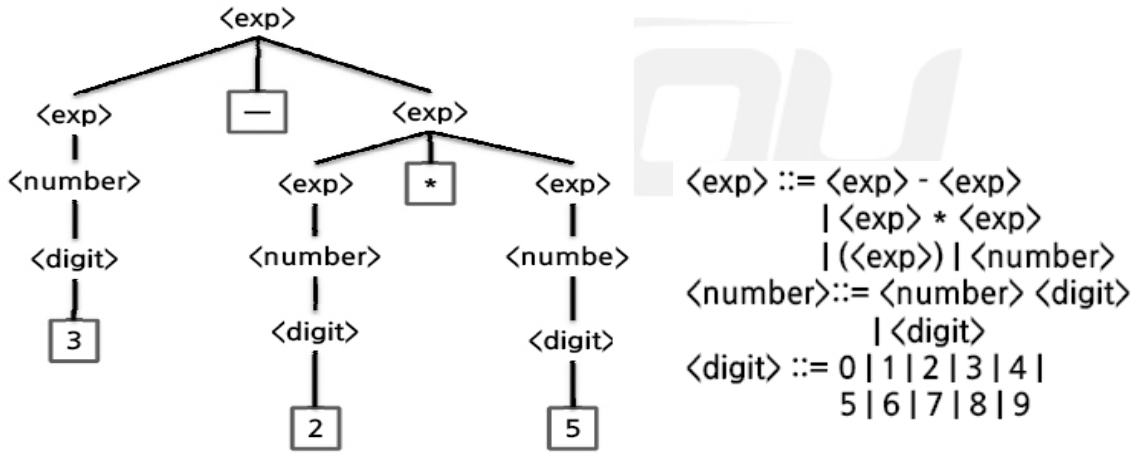
단말 X(::=x): 원 또는 타원 안에 표기하고 나가는 지시선을 그림 비단말 X(::=<x>): 사각형 안에 표기하고 지시선 그음

(3) 파스 트리와 프로그램 문법의 모호성
1) 파스트리
① 주어진 BNF에 의해 어떤 표현이 생성될 수 있는지 확인하기 위해 작성하는 트리구조 단말노드 나열

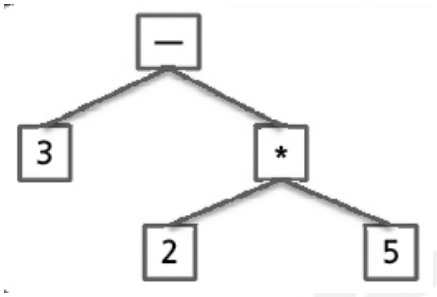
식별자 test 1에 대한 하스 트리



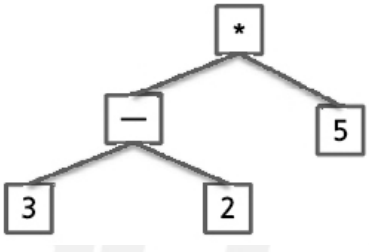
3-2*5에 대한 파스 트리



② 추상구문트리(구문트리): 파스트리의 본질적인 구조를 타내는 트리(불필요한 비단말 기호제거, 동일한 파스 트리, 서로 다른 파스 트리 등)
③ 추상구문 트리 비교



-> 결과값: -7



결과 값: 5

=> 위 경우와 같이 서로 다른 파스트리가 만들어지는 문법을 모호하다고 함(모호성)

④ 모호성 제거법칙
- 비단말 기호와 문법규칙 추가
- BNF 문법에 좌순환 규칙을 사용하여 좌결합 지원

2) 프로그램 문법의 모호성

① 프로그램 문법의 신뢰성: 프로그래밍 언어의 구문이 사람과 기계에 의해 쉽게 분석될 수 있어야 함
㉠ A=B=C; : B에 C의 값을 할당하고, A에는 B의 값을 할당하는 다중 할정
- A, B= C; : PL/1에서의 다중 할정문은 A, B = C;로 표현됨
- A=(B=C) : 사용한 의도의 다중 할당이 아닌, B와 C가 같은지 틀린지에 따라 A에 참 또는 거짓을 할당되는 것으로 해석

㉡ 변수의 선언을 명확하지 않은 언어의 경우 철자 오류가 묵시적 선언으로 간주되고 처리되어 올바른 프로그램을 어렵게 만듦

② 프로그램 문법의 모호성
㉠ IF절에서 else 문 처리
- else 문제 -> 현수 else(dangling else)
- BNF 문법과 관계없이 언어 자체가 포함하고 있는 모호성에 해당
- 현수 else 모호성: 두 가지 의미를 가질 수 있는 문제 발생, 이러한 else 문제를 현수 else이라 함. BNF 문법과 관계없이 언어 자체가 포함하고 있는 모허성에 해당

2. 프로그래밍 언어 구현기법

(1) 컴파일 기법

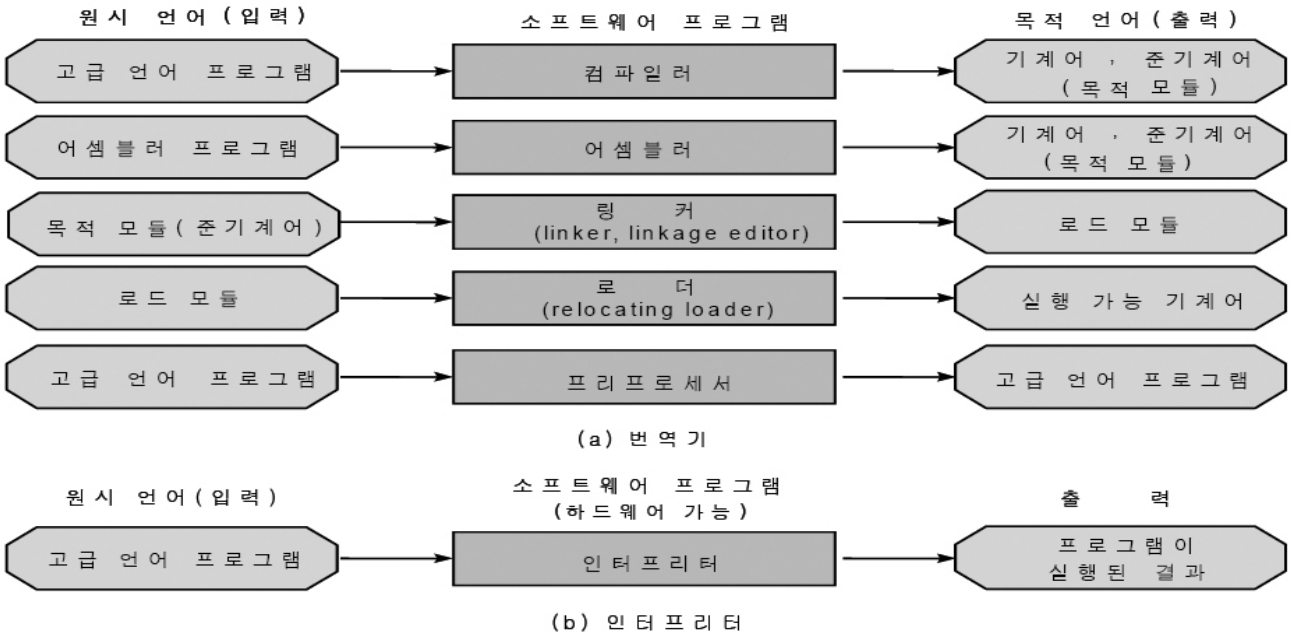
① 컴파일러: 고급언어를 저급언어인 목적언어로 만들어주는 번역기
② 어셈블러: 원시언어가 어셈블리 언어인 번역기
③ 링커: 재배치 형태의 기계어로 구성된 여러개의 프로그램을 묶어서 어느 정도 실행 가능한 하나의 기계어로 번역함
④ 로더: 로드 모듈의 기계어 프로그램을 실행 가능한 기계어로 번역하여 주기억 장치에 적재함
⑤ 프리프로세서: 원시언어와 목적언어가 모두 고급

(2) 인터프리트 기법

① 고급언어를 기계어로 취급하여 이를 실행할 수 있는 고급언어 기계를 소프트웨어로 시뮬레이션하여 구성하는 방법
② 인터프리터로 프로그램을 실행시키는 과정

입력자료
↓
고급언어로 작성된 원시프로그램 → 인터프리터 ← 실행결과

③ 번역기종류와 인터프리터



(3) 컴파일과 인터프리트 기법 비교

컴파일 기법	인터프리터 기법
입력프로그램과 동일한 목적 언어로 된 목적 코드 출력만 가능	직접 고수준의 프로그램을 실행 가능
각 문장을 입력된 순서대로 한 번씩 처리	논리적 순서에 따라 문장들 실행
입력(원시)언어가 어셈블리 언어처럼 기계어에 가까운 언어일 경우 주로 사용	운영체제의 작업 제어 언어나 APL과 같은 대화용 언어에 주로 사용
고급언어 번역 -> 목적 모듈 출력 -> 링크, 로드 -> 실행	고급언어 -> 중간코드까지만 번역 -> 실행
번역된 코드 -> 다시 번역할 필요 x	실행시마다 실행 시뮬레이션을 통해 실행
컴퓨터의 실행시간을 중시하는 경우 사용	사용자의 유연성을 중시하는 경우 사용
추가 기억장소 필요 없으며 유연성 높은 언어의 구현에 있어 편리함	많은 횟수로 반복처리되는 프로그램을 실행할 때 매우 효율적
번역된 프로그램이 매우 큰 기억장치를 요구할 수 있음	실생시간을 매우 많이 요구할 경우 발생
Fortran, Algol, PL/I, Pascal, Cobol, C 등	Lisp, Snobol4, APL, Prolog

<2장 출제예상문제>

1. 인터프리터 기법에 대한 설명이 아닌 것은?
① 직접 고수준의 프로그램을 실행 가능
② 논리적 순서에 따라 문장들 실행
③ 컴퓨터의 실행시간을 중시하는 경우 사용
④ 운영체제의 작업 제어 언어나 APL과 같은 대화용 언어에 주로 사용

[해설]
③ 컴파일 기법에 대한 설명. 인터프리터 기법은 사용자의 유연성을 중시하는 경우 사용됨

[정답] 3

2. 예약어와 관련 없는 것은?
① 프로그램의 가독성 향상
② 예약어 관리의 어려움
③ 컴파일러 탐색 시간 단축
④ 오류 검색 시간 증가

[해설] 예약어의 장단점
- 장점: 프로그램 가독성 향상, 컴파일러의 탐색 시간 단축, 오류 검색 시간 단축
- 단점: 예약어 관리의 어려움, 프로그래밍 언어 확장시의 예약어와 식별자의 충돌

[정답] 4

3. 다음 중 좌순환적인 BNF 규칙은?
① <식> ::= <식> + <식>
<식> ::= <식> * <식>
② <식> ::= <식> + <식>|<항>
<식> ::= <항> * <항>|<수>
③ <식> ::= <식> * <식>
<식> ::= <식> + <식>
④ <식> ::= <식> * <식>|<항>
<식> ::= <항> + <항>|<수>

[정답] 2

4. 다음 중 비단말기호는?
① {}
② []
③ ()
④ <>

[해설] 비단말기호
각진 괄호 '<>'로 묶인 기호, 다시 정의될 대상

[정답] 4

5. 모호한 문법의 문제점은 무엇인가?
① 분석나무를 만들 수가 없다.
② 문맥 무관형 문법이 아니다.
③ 간결한 코드를 만들기가 어렵다.
④ 여러 분석나무 중에서 실제로 쓰이는 분석나무에 따라서 다른 의미의 코드가 만들어질 수 있다.

[정답] 4

6. 다음 중 형태가 순서도와 비슷하고 EBNF 선언과 대응시킬 수 있으며 사각형, 타원 등의 도형과 화살표로 구성된 것은 무엇인가 ?
① 단말도표
② 구문도표
③ 문맥자유 도표
④ 어휘분석 도표

[해설]
구문도표는 EBNF 방법 외에 구문에 대한 형식 정의하는 또 다른 방법이며, 순서도와 비슷하며 EBNF 선언과 바로 대응시킬 수 있다.

[정답] 2

제3장 변수, 바인딩, 식 및 제어문

1. 변수

선언문 또는 묵시적 선언으로 생성되며 식별자, 자료속성들의 집합, 하나 이상의 주소 그리고 자료 값들의 네 가지 요소로 구성되는데, 주소와 자료 값들의 관계는 변할 수 있음

2. 바인딩

(1) 개념

① 변수의 네 가지 요소에 값을 확정하는 것
② 프로그램 작성시 변수 속성은 변할 수 있으나 번역시간에 결정된 변수의 속성은 변할 수 없음
③ 일반적으로 변수 속성은 선언문을 통해 이루어짐
④ 변수의 주소란 변수값이 저장되는 기억장소 위치
⑤ 이름에 어떤 속성을 연결하는 과정

(2) 바인딩 시간

① 속성이 이름에 연결되고 계산되는 과정이 어느 시점에서 이루어지는 가에 따라 바인딩이 분류되는 시간
② 종류: 실행시간, 번역시간, 언어의 구현시간, 언어의 정의시간
③ 중요성: 효율성, 유연성

3. 선언

(1) 정의

실행 시 사용될 자료의 속성을 컴파일러 등에게 알려주는 프로그램 문장(바인딩을 제공하는 중요한 방법)

(2) 선언문의 목적

① 주기억장치 사용과 접근 방법의 효율성
② 주기억 장치 경영의 효율성
③ 정적 형 검사 가능

4. 할당문

(1) 정의

변수의 내용을 변경할 수 있는 연산

(2) 종류

단순 할당문, 다중목적 할당문, 조건 목적변수 할당문, 복합할당 연산자, 단항 할당 연산자, 식으로서의 할당문, 혼합형 할당형

(3) 식으로서의 할당문

① 식으로 혹은 다른 식에 포함된 피연산자로 사용가능
② 할당 연산자를 다른 이항 연산자와 유사하게 취급
③ 할당 연산자가 왼쪽 피연산자를 변경시키는 식 부작용을 가질 수 있음

④ 단점: 할당문을 식의 피연산자에 허용하는 단점으로 다른 종류의 식 부작용 유발, 프로그램 가독성 저하

(4) 혼합형 할당형
① 할당문의 양편 자료형이 서로 다를 경우
② 설계시 고려사항: 식의 타입이 할당된 변수의 형과의 동일성 검사, 두 형이 일치하지 않는 경우 묵시적 형변환의 사용여부

5. 상수 및 변수 초기화
① 상수: 변하지 않는 값을 갖는 변수의 사용을 지원하기 위한 개념
② 컴파일러가 쉽게 인식할 수 있으므로 프로그램의 신뢰성 증가
③ 식별자로 주어지며 프로그램 수행 중 변하지 않음
④ 주소를 가질 수 있지만 오직 한 개의 자료 값만을 갖기 때문에 내용이 변할 수 없음

6. 표현식, 조건문
(1) 표현식
1) 식의 개요
① 표현식: 하나 이상의 피연산자를 가지고 자료 값의 계산을 기술
② 식평가: 피연산자 값에 대하여 주어진 연산을 실행함으로써 이루어짐
③ 참조투명성: 프로그래밍 환경을 변화시키지 않고 오직 값만을 생성
④ 좌결합법칙: 동일 우선순위를 가지는 두 개의 연산자가 함께 이웃하였을 때 왼쪽 것을 먼저 수행

2) 논리조건
① 적용순서: 피연선자1 연산자 피연산자2와 같은 식의 평가에서 두 개의 피연산자를 평가한 후 결과를 얻기 위해 연산자를 적용하는 방법
② 단락 회로 평가기법: 컴파일러의 불일치를 제거하기 위하여 언어 설계자들은 언어 내부에 새로운 단락 회로 평가 기법 도입

(2) 조건문
① 정의: 조건에 따라 실행되는 부분이 달라질 때 사용하는 문장
② fortran 언어, 내포된 if 문, case 문, switch 문

7. 반복문
(1) 정의
① 한 개 이상의 문장을 0번 이상 반복하여 실행시키는 문장
② 필요성: 반복기능이 없으면 모든 동작을 순차적으로 기술, 복잡성 증가와 유연성의 저하와 가독성의 저하와 유지보수의 어려움이 있음

(2) 사용자 지정 반복
① 반복수행을 하려는 일련의 문장들을 괄호로 묶어서 단위화시켜 반복 수행시키는 것
② C/C++에서는 break 문, Continue 문

(3) 논리제어 반복문
① 초기 조건 검사 하고, 그 결과가 참이면 반복문 몸체를 한번 수행하고 다시 조건 검사를 행하는 과정 반복
② 조건문 결과가 거짓일 경우 반복문 영역을 벗어나 다음 문장으로 수행제어가 넘어감
③ while 문, until문, do-while 문, loop-repeat 와 exit

(4) 제어변수 반복문
① for 문: 반복제어(제어변수)를 사용하여 고정된 횟수의 반복 표시

<3장 출제예상문제>

1. 바인딩에 대한 개념으로 잘못된 것은?
① 변수의 네 가지 요소에 값을 확정하는 것
② 이름에 어떤 속성을 연결하는 과정
③ 일반적으로 변수 속성은 제어문을 통해 이루어짐
④ 변수의 주소란 변수 값이 저장되는 기억장소 위치

[해설]
③ 일반적으로 변수 속성은 선언문을 통해 이루어짐

[정답] 3

2. 바인딩 시간의 종류에 속하지 않는 것은?
① 번역 시간 ② 언어의 설계 시간
③ 실행시간 ④ 언어의 구현 시간

[해설] 바인딩 시간의 종류
- 실행시간: 변수의 값을 확정, 자료 구조에 기억 장소를 배당
- 번역시간: 구성은 컴파일시간, 링크시간, 로드시간, 종류로는 변수의 형, 자료구조의 형과 크기, 레코드의 각 항목들의 형 등을 확정
- 언어의 구현시간: 정수의 자릿수, 실수의 유효숫자 개수, 수의 기계 내에서의 표기법 등
- 언어 정의 시간: 혼합형 연산이 허용되는 연산에서 연산해야 될 두 피연선자의 형에 따라 어떤 형의 연산을 해야 되는지를 확정

[정답] 2

3. 다음 중 선언문의 목적으로 옳지 않은 것은?
① 주기억 장치 사용과 접근 방법의 효율성
② 주기억 장치 경영의 효율성
③ 정적 형 검사 가능
④ 번역기의 구현이 용이

[해설] 선언문의 목적
- 주기억 장치 사용과 접근 방법의 효율성: 프로그램 실행 동안 변하지 않는 자료 구조의 속성들을 한정해 주는 것, 컴파일러가 자료구조에 접근하기 위한 계산을 최적화함, 주기억 장치의 절약 및 프로그램 실행 시간 절약
- 주기억 장치 경영의 효율성: 자료 구조의 크기, 생성 시기, 소멸 시기 등을 번역시간에 알게됨으로써 보다 효율적인 기억장소 배당 기법 제공
- 정적 형 검사 가능: 잘못 사용한 자료형 등을 번역시간에 낮아낼 수 있어 프로그램의 신뢰성을 높일 수 있음

[정답] 4

4. 다음 중 동적 형 바인딩에 대해 바르게 설명한 것은?
① 프로그램 실행시에 프로그램의 외부 구조에 의해 결정된다.
② 프로그램 실행시에 실행 과정에 의해 결정된다.
③ 프로그램 번역시에 프로그램의 외부 구조에 의해 결정된다.
④ 프로그램 번역시에 실행 과정에 의해 결정된다.

[해설]
동적 형 바인딩은 실행 시간에 이루어짐

[정답] 2

5. 피연산자 값에 대해 주어진 연산을 실행함으로써 이루어지는 것은?
① 표현식 ② 참조 투명성
③ 좌결합 법칙 ④ 식 평가

[해설]
① 표현식: 하나 이상의 피연산자를 가지고 자료값의 계산을 기술하는 것
② 참조 투명성: 프로그래밍 환경을 변화시키지 않고 오직 값만을 생성하는 것
③ 좌결합 법칙: 동일 우선순위를 가지는 두 개의 연산자가 함께 이웃했을 때 왼쪽 것을 먼저 수행하는 법칙

[정답] 4

6. 상수에 대한 설명으로 틀린 것은?
① 상수는 변하지 않는 값을 갖는 변수의 사용을 지원하기 위한 개념이다.
② 식별자로 주어지며, 프로그램 수행 중에는 변하지 않는다.
③ 주소를 가질 수 있지만 오직 한 개의 자료 값만을 갖기 때문에 내용이 변할 수 없다.
④ 컴파일러가 쉽게 인식할 수 없으므로 프로그램의 신뢰성이 증가한다.

[해설]
④ 컴파일러가 쉽게 인식할 수 있으므로 프로그램의 신뢰성이 증가함

[정답] 4

7. 다음 중 실행 시 사용할 자료속성을 컴파일러 등에 알려주는 프로그램 문장으로 바인딩을 제공하는 중요한 방법은 ?

① 표현식
② 바인딩
③ 할당
④ 선언

[해설]
선언은 실행 시 사용될 자료의 속성을 컴파일러 등에게 알려주는 프로그램 문장이며 바인딩을 제공하는 중요한 방법이다.

[정답] 4