소프트웨어 공학개론

Introduction to Software Engineering

2022-1학기 (Spring)
선문대학교 AI소프트웨어학과

5월 17일 (화)

강의 목차

- 10. 프로그래밍
  - 10.1 프로그래밍이란?
  - 10.2 구조화프로그래밍
  - 10.3 데이터중심형기법
- 연습 문제

10.1 프로그래밍이란?

개요

- 프로그래밍
  - 분석공정과 설계공정에 있어서 작성된 요구사양 및 설계사양을 토대로, 모듈의 논리설계 (데이터구조와 알고리즘의 결정)를 수행하여 프로그램의 소스코드를 기술하는 작업

목차

10.1 프로그래밍이란?

선문대학교

10.1.1 프로그래밍이란 무엇인가?

- 프로그램: 개발시스템이 고객과 이용자의 요구사항을 달성할 때 실제로 실행되는 성과물
  - 간결하고 이해하기 쉬운 프로그램을 오류없이 구축하는 것이 중요
- 그림 예: 프로그래밍 단계에서의 작업(소프트웨어구축)
  - 구현공정에서 프로그래밍은 주로 코딩작업을 의미
  - 프로그래머
    - 주어진 모듈기능사양과 모듈구조도를 토대로 모듈의 논리설계(데이터구조와 알고리즘의 결정)를 수행하고, 프로그램의 소스코드를 기술
    - 완성된 소스코드에 대해 스스로 단위테스트를 수행하고 모듈의 동작이 올바른지 확인
      - 만약 모듈이 사양에 있는대로 동작하지 않는 경우, 오류 검출 및 제거를 위해 소스코드를 수정
    - 단위테스트에서 더이상 오류를 찾아내지 못하는 시점에서 프로그래밍작업이 종료 및 완성됨

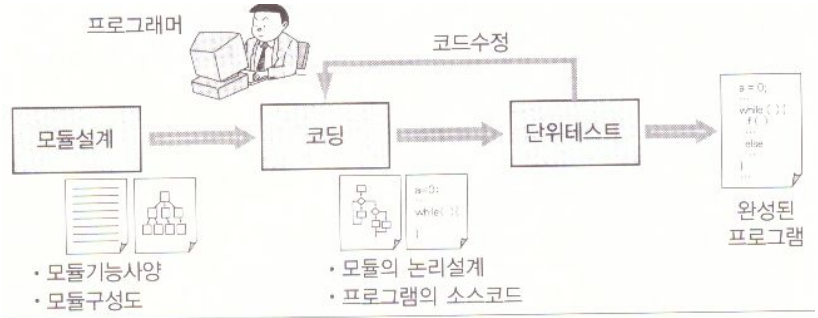

그림 10.1 프로그래밍단계에서의 작업

10.1.2 통합개발환경

- 소프트웨어개발환경
  - 소프트웨어개발작업의 자동화와 부담경감을 목적으로 소프트웨어개발을 지원하는 도구 및 이용방법을 결정하는 체계
    - cf. 프로그래밍 환경: 특히 프로그래밍을 주요지원대상으로 하고 있는 경우
- 통합개발환경 (IDE: Integrated Development Environment)
  - 다양한 개발도구가 개별적으로 존재하는 것이 아니라, 도구들을 연계시켜서 실행할 수 있는 소프트웨어개발환경
  - 도구들의 기반이라는 관점에서 도구 플랫폼(Tool Platform)이라고 부름
  - 소스코드를 편집하는 editor, 소스코드를 실행코드로 변환시키는 compiler, 소스코드의 실행을 감시하는 profiler, debugger, tester, code checker 등이 통합되어 있음
  - 예: Eclipse, Microsoft Visual Studio, Adobe Flex 등이 있음

10.1 프로그래밍이란?

10.1.3 프로그래밍 패러다임

- 프로그래밍 패러다임
  - (1) 프로그램의 작성방법에 관한 규범
  - (2) 설계순서와 프로그램구조 및 프로그램의 기술방법에 대해 규정한 사항들
  - (3) 프로그래밍을 수행할 때 무엇에 초점을 맞추어서 문제를 정리하는지, 무엇을 중심으로 프로그램을 구성하는지에 대한 방향을 설정한 것

- 프로그램의 논리, 제어, 데이터구조 등을 파악하는 방법이나 표현방법의 차이에 의한 주요 프로그래밍 패러다임의 분류

  - (1) 절차형 프로그래밍(명령형 프로그래밍)
    - 컴퓨터 처리절차를 명령문으로 기술
    - 구조화 프로그래밍을 토대로 함
    - 주요 언어(대부분의 절차형 언어): FORTRAN, COBOL, BASIC, PASCAL, C, Ada 등

  - (2) 함수형 프로그래밍
    - 입출력관계를 표현하는 함수와 그에 대한 호출로서 기술
    - 주요언어: Lisp, Scheme, ML, Haskell 등

10.1.3 프로그래밍 패러다임

- 프로그램의 논리, 제어, 데이터구조 등을 파악하는 방법이나 표현방법의 차이에 의한 주요 프로그래밍 패러다임의 분류
  - (3) 논리형 프로그래밍
    - 입출력관계를 술어논리(사실과 규칙)로 기술
    - 주요언어: Prolog
  - (4) 객체지향 프로그래밍
    - 데이터와 데이터를 처리하는 동작을 캡슐화한 객체들, 그리고 객체들 사이의 메시지송수신으로 기술
    - 주요언어: Smalltalk, C++, Java, C#, Ruby 등
  - (5) 관점지향프로그래밍 (Aspect Oriented Programming)
    - 여러 객체들에 관련되어 있는 횡단적인 관심사항들을 aspect에 모아서 기술하고 나중에 조합
    - 주요언어: AspectJ, Hyper/J 등

10.2 구조화 프로그래밍

개요

- 현재와 같이 대규모의 높은 신뢰성을 갖는 소프트웨어가 요구되는 상황에서는, 이해하기 수월한 프로그램을 작성하는 것이 중요
  - 이해하기 수월하다는 것은 프로그램의 구조와 동작을 수월하게 파악할 수 있다는 뜻
- 이해하기 수월한 프로그램
  - 올바른 동작을 수행하는지 검사하기 수월하며, 오류를 찾아내기 용이
  - 미래에 변경과 확장이 수월하게 되며 유지보수 비용을 절감할 수 있음
- 위와 같은 사항을 토대로 좋은 구조를 갖는 프로그램을 작성하기 위한 기법으로서 구조화 프로그래밍이 등장함

목차

10.2 구조화 프로그래밍

10.2.1 프로그램의 구조

- 일반적으로 프로그램의 구조는 채용된 프로그래밍언어에 의해 규정됨
  - 절차형언어(C, FORTRAN, COBOL등)에서는 함수, 서브루틴, 프로시저가 모듈에 해당됨
  - 객체지향언어(Smalltalk, Java, C++ 등)에서는 클래스를 모듈이라고 생각하는 것이 일반적

- 모듈의 논리설계에 있어서 프로그래머는 데이터구조와 제어구조를 결정할 필요가 있음
  - 프로그래밍언어에 의해 제공되는 데이터형과 데이터형을 정의/확장/구조화하는 체계가 서로 다름
    - C: 정수형(int)과 메모리를 직접 참조가능한 포인터형, 각 데이터형을 구조화할 수 있는 배열이나 구조체 등이 제공
    - Java: Java가 제공하는 기본형과 프로그래머가 형(type)을 직접 정의할 수 있는 체계를 제공
  - 제어구조에 관해서도 각 프로그래밍 언어마다 서로 체계가 다름
    - C: 단순한 선택구조(if, switch), 반복구조(do, while), 분기처리(break, continue, return), 함수호출 등이 자주 사용
    - Java: 이와 함께 예외처리, 동적바인딩, 다중쓰레드실행 등도 제공

10.2 구조화 프로그래밍

10.2.1 프로그램의 구조

- 프로그래머는 채용할 프로그래밍언어에 정통해 있어야 함
  - 사양에 기초한 올바른 프로그램을 작성을 위해
- 프로그래머는 다양한 알고리즘을 사용할 수 있어야 함
  - 고객과 이용자의 요구사항이 현실적인지를 판단하기 위해
- 알고리즘은 다음과 같은 처리절차를 의미
  - 데이터 정렬
  - 대량의 데이터들로부터 특정한 데이터 탐색
  - 그래프상에서 최단경로를 구하기

10.2.2 구조화 정리

- Edsger W. Dijkstra에 의해 제안된 “구조화 프로그래밍”의 기본적인 개념
  - 이해하기 쉬운 프로그램은 제어구조가 단순하고, 프로그램의 실행순서가 소스코드내부의 명령이나 문장의 기술순서와 거의 같은 연속성을 갖는다는 조사결과를 기반으로 함
    - 프로그램내부의 임의의 지점(명령어 또는 문장)에 제어를 이동시킬 수 있는 GOTO문이 많이 사용되면, 프로그램의 실행지점이 무질서하게 소스코드 내부의 여기저기로 이동하게 됨
      → 이와같은 프로그램의 실행순서는 추적하기 어렵게 되고, 소스코드에서 구현되고 있는 기능을 이해하기 어렵게 됨
    - 소스코드에 버그가 존재하는 경우, 버그의 영향범위를 파악하는 것도 어렵게 됨
      → 구조가 복잡한 프로그램에 대한 작업 부담이 크게 증가함

10.2.2 구조화 정리

- 구조화프로그래밍에서는 GOTO문을 가능한 한 작성하지 않고, 프로그램을 그림과 같이 3가지 기본제어구조를 조합하여 기술
- 그림과 같은 도면을 **FlowChart**라고 부름
  - 직사각형: 프로그램 내부의 명령어(문)
  - 마름모: 조건판단
  - 화살표: 프로그램의 제어흐름
- 그림 예: 구조화프로그래밍의 기본제어구조
  - (a) 순차: 명령문A를 실행하고나서, 명령문B를 실행
  - (b) 선택: 조건P의 참/거짓에 의해, 명령문A 또는 명령문B 중에서 어느 하나를 실행
  - (c) 반복: 조건P가 참인 동안 명령문A를 반복실행
    - 명령문A를 실행하기 전에 조건을 판정하는 경우와 명령문A의 실행 후에 조건판단을 수행하는 경우가 있음

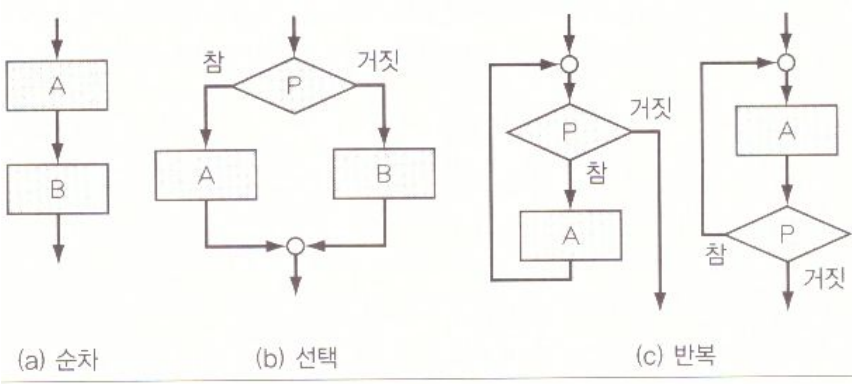

그림 10.3 구조화프로그래밍의 기본제어구조

10.2.2 구조화 정리

- “적정한 프로그램”: 아래 2가지 조건을 만족하는 프로그램
  - (1) 제어의 흐름에 대해서 입구와 출구가 반드시 한 개씩 있다.
  - (2) 입구에서 출구로 향하는 제어흐름에서 모든 명령문들이 관여한다.
    - 절대로 실행되지 않는 명령문이 존재하지 않는다는 것을 의미

- “구조화 정리”
  - 모든 적정한 프로그램이 3가지 제어구조(순차, 선택, 반복)의 조합으로 기술가능하다는 것을 나타낸 것
  - Corrado Bohm과 Giuseppe Jacopini에 의해 증명됨

10.2.2 구조화 정리

- 그림 예: 기본제어구조를 조합한 적정한 프로그램의 예
  - (a): 선택과 반복이 순차적으로 실행됨
  - (b): 반복구조의 내부에 선택구조가 포함되어있고, 다시 선택구조에서 거짓 방향의 처리에서 순차구조가 포함되어 있음
- 구조화프로그래밍은 제어구조를 파악하기 수월한 프로그램을 기술하는 것이 목적
  - 반드시 GOTO문을 배제하는 것을 목적으로 하고 있지 않다는 사실에 주의필요
    - GOTO문을 배제하기만 하면 이해하기 쉬운 프로그램을 기술할 수있는 것이 아님
  - 최근, 함수로부터의 반환(return), 반복으로의 탈출(break), 예외처리 등에서 GOTO문의 필요성이 인정됨

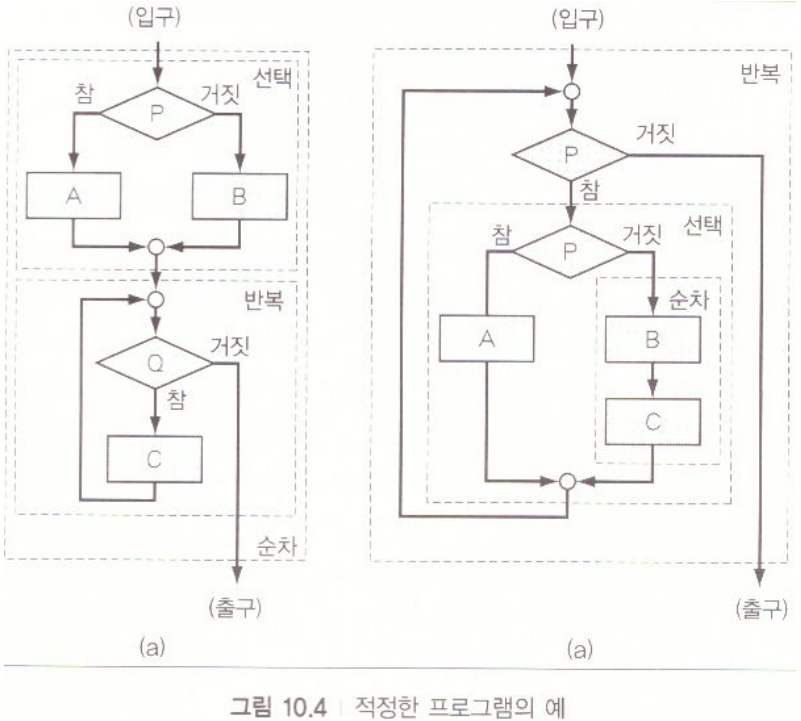

그림 10.4 | 적정한 프로그램의 예

10.2.3 프로그래밍 스타일

- 소스코드를 기술하는 경우에는 정해진 프로그래밍 스타일을 엄수해야 함
  - 일반적으로 프로그램은 에디터를 사용해서 소스코드를 텍스트형식으로 입력
  - 구조화 프로그래밍의 원칙에 따라 소스코드를 기술하는 것을 전제로 하면
    - 각각의 명령문의 제어범위에 대응하는 적절한 들여쓰기가 가능
    - 데이터와 함수에는 각각의 내용을 적절하게 나타낼 수 있는 이름을 사용하는 것이 중요
  - 다수의 개발자가 프로그래밍 작업을 수행하는 경우
    - 조직과 프로젝트에서 정한 들여쓰기 및 이름제정에 관한 코딩규약을 따를 필요가 있음
- **설계패턴(Design Pattern)**
  - 객체지향에 있어서 좋은 설계의 집합
  - 특정한 프로그래밍언어에 의존하지 않는 추상적인 표현으로 정의되어 있는 것이 일반적
  - 같은 처리를 반복하는 경우에 사용하는 Iterator 등과 같은 구현코드를 의식하여 만들어진 것도 존재
    - 코딩패턴(Coding Pattern): 구현코드를 의식하여 만들어진 설계패턴
      - 예: 프로그래밍 정석, 널리 알려진 알고리즘, Idiom(특정한 프로그래밍언어에서 자주 사용되는 구현코드와 알고리즘)

10.3 데이터중심형기법

개요

- 프로세스중심형기법 (POA: Process Oriented Approach)
  - 시스템의 처리(프로세스)를 중심으로 생각하여 데이터흐름을 기반으로 모듈구조를 설계하는 기법
  - 예: STS분할, TR분할

- 데이터중심형기법 (DOA: Data Oriented Approach)
  - 데이터의 기본적인 특성을 중심삼고, 모듈구조를 설계하는 기법
  - 명확한 데이터구조를 갖는 입출력을 다루는 사무처리용시스템 설계에 적합
  - 장점: 업무가 변경되더라도 데이터 자체에 변경이 발생되지 않는다면, 설계의 재수정작업을 최소화할 수 있음
  - 문제점: 처리를 중심으로 설계를 수행하게 되면 각각의 시스템과 모듈에서 구조가 미묘하게 상이한 데이터가 정의될 가능성이 높아짐 → 이러한 데이터를 서로다른 시스템들 사이 또는 모듈들 사이에서 공유하게되면 문제가 발생함
  - 예: Jackson기법 Warnier기법

10.3 데이터중심형기법

목차

10.3 데이터중심형기법

10.3.1 Jackson기법

- Michael Jackson에 의해 제안된 구조화설계기법, JSP(Jackson Structured Programming)이라고도 부름
- 입력데이터의 구조와 출력데이터의 구조에 주목하여, 프로그램의 논리구조를 데이터구조로부터 도출

- 프로그램은 입력데이터를 출력데이터로 변환하는 기계이며, 해당처리는 데이터를 가공하는 것으로 간주
- 데이터의 가공처리 순서, 즉 프로그램의 제어구조는 데이터구조에 의해 결정할 수 있다는 개념을 기반으로 함
  - 데이터구조에 선택이 발견되면, 이를 처리하는 프로그램의 논리구조도 선택구조가 됨
  - 데이터구조에 반복이 발견되면, 이를 처리하는 프로그램의 논리구조도 반복구조가 됨

10.3.1 Jackson기법

- 그림 예:데이터구조를 표현하는 4가지 구성요소
  - (a) **기본**: 더이상 분할할 수 없는 구성요소
    - 한 개의 데이터항목에 대응
  - (b) **연접(連接)**: 서로다른 기본요소들로 구성되는 구성요소
    - 각각의 구성요소들은 왼쪽에서 오른쪽으로 한번씩 순차적으로 나타남
    - 여러 개의 데이터항목을 갖는 레코드에 대응
  - (c) **선택**: 여러개의 구성요소들 중 어느 한 개를 선택하는 구성요소
  - (d) **반복**: 동일한 구성요소들이 반복해서 나타나는 구성요소

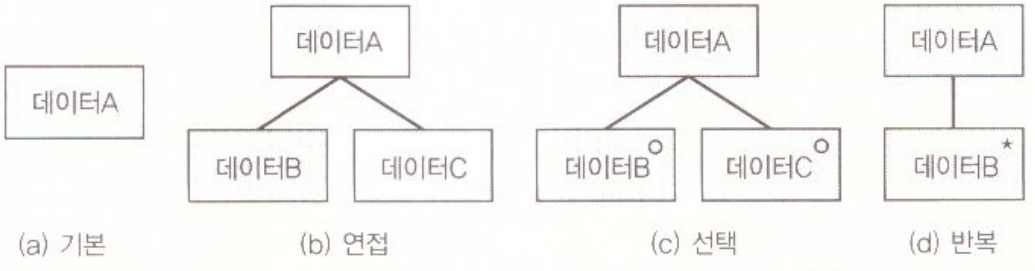

그림 10.5 Jackson기법에서의 데이터구조 표현

10.3.1 Jackson기법

- 예제: 입력데이터와 출력데이터를 다루는 시스템
  - 학생파일(입력데이터)에는 각 학생에 대하여 수강신청과목의 합격/불합격 여부가 기록되어 있음
  - 이 시스템은 학생파일로부터 합격자만을 추출하여 합격자 일람표(출력데이터)를 인쇄함

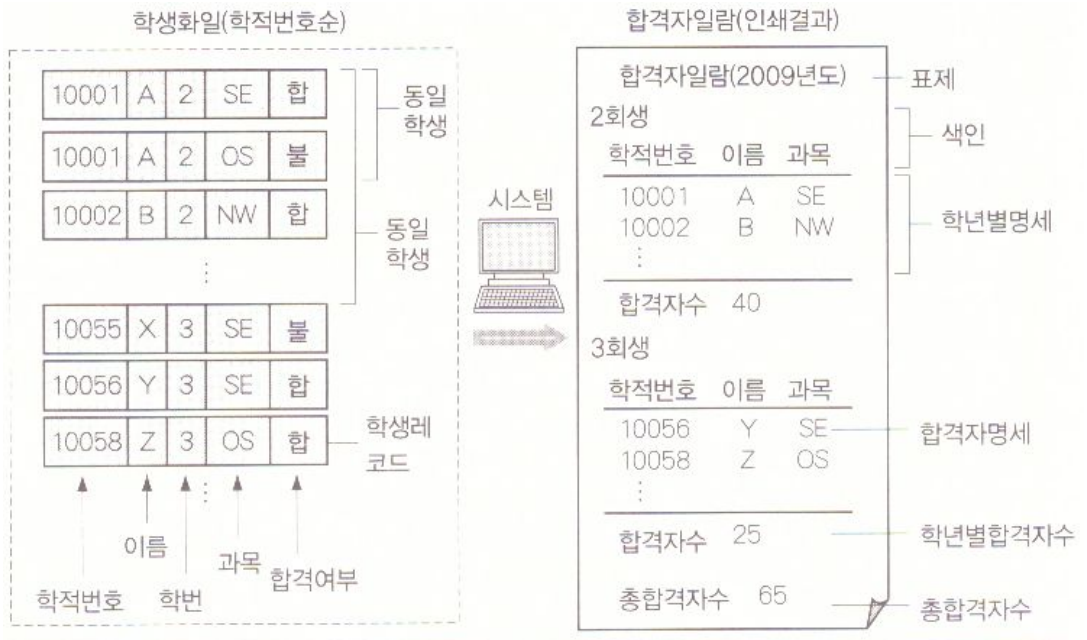

그림 10.6 입력데이터와 출력데이터

10.3.1 Jackson기법

Jackson기법의 절차

- **① 데이터구조의 결정**
  - 입력데이터구조와 출력데이터구조를 분석하고 4가지 구성요소들을 사용하여 각각의 데이터구조도를 정의
  - 그림 예: 입력파일 및 출력결과로부터 작성된 입력데이터구조도와 출력데이터구조도
    - 학생레코드를 최소단위로 취급하고 있고, 학생레코드의 내부에 있는 데이터항목들에 대한 상세사항은 생략

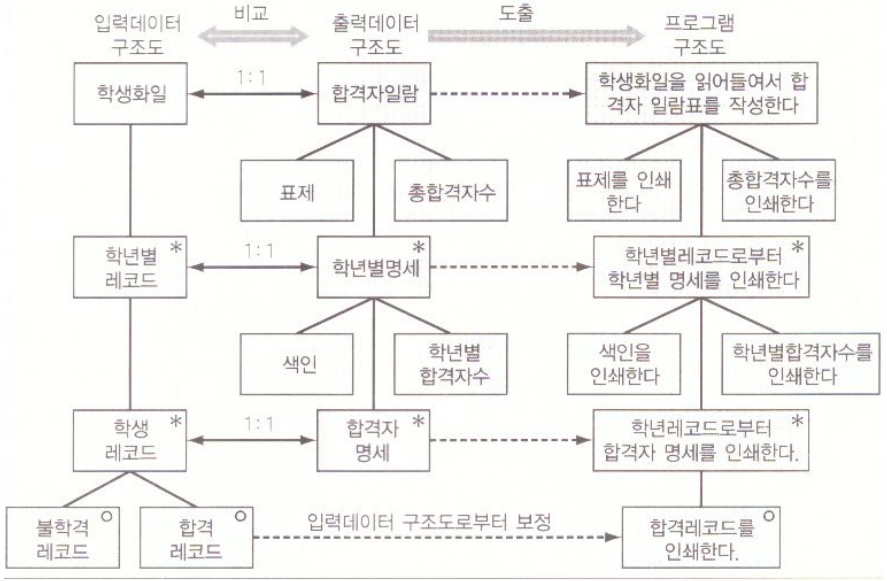

그림 10.7 | Jackson기법에 의한 프로그램구조도의 도입

10.3.1 Jackson기법

Jackson기법의 절차

- **② 데이터구조의 대응관계 결정**
  - 입력데이터구조도와 출력데이터구조도의 구성요소들 사이의 대응관계(입력으로부터 출력으로의 매핑)를 결정함
  - 그림 예: 반복구조에 주목(대응)
    - 학생파일 ⟺ 합격자일람
    - 학년별레코드 ⟺ 학년별명세
    - 학생레코드 ⟺ 합격자명세

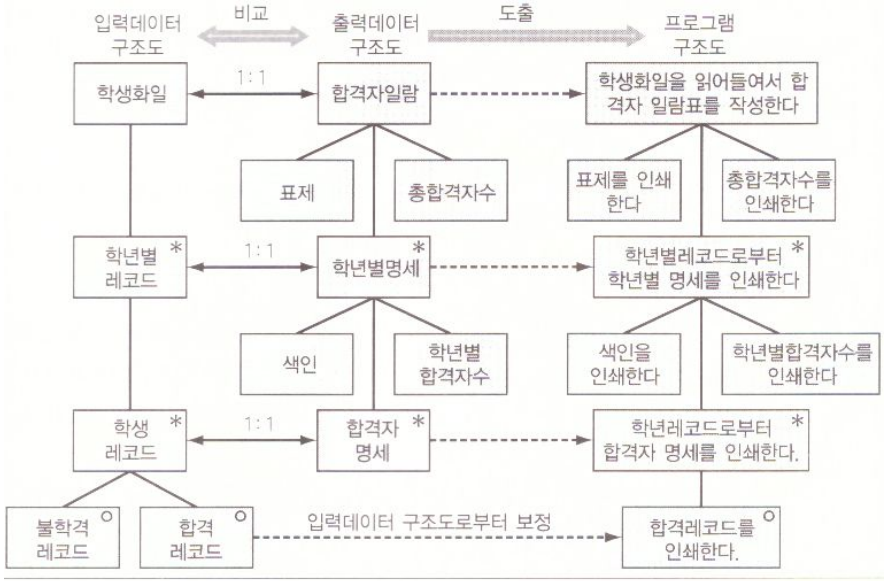

그림 10.7 | Jackson기법에 의한 프로그램구조도의 도입

10.3.1 Jackson기법

Jackson기법의 절차

- **③ 프로그램의 논리구조 결정**
  - 기본적으로 출력데이터구조도를 기반으로 프로그램의 논리구조를 결정
    - 이와 같이 작성된 도면을 “프로그램구조도”라고 부름
  - 프로그램구조도에서는 각각의 구성요소들에 대한 처리를 입력데이터로부터 출력데이터로 변환하는 조작으로 기술
  - 입력데이터구조도에만 나타나는 구성요소에 관해서는 해당 데이터가 프로그램의 출력에 필요한지 검토하고, 프로그램구조도를 수정/보완
  - 출력데이터구조도에만 나타나는 구성요소에 관해서는 출력데이터에 대한 조작으로서 기술

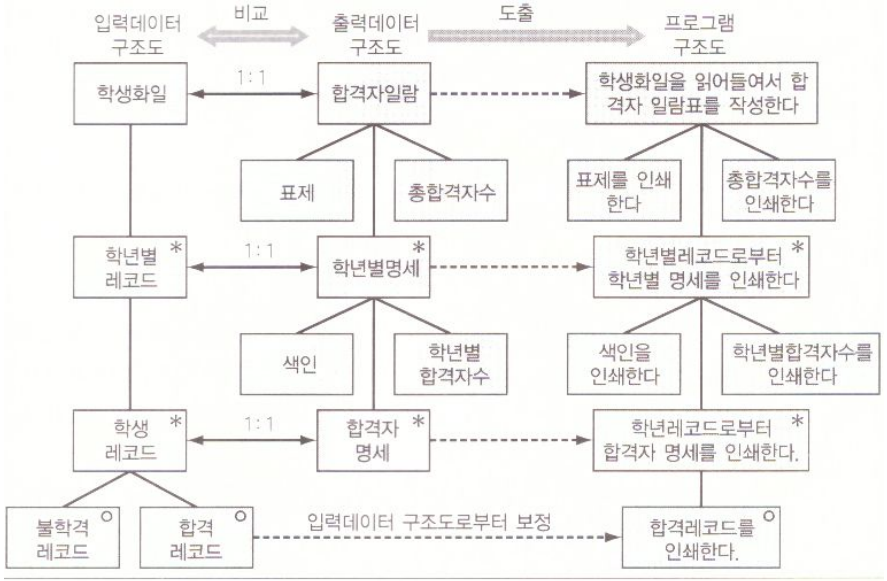

그림 10.7 | Jackson기법에 의한 프로그램구조도의 도입

10.3.1 Jackson기법

Jackson기법의 절차

- ③ **프로그램의 논리구조 결정**(그림 예)
  - 입력데이터구조도에서만 “합격레코드”, “불합격레코드”가 나타남
    - 출력데이터구조에는 “합격레코드”에 대응하는 “합격자명세”가 묵시적 존재
  - 이러한 구성요소들에 관한 수정/보완을 적용하고,“합격레코드”에 관한 조작만을 프로그램구조도에 추가
  - 출력데이터구조도에만 나타나는 구성요소 “표제”,“총합격자수”,“목차”,“학년별합격자수” 에 관해서는 각 데이터에 대한 조각을 프로그램구조도에 그대로 추가
  - 그림의 오른쪽: 완성된 프로그램구조도

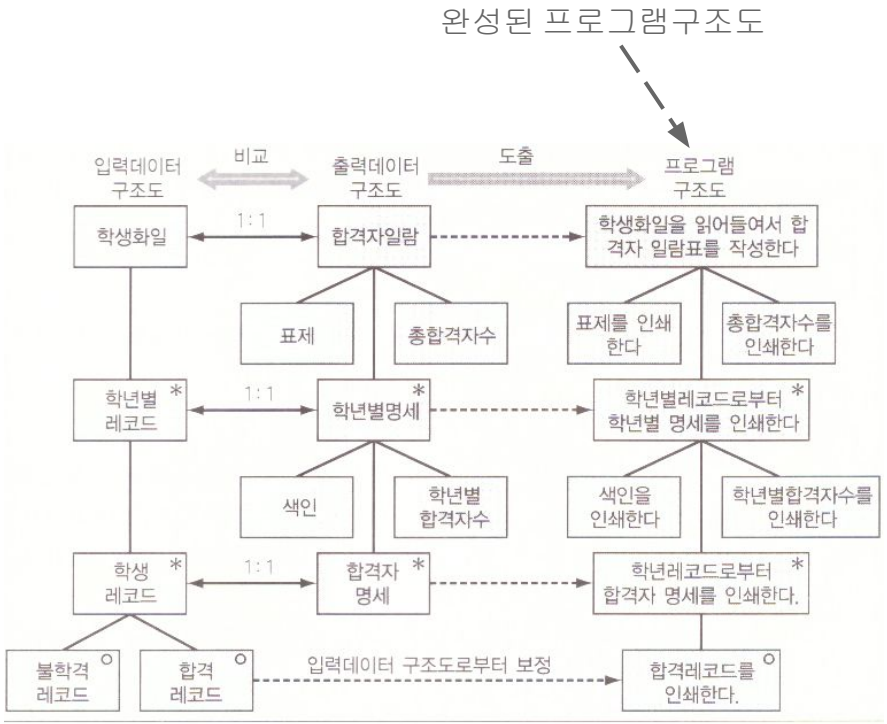

그림 10.7 | Jackson기법에 의한 프로그램구조도의 도입

10.3.2 Warnier기법

- 데이터구조로부터 프로그램의 제어구조를 직접 도출함
  - cf. Jackson기법: 데이터구조를 기반으로 프로그램의 논리구조를 도출
- Jean D. Warnier에 의해 제안되어 Ken Orr가 확장한 기법

Warnier기법의 절차

- **① 데이터구조의 결정**
  - 입력데이터구조와 출력데이터구조를 분석하여 Jackson기법과 유사하게 4가지 구성요소들을 사용하여 각각의 데이터구조도를 정의

10.3.2 Warnier기법

Warnier기법의 절차

- **② 프로그램의 제어구조 결정**
  - 아래의 규칙에 따라서 입력데이터구조를 토대로 프로그램의 제어구조를 결정
    - 선택의 데이터구조로부터 선택의 제어구조를 도출(그림 (a))
    - 반복의 데이터구조로부터 반복의 제어구조를 도출(그림 (b))
    - 입력데이터구조도에만 나타나는 구성요소는 무시
    - 출력데이터구조도에만 나타나는 구성요소는 처리를 추가하여 보완
    - 프로그램의 제어 입구와 출구에 시작처리와 종료처리를 추가

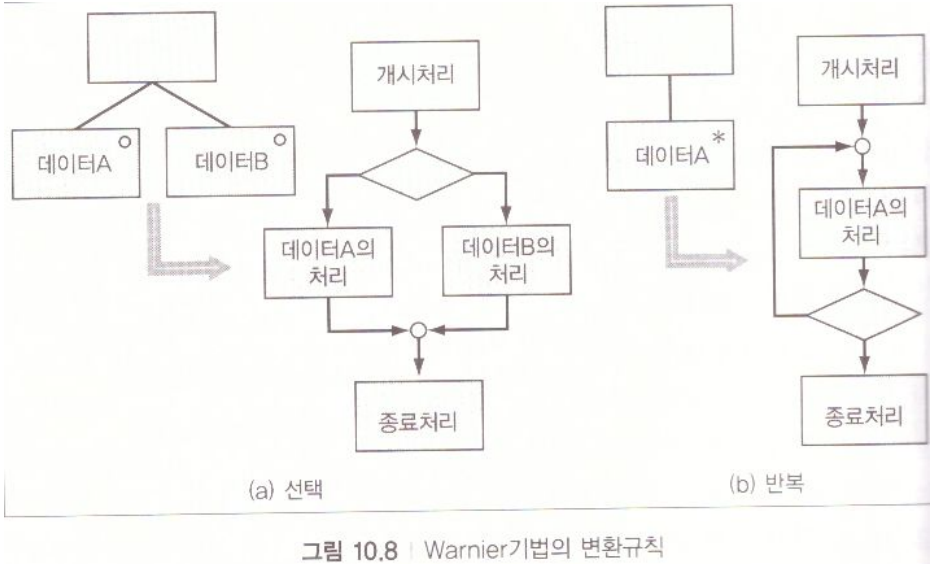

그림 10.8 | Warnier기법의 변환규칙

10.3.2 Warnier기법

Warnier기법의 절차

- **② 프로그램의 제어구조 결정**(그림 예)
  - 입력구조도에서 “학생레코드”는 “학년별레코드”의 반복으로 구성
    - “학생파일”에 대응하는 프로그램은 “학년별레코드”에 대한 처리의 반복으로 도출
  - “학년별레코드”는 “학생레코드”의 반복으로 구성
    - “학년별레코드”에 대한 처리는 “학생레코드”의 반복으로 도출
  - 이와같이 작성된 제어구조들을 조합하여 결합함으로써 최종적인 프로그램의 제어구조가 완성됨

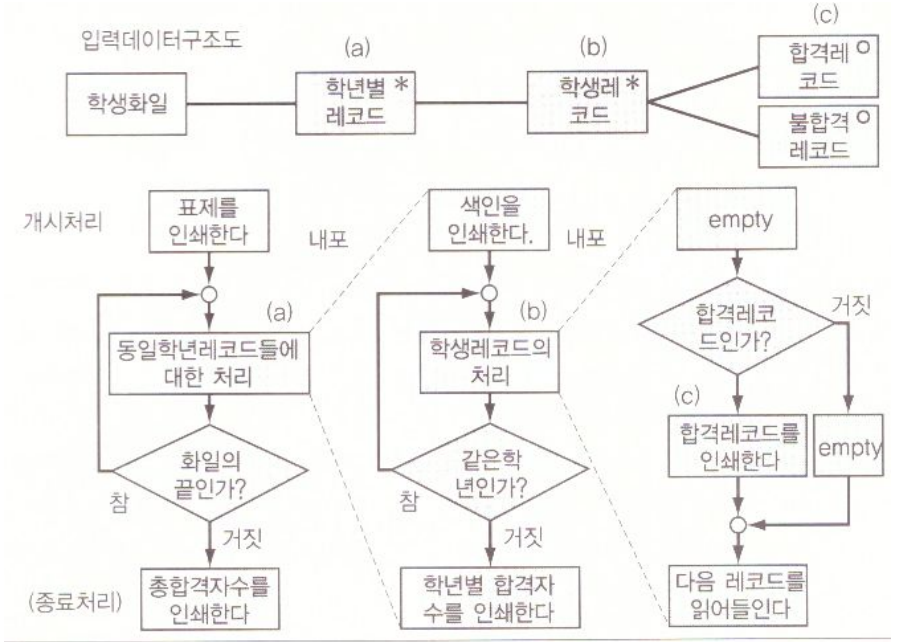

그림 10.9 | Warnier기법에 의한 프로그램 제어구조의 도출

10.3.3 Jackson기법과 Warnier기법을 사용하는 경우의 주의사항

- 데이터구조로부터 프로그램의 논리구조와 제어구조를 도출
- 기법을 적용하기 전에 다양한 시스템에서 데이터를 공유할 수 있도록 데이터구조의 표준화 필요
  - 데이터의 표준화에는 데이터의 정규화를 사용하는 것이 일반적
  - 데이터를 정규화함으로써 데이터의 중복이 제거되어 데이터 사이의 종속관계가 간결하게 됨
- Jackson기법에서는 모듈의 논리구성이 도출됨
  - 프로그래밍기법이라기보다는 모듈분할기법에 가까움
    - 그러나 Jackson기법의 프로그램구조도에 나타나는 구성요소는 모듈에 비해 매우 작음
  - 소규모 소프트웨어의 개발에서 모듈설계에 사용하는 것이 가능
  - 중간규모 내지는 대규모 소프트웨어개발에서는 설계기법으로 사용하는 것보다는 프로그래밍기법으로 적용해도 좋음
- Warnier기법에서는 프로그램의 제어구조가 명령어 수준까지 추출되므로 일종의 프로그래밍기법임

연습문제 (객관식)

각 질문에 대한 알맞은 답의 번호(숫자)를 괄호안에 표시하시오.

- 1. 다음 중 프로그래머가 프로그래밍 단계에서 작업을 할 때의 설명으로 옳지 않은 것은 무엇인가? ( )
  (1) 프로그램의 소스코드를 기술한 것을 토대로 모듈기능사양과 모듈구조도를 작성한다.
  (2) 완성된 소스코드에 대해 스스로 단위테스트를 수행하고 모듈의 동작이 올바른지 확인한다.
  (3) 만약 모듈이 사양에 있는대로 동작하지 않는 경우, 오류 검출 및 제거를 위해 소스코드를 수정한다.
  (4) 단위테스트에서 더이상 오류를 찾아내지 못하는 시점에서 프로그래밍작업이 종료 및 완성된다.

- 2. '프로그래밍 패러다임'의 분류 5가지 중에서, "주요언어로는 Smalltalk, C++, Java, C#, Ruby 등이 있으며, 데이터와 데이터를 처리하는 동작을 캡슐화한 객체들, 그리고 객체들 사이의 메시지송수신으로 기술"하는 분류는 무엇인가? ( )
  (1) 함수형 프로그래밍 (2) 절차형 프로그래밍 (3) 논리형 프로그래밍
  (4) 객체지향 프로그래밍 (5) 관점지향 프로그래밍

객관식 문제 (답안지)

- 1. (1) → 주어진 모듈기능사양과 모듈구조도를 토대로 모듈의 논리설계 (데이터구조와 알고리즘의 결정)를 수행하고 프로그램의 소스코드를 기술한다.

- 2. (4) → 객체지향 프로그래밍

연습문제 (객관식)

선문대학교

각 질문에 대한 알맞은 답의 번호(숫자)를 괄호안에 표시하시오.

- 3. '프로그래밍 패러다임'의 분류 5가지 중에서, "주요언어로는 AspectJ, Hyper/J 등이 있으며, 여러 객체들에 관련되어 있는 횡단적인 관심사항들을 aspect에 모아서 기술하고 나중에 조합"하는 분류는 무엇인가? (　　)
  (1) 객체지향 프로그래밍 (2) 관점지향 프로그래밍 (3) 함수형 프로그래밍 (4) 절차형 프로그래밍 (5) 논리형 프로그래밍

- 4. Jackson기법 등에서 데이터구조를 표현할 때 사용하는 4가지 구성요소 중, "서로다른 기본요소들로 구성되는 구성요소로서, 각각의 구성요소들은 왼쪽에서 오른쪽으로 한번씩 순차적으로 나타나며 여러 개의 데이터항목을 갖는 레코드에 대응"하는 것은 무엇인가? (　　)
  (1) 순차 (2) 기본 (3) 연접 (4) 선택 (5) 반복

객관식 문제 (답안지)

- 3. (2) → 관점지향 프로그래밍 (또는 Aspect Oriented Programming)

- 4. (3) → 연접 (連接)

  'ㅁ'(미음)이 아닌 'ㅇ'(이응) 임에 주의요망

연습문제 (객관식)

선문대학교

각 질문에 대한 알맞은 답의 번호(숫자)를 괄호안에 표시하시오.

- 5. Jackson기법 등에서 데이터구조를 표현할 때 사용하는 4가지 구성요소 중, "더이상 분할할 수 없는 구성요소"는 무엇인가? ( )
  (1) 순차 (2) 기본 (3) 연접 (4) 선택 (5) 반복

- 6. Jackson기법에 대한 설명 중 옳지 않은 것은 무엇인가? ( )
  (1) 데이터의 구조에 주목하여 프로그램의 논리구조를 데이터구조로부터 도출하는 기법이다.
  (2) 프로그램은 입력데이터를 출력데이터로 변환하는 기계이며, 해당처리는 데이터를 가공하는 것으로 간주한다.
  (3) 데이터의 가공처리 순서, 즉 프로그램의 제어구조는 데이터구조에 의해 결정할 수 있다는 개념을 기반으로 한다.
  (4) 데이터구조로부터 프로그램의 제어구조를 직접 도출하는 기법이다.

객관식 문제 (답안지)

- 5. (2) → 기본
- 6. (4) → Warnier기법에 대한 설명

연습문제 (주관식)

선문대학교

주관식 문제 (괄호안에 알맞은 단어를 입력하시오.)

- 1. (　　)이란 분석공정과 설계공정에 있어서 작성된 요구사양 및 설계사양을 토대로, 모듈의 논리설계(데이터구조와 알고리즘의 결정)를 수행하여 프로그램의 소스코드를 기술하는 작업이다.
- 2. 최근의 소프트웨어개발환경은 다양한 개발도구가 개별적으로 존재하는 것이 아니라, 도구들을 연계시켜서 실행할 수 있다. 이와 같은 소프트웨어개발환경을 (　　)이라고 부른다.
- 3. 현재와 같이 대규모의 신뢰도가 높은 소프트웨어가 요구되는 상황에서는 이해하기 쉬운 프로그램을 작성하는 것이 중요하다. 프로그램이 이해하기 쉽다는 것의 의미는 무엇인가?
  (　　)
- 4. 프로그램의 작성방법에 관한 규범으로서 설계순서와 프로그램구조 및 프로그램의 기술방법에 대해 규정한 사항들을 말하며, 프로그래밍을 수행할 때 무엇에 초점을 맞추어서 문제를 정리하는지, 무엇을 중심으로 프로그램을 구성하는지에 대한 방향을 설정한 것을 무엇이라고 하는가?
  (　　)

주관식 문제 (답안지)

- 1. 프로그래밍
- 2. 통합개발환경
- 3. 프로그램의 구조와 동작을 수월하게 파악할 수 있다.
- 4. 프로그래밍 패러다임

연습문제 (주관식)

주관식 문제 (괄호안에 알맞은 단어를 입력하시오.)

- 5. 구조화프로그래밍에서 모든 적정한 프로그램은 3가지 기본제어구조를 조합하여 기술한다. 기본제어구조 3가지는 아래의 ([1]), ([2]), ([3]) 이다.
  ([1])는 명령문A를 먼저 실행한 이후에 명령문B를 실행한다.
  ([2])은 조건P의 참/거짓에 의해, 명령문A 또는 B 중에서 하나를 실행한다.
  ([3])은 조건P가 참인 동안 명령문A를 계속 실행한다.

- 6. 객체지향에 있어서 좋은 설계의 집합으로 일반적으로 특정한 프로그래밍언어에 의존하지 않는 추상적인 표현으로 정의되어 있는 것을 ( )이라고 한다.

- 7. ([1])은 시스템의 처리(프로세스)를 중심으로 생각하여 데이터흐름을 기반으로 모듈구조를 설계하는 기법이고, ([2])은 데이터의 기본적인 특성을 중심삼고, 모듈구조를 설계하는 기법이다.

주관식 문제 (답안지)

- 5. (1) 순차
  (2) 선택
  (3) 반복

- 6. 설계패턴 (Design Pattern)

- 7.

  (1) 프로세스중심형기법
  (또는 POA: Process Oriented Approach)

  (2) 데이터중심형기법
  (또는 DOA: Data Oriented Approach)

연습문제 (주관식)

주관식 문제 (괄호안에 알맞은 단어를 입력하시오.)

- 8. 데이터중심형기법 중 입력데이터의 구조와 출력데이터의 구조에 주목하여, 프로그램의 논리구조를 데이터구조로부터 도출하는 기법을 무엇이라고 하는가?
  ( )

- 9. 데이터중심형기법 중 데이터구조로부터 프로그램의 제어구조를 직접 도출하는 기법을 무엇이라고 하는가?
  ( )

- 10. Jackson기법을 사용하여 프로그램의 논리구조를 추출하는 절차를 순서대로 기술하시오.
  ([1]),
  ([2]),
  ([3])

주관식 문제 (답안지)

- 8. Jackson기법
- 9. Warnier기법
- 10.
  (1) 데이터구조의 결정
  (2) 데이터구조의 대응관계 결정
  (3) 프로그램의 논리구조 결정

Any questions?