5주차 3차시 모듈 설계

【학습목표】

1. 모듈의 논리설계 과정 및 구조도와 모듈설계의 관계를 파악할 수 있다.
2. 논리설계 도구인 의사코드와 N-S Chart, 구조적 코딩을 각각의 특징을 통해서 구분할 수 있다.

학습내용1 : 모듈의 외부설계

*** 모듈설계 (Module Design)**
- 구조도에 나타난 각 모듈의 내용을 상세히 표현하는 작업
- 외부설계
- 논리설계
〈모듈설계는 다른 모듈이 해당 모듈 호출 시 요구되는 모든 정보를 나타내야 하기 때문에 다음과 같은 모듈의 「외부적 특성」을 정의해야 함〉

① 모듈명(이름)
- 해당 모듈을 호출 시에 사용함

② 기능
- 모듈이 수행할 기능을 요약된 문장으로 기술함
- 모듈의 내부적 논리는 언급하지 않음

③ 파라미터 리스트
- 모듈 사이 정보를 수수할 파라미터의 개수와 순서를 정의함

④ 입력 · 출력
- 파라미터 리스트를 입출력으로 구분하여 양식 · 크기 · 속성 · 단위 · 타당성의 식별조건을 기술함
- 입력 호출모듈에서 피호출모듈로 넘겨주는 정보
- 출력 피호출모듈에서 호출모듈로 돌려주는 정보

⑤ 외부효과
- 모듈의 수행결과가 시스템, 다른 모듈에 미치는 영향, 효과를 약술함

학습내용2 : 모듈의 논리 설계

1. 논리 설계

<모듈기능의 처리절차와 내용을 상세히 설계하는 것을 의미함>
- 논리설계는 담당자의 경험 · 능력에 따라서 많은 차이가 발생하는 속인성(屬人性)의 특징을 보임
- 구조적 설계는 생산성이 높고, 고품질의 소프트웨어 생산을 위하여 표준화된 절차와 방안을 제공하고 있음

2. 모듈의 논리설계 과정 (2단계로 요약됨)

논리의 구성	데이터 정의
• 기능수행에 필요한 처리절차를 상세히 설계하는 단계임 • 이해 · 사용의 용이성 확보를 위하여 요구기능을 단계적으로 상세화 시킴 • 논리설계 도구 - 의사코드(pseudo code) - N-S Chart - Structured Coding	• 모듈에 관련되는 데이터를 정확하게 파악하기 위하여 • 모듈과 관련되는 인터페이스 정의 • 모듈 내부에서 사용되는 데이터를 정의해야함

3. 구조도에 나타난 모든 모듈에 대해서

<모듈 설계를 해야함>

외부설계	논리설계

4. 구조도와 모듈설계의 관계

- 교재「P. 264」의「그림 10-1」 참조

5. 모듈의 특성

① 실행은 상호 결합하여 종속적으로 이루어짐 - 그러나 컴파일은 독립적으로 이루어짐
② 시스템의 복잡성 해소에 효과적이고, 다수가 동시에 독립적으로 병행작업이 가능함
③ 유사업무에 부품처럼 공통으로 사용 가능함
④ 모듈의 상속성 · 다형성을 가짐
⑤ 모듈 사이 수수되는 값은 파라미터로 주고받을수있음

6. 모듈설계 시 고려사항

① 적절한 모듈의 크기(Module Size)
② 모듈의 재사용성(Reusability)
③ 판독 · 이해의 용이성
④ 응집도(Cohesion)는 최대화, 결합도(Coupling)는 최소화되도록 배려해야함
⑤ 자료의 추상화(Encapsulation) · 상속성(Inheritance) 개념을 도입해야함

학습내용3 : 모듈의 논리설계 도구와 방법

1. 논리설계 도구

① 의사코드(Pseudo Code)
② N-S Chart
③ 구조적 코딩(Structured Coding)

2. 프로그램 기술언어

<프로그램 기술언어(Program Description Language : PDL) 의사코드(Pseudo Code)>

1) 기본적인 제어구조
- 순차구조
- 선택구조
- 반복구조
- CASE 구조
- 교재「PP. 265~268」참조

3. N-S Chart

<논리기술(Logic Description)에 중점을 두고 있는 그림형식의 표현도구>

1) 논리의 제어구조
- 순차구조
- 선택구조
- 반복구조
- CASE 구조
- 논리의 제어구조 사용 예 교재「PP. 269~270」참조

4. 구조적 코딩

1) Dijkstra가 주장한 구조적 프로그래밍」의 핵심적인 내용
- 프로그램 이해 · 작성을 어렵게 하는 주된 원인 GOTO문의 부적절한 사용임
- 제어구조
    - 순차구조
    - 선택구조
    - 반복구조로 표현 가능함 Boehm, Jacopini에 의해서 증명됨
- GOTO문 배제 프로그램이 위에서 아래로 처리되어 이해가 용이함

<내부논리를 프로그램 기술언어, N-S Chart로 설계하면 그 내용이 구조적 프로그래밍의 구조로 되기 때문에 「구조적 코딩(Structured Coding)」을 취급하는 것은 의미 있음>

* 구조적 코딩의 효율향상 방안
- GOTO 명령의 배제 문제
- 세그먼트 개념의 도입 문제
- 구조적 코딩의 문서화 문제

1) GOTO 명령의 배제 문제

㉮ GOTO문의 배제 프로그램이 위에서 아래로 차례로 처리되어 이해 · 판독 · 관리가 용이해짐

㉯ GOTO문 사용이 허용되는 경우는 다음과 같음 4 가지가 있음
- 모듈의 끝 부분으로 GOTO 하는 경우
- DO 루프의 범위를 벗어나는 경우
- GOTO 명령을 배제하면 모듈 내에 여러 가지 무리가 야기되는 경우
- PL/I의 ON-UNIT에서 모듈의 끝 부분으로 분기하는 경우

2) 세그먼트(Segment) 개념의 도입 문제

㉮ 프로그램 분량이 방대할 경우 취급의 편의를 위해서 적정한 논리량으로 분할하는 세그먼트 개념의 도입이 바람직함

㉯ 세그먼트로 분할 시 유의사항은 다음과 같음
- 하향식으로 분할 트리구조를 유지토록 분할함 판독 · 이해의 용이함
- 세그먼트는 각각 독립된 기능을 갖도록 변경 시 다른 세그먼트에 영향을 최소화시키기 위해서
- 세그먼트 사이는 「블랙박스」 상호간섭 배제를 위해서
- 세그먼트 사이 정보의 수수 「파라미터 리스트」로 해야함

3) 구조적 코딩의 문서화 문제
- 적정한 세그먼트로 구조적 코딩한 소스 리스트 이해 · 판독 · 관리가 용이하여 문서화(Documentation) 대신으로 사용 가능함

㉮ 제어논리 범위를 표현하는 코딩
- Indentation 적용,
- 제어범위의 파악·이해가 용이함

㉯ Label, 데이터 이름은
- 해당 내용 전체를 함축적으로 나타내게 부여함
- 사용언어의 허용 한도 내에서 부여해야함

【학습정리】

1. 모듈의 외부설계를 학습한다.

2. 모듈의 논리 설계를 설명할 수 있다.

3. 모듈의 논리설계 도구와 방법을 학습한다.