

정보처리기사 필기 요약본

* 표기 방법

과목 번호	주제
-------	----

** 정보처리기사 과목

I	소프트웨어 설계
II	소프트웨어 개발
III	데이터베이스 구축
IV	프로그래밍 언어 활용
V	정보시스템 구축 관리

--<개발 사전지식>-----

I	소프트웨어 생명주기 : 그래서 소프트웨어를 어떤 과정으로 개발한다는거야?
---	------------------------------------------

1) 소프트웨어 생명 주기 (소프트웨어 수명 주기)

- 정의 : 소프트웨어를 개발하기 위한 과정 (요구사항 분석, 설계, 개발, 유지보수 등)
- 종류 : 폭포수 모형, 프로토타입 모형, 나선형 모형, 애자일

(1) 폭포수 모형 (고전적 생명 주기 모형)

- a. 이전 단계로 돌아갈 수 없음 => 앞단계가 끝나야 다음단계로 넘어감 (병행X)

(2) 프로토타입 모형 (원형 모형)

- a. 프로토타입(건본)을 만들어 최종 결과물을 예측
- b. 새로운 요구사항이 도출될 때마다 프로토타입 구축 => 새로운 요구사항 반영 OK

(3) 나선형 모형 (점진적 모형)

- a. 여러 번의 소프트웨어 개발과정 (계획-> 분석-> 개발-> 평가)의 반복
- b. 위험 관리 중심, 요구사항 첨가 가능, 유지보수 필요X

(4) 애자일

- a. 고객과의 소통에 초점을 맞춘 개발 방법론
- b. 스프린트 또는 이터레이션같은 짧은 개발 주기를 반복 -> 테스트

=> 고객의 요구사항 변화에 유연하게 대응 가능

- c. 애자일 모형의 종류 : 스크럼, XP, 기능 중심 개발(FDD)

	폭포수 모형	애자일
새로운 요구사항 반영	어려움	지속적으로 반영
고객과 의사소통	적음	지속적
테스트	마지막에 모든 기능 테스트	반복적으로 테스트
개발 중심	계획, 문서(메뉴얼)	고객

d. XP 기법

- 이터레이션(짧은 개발-피드백 단위) 반복
- 핵심가치 : 피드백, 존중, 용기, 단순성, 소통 (피존용 단소)
- 주요 실천 방법 : 짝 프로그래밍(함께 프로그래밍 수행),

코드 공동 소유 (권한과 책임의 공동 소유),

테스트 주도 개발, 전체 팀 (모두가 각자의 역할이 있다!)

I	UML : 개발 과정에서 의사소통은 어떻게해?
---	---------------------------






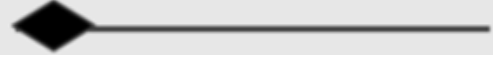
1, UML

☞ 정의 : 의사 소통이 원활하게 이루어지도록 표준화한 객체 지향 모델링 언어

☞ 구성 요소 : 사물, 관계, 다이어그램

1) 사물 : 구조사물, 행동사물, 그룹사물, 주해사물

2) 관계 : 사물과 사물 사이의 연관성 표현

종류	의미	기호
연관 관계	서로 연관됨	
의존 관계	일시적으로 연관 있음	
일반화 관계	상위 개념	
실체화 관계	공통적인 기능	
집합 관계	다른 사물에 포함됨(종속X)	
포함 관계	다른 사물에 포함됨(종속)	

3) 다이어그램 : 사물과 관계를 도형으로 표현한 것

구조적 다이어그램	행위 다이어그램
☞ 클래스 다이어그램 ☞ 객체 다이어그램 ☞ 컴포넌트 다이어그램 ☞ 배치 다이어그램 ☞ 복합체 구조 다이어그램 ☞ 패키지 다이어그램	☞ 유스케이스 다이어그램 ☞ 순차 다이어그램 ☞ 커뮤니케이션 다이어그램 ☞ 상태 다이어그램 ☞ 활동 다이어그램 ☞ 상호작용 개요 다이어그램 ☞ 타이밍 다이어그램

5) 자주 사용되는 다이어그램

a. 클래스 다이어그램

- 정의 : 시스템을 구성하는 클래스, 클래스의 특성과 오퍼레이션(동작), 속성과 오퍼레이션에 대한 제약조건, 클래스 사이의 관계를 나타냄
- 구성요소 : 클래스(객체의 속성 + 동작), 제약조건, 관계

b. 유스케이스 다이어그램

- 정의 : 수행할 수 있는 기능을 사용자 관점에서 표현함

구성요소	기능
시스템	시스템 내부의 유스케이스를 사각형으로 묶어 시스템의 범위 표현
액터	시스템과 상호작용하는 모든 외부 요소 (사람, 외부시스템 등)
유스케이스	시스템이 액터에게 제공하는 기능
관계	구성요소들간의 관계

c. 순차 다이어그램

- 정의 : 시스템이나 객체들이 시간의 흐름에 따라 상호작용하는 과정을 표현

--<개발 과정>-----

I	요구사항 분석 : 개발 전, 고객이 원하는게 뭔지 파악하자!
---	-----------------------------------

1) 요구사항

☞ 정의 : 소프트웨어에 대한 필요 조건이나 제약사항

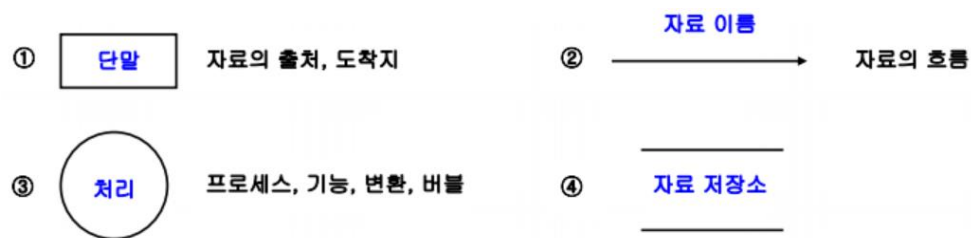
- a. 기능 요구사항 : 소프트웨어가 반드시 수행해야하는 기능
- b. 비기능 요구 사항 : 성능, 보안, 품질, 제약사항 등

2) 요구사항 분석 과정

- (1) 도출 : 질문을 바탕으로 요구사항을 수집함 (프로토타이핑, 유스케이스 등 활용)
- (2) 분석 : 요구사항의 타당성 분석, 상충될 경우 중재
- (3) 명세 : 문서화(명세서 작성)
- (4) 확인 : 방법 - 동료검토 / 워크스루 (사전 검토 후 회의) / 인스펙션 (전문가 검토)

3) 요구사항 분석 시 이용되는 도구

a. 자료흐름도 (Data Flow Diagram, DFD) : 자료의 흐름을 도형 중심으로 기술



b. 자료 사전 : 자료 흐름도에 있는 자료를 더 자세히 정의하고 기록

기호	의미
=	자료의 정의 (~is composed of)
+	자료의 연결 (and)
[]	자료의 선택 (or)
{ }	자료의 반복
()	생략 가능
* *	주석

3) 객체지향 분석 : 요구 사항을 객체 기준으로 분석

(1) 정의

- 객체지향 : 현실 세계의 대상을 하나의 모듈로 만들어서 조립
- 객체: (데이터 + 기능)을 캡슐화 한 것(하나로 묶음)
- 클래스 : 객체의 설계도 (속성, 연산을 정의한 틀)

(2) 객체 지향의 특징

- 캡슐화 : 데이터 + 함수(기능)을 하나로 묶어서 외부 은닉 효과
- 다형성 : 여러가지 형태를 가질 수 있음
(차 클래스가 생성한 객체가 모닝이든 소나타든 벤츠든 OK)
- 상속 : 부모 클래스의 모든 속성과 연산을 하위 클래스가 물려받음
(차의 설계도를 가지고 소나타 설계도를 만들고 소나타 객체 생산)
- 연관성 : 객체들이 상호 참조함

(3) 객체지향 분석 기법 : 림바우 기법(객체 모델링 기법)

- 순서 : 객체 모델링 -> 동적 모델링 -> 기능 모델링 (객동기)
 - 객체 모델링 : 객체와 객체간의 관계를 찾아내 객체 다이어그램으로.
 - 동적 모델링 : 동적인 행위를 상태 다이어그램으로
(시간의 흐름에 따른 제어 흐름, 상호작용, 동작 순서 등)
 - 기능 모델링 : DFD 이용, 자료 흐름 중심의 모델링

I	설계 -> SW 품질을 높일 수 있도록 설계해야!
----------	-----------------------------

1. SW의 품질 특성

- 1) 기능성 : 사용자의 요구사항을 정확하게 만족하는 기능을 제공하는지
- 2) 신뢰성 : 기능을 오류없이 수행하는지
- 3) 사용성 : 쉽게 배우고 사용할 수 있는지
- 4) 효율성 : 얼마나 빨리 기능이 수행되는지
- 5) 유지 보수성 : 변화에 따라 개선하거나 확장할 수 있는지
- 6) 이식성 : 다른 환경에서도 얼마나 쉽게 적용되는지

2. 화면설계 (UIUX 설계)

1) 사용자 인터페이스 (UI)

- (1) 정의: 사용자와 시스템 간의 상호작용이 원활하게 이뤄지도록 도와주는 장치/SW

=> 사용자 중심으로 설계되어야함

(2) 사용자 인터페이스 구분

a. CLI (Command Line Interface) : 텍스트 형태 인터페이스 (터미널)

b. GUI (Graphical User Interface) : 그래픽 환경의 인터페이스

=> 아이콘이나 메뉴를 마우스로 선택하여 작업을 수행

c. NUI (Natural User Interface) : 말이나 행동으로 기기를 조작

d. VUI (Voice User Interface) : 음성으로 조작

e. OUI (Organic User Interface)

: 모든 사물과 사용자간의 상호작용을 위한 인터페이스

(3) 기본 원칙

- a. 직관성 : 누구나 쉽게 사용할 수 있어야함
- b. 유효성 : 사용자의 목적을 정확하고 완벽하게 달성해야함
- c. 학습성 : 누구나 쉽게 배울 수 있어야함
- d. 유연성 : 실수를 최소화해야함

(4) 설계 지침

- 사용자 중심, 사용성(사용자의 편리한 사용) 심미성(가독성을 높이는 설계), 오류 발생(오류가 발생하면 사용자가 쉽게 인지할 수 있도록)

2. 애플리케이션 설계

1) 소프트웨어 아키텍처

(1) 정의 : 소프트웨어의 골격이 되는 기본 구조

(2) 설계 과정 : 목표설정 -> 시스템 타입 결정 -> 패턴 적용 -> 구체화 -> 검토

(3) 설계의 기본 원리

- 추상화: 전체적이고 포괄적인 개념 설계 후 구체화

=> 유형 : 제어 추상화, 과정 추상화, 자료(데이터) 추상화 (제과자)

- 단계적 분해: 하향식 설계 (상위 중요 개념 -> 하위 개념)

- 정보 은닉 : 모듈 내의 데이터나 기능에 다른 모듈이 접근하지 못하도록

- 모듈화 : 시스템이 모듈 단위로 분해되어야함

a. 모듈: 시스템의 기능적 단위

b. 팬인(입력)은 높게, 팬아웃(출력)은 낮게!

c. 약한 결합도, 강한 응집도 필요 (표의 위일수록 좋은거)

결합도 : 모델 사이의 의존정도	응집도 : 내부요소들의 관련 정도
자료 결합도: 자료를 주고받음	기능적 응집: 전 기능이 단일 문제와 관련
스탬프 결합도: 자료구조를 주고받음	순차적 응집 : A->B 순차적 수행
제어 결합도: 다른 모듈의 흐름 제어	교환(통신)적 응집 : 동일 입력 다른 기능
외부 결합: 모듈의 데이터를 외부에서 참조	절차적 응집 : 순차가 많음
공통 결합도 : 공통 데이터 영역 사용	시간적응집: 특정 시간에 필요한 기능 모음
내용 결합도 : 기능이나 자료를 직접 참조	논리적 응집 : 유사한 성격의 기능 모음
	우연적 응집 : 관련 X

(4) 아키텍처 패턴 : 전형적인 소프트웨어 아키텍처 설계 방법

- 파이프-필터 : 데이터 흐름(스트림)의 각 단계별로 캡슐화(묶어서 관리)
- 모델-뷰-컨트롤러 :
 - a. 모델 : 핵심기능과 데이터보관
 - b. 뷰 : 사용자계 정보 표시
 - c. 컨트롤러 : 사용자의 요청을 처리하기 위해 모델에게 명령
- 마스터-슬레이브 : 동일한 구조의 슬레이브 컴포넌트들이 작업을 분할 수행

2) 디자인 패턴

- (1) 정의 : 모듈의 역할, 모듈 간 인터페이스같은 세부적인 구현 방안 설계시의 패턴
- (2) 종류 : 생성 패턴, 구조 패턴, 행위 패턴
- (3) 생성 패턴 : 객체의 생성과 관련된 패턴
 - 추상팩토리 : 인터페이스를 조합해 객체 생성
 - 빌더 : 다른 객체를 조합해서 객체 생성

- 팩토리 메소드 : 상위 클래스를 바탕으로 하위 클래스에서 구체화
- 프로토 타입 : 원본 객체를 복제해서 생성 (비용 저렴)
- 싱글톤 : 클래스로 하나의 객체만 생성

(4) 구조패턴 : 클래스나 객체를 조합하여 더 큰 구조 만드는 패턴

- 어댑터 : 호환되지 않는 클래스를 다른 클래스가 이용할 수 있게 변환
- 브리지 : 기능과 구현을 두 개의 별도 클래스로 구현
- 컴포지트 : 객체들을 트리 구조로 구성
- 프록시 : 접근이 어려운 객체 사이의 인터페이스 역할 수행
- 기타 : 데코레이터, 퍼싸드, 플라이웨이트

(5) 행위 패턴 : 객체들의 서로 상호작용 하는 방법 정의

- 커맨드 : 요청을 캡슐화해서 저장해둠
- 중재자(Medidator) : 상호작용을 캡슐화하여 객체로 정의
- 옵서버 : 한 객체의 상태가 변화하면 다른 객체들에게 전달 (이벤트)
- 상태 : 상태에 따라 다르게 처리
- 전략 : 동일 계열 알고리즘을 개별적 캡슐화 (상호교환 가능)
- 템플릿 메소드 : 상위 클래스는 골격정의, 하위 클래스는 세부 처리
- 방문자 : 처리 기능을 분리하여 별도의 클래스로
- 기타 : 책임 연쇄, 인터프리터, 반복자(iterator), 메멘토

3) 객체 지향 설계 원칙 (SOLID)

- SRP(Single Responsibility, 단일 책임) : 객체는 하나의 책임만 가져야함
- OCP(Open-Closed, 개방-폐쇄) : 기존 코드 변경없이 기능 추가 가능해야

- LSP(Liskov Substitution, 리스코프 치환)

: 자식 클래스는 부모 클래스의 행위를 할 수 있어야함 => 확장만

- ISP(Interface Segregation, 인터페이스 분리): 인터페이스 버전의 SPR

- DIP(Dependency Inversion, 의존 역전): 추상성이 높은 클래스를 의존

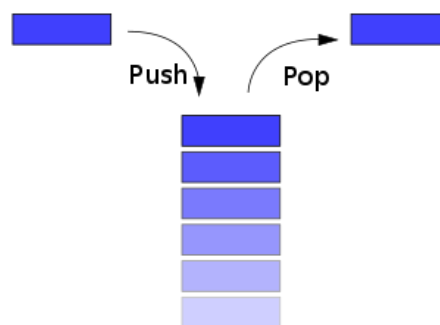
II	개발 + 테스트 : 개발하면서 테스트 진행 (개발 끝나고만 하는거 아님!)
-----------	-------------------------------------------

1. 기본적인 구현 방법

1) 자료 구조

(1) 선형 자료 구조 : 배열, 선형 리스트(연속리스트와 연결리스트), 스택, 큐, 덱

a. 스택



- 특징 : 한 쪽으로만 입출력 => 후입선출 (LIFO, Last In First Out)
- 오버플로우 : 스택이 가득 찬 상태에서 데이터 삽입시 발생
- 언더플로우 : 스택이 텅 빈 상태에서 데이터 삭제시 발생
- 활용 : 인터럽트의 처리, 수식의 계산, 서브루틴 호출 작업 등

b. 큐 : 선입선출 (FIFO, First In First Out)

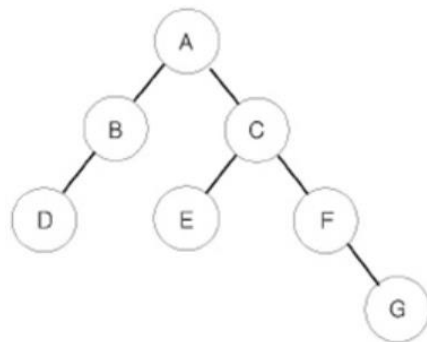
c. 덱 : 삽입과 삭제가 양쪽 끝에서 모두 발생 가능

(2) 비선형 자료 구조 : 트리, 그래프

a. 트리

- 정의 : 사이클이 없는 그래프

노드(정점)	트리의 기본 요소 (하나의 기억 공간)
링크(간선)	노드와 노드를 연결하는 선
차수(Degree)	연결된 하위 노드(자식 노드)의 개수 = 각 노드가 가진 가지 수
트리의 차수	노드의 차수 중 가장 많은 수
루트 노드	맨 위에 있는 노드
리프 노드	차수가 0인 노드



트리의 운행법(순회 방법)	순회 순서
Preorder : root -> left -> right	ABDCEFG
Inorder : left -> root -> right	DBAECFG
Postorder : left -> right -> root	DBEGFCA

2) 정렬 구현 (=> 실제로 구현했을 때의 결과를 주로 물어봄, 괄호는 시간복잡도)

(1) 선택정렬: n개중 최소값을 1번에, n-1개중 최소값을 2번에 두는 것을 반복 (n^2)

(2) 버블정렬: 인접한 두 개의 값을 비교해서 서로 교환하기를 반복 (n^2)

(3) 삽입정렬: n번째 키를 적절한 위치(1 ~ n)에 삽입해서 정렬

(3) 퀵정렬 : 하나의 파일을 부분적으로 나누어가며 정렬 ($n \log n \sim n^2$)

=> 키(피벗)보다 작으면 왼쪽, 크면 오른쪽으로 분해하는 방식

(4) 힙정렬 : 완전이진트리(Complete Binary Tree)를 이용한 정렬 ($n \log n$)

3) 검색 구현

(1) 이진 검색 : 찾는 값을 파일의 중간 값과 비교하면서 검색 반복 (정렬필수)

=> 중간 번호 = (첫번째 번호 + 마지막 번호) /

(2) 해싱 : 해싱 함수를 통해 주소값을 계산하여 해당 주소값에 저장/검색

해싱 함수	주소값
제산법	키 값을 소수로 나눈 나머지
제공법	키 값을 제공한 값의 중간 부분
폴딩법	키를 부분으로 나눠 각 부분을 더하거나 XOR한 값
숫자분석법	키 값을 이루는 숫자의 분포를 분석해서 나온 값

4) 알고리즘 시간 복잡도

(1) 빅오표기법 : 알고리즘 실행시 최악의 경우를 표기

- $O(1)$: 입력값과 관계없이 하나의 단계만을 거침

- $O(n \log n)$: 문제 해결에 필요한 단계($O(\log n)$)가 n번 실행됨

5) 클린코드 : 단순 명료하게 잘 작성된 코드

- 배드코드 : 스파게티 코드(복잡한 로직), 외계인 코드(유지보수 어려움)

(1) 작성원칙

가독성	누구든지 쉽게 읽을 수 있게 => 쉬운 용어, 들여쓰기
단순성	간단하게 작성 => 클래스, 메서드 등을 최소 단위로 분리 (한가지만 처리)
의존성 배제	다른 모듈에 미치는 영향 최소화 => 코드를 변경해도 영향주지 않도록
중복성 최소화	중복된 코드 삭제
추상화	상위클래스, 메서드는 간략하게, 구현은 하위클래스나 메서드에서

(2) 소스 품질 분석 도구

- 정적 분석 도구 : 실행x, 코딩 표준이나 스타일, 결함 분석
- 동적 분석 도구 : 실행 => 메모리 누수, 스레드 결함 분석

5) 개발 지원 도구

(1) 통합 개발 환경 (IDE) : 개발 관련 모든 작업을 하나의 프로그램에서 처리

코딩	프로그램 작성
컴파일	프로그램을 컴퓨터에서 실행가능한 형태로 변환
디버깅	버그를 찾아 수정
배포	사용자에게 배포

(2) 빌드 도구 : 코드를 제품 소프트웨어로 변환

- 종류 : Ant, Maven, Gradle(Groovy 기반)

2. 인터페이스 구현

1) 모듈 연계 방법 (데이터 교환을 위한 관계 설정) : EAI

EAI 구축 유형	설명
Point-to-Point	애플리케이션의 1:1 연결
Hub & Spoke	허브를 통해 데이터를 전송하는 중앙 집중형 방식
Message Bus (ESB 방식)	애플리케이션 사이에 미들웨어를 두어 처리
Hybrid	Hub & Spoke(그룹내) + Message Bus(그룹간)

2) 인터페이스 구현

(1) 통신 기술 : AJAX (Javascript를 이용한 비동기 통신 기술)

- 데이터 포맷: JSON (데이터 객체를 속성-값의 쌍 형태로 표현), XML (마크업 언어)

(2) 보안 : 스니핑(수동적 공격)을 막기 위해 네트워크 트래픽에 대한 암호화 설정

=> 방식 : IPSec, SSL, S-HTTP 등

3) 구현 검증 도구

xUnit	Java(Junit), C++(CppUnit) 등 다양한 언어를 지원
STAF	다양한 환경을 지원
FitNess	웹기반
NTAF	FitNess + STAF
watir	Ruby 기반

3. 테스트

1) 애플리케이션 테스트 : 애플리케이션에 잠재되어 있는 결함을 찾아내는 행위

(1) 확인과 검증

- 확인 : 요구사항 만족 확인 (사용자 중심)
- 검증 : 명세서에 맞게 만들어졌는지 검증 (개발자 중심)

(2) 기본 원리

- 파레토법칙 : 오류의 80%는 20%의 모듈에서 발견된다 => 결함 집중

(3) 테스트 오라클: 결과를 판단하기 위해 테스트케이스에 대한 예상결과 계산, 확인

참오라클	모든 테스트 케이스의 입력값에 대한 기대값 제공
샘플링 오라클	특정 케이스에 대한 기대값 제공
추정 오라클	특정 케이스에 대한 기댓값 + 나머지는 추정으로 처리
일관성 검사 오라클	애플리케이션 변경 시, 변경 전과 후의 결과값이 동일한지 비교

2) 애플리케이션 테스트의 종류 : 테스트 기법과 개발 단계에 따른 분류

(1) 기법에 따른 구분

a. 화이트 박스 테스트 : 내부의 논리적인 경로를 테스트

- 기초 경로 검사 : 논리적 복잡성 측정을 위해, 모든 경로 검사
- 검증 기준 : 테스트 케이스가 적정한가 판단

문장 검증 기준	모든 구문이 한 번 이상 수행되었는가
분기 검증 기준	모든 조건문에 대해 조건이 참일때와 거짓일 때가 모두 수행되었는가
조건 검증 기준	조건문에 포함된 조건식의 결과가 참/거짓일 때가 모두 수행되었는가
분기/조건 기준	분기 검증 + 조건 검증

b. 블랙 박스 테스트 : 결과물이 정확한지 테스트

- 종류 : 동치분할 검사, 경계값 분석,

원인-효과 그래프 검사(효용성 높은 테스트 케이스 선정),

오류 예측 검사(감으로), 비교검사 (버전 여러개에 적용해봄)

동치 분할 검사	타당한 입력 자료와 타당하지 않은 입력자료를 균등하게 함
경계값 분석	입력 조건의 경계값을 테스트 케이스로 선정

(2) 개발 단계에 따른 분류

단계	설명
단위 테스트	- 모듈이나 컴포넌트에 초점을 맞춰 테스트
통합 테스트	- 모듈간 상호작용 오류 검사 (1) 하향식 : 상위 모듈 -> 하위 모듈 통합 (넓이우선/깊이우선 등) => 하위 모듈을 대체하는 stub 활용 (2) 상향식 : 하위모듈 -> 상위 모듈 통합 => 상위 모듈을 대체하는 driver 활용
시스템 테스트	- 실제 사용환경과 유사한 환경에서 완벽하게 수행되는가
인수 테스트	- 사용자의 요구사항을 충족하는지 (1) 알파테스트 : 개발자의 장소에서, 개발자 앞에서 (2) 베타테스트 : 개발자 없이 고객의 사용 환경에서

3) 테스트 케이스 생성 도구 (테스트 케이스 자동화 도구의 일종)

: 자료흐름도 작성, 기능테스트(가능한 모든 입력 작성), 입력 도메인 분석, 랜덤 테스트

II	패키징
----	-----

1. 소프트웨어 패키징

1) 정의: 모듈별로 생성한(모듈화한) 실행 파일을 묶어 배포용 설치파일을 만드는 것

2) 패키징 시 고려사항 => 사용자 중심!

(1) 사용자에게 필요한 최소 환경 정의

(2) 내부 콘텐츠에 대한 암호화 및 보안

(3) 다른 여러 콘텐츠 및 단말기 간의 DRM(디지털 저작권 관리) 연동 고려

3) DRM(디지털 저작권 관리)

(1) 디지털 콘텐츠와 디바이스의 사용을 제한하기 위한 접근 제어 기술

=> 사용 권한, 과금, 유통 관리

(2) 구성 요소

- 클리어링 하우스 : 권한정책 및 라이선스 관리

- 콘텐츠 제공자, 분배자(유통), 소비자

- 패키지, DRM 컨트롤러, 보안 컨테이너

(3) 관리 기술

암호화	콘텐츠 및 라이선스를 암호화하고 전자 서명을 할 수 있는 기술
키 관리	콘텐츠를 암호화한 키에 대한 저장 및 분배
암호화 파일 생성	콘텐츠를 암호화된 콘텐츠로 생성
식별 기술	콘텐츠에 대한 식별 체계 표현
저작권 표현	라이선스의 내용 표현
정책 관리	라이선스 발급 및 사용에 대한 정책 및 관리
크랙 방지	무단으로 사용하는 것을 방지
인증	사용자 인증

--<데이터 베이스>-----

III	관계형 데이터 베이스
-----	-------------

1. 관계형 데이터 베이스

1) Relation = 릴레이션 스키마 (구조) + 릴레이션 인스턴스 (데이터)

(1) 정의 : 데이터를 표의 형태로 구현한 것

-> 동일한 튜플이 존재하지 않으며, 순서가 없음

(2) 튜플 : 릴레이션을 구성하는 각각의 행, (= 속성의 모임, 레코드)

- 카디널리티 : 튜플의 수

(3) 속성 : 데이터 항목 또는 데이터 필드

- 디그리(차수) : 속성의 수

(4) 도메인 : 하나의 속성이 취할 수 있는 원자값들의 집합

2) key

(1) 후보키 : 튜플을 유일하게 식별하기위해 사용하는 속성들의 부분집합 (기본키 후보)

=> 유일성, 최소성 (꼭 필요한 속성으로만 구성)

(2) 기본키 : 후보키 중에서 특별히 선정된 주키

(3) 대체키 : 기본키를 제외한 나머지 후보키

(4) 슈퍼키 : 한 릴레이션 내에 있는 속성들의 집합 (유일성O 최소성X)

(5) 외래키 : 다른 릴레이션의 기본키를 참조하는 속성

3) 무결성

(1) 개체 무결성 : 기본키를 구성하는 어떤 속성도 Null이나 중복값을 가질 수 없음

(2) 참조 무결성 : 외래키의 값은 null이거나 참조릴레이션의 기본키 값과 동일해야함

4) 시스템 카탈로그 : 시스템 자체에 관련 있는 객체에 관한 정보를 포함함 (데이터 사전)

=> 메타데이터가 저장됨 (DBMS 스스로 생성, 유지하며 사용자는 조회만 가능)

2. 관계대수

1) 개요

- 관계형DB에서 원하는 정보를 검색하기 위한 유도 방법을 기술하는 절차적 언어

순수 관계 연산자 + 일반 집합 연산자

- 관계해석(원하는 정보가 무엇이라는것만 정의하는 비절차적 방법)과 구분이 필요함

기호	구성요소	설명
\forall	전칭 정량자	가능한 모든 튜플에 대하여 (For All)

2) 순수 관계 연산자

	개요	기호
Select	조건을 만족하는 튜플의 부분집합을 구하여 새로운 릴레이션을 만듦	σ
Project	속성리스트에 제시된 속성값만을 추출하여 새로운 릴레이션을 만듦(중복x)	π
Join	공통 속성을 중심으로 두 개의 릴레이션을 하나로 합침	\bowtie
Division	두 개의 릴레이션(R, S)에서 L가 가진 속성을 제외한 속성만을 구함	\div

3) 일반 집합 연산자

	개요	기호
합집합 (Union)	릴레이션 R 또는 S에 존재하는 튜플 (중복 제거)	\cup
교집합 (intersection)	릴레이션 R과 S에 동시에 존재하는 튜플의 집합	\cap
차집합 (different)	릴레이션 R에는 존재하고 S에는 없는 튜플의 집합	$-$
교차곱 (Cartesian product)	두 릴레이션에 있는 튜플들의 순서쌍	\times

III	데이터 베이스 설계
-----	------------

1. 데이터베이스 설계

1) 도구 : 데이터 모델

(1) 데이터 모델이 표시할 요소

- 구조(Structure) : 개체간의 관계
- 연산(Operation) : 실제 데이터를 처리하는 작업에 대한 명세
- 제약조건(Constraint)

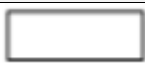



2) 과정 : 요구조건 분석 -> 개념적 설계 -> 논리적 설계 -> 물리적 설계 -> 구현

(1) 개념적 설계: 개념적 데이터 모델(DBMS에 독립) + 트랜잭션 모델링=>ER다이아그램

(2) 논리적 설계 : 사용할 DBMS에 맞는 논리적 데이터 모델 설계

(3) 물리적 설계 : 물리적 저장 장치에 저장할 수 있도록 물리적 데이터 모델로 변환

2. 개념적 모델링 - E-R 다이어그램 (Entity-Relationship)

기호	의미
	개체 (Entity)
	관계 (Relationship)
	속성 (Attribute)
	개체타입과 속성을 연결

3. 논리적 모델링 - 정규화 수행

1) 목적 : 데이터의 중복성을 최소화하고 일관성을 보장하여 이상(anomaly) 발생 방지

=> 이상 : 데이터의 불필요한 중복으로 인한 이상 현상 (삽입 이상, 삭제 이상, 갱신 이상)

2) 정규화 과정 (도부이걸다조)

단계	개요
1NF	도메인이 원자값
2NF	부분적 함수적 종속성 제거 => 완전 함수적 종속 관계 만족 ($X \rightarrow Y$)
3NF	이행적 함수 종속 제거 ($A \rightarrow B$ 이고 $B \rightarrow C$ 일 때 $A \rightarrow C$ 를 만족하는 관계 제거)
BCNF	결정자이면서 후보키가 아닌 것 제거
4NF	다치 종속 제거
5NF	조인 종속성 이용

3) 반정규화 : 정규화된 엔티티, 속성, 관계를 중복, 통합, 분리(의도적인 정규화 원칙 위배)

4. 물리적 모델링

1) 트랜잭션 분석

- (1) 트랜잭션 : 하나의 논리적 기능을 수행하기 위한 작업의 단위 (DB 상태 변화)
- (2) 트랜잭션 상태 : 활동, 실패, 철회(롤백 수행), 부분완료(커밋 직전), 완료(커밋)
- (3) 분석 방법 : CRUD 분석 (Create, Read, Update, Delete)
- (4) 트랜잭션의 특성 (ACID)

Atomicity(원자성)	모두 반영되거나(Commit) 전혀 반영되지 않고 복구(Rollback)
Consistency (일관성)	시스템이 가진 고정요소는 트랜잭션 수행전과 수행후가 동일
Isolation (독립성)	하나의 트랜잭션 실행 중에 다른 트랜잭션이 끼어들 수 없음
Durability (영속성)	완료된 트랜잭션 결과는 영구적으로 반영

2) 뷰(View) 설계 : 물리적으로 존재하지 않는 가상테이블 설계 (변경 불가, 독립적 인덱스 x)

3) 파티션 설계: 범위분할, 해시분할, 조합분할(범위+해시), 목록분할, 라운드로빈분할(균일배분)

4) 분산 데이터베이스 설계

(1) 분산 데이터베이스 : 논리적으로는 하나, 물리적으로는 분산

(2) 구성요소 : 분산 처리기, 분산 데이터베이스, 통신 네트워크

(3) 분산 데이터베이스의 모교

위치 투명성	논리적 명칭만으로 액세스 (실제위치 몰라도 됨)
중복 투명성	동일 데이터가 여러곳에 중복되어 있더라도 마치 하나만 존재하듯이 사용
병행 투명성	분산 DB관련 다수의 트랜잭션이 동시 실현되어도 결과에 영향x
장애 투명성	장애에도 불구하고 트랜잭션을 정확하게 처리

5. 접근 통제 : 보안을 위해 데이터가 저장된 객체와 사용하려는 주체 사이의 정보 흐름 제어

임의 접근통제(DAC)	신원에 따라 접근 권한 부여
강제 접근통제(MAC)	보안 레벨을 비교하여 접근 권한 부여
역할기반 접근통제 (RBAC)	개인의 역할 (직무, 직책 등)에 따라 접근 통제

--<언어>-----

IV	프로그래밍 언어의 기초
-----------	--------------

1. 데이터 타입

(1) C언어와 자바

자료형	C/C++		Java	
	키워드	크기	키워드	크기
논리형	BOOL	4byte	boolean	1bit/1byte
자료형	char	1byte	char	2byte
정수형			byte	1byte
	short	2byte	short	2byte
	int	4byte	int	4byte
	long	4byte	long	8byte
실수형	float	4byte	float	4byte
	double	8byte	double	8byte

2. 변수

(1) 정의 : 데이터를 저장할 수 있는 이름이 부여된 기억 장소

(2) 변수 이름 규칙

- 첫글자는 영문자나 _로 시작(숫자X)
- 공백 / 특수문자/예약어 사용 불가 및 대소문자 구분

3. 연산자

1) 산술 연산자

연산자	의미	비고
+	덧셈	
-	뺄셈	
*	곱셈	
/	나눗셈	
%	나머지	
++	증가 연산자	• 전치 : 변수 앞에 증감 연산자가 오는 형태로 먼저 변수의 값을 증감시킨 후 변수를 연산에 사용한다(++a, --a). • 후치 : 변수 뒤에 증감 연산자가 오는 형태로 먼저 변수를 연산에 사용한 후 변수의 값을 증감시킨다(a++, a--).
--	감소 연산자	

2) 비트 연산자 (2진수로 연산)□

연산자	의미	비고
&	and	모든 비트가 1일 때만 1
^	xor	모든 비트가 같으면 0, 하나라도 다르면 1
*	or	모든 비트 중 한 비트라도 1이면 1
~	not	각 비트의 부정, 0이면 1, 1이면 0

3) 논리 연산자 (0은 false, 나머지는 true)

연산자	의미	비고
!	not	부정
&&	and	모두 참이면 참
	or	하나라도 참이면 참

4) 조건 연산자

조건 ? 수식1 : 수식2 '조건'의 수식이 참이면 '수식1'을, 거짓이면 '수식2'를 실행한다.

5) 연산자 우선 순위

대분류	중분류	연산자	결합규칙	우선 순위
단항 연산자	단항 연산자	!(논리 not) ~ (비트 not) ++ (증가) -- (감소) sizeof (기타)	←	높음 ↑
이항 연산자	산술 연산자	* / % (나눗셈) + -	→	
	시프트 연산자	<< >>		
	관계 연산자	< <= > >=		
	비트 연산자	& (비트 and) ^ (비트 xor) (비트 or)		
	논리 연산자	&& (논리 and) (논리 or)		
삼항 연산자	조건 연산자	? :	→	
대입 연산자	대입 연산자	= += -= *= /= %= <= >= 등	←	
순서 연산자	순서 연산자	,	→	

IV	C언어와 자바 (둘이 비슷함)
-----------	------------------

1. 데이터 입출력

(1) C언어 : 입력 - scanf(서식문자열, &변수) / 출력 - printf(서식문자열, 변수)

(2) JAVA : 입력 - scanner.next() / 출력 - System.out.printf(서식문자열, 변수)

(3) 주요 서식 문자열 : %d(정수), %c(문자), %s(문자열), %f(실수)

2. 반복문

1) for 문

for(식1; 식2; 식3)

실행할 문장;

- for는 반복문을 의미하는 예약어로 그대로 입력한다.
- 식1 : 초기값을 지정할 수식을 입력한다.
- 식2 : 최종값을 지정할 수식을 입력한다.
- 식3 : 증가값으로 사용할 수식을 입력한다.

식2가 참일 동안 실행할 문장을 입력한다. 실행할 문장이 두 문장 이상일 경우 { }를 입력하고 그 사이에 처리할 문장들을 입력한다.

2) while 문

while(조건)

- while은 반복문에 사용되는 예약어로 그대로 입력한다.
- (조건) : 참이나 거짓을 결과로 갖는 수식을 '조건'에 입력한다. 참()을 직접 입력할 수도 있다.

실행할 문장;

조건이 참인 동안 실행할 문장을 입력한다. 문장이 두 문장 이상인 경우 { }를 입력하고 그 사이에 처리할 문장들을 입력한다.

3) do ~ while 문

do

do는 do~while문에 사용되는 예약어로, do~while의 시작 부분에 그대로 입력한다.

실행할 문장;

조건이 참인 동안 실행할 문장을 입력한다. 문장이 두 문장 이상인 경우 { }를 입력하고 그 사이에 실행할 문장들을 입력한다.

while(조건);

- while은 do~while문에 사용되는 예약어로, do~while의 끝 부분에 그대로 입력한다.
- (조건) : 참이나 거짓을 결과로 갖는 수식을 '조건'에 입력한다. 참()을 직접 입력할 수도 있다.

4) break, continue

- break : 반복문 종료

- continue : continue 이후를 수행하지 않고 다음 반복 수행

3. 배열

1) 특징 : 인덱스는 0부터 시작

2) 1차원 배열 : 변수를 일직선상의 개념으로 조합

자료형 변수명[개수];

- 자료형 : 배열에 저장할 자료의 형을 지정한다.
- 변수명 : 사용할 배열의 이름으로 사용자가 임의로 지정한다.
- 개수 : 배열의 크기를 지정하는 것으로 생략할 수 있다.

3) 2차원 배열 : 변수를 행과 열로 조합

자료형 변수명[행개수][열개수]

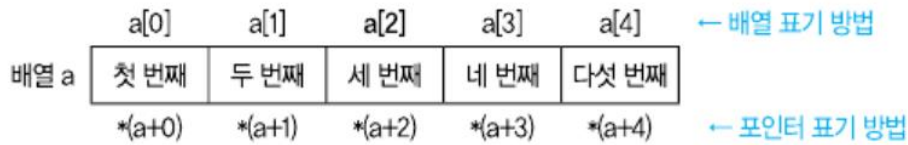
- 자료형 : 배열에 저장할 자료의 형을 지정한다.
- 변수명 : 사용할 배열의 이름으로 사용자가 임의로 지정한다.
- 행개수 : 배열의 행 크기를 지정한다.
- 열개수 : 배열의 열 크기를 지정한다.

4) 배열 형태의 문자열 변수 (C언어) : 마지막은 널문자('w0')

char 배열이름[크기] = "문자열"

4. 포인터 (변수의 주소, *변수) - c언어에서 가능

1) 배열을 포인터 변수에 저장한 후 포인터를 이용해 배열의 요소에 접근 가능



5. 라이브러리 : 자주 사용하는 함수나 데이터의 집합체

- C언어에서 헤더 파일을 사용하려면 '#include <stdio.h>'와 같이 include문을 이용해 선언한 후 사용해야 한다.

헤더 파일	기능
23.5, 23.2 stdio.h	<ul style="list-style-type: none"> 데이터의 입 · 출력에 사용되는 기능들을 제공한다. 주요 함수 : printf, scanf, fprintf, fscanf, fclose, fopen 등
23.2 math.h	<ul style="list-style-type: none"> 수학 함수들을 제공한다. 주요 함수 : sqrt, pow, abs 등
23.2 string.h	<ul style="list-style-type: none"> 문자열 처리에 사용되는 기능들을 제공한다. 주요 함수 : strlen, strcpy, strcmp 등
22.7, 21.5, 21.3 stdlib.h	<ul style="list-style-type: none"> 자료형 변환, 난수 발생, 메모리 할당에 사용되는 기능들을 제공한다. 주요 함수 : atoi, atof, srand, rand, malloc, free 등
23.2 time.h	<ul style="list-style-type: none"> 시간 처리에 사용되는 기능들을 제공한다. 주요 함수 : time, clock 등

IV	Python
----	--------

1. 데이터 입출력 : input(), print()

2. 자료구조

- 리스트 : C언어의 배열과 비슷하지만 크기 지정X
- 딕셔너리 : (key : value) 로 묶어서 저장 -> 데이터에는 key로 접근
- slice : 문자열이나 리스트에서 일부를 잘라서 반환

객체명[초기위치:최종위치]

'초기위치'에서 '최종위치'-1까지의 요소들을 가져온다.

객체명[초기위치:최종위치:증가값]

- '초기위치'에서 '최종위치'-1까지 '증가값'만큼 증가하면서 해당 위치의 요소들을 가져온다.
- '증가값'이 음수인 경우 '초기위치'에서 '최종위치'+1까지 '증가값' 만큼 감소하면서 해당 위치의 요소들을 가져온다.

3. 조건문

if 조건1:	
실행할 문장1	조건1이 참일 경우 실행할 문장을 적는다.
elif 조건2:	
실행할 문장2	조건2가 참일 경우 실행할 문장을 적는다.
elif 조건3:	
실행할 문장3	조건3이 참일 경우 실행할 문장을 적는다.
⋮	
else:	
실행할 문장4	앞의 조건이 모두 거짓일 경우 실행할 문장을 적는다.

4. 반복문

1) for문

for 변수 in range(최종값):	0에서 '최종값'-1까지 연속된 숫자를 순서대로 변수에 저장하며 '실행할 문장'을 반복 수행한다.
실행할 문장	반복 수행할 문장을 적는다.

for 변수 in 리스트	리스트의 0번째 요소에서 마지막 요소까지 순서대로 변수에 저장하며 실행할 문장을 반복 수행한다.
실행할 문장	반복 수행할 문장을 적는다.

2) while문

while 조건:	• while은 예약어로, 그대로 입력한다.
	• 참이나 거짓을 결과로 갖는 수식을 조건에 입력한다. 참(1 또는 True)*을 직접 입력할 수도 있다.
실행할 문장	조건이 참인 동안 반복 수행할 문장을 적는다.

5. 함수 선언

def 메소드명(self, 인수):	• def는 메소드를 정의하는 예약어로, 그대로 입력하고, 메소드명은 사용자가 임의로 지정한다.
	• self는 메소드에서 자기 클래스에 속한 변수에 접근할 때 사용하는 명칭으로, 일반적으로 self를 사용하지만 사용자가 임의로 지정해도 된다.
	• '인수'는 메소드를 호출하는 곳에서 보낸 값을 저장할 변수로, 사용자가 임의로 지정한다.

III	SQL : DB를 위한 언어
-----	-----------------

1. SQL의 구분

DDL (Data Define)	논리적, 물리적 구조 정의 => CREATE (테이블 생성), ALTER(테이블 수정), DROP (테이블 삭제)
DML (Data Manipulation)	데이터 처리 => SELECT, INSERT, DELETE, UPDATE
DCL (Data Control)	데이터 관리 => COMMIT (반영), ROLLBACK(이전 상태로 되돌림), GRANT(권한 부여), REVOKE (권한 부여 취소)

2. DML

- (1) INSERT INTO ~ VALUES (삽입)
- (2) DELETE FROM (삭제) : 구조가 아닌 데이터 삭제!
- (3) UPDATE ~ SET ~ (갱신)
- (4) SELECT 형식 ([] : 필수 아님)

SELECT (속성명)

FROM 테이블명

[WHERE 조건]

[ORDER BY 속성명 [ASC | DESC]]

--<기타>-----

ALL	소프트웨어 만드려면 이 정도는 알고 있어야 한다!
-----	-----------------------------

1. 미들웨어 : 서로 다른 기종간의 원만한 통신을 위한 소프트웨어

1) 종류

(1) DB : 클라이언트와 원격의 데이터베이스를 연결하는 미들웨어

(2) PRC(Remote Procedure Call) : 원격 프로시저를 로컬 프로시저처럼 호출

(3) MOM(Message Oriented Middleware)

: 비동기형(응답을 기다리지 않고 작업처리) 메시지를 전달하는 방식

=> 서로 다른 기종의 분산 데이터 시스템의 데이터 동기를 위해 사용

(4) TP-Monitor(Transaction Processing) : 트랜잭션 처리, 감시 (빠른 속도)

* 트랜잭션 : 한 번에 실행하는 여러 작업들

(5) ORB(Object Request Broker) : 객체 지향 미들웨어 - CORBA 구현

(6) WAS(Web Application Server) : 동적 콘텐츠를 처리하기 위한 미들웨어

2. 스토리지 : 서버와 저장 장치를 연결하는 기술

DAS	전용 케이블로 직접 연결 -> 서버를 통해야만 파일에 접근 가능
NAS	네트워크를 통해 연결
SAN	광채널 스위치를 이용하여 전용 네트워크 구성

IV	운영 체제
----	-------

1. 운영체제의 역할

1) 기억장치 관리 => 가상 기억 장치

(1) 가상기억장치: 용량이 작은 주기억장치를 마치 큰 용량을 가진 것처럼 사용하는 기법

(2) 페이지징 기법 : 프로그램을 일정한 크기로 나뉘(페이지) 주기억장치(페이지프레임)에 할당

=> 내부 단편화 발생 가능

(3) 페이지 교체 알고리즘 :

- 주기억장치에 필요한 페이지가 없을 때(Page Fault) 모든 페이지 프레임이 사용중이라면 어떤 페이지를 선택하여 교체할지 결정하는 기법

FIFO (First In, First Out)	가장 먼저 들어온 페이지를 교체
LRU (Least Recently Used)	가장 오랫동안 사용하지 않은 페이지를 교체

- 스래싱 (Thrashing) 발생 가능 : 프로세스 처리시간 < 페이지 교체 시간

(4) 세그멘테이션 기법 : 프로그램을 다양한 크기의 논리적인 단위(세그먼트)로 나눔

=> 외부 단편화 발생 가능

2) 프로세스 관리 (= 자원할당)

(1) 프로세스 = 실행중인 프로그램 (스레드 : 프로세스 내에서의 작업 단위)

: 디스패치 : 프로세스가 준비상태에서 프로세서가 배당되어 실행상태로 변화하는 것

(2) 스케줄링 알고리즘 (프로세스에게 자원을 할당하는 작업)

HRN (Highest Response-ratio Next)	우선순위 계산식 = $\frac{\text{대기 시간} + \text{서비스 시간}}{\text{서비스 시간}}$
--------------------------------------	-----------------------------------------------------------------

IV	네트워크
----	------

1. 인터넷

1) IP주소 : 인터넷에 연결된 컴퓨터 자원을 구분하기 위한 고유한 주소(예 - 192.168.0.1)

=> A클래스 ~ E클래스로 구성. C클래스의 경우 192~223으로 시작됨

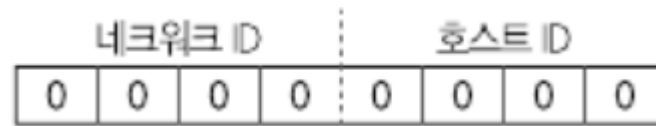
(1) IPv6 : IPv4(32비트)의 주소 부족 문제를 해결하기 위해 개발됨 (128비트)

- 특징 : 패킷 크기 제한 X, 패킷의 헤더는 40 Octet(Byte)로 고정

- 주소 체계 : 유니캐스트, 멀티캐스트, 애니캐스트

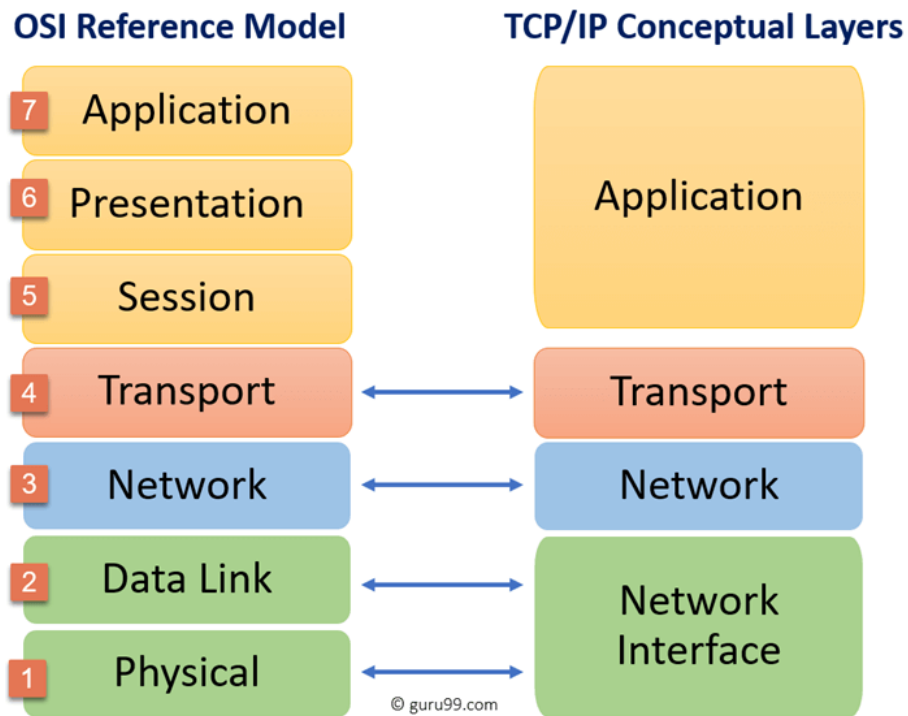
2) 서브네팅 : IP 주소 중 네트워크 주소와 호스트 주소를 구분하기 위한 비트 (for 분할사용)

- 마지막 8비트를 네트워크와 호스트ID로 구분, 네트워크에는 서브넷의 개수만큼 할당



- FLSM 방식 : 고정된 크기의 주소 할당
- ip subnet-zero : 서브넷 부분이 모두 0인 네트워크도 IP주소로 사용함

2. OSI 7계층 (물데네전세표응) : 다른 시스템간의 원활한 통신을 위한 통신규약



1) 계층별 개요

- (1) 물리계층 : 전송에 필요한 장치 간의 실제 접속과 절단 등 물리적 특성 정의
- (2) 데이터 링크 계층 : 링크 설정, 유지 및 종료 담당 (프레임 전송 에러 제어)
- (3) 네트워크 계층 : 네트워크 연결 관리 및 데이터의 교환 및 중계 담당
 - 패킷을 최종 목적지까지 전달 => 경로설정 (라우팅)
- (4) 전송계층 : 종단 시스템(End-to-End)간의 균일한 데이터 전송 서비스 제공

2) TCP/IP : 데이터 교환을 위한 표준 프로토콜

(1) 네트워크 액세스 계층 (물리+데이터링크) 계층의 주요 프로토콜

A. Ethernet : CSMA/CD(전송 매체 접속 제어(MAC)) 방식의 LAN

(2) 인터넷 계층(네트워크 계층)의 주요 프로토콜

IP	- 비연결형 서비스 제공 - Best Effort 원칙에 따른 전송 기능을 제공 - 주소 지정, 경로 선택 기능 제공 - 헤더의 길이는 20 ~ 60Byte
ICMP	오류 처리와 전송 경로 변경등을 위한 제어메시지 관리 (헤더 : 8byte)
ARP	Address Resolution Protocol, 주소 분석 프로토콜 : 호스트의 IP주소를 물리적 주소로 변환 (MAC Address)
RARP	물리적 주소를 IP주소로 변환 (Reverse ARP)

(3) 전송 계층의 주요 프로토콜

TCP	- 신뢰성/안정성 있는 양방향 연결형 서비스 제공 - 순서제어, 오류제어, 흐름 제어 및 스트림 전송 기능 제공
UDP	- 흐름이나 순서제어가 없어 전송 속도가 빠름

(4) 응용 계층의 주요 프로토콜

A. MQTT : 발행-구독 기반의 메시징 프로토콜로, IoT 환경에서 자주 사용

-----<정보시스템 구축 관리>-----

V	소프트웨어 개발 방법론 활용
---	-----------------

1. SW 공학의 발전 추세

1) 소프트웨어 재공학: 기존 시스템을 이용하여 더 나은 시스템 구축

- (1) 이식(Migration) : 기존 소프트웨어를 다른 환경에서 사용할 수 있도록 변환하는 기술
- (2) 역공학(Reverse Engineering) : 기존 소프트웨어를 분석하여 다시 만들어내는 활동
- (3) 소프트웨어 재사용방법 : 합성 중심 (모듈화를 통한 소프트웨어 완성)

2) CASE : 개발 과정을 컴퓨터와 전용 소프트웨어를 사용하여 자동화

(1) 원천 기술 : 구조적 기법, 프로토 타이핑, 자동 프로그래밍, 정보 저장소, 분산처리

2. 비용 산정 기법 : 상향식 비용 산정

1) LOC 기법 (source Line Of Code) : 비관치, 낙관치, 기대치를 측정하여 예측치를 구함

- 생산성 : 원시라인 코드수 / 노력(=투입인원 x 개발기간)

2) 수학적 산정 : COCOMO, Putnam, 기능점수(FP)

(1) COCOMO 모형 : 원시 프로그램의 규모인 LOC에 의한 비용 산정 기법

- 유형 : Organic(5만라인 이하), Semi-Detached(30만 이하), Embedded(30만 이상)

=> 같은 규모의 소프트웨어도 유형에 따라 비용이 다르게 산정됨

3. 프로젝트 일정 계획 도구 : PERT/CPM, 간트 차트

1) CPM : 소요기간을 예측하는 기법 (임계경로 = 최장 경로)

2) 간트 차트(시간선 차트) : 각 작업 일정과 기간을 막대도표를 이용해서 나타냄

4. 프로젝트 개발 프레임워크

1) 프레임 워크 : 개발해야하는 애플리케이션의 일부분이 이미 구현된 뼈대, 골조

: 모듈화, 재사용성, 확장, 제어의 역흐름(프레임워크가 흐름을 제어)

V	네트워크 구축
---	---------

1. 네트워크 구축

1) 네트워크 설치 구조

버스형	<p>1. 한 개의 통신 회선에 여러 단말장치가 연결됨</p> <p>2. 매체 접근 제어 방식 : CSMA/CD : 전송로의 상태를 확인하여 충돌을 예방 (유선) * CSMA/CA (무선)</p>	
-----	-----------------------------------------------------------------------------------------------------------------------	--

2. 경로제어 : 최적의 패킷 교환 경로를 결정하는 기술

1) 경로제어 프로토콜(라우팅 프로토콜)

(1) RIP : 최대 홉(거치는 네트워크의 수)를 15로 제한 => 거리 벡터 라우팅 프로토콜

V	소프트웨어 개발 보안 구축
---	----------------

1. Secure SDLC : SDLC(소프트웨어 개발 생명 주기)에 보안 강화를 위한 프로세스 포함

1) 보안 요소

기밀성	인가된 사용자에게만 접근 허용
무결성	인가된 사용자만 수정 허용
가용성	인가 받은 사용자는 언제라도 사용 가능

2. 입력 데이터 검증

1) 보안 약점 : 크로스사이트 스크립팅 (XSS, 악의적인 스크립트 삽입)

3. 암호 알고리즘

1) 개인키 암호화(대칭 암호 기법) : 동일한 키로 데이터를 암호화, 복호화

2) 공개키 암호화(비대칭 암호 기법) : 암호화는 공개키로, 복호화는 비밀키로

3) 해시 : 임의의 입력데이터를 받아 고정된 길이의 해쉬값으로 변환

- Salt : 똑 같은 패스워드들이 다른 암호값으로 저장되도록 추가되는 값

V	시스템 보안 구축
---	-----------

1. 공격 유형

1) 서비스 거부 공격 : 대량의 데이터를 한 곳의 서버에 집중 전송하여 정상 기능 방해

(1) Ping of Death : 매우 큰 패킷을 전송하여 네트워크 마비

- (2) SMURFING : IP또는 ICMP의 특성을 악용하여 특정 사이트에 집중적으로 데이터 전송
- (3) Land : 패킷 전송시 송신 IP와 수신 IP를 모두 공격대상의 IP 주소로 하여 전송
- (4) SYN Flooding : 가상의 클라이언트로 위장하여 서버를 대기상태로 만듦

3) 정보 보안 침해 공격

- (1) Worm : 연속적으로 자신을 복제하여 시스템 부하를 높임
- (2) 키로거 공격 : 키보드 움직임을 탐지해 정보 탈취
- (3) 랜섬웨어 : 문서나 파일을 암호화해 열지 못하게 함
- (4) 백도어 : 시스템 설계자가 액세스 편의를 위해 보안을 제거해둔 비밀 통로
 - A. 탐지 방법 : 무결성 검사, 열린 포트 확인, 로그분석, SetUID 검사
- (5) 트로이 목마 : 정상적인 프로그램으로 위장하다가 대상 프로그램이 동작할 때 활성화
- (6) 파밍 : 도메인을 탈취해 가짜 사이트로 접속하게 하여 정보 탈취

2. 서버 인증

- 1) 인증 : 자신의 신원을 시스템에 증명
- 2) 인가 : 클라이언트로부터 요청된 정보에 대한 사용 권한을 부여

V	네트워크 / SW / 보안 / DB 신기술
----------	-------------------------

1. 네트워크 관련 기술

1) 소프트웨어 정의 기술

- (1) 소프트웨어 정의 네트워킹 : 여러 사용자가 네트워크를 가상화하여 제어, 관리

2) IoT

- (1) 메시 네트워크 : 대규모 디바이스의 네트워크 생성에 최적화된 네트워크 기술
- (2) 올조인(AllJoyn) : 오픈 소스 기반의 IoT 플랫폼

3) 클라우드 컴퓨팅

(1) 클라우드 기반 HSM : 클라우드를 기반으로 암호화를 처리하는 보안기기

(2) 파스타(PaaS) : 개방형 클라우드 컴퓨팅 플랫폼

4) 기타

(1) SSO : 한 번의 로그인으로 개인이 가입한 모든 사이트를 이용할 수 있는 시스템

2. SW 관련 기술

1) 매시업 : 웹에서 제공하는 정보 및 서비스를 이용하여 새로운 것을 만들어 냄

2) 디지털 트윈 : 현실속의 사물을 소프트웨어로 가상화한 모델

3) 텐서플로 : 데이터 흐름 프로그래밍을 위한 오픈소스 sw라이브러리 (from 구글)

4) 도커 : 컨테이너 기술을 자동화하여 쉽게 사용할 수 있게 함

5) 스크래피(Scrapy) : 파이썬 기반의 웹 크롤링 프레임워크

6) 증발품 (Vaporware) : 배포 계획은 발표되었으나 실제로 배포되지 않고 있는 sw

3. 보안 관련 기술

1) 블록체인 : P2P 네트워크를 이용하여 정보를 네트워크 참여자의 장비에 분산 저장

2) 서비스형 블록체인(BaaS) : 블록체인 개발 환경을 클라우드로 서비스하는 개념

3) OWASP (오픈 웹 애플리케이션 보안 프로젝트)

4) 허니팟 : 비정상적인 접근의 탐지를 위해 의도적으로 설치해둔 취약점

5) TCP Wrapper : 외부 컴퓨터의 접속 인가 여부를 점검하여 접속을 허용 및 거부

4. DB 관련 기술/작업

1) 하둡 : 오픈 소스를 기반으로 한 분산 컴퓨팅 플랫폼 => 일반 pc로 대형 스토리지 생성

- sqoop : 하둡과 관계형 데이터베이스간에 데이터를 전송하는 도구

2) 맵리듀스 : 대용량 데이터를 분산처리하기 위함 map(분산 처리) + reduce(합침)

3) 데이터 마이닝 : 데이터 속에서 일정한 패턴을 찾아내는 빅데이터 분석 기술

4) 회복 : 트랜잭션 수행 중 DB가 손상되었을 때 정상상태로 복구하는 작업

- 즉각 갱신 : 트랜잭션이 데이터를 갱신하면 부분 완료 전이여도 즉시 반영 (로그 필요)

5) 병행제어 : 트랜잭션 병행 수행 시 일관성을 파괴하지 않도록 상호작용을 제어

- 로킹 : 주요 데이터의 액세스를 상호 배타적으로 하는 것 => 교착 상태 발생 가능

- 교착상태

발생 조건	해결 방법
상호 배제	회피기법 : 교착 상태가 발생하면 적절히 피함 ➔ 은행원 알고리즘 적용
점유와 대기	
비선점	
환형 대기	