

정보처리기사 실기 이론 요약 PDF

(코딩, SQL 미포함)

작성자
잡지식 정보 블로거 잇티제

Prologue

기존에 있던 책들로 공부하다 보니 어떤 특정한 이론을 바탕으로 하는 내용들이라 이야기처럼 유기적으로 연결돼있는 내용임에도 불구하고 시험에 나오는 부분만 서술하기 위해 각각의 이론들의 연결고리가 끊어져 중구난방으로 섞여 있는 느낌을 받았고 각각의 이론이 머릿속에서 뒤죽박죽 섞여 정리가 전혀 되지 않았습니다.

그래서 그냥 제가 한번 요약 pdf를 만들어보았습니다.

제가 요약 pdf를 만들면서 사용한 방법은 연결고리 암기법입니다. 그냥 무작정 외우는 것보다 각각의 내용에 연결고리를 만들면 더 외우기 쉬워지는 것을 이용했습니다.

물론 정보처리기사 시험은 한국사 시험처럼 흐름을 알아야 하는 시험이 아니고 굳이 연결고리를 통해서 공부하지 않아도 됩니다.

하지만 좀 더 빠르고 효율적으로 공부할 수는 있습니다

중구난방으로 돼 있는 이론을 10번 보는 것보다 유기적으로 단단히 연결돼있는 이론 1번을 공부하는 것이 훨씬 수월하고 효율적이기 때문입니다.

연결고리를 이용하기 위해서 저는 소프트웨어 공학을 이용했습니다. 소프트웨어 공학에 대한 내용을 바탕으로 개념들이 중구난방으로 흩어져있는 거라면 다시 그 개념들을 소프트웨어 공학에 넣으면 된다고 생각했기 때문이었습니다.

기존의 소프트웨어 공학에 대한 내용을 바탕으로 서술하되 정보처리기사에 출제될 만한 내용들만 소프트웨어 공학 순서에 맞게 기술하였습니다.

그 외에도 정보처리기사 범위에 있는 데이터베이스, 네트워크, 웹, 보안, 운영체제, 자료구조와 정렬 파트도 정리를 했습니다.

그냥 공부가 아니라 책 읽는다고 생각하고 편하게 한번 훑어보셔도 좋습니다.

기존의 자격증 책보다 훨씬 보기 편하고 개념들이 머릿속으로 잘 들어갈 것이라고 확신합니다.

제 요약 pdf를 2~3번 정도 읽으시고 실기 기출문제나 필기 기출문제를 다시 풀면서 잘 모르는 개념이 나오면 다시 한번 더 정리하시면 됩니다.

비전공자, 노베이스 분들도 이해할 수 있도록 최대한 쉽게 기술하였습니다.

간략하게 적으려다 보니 생략된 부분도 꽤 있으므로 부족한 부분은 시나공 교재를 참고해주면 될 것 같습니다

□ 실기시험

간단 목차

- I. 소프트웨어 공학 파트
- II. 데이터베이스(DB) 파트
- III. 네트워크 파트
- IV. 웹 파트
- V. 보안 파트
- VI. 운영체제 파트
- VII. 자료구조와 정렬 파트

과목명	활용 NCS 능력단위	NCS 세분류
정보처리 실무	요구사항 확인	응용SW엔지니어링
	데이터 입출력 구현	
	통합 구현	
	제품소프트웨어 패키징	
	서버프로그램 구현	
	인터페이스 구현	
	프로그래밍 언어 활용	
	응용 SW 기초 기술 활용	
	화면 설계	
	애플리케이션 테스트 관리	
	SQL 응용	DB엔지니어링
	소프트웨어 개발 보안 구축	보안엔지니어링

출처 : 큐넷

상세 목차(형광펜 칠한 부분은 출제 빈도 높은 파트)

I. 소프트웨어 공학 파트

- 소프트웨어의 뜻
- 소프트웨어 개발 프로세스(폭포수, 애자일 등등)
- 계획 단계
- 요구 사항 분석 단계(UML 다이어그램 등등)
- 설계 단계(디자인 패턴, 객체지향, 응집도/결합도 등등)
- 구현 단계
- 테스트 단계(블랙/화이트박스 테스트 등등)
- 품질
- 소프트웨어 패키징
- 화면 설계
- 기타 용어

II. 데이터베이스(DB) 파트

- 데이터베이스 뜻
- 데이터 모델링
- E-R모델
- 관계형 데이터베이스(후보키, 기본키 등등)
- 데이터베이스 설계
- 정규화
- 트랜잭션, 회복, 병행 제어

III. 네트워크 파트

- 네트워크의 뜻
- 정보 전송(비동기식, 동기식, 패킷교환)
- OSI 계층, TCP/IP 계층에서 각 계층별 특징과 프로토콜(캡슐화, TCP헤더)
- 데이터링크 계층 심화(흐름제어, 오류제어, HDLC)
- 네트워크 계층 심화(IP, 서브네팅)

IV. 웹 파트

- 웹 서비스

V. 보안 파트

- 해킹 용어(스푸핑, 시니핑 등등)
- 암호 알고리즘
- 보안 솔루션(SSO, SIEM 등등)

VI. 운영체제 파트

- 운영체제의 기능
- 페이징, 세그먼테이션 기법
- 페이지 교체 알고리즘
- 프로세스, 스케줄링

VII. 자료구조와 정렬 파트

- 자료구조와 정렬(이진 트리, 삽입/선택/버블 정렬 등등)

I. 소프트웨어 공학 파트

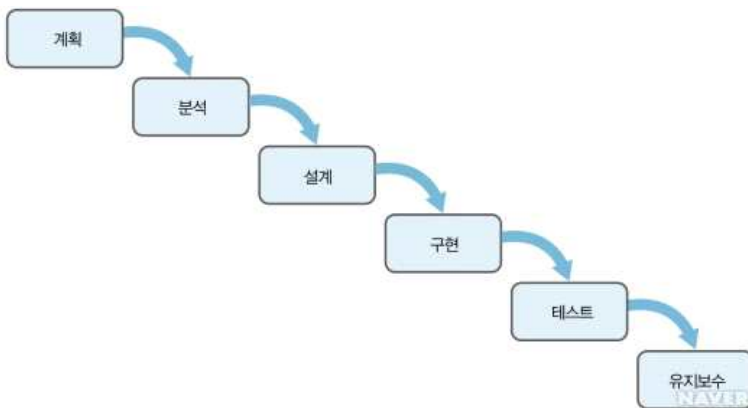
1) 소프트웨어란?

소프트웨어는 컴퓨터 시스템, 프로그램, 데이터에 의해 처리된 모든 정보를 말하는데 컴퓨터의 시스템을 구성하는 주요 요소 중 하나이다(컴퓨터=하드웨어+소프트웨어+펌웨어)

우리가 컴퓨터를 사용하는 목적은 소프트웨어를 이용하거나 처리하기 위함이며, 소프트웨어가 없는 컴퓨터(즉 하드웨어)는 그냥 빈 껍질이다.

이런 소프트웨어는 계획 단계를 시작으로 분석, 설계, 구현, 테스트, 유지보수 단계를 거쳐 개발된다. 이 단계를 소프트웨어 개발 생명주기라 하고, 소프트웨어 공학은 이 생명주기의 각 단계에서 필요한 이론, 방법, 도구 등을 배우는 학문이다.

소프트웨어 개발과정은 하나의 제품인 소프트웨어를 만들기 위해 계획 단계에서 유지보수 단계에 이르기까지 일어나는 일련의 과정을 말하고 이를 소프트웨어 공학에서는 소프트웨어 개발 생명주기라고 한다



소프트웨어 개발 생명주기(SDLC(Software Development Life Cycle))

출처 : 네이버 지식백과(쉽게 배우는 소프트웨어 공학)

간단 요약

- ①**계획 단계**에서 여행계획 짜듯이 비용, 기간 등 프로젝트를 수행하는 데 필요한 것에 대한 계획을 한다.
- ②**요구 분석 단계**에서 기존 시스템의 문제점을 파악하고 사용자 인터뷰를 통해 새로운 요구사항을 도출하여 수집하며 이 요구사항을 최적화된 상태로 정리한 후 특정 표현 도구를 사용하여 다이어그램 등으로 나타낸다.
- ③**설계 단계**에서 표현한 다이어그램을 가지고 코딩할 수 있는 수준으로 환경에 밀접하게 구체화한다.
- ④**구현 단계**에서 프로그램을 작성한다.
- ⑤**테스트 단계**에서 코딩이 완료된 후 제품을 출시하기 전에 여러 테스트 기법을 사용하여 오류를 찾아낸다.
- ⑥**유지보수 단계**에서 사용자가 소프트웨어를 사용하다가 추가, 변경 요구를 하면 이에 대해 적절한 조치를 해준다.

2) 소프트웨어 개발 프로세스

프로세스는 어떤 작업에 대한 수많은 반복과 시행착오를 통하여 얻은 방법이나 도구 등에 관한 지식을 같은 작업을 수행하는 다른 사람들에게 전달함으로써 시행착오를 줄이고 빠르게 적응하여 일을 할 수 있도록 가이드 역할을 한다고 볼 수 있다.

규모가 작은 소프트웨어는 개발할 때 굳이 분석, 설계, 코딩의 과정을 거치지 않고 바로 코딩을 한 후 필요할 때마다 수정해 사용할 수 있지만, 대규모 소프트웨어는 개발에 참여하는 개발자가 많고 기간도 길며 예산도 많이 필요하다. 따라서 개발 전에 여러 소프트웨어 프로세스 모델 중에서 가장 적합한 모델을 프로젝트의 표준으로 정하고 모든 참여자가 그 표준의 절차에 따라 개발해야 한다.

소프트웨어 프로세스 모델은 소프트웨어 개발 생명주기(SDLC)라고도 한다.

소프트웨어 프로세스 모델은 공장에서 제품을 생산하듯이 소프트웨어를 개발하도록 개발의 전 과정을 하나의 프로세스로 정의한다. 따라서 주어진 예산과 자원으로 개발하고 관리하는 방법을 구체적으로 정의한다. 이런 소프트웨어 프로세스 모델은 프로젝트에 대한 전체적인 기본 골격을 세워주며 그에 따른 일정 계획을 수립할 수 있고, 개발 비용 산정뿐 아니라 여러 자원을 산정하고 분배할 수 있다. 또한 참여자 간의 의사소통 기준을 정할 수 있고 용어의 표준화를 가능케 할 뿐만 아니라 개발 진행 상황도 명확히 파악할 수 있다.

대표적인 소프트웨어 프로세스 모델(생명주기 모형)로는 폭포수, 프로토타입, 나선형, 애자일 모형이 있다.

㉠폭포수 모형(Waterfall Model)

이전 단계로 돌아갈 수 없다는 전제하에 각 단계를 확실히 매듭짓고 그 결과를 철저하게 검토하여 승인 과정을 거친 후에 다음 단계를 진행하는 개발 방법론

- ▶가장 오래되고 가장 폭넓게 사용된 전통적인 모형
- ▶고전적 생명주기 모형

㉡프로토타입 모형(Prototype Model, 원형 모형)

사용자의 요구사항을 파악하기 위해 실제 개발될 소프트웨어에 대한 견본품(Prototype)을 만들어 최종 결과물을 예측하는 모형

㉢나선형 모형(Spiral Model, 점진적 모형)

나선을 따라 돌 듯이 여러 번의 소프트웨어 개발 과정을 거쳐 점진적으로 완벽한 최종 소프트웨어를 개발하는 모형

- ▶보ehm(Boehm)이 제안
- ▶폭포수 모형과 프로토타입 모형의 장점에 위험 분석 기능을 추가한 모형
- ▶누락되거나 추가된 요구사항을 첨가 가능
- ▶유지보수 과정 불필요

㉣애자일 모형(Agile Model)

고객의 요구사항 변화에 유연하게 대응할 수 있도록 일정한 주기를 반복하면서 개발하는 모형

- ▶어느 특정 개발 방법론이 아니라 좋은 것을 빠르고 낭비 없게 만들기 위해 고객과의 소통에 초점을 맞춘 방법론
- ▶폭포수 모형과 대조적
- ▶대표적인 개발 모형(스크럼(Scrum), Xp(eXtreme Programming), 칸반, Lean, 기능 중심 개발(FDD))

◆스크럼(Scrum) 기법

팀이 중심이 되어 개발의 효율성을 높이는 기법

▶팀원 스스로가 스크럼 팀을 구성하고 개발 작업에 관한 모든 것을 스스로 해결할 수 있어야 한다

◆Xp(eXtreme Programming) 기법

수시로 발생하는 고객의 요구사항에 유연하게 대응하기 위해 고객의 참여와 개발과정의 반복을 극대화하여 개발 생산성을 향상시키는 방법

▶XP의 5가지 핵심 가치

의사소통(Communication), 단순성(Simplicity), 용기(Courage), 존중(Respect), 피드백(Feedback)

▶XP의 주요 실천 방법

실천 방법	내용
Pair Programming (짝 프로그래밍)	다른 사람과 함께 프로그래밍을 수행함으로써 개발에 대한 책임을 공동으로 나눠 갖는 환경을 조성함
Collective Ownership (공동 코드 소유)	개발 코드에 대한 권한과 책임을 공동으로 소유함
Test-Driven Development (테스트 주도 개발)	개발자가 실제 코드를 작성하기 전에 테스트 케이스를 먼저 작성하므로 자신이 무엇을 해야 할지를 정확히 파악함
Whole Team (전체 팀)	개발에 참여하는 모든 구성원들은 각자 자신의 역할이 있고 그 역할에 대한 책임을 가져야함
Continuous Integration (계속적인 통합)	모듈 단위로 나눠서 개발된 코드들을 하나의 작업이 마무리 될 때마다 지속적으로 통합됨
Refactoring (리팩토링)	프로그램 기능의 변경 없이 시스템을 재구성함
Small Releases (소규모 릴리즈)	릴리즈 기간을 짧게 반복함으로써 고객의 요구 변화에 신속히 대응할 수 있음

2-1) 계획 단계

소프트웨어를 개발할 때 비용, 기간, 자원 등을 잘 고려해 계획을 세워야 원하는 결과를 얻을 수 있다. 계획을 잘 세우지 않고 수행하는 소프트웨어 개발은 일정 지연, 비용 초과, 품질 저하라는 결과로 이어지기 쉽고 그로 인해 유지보수 비용도 더욱 증가하게 된다.

현재 상황과 구현될 시스템의 목표 및 제약조건 등을 포함해 무엇을 개발할 것인지 명확히 정의하고 개발 범위를 결정한다. 개발할 시스템을 정의하고 신규 시스템 실현 방안을 모색하면서 투자 효율성이 얼마나 높은지 시장성은 얼마나 큰지 등을 검토해야 한다. 그리고 사용자가 원하는 수준으로 개발하기 위해 기술적인 어려움은 없는지 개발과정에서 사용하는 프로그램이나 도구가 소유권 등의 법적인 문제는 없는지 면밀히 검토해 본다.

◆소프트웨어 비용 산정

개발에 소요되는 인원, 자원, 기간 등으로 소프트웨어의 규모를 확인하여 개발 계획 수립에 필요한 비용을 산정하는 것

소프트웨어 개발은 가전제품 생산이나 건축 공사와 달리 사람(개발자)이 중심이 되고 사람에 매우 의존적이라 비용 결정 요소(ex) 개발자 능력(초급인지 고급인지), 개발기간 등등)에 따라 천차만별이다 하지만, 누가 산출해도 거의 동일한 결과를 얻을 수 있는 개발 비용 산정 방법이 있다. 크게 하향식 비용 산정 기법과 상향식 산정 기법으로 나뉜다.

◆하향식 비용 산정 기법

과거의 유사한 경험을 바탕으로 전문 지식이 많은 개발자들이 참여한 회의를 통해 비용을 산정하는 비과학적인 방법

ex) 전문가 감정 기법, 델파이 기법

▶전문가 감정 기법

전문가 감정 기법조직 내에 있는 경험이 많은 두 명 이상의 전문가에게 비용 산정을 의뢰하는 기법

▶델파이 기법

전문가 감정 기법의 주관적인 편견을 보완하기 위해 많은 전문가의 의견을 종합하여 산정하는 기법

◆상향식 비용 산정 기법

프로젝트의 세부적인 작업 단위별로 비용을 산정한 후 집계하여 전체 비용을 산정하는 방법
ex) LOC(원시 코드 라인 수) 기법, 개발 단계별 인월수 기법, 수학적 산정 기법

▶LOC(원시 코드 라인수, source Line Of Code) 기법

소프트웨어 각 기능의 원시 코드 라인 수의 비관치, 낙관치, 기대치를 측정하여 예측치를 구하고 이를 이용하여 비용을 산정하는 기법

*비관치 : 가장 많이 측정된 코드 라인 수

*낙관치 : 가장 적게 측정된 코드 라인 수

*기대치 : 측정된 모든 코드 라인 수의 평균

*여기서 말하는 코드 라인 수는 그냥 코딩할 때 쓰는 그 라인 수(아래 그림 왼쪽 편에 있는 숫자)이다

```
1  #include <stdio.h>
2
3  int main() {
4      printf("Hello, world!");
5      return 0;
6  }
7
```

♣산정 공식

㉠노력(인월)=개발기간 x 투입기간 = LOC / 1인당 월평균 생산 코드 라인 수

㉡개발 비용= 노력(인월) x 단위 비용(1인당 월평균 인건비)

㉢개발기간 = 노력(인월)/ 투입 인원

㉣생산성 = LOC / 노력(인월)

예제) LOC 기법에 의하여 예측된 총 라인 수가 30,000라인, 개발에 참여할 프로그래머가 5명, 프로그래머들의 평균 생산성이 월간 300라인일 때 개발에 소요되는 기간은?

노력(인월)이라는 건 쉽게 말해 한 사람이 한 달 동안 할 수 있는 양이고 M/M(Man Month)=인월수라고 표현할 수도 있다

식이 아니라 한국말로 생각하면 전혀 어려운데 아니다.

노력(인월)은 즉 한 달 동안 한 사람이 할 수 있는 양이다, 총 30,000라인이 있고 프로그래머 1명당 평균 생산성이 월간 300라인이라고 하니까 그냥 30,000에서 300을 나누면 되는 것이다

한 사람당 30,000/300= 100라인이고 참여할 프로그래머가 5명 있으니 100에서 5를 또 나누면 그게 개발 기간이 된다.

따라서 100/5 = 20개월

▶개발 단계별 인월수(Effort Per Task) 기법

LOC 기법을 보완하기 위한 기법으로 각 기능을 구현시키는 데 필요한 노력을 생명 주기의 각 단계별로 산정하며 LOC 기법보다 더 정확하다.

▶수학적 산정 기법

경험적 추정 무형, 실험적 추정 모형이라고도 하며 개발 비용 산정의 자동화를 목표로 한다
ex) COCOMO 모형, Putnam 모형, 기능 점수(FP) 모형

㉠COCOMO(ConStructive COst MOdel) 모형

원시 프로그램의 규모인 LOC에 의한 비용 산정 기법이다.

개발할 소프트웨어 규모(LOC)를 예측한 후 이를 소프트웨어 종류에 따라 다르게 책정되는 비용 산정 방정식에 대입하여 비용을 산정한다.

비용 산정 결과 프로젝트를 완성하는 데 필요한 노력(Man-Month)으로 나타나며 보렘이 제안했다.

유형	특징
조직형 (Organic Mode)	<ul style="list-style-type: none"> ▶기관 내부에서 개발된 중·소 규모의 소프트웨어 ▶일괄 자료 처리나 과학 기술 계산용, 비즈니스 자료 처리용 등의 5만(50KDSI) 라인 이하의 소프트웨어를 개발하는 유형 ▶사무 처리용, 업무용, 과학용 응용 소프트웨어 개발에 적합
반분리형 (Semi-Detached Mode)	<ul style="list-style-type: none"> ▶조직형과 내장형의 중간형 소프트웨어 ▶트랜잭션 처리 시스템이나 운영체제, 데이터베이스 관리 시스템 등의 30만(300KDSI) 라인 이하의 소프트웨어를 개발하는 유형 ▶컴파일러, 인터프리터와 같은 유틸리티 개발에 적합
내장형 (Embedded Mode)	<ul style="list-style-type: none"> ▶초대형 규모의 소프트웨어 ▶트랜잭션 처리 시스템이나 운영체제 등의 30만(300KDSI) 라인 이상의 소프트웨어를 개발하는 유형 ▶신호기 제어 시스템, 미사일 유도 시스템, 실시간 처리 시스템 등의 시스템 프로그램 개발에 적합

㉡Putnam 모형

소프트웨어 생명 주기의 전 과정 동안에 사용될 노력의 분포를 예상하는 모형
푸트남(Putnam)이 제안한 것으로 생명주기 예측 모형이라고도 한다

㉢기능 점수(FP: Function Point) 모형

소프트웨어의 기능을 증대시키는 요인별로 가중치를 부여하고 요인별 가중치를 합산하여 총 기능 점수를 산출하며 총 기능 점수와 영향도를 이용하여 기능 점수(FP)를 구한 후 이를 이용해서 비용을 산정하는 기법
알브레히트(Albrecht)가 제안했다.

■비용 산정 자동화 추정 도구

SLIM	Rayleigh-Norden 곡선과 Putnam 예측 모델을 기초로 하여 개발된 자동화 추정 도구
ESTIMACS	다양한 프로젝트와 개인별 요소를 수용하도록 FP모형을 기초로 하여 개발된 자동화 추정 도구

◆프로젝트 일정 계획

소프트웨어를 개발하기 위해 어떤 작업이 필요한지 찾은 후, 이를 진행할 순서를 결정하거나 주어진 개발 기간에 소작업의 개발기간 및 그들 간의 순서, 필요한 자원 등과 같은 일정을 계획하는 것을 말한다.

ex) WBS, PERT/CPM, 간트 차트 등

▶작업 분할 구조도(WBS(Work Breakdown Structure))

프로젝트 목표를 달성하기 위해 필요한 활동과 업무를 세분화하는 작업이다. 그냥 쉽게 말해 업무 분장이다

▶네트워크 차트(PERT/CPM)

프로젝트를 완료할 수 있는 최소 기간은 얼마인지, 완료 기간을 맞추기 위해서는 각 작업을 언제 시작하고 완료해야 하는지, 지연되지 않으려면 어떤 작업에 특히 주의를 기울여야 하는지, 또 전체 프로젝트 완료 기간을 단축하기 위해서는 어떤 작업들을 단축하는 것이 가장 경제적인지 등을 알아보기 위해 필요한 도구

○PERT(Program Evaluation and Review Technique, 프로그램 평가 및 검토 기술)

프로젝트에 필요한 전체 작업의 상호 관계를 표시하는 네트워크

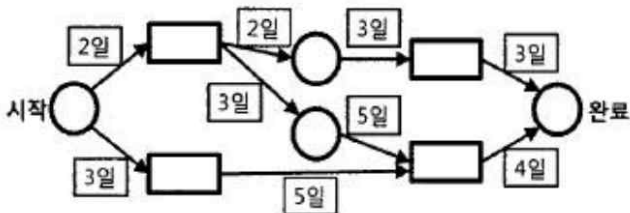
프로그램을 평가하고 검토하는 프로젝트 관리 기법으로 프로젝트 진행 상황을 통계적인 방법으로 파악하고 이를 통해 일정 계획 및 통제를 할 수 있도록 고안되었다.

○CPM(Critical Path Method, 임계 경로 기법)

프로젝트 완성에 필요한 작업을 나열하고 작업에 필요한 소요 기간을 예측하는데 사용하는 기법

미 듀폰사에서 화학 처리 공장의 건설 계획을 조직적으로 추진하기 위해 개발하였고, 건설 공사와 같이 단위 작업이 확정적 소요 시간을 갖는 프로젝트인 경우에 적합하다.

예제) 다음은 CPM 네트워크이다. 임계 경로의 소요 기일을 구하시오.



출처 : 20년 3회 정보처리기사 필기

원형 노드는 각각의 작업을 의미하며, 작업 이름과 소요 기간을 표시한다.

박스 노드는 이정표를 의미하며, 이정표 이름과 예상 완료 시간을 표시한다.

간선을 나타내는 화살표의 흐름에 따라 각 작업이 진행되며, 전 작업이 완료되어야 다음 작업을 진행할 수 있다.

임계 경로(Critical path)는 그래프에서 여유 시간이 없는 경로를 말하는데 쉽게 말해 그냥 최장 경로이다.

임계 경로를 구하는 이유는 여유 시간이 없으므로 모든 일정 계획이 임계 경로에 좌우되고 임계 경로에 작업 하나가 1일 지연되면 전체 일정도 똑같이 1일 지연되는 것이다.

반대로 전체 프로젝트 완료 시간을 단축하고 싶다면 임계 경로의 작업 시간을 줄이면 된다.

간단하게 예를 들면 친구 3명이 교실 2개와 강당을 청소해야 되는 상황이고 청소가 끝나면 집에 갈 수 있다. 인당 1개씩 맡아서 하기로 했는데 강당은 교실보다 넓으므로 청소 시간이 더 오래 걸린다. 이때 강당 청소가 임계 경로이다. 강당 청소가 귀가 시간을 좌우하게 된다. (어차피 강당 청소가 끝나야 집에 가므로 교실 청소하는 친구들은 천천히 여유 시간을 가지면서 해도 되지만 강당을 맡은 친구는 여유 시간 없이 열심히 청소를 해야 한다)

▶간트 차트

프로젝트의 각 작업들이 언제 시작하고 언제 종료되는지에 대한 작업 일정을 막대 도표를 이용하여 표시하는 프로젝트 일정표

시간선(Time-Line) 차트라고도 한다.



간트 차트를 이용한 프로젝트 일정 계획표
출처 : 네이버 지식백과(쉽게 배우는 소프트웨어 공학)

2-2) 요구 사항 분석 단계

소프트웨어 개발의 궁극적인 목적은 개발된 소프트웨어를 사용하는 고객이 만족하도록 하는 것이다. 고객 만족을 위해서는 원하는 품질의 제품을 정해진 개발기간과 주어진 예산 범위 안에서 개발해야 한다. 그러려면 먼저 사용자의 요구사항을 정확히 파악하고 분석하는 작업이 필요하다.

◆요구사항

소프트웨어가 어떤 문제를 해결하기 위해 제공하는 서비스에 대한 설명과 정상적으로 운영되는데 필요한 제약조건

ex) 기능 요구사항, 비기능 요구사항, 사용자 요구사항, 시스템 요구사항

▶기능 요구사항(Functional requirements)

시스템이 무엇을 하는지, 어떤 기능을 하는지 등의 기능이나 수행과 관련된 요구사항

▶비기능 요구사항(Non-Functional requirements)

품질이나 제약사항과 관련된 요구사항

▶사용자 요구사항(User requirements)

사용자 관점에서 본 시스템이 제공해야 할 요구사항

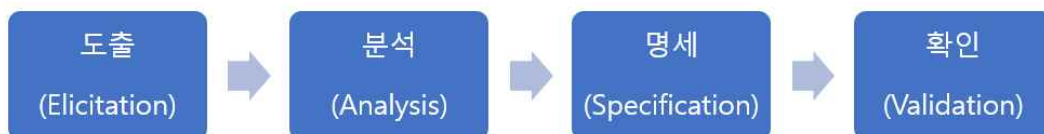
▶시스템 요구사항(System requirements)

개발자 관점에서 본 시스템 전체가 사용자와 다른 시스템에 제공해야 할 요구사항(소프트웨어 요구사항이라고도 함)

개발 초기에 사용자의 요구사항을 추출하여 정리한 문서를 요구 분석 명세서라고 한다.

◆요구사항 개발 프로세스

개발 대상에 대한 요구사항을 체계적으로 도출하고 분석한 후 명세서에 정리한 다음 확인 및 검증하는 일련의 구조화된 활동



요구사항 개발 프로세스

㉠요구사항 도출(Requirement Elicitation, 요구사항 수집)

시스템, 사용자, 개발자 등 시스템 개발에 관련된 사람들이 서로 의견을 교환하여 요구사항을 어떻게 수집할 것인지를 식별하고 이해하는 과정

ex) 청취와 인터뷰, 설문, 브레인스토밍, 워크샵, 프로토타이핑, 유스케이스

요구사항이 정리되면 어떠한 형태로든 정리된 내용을 한눈에 파악할 수 있도록 기록하거나 표현해놓아야 한다. 자연어(한글)로 작성하여 문서화해놓을 수도 있고 도표나 그래프로 표현할 수도 있다

㉠요구사항 분석(Requirement Analysis)

개발 대상에 대한 사용자의 요구사항 중 명확하지 않거나 모호하여 이해되지 않는 부분을 발견하고 이를 걸러내기 위한 과정

소프트웨어 개발의 실제적인 첫 단계로, 개발 대상에 대한 사용자의 요구사항을 이해하고 문서화하는 활동
ex) 자료 흐름도(DFD), 자료 사전(DD)

▶구조적 분석 기법

자료의 흐름과 처리를 중심으로 하는 요구사항 분석 방법



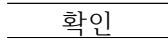
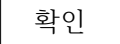
ex) 자료 흐름도(DFD), 자료 사전(DD), 소단위 명세서(Mini-Spec), 개체 관계도(ERD), 상태 전이도(STD), 제어 명세서

①자료 흐름도(DFD: Data Flow Diagram)

자료의 흐름 및 변환 과정과 기능을 도형 중심으로 기술하는 방법

자료 흐름 그래프, 버블 차트라고도 한다.

*자료 흐름도 기본 기호

기호	의미	표기법 (Yourdon/DeMacro)
프로세스 (Process)	자료를 변환시키는 시스템의 한 부분을 나타내며 처리, 기능, 변환, 버블이라고도 함	
자료 흐름도 (Data Flow)	자료의 이동이나 연관 관계를 나타냄	
자료 저장소 (Data Store)	시스템에서의 자료 저장소(파일, 데이터베이스)를 나타냄	
단말 (Terminator)	시스템과 교신하는 외부 개체로, 입력 데이터가 만들어지고 출력 데이터를 받음	

②자료 사전(DD: Data Dictionary)

자료 흐름도에 있는 자료를 더 자세히 정의하고 기록한 것

데이터를 설명하는 데이터로, 데이터의 데이터 또는 메타 데이터라고도 한다.

*자료 사전에서 사용되는 표기 기호

기호	의미
=	자료의 정의 : ~로 구성되어 있다(is composed of)
+	자료의 연결 : 그리고(and)
()	자료의 생략 : 생략 가능한 자료(Optional)
[]	자료의 선택 : 또는(or)
{ }	자료의 반복 : Iteration of
* *	자료의 설명 : 주석(Comment)

▶요구사항 분석용 도구

㉔요구사항 분석용 CASE(자동화 도구)

요구사항을 자동으로 분석하고 요구사항 분석 명세서를 기술하도록 개발된 도구

SADT	-시스템 정의, 소프트웨어 요구사항 분석, 시스템/소프트웨어 설계를 위한 도구 -SoftTech사에서 개발 -구조적 요구 분석을 하기 위해 블록 다이어그램을 채택한 자동화 도구
SREM= RSL/REVS	TRW사가 실시간 처리 소프트웨어 시스템에서 요구사항을 명확히 기술하도록 할 목적으로 개발한 도구
PSL/PSA	PSL과 PSA를 사용하는 자동화 도구
TAGS	시스템 공학 방법 응용에 대한 자동 접근 방법

㉕HIPO(Hierarchy Input Process Output)

시스템 분석 및 설계, 또는 문서화에 사용되는 기법으로 시스템 실행 과정인 입력, 처리, 출력의 기능을 표현한 것

하향식 소프트웨어 개발을 위한 문서화 도구

기능과 자료의 의존 관계를 동시에 표현 가능

기호, 도표 등을 사용하므로 보기 쉽고 이해하기 쉬움

▶모델과 모델링

모델은 어떤 복잡한 대상의 핵심 특징만 선별하여 일정한 관점으로 단순화시켜 기호나 그림 등을 사용해 체계적으로 표현한 것이다.

모델은 하나의 사물을 여러 관점에서 바라볼 수 있게 해준다. 원통을 예로 들 수 있는데 정면에서 보면 사각형, 위에서 보면 원으로 보인다. 이처럼 관점을 다르게 하면 다르게 보이듯이 소프트웨어를 개발할 때도 여러 관점의 모델에서 필요한 정보를 얻어 개발한다.

모델링은 모델을 제작하는 작업을 말하는데 객체지향 개발에서는 UML의 다양한 다이어그램을 통해 개발하려는 소프트웨어의 범위나 개략적인 구조와 기능을 이해할 수 있다. 이와 같이 UML의 수많은 다이어그램들이 소프트웨어 개발 과정에서 하나의 모델로 사용된다.

▶모델링 언어

소프트웨어 개발에서 모델링 언어는 요구사항 정의 및 분석, 설계의 결과물을 다양한 다이어그램으로 표현하는 표기법으로 애매모호한 표현이 없고 일관되어 모델링하는 데 매우 유용하며 개발자 간에 원활한 의사소통이 가능해진다.

모델링 개발 방법론에 따라 사용하는 도구가 다르다.

ex)

구조적 방법론-자료 흐름도(DFD)/자료 사전(DD)/처리명세서

정보공학 방법론-개체-관계 다이어그램(ERD: Entity-Relationship Diagram)

객체지향 방법론-UML(특히 요구사항은 유스케이스 다이어그램 사용)

*구조적 방법론 : 정형화된 분석 절차에 따라 사용자 요구사항을 파악하여 문서화하는 처리 중심의 방법론(1960년대까지 가장 많이 적용되었던 소프트웨어 개발 방법론)

*정보공학 방법론 : 정보 시스템의 개발을 위해 계획, 분석, 설계, 구축에 정형화된 기법들을 상호 연관성 있게 통합 및 적용하는 자료(Data) 중심의 방법론

*객체지향 방법론 : 현실 세계의 개체(Entity)를 기계의 부품처럼 하나의 객체(Object)로 만들어 소프트웨어를 개발할 때 기계의 부품을 조립하듯이 객체들을 조립해서 필요한 소프트웨어를 구현하는 방법론

*컴포넌트 기반(CBD: Component Based Design) 방법론 : 기존의 시스템이나 소프트웨어를 구성하는 컴포넌트를 조합하여 하나의 새로운 애플리케이션을 만드는 방법론

▶UML(Unified Modeling Language)

시스템 분석, 설계, 구현 등 시스템 개발 과정에서 시스템 개발자와 고객 또는 개발자 상호간의 의사소통이 원활하게 이루어지도록 표준화된 대표적인 객체지향 모델링 언어

Rumbaugh(OMT), Booch, Jacobson 등의 객체지향 방법론의 장점을 통합

OMG(Object Management Group)에서 표준으로 지정

♣UML 구성요소 : 사물(Things), 관계(Relationships), 다이어그램(Diagram)

@사물(Things)

다이어그램 안에서 관계가 형성될 수 있는 대상들

구조 사물 (Structural Things)	-시스템의 개념적, 물리적 요소를 표현 -클래스, 유스케이스, 컴포넌트, 노트 등
행동 사물 (Behavioral Things)	-시간과 공간에 따른 요소들의 행위를 표현 -상호작용, 상태 머신 등
그룹 사물 (Grouping Things)	-요소들을 그룹으로 묶어서 표현 -패키지
주해 사물 (Annotation Things)	-부가적인 설명이나 제약조건 등을 표현 -노트

⑤관계(Relationships)

사물과 사물 사이의 연관성을 표현하는 것

ex) 연관 관계, 집합 관계, 포함 관계, 일반화 관계, 의존 관계, 실체화 관계

①연관(Association) 관계

2개 이상의 사물이 서로 관련되어 있는 관계

ex) 사람과 집, 선생님과 학생

②집합(Aggregation) 관계

하나의 사물이 다른 사물에 포함되어 있는 관계

ex) 컴퓨터와 프린터(프린터는 컴퓨터에 연결해서 사용할 수 있으며, 다른 컴퓨터에 연결해서 사용도 가능)

③포함(Composition) 관계

포함하는 사물의 변화가 포함되는 사물에게 영향을 미치는 관계

ex) 문과 열쇠(문을 열 수 있는 열쇠는 하나이며, 해당 열쇠로 다른 문은 열 수 없음, 문이 없으면 열쇠도 필요 없어짐)

④일반화(Generalization) 관계

하나의 사물이 다른 사물에 비해 더 일반적이거나 구체적인 관계

ex) 동물/코끼리, 치타(코끼리와 치타는 동물이다)

⑤ 의존(Dependency) 관계

연관 관계와 같이 사물 사이에 서로 연관은 있으나 필요에 의해 서로에게 영향을 주는 짧은 시간 동안만 연관을 유지하는 관계

ex) 등급, 할인율(등급이 높으면 할인율을 적용하고 등급이 낮으면 할인율을 적용하지 않음)

⑥실체화(Realization) 관계

사물이 할 수 있거나 해야 하는 기능으로, 서로를 그룹화 할 수 있는 관계

ex) 헤엄칠 수 있다/수영 선수, 물고기(수영 선수와 물고기는 헤엄칠 수 있다는 행위로 그룹화할 수 있다)

◎다이어그램(Diagram)

사물과 관계를 도형으로 표현한 것

정적 모델링에서는 주로 구조적 다이어그램을 사용하고

동적 모델링에서는 주로 행위 다이어그램을 사용한다.

①구조적(Structural) 다이어그램

종류	내용
클래스 다이어그램 (Class Diagram)	클래스와 클래스가 가지는 속성, 클래스 사이의 관계를 표현
객체 다이어그램 (Object Diagram)	클래스에 속한 사물(객체)들, 즉 인스턴스(Instance)를 특정 시점의 객체와 객체 사이의 관계로 표현 럼바우(Rumbaugh) 객체지향 분석 기법에서 객체 모델링에 활용됨
컴포넌트 다이어그램 (Component Diagram)	실제 구현 모듈인 컴포넌트 간의 관계나 컴포넌트 간의 인터페이스를 표현함(구현 단계에서 사용)
배치 다이어그램 (Deployment Diagram)	결과물, 프로세스, 컴포넌트 등 물리적 요소들의 위치를 표현 (구현 단계에서 사용)
복합체 구조 다이어그램 (Composite Structure Diagram)	클래스나 컴포넌트가 복합 구조를 갖는 경우 그 내부 구조를 표현
패키지 다이어그램 (Package Diagram)	유스케이스나 클래스 등의 모델 요소들을 그룹화한 패키지들의 관계를 표현

② 행위(Behavioral) 다이어그램

종류	내용
유스케이스 다이어그램 (Use Case Diagram)	사용자의 요구를 분석하는 것으로, 기능 모델링 작업에 사용함
시퀀스 다이어그램 (Sequence Diagram)	상호 작용하는 시스템이나 객체들이 주고받는 메시지를 표현함
커뮤니케이션 다이어그램 (Communication Diagram)	동작에 참여하는 객체들이 주고받는 메시지와 객체들 간의 연관 관계를 표현함
상태 다이어그램 (State Diagram)	하나의 객체가 자신이 속한 클래스의 상태 변화 혹은 다른 객체와의 상호 작용에 따라 상태가 어떻게 변화하는지를 표현 (럼바우 객체지향 분석 기법에서 동적 모델링에 활용)
활동 다이어그램 (Activity Diagram)	시스템이 어떤 기능을 수행하는지 객체의 처리 로직이나 조건에 따른 처리의 흐름을 순서에 따라 표현
상호작용 개요 다이어그램 (Interaction Overview Diagram)	상호작용 다이어그램 간의 제어 흐름을 표현
타이밍 다이어그램 (Timing Diagram)	객체 상태 변화와 시간 제약을 명시적으로 표현

*스테레오 타입(Stereotype)

UML에서 표현하는 기본 기능 외에 추가적인 기능을 표현하는 것

길러멧(Guillemet)이라고 부르는 겹화살괄호(«») 사이에 표현할 형태를 기술한다.

㉔요구사항 명세(Requirement Specification)

분석된 요구사항을 바탕으로 모델을 작성하고 문서화하는 것

구분	정형 명세 기법	비정형 명세 기법
기법	수학적 원리 기반, 모델 기반	상태/기능/객체 중심
작성 방법	수학적 기호, 정형화된 표기법	일반 명사, 동사 등의 자연어를 기반으로 서술 또는 다이어그램으로 작성
특징	요구사항을 정확하고 간결하게 표현 가능 표기법이 어려워 사용자가 이해하기 어려움	자연어의 사용으로 인해 요구사항 결과가 작성자에 따라 다를 수 있어 일관성이 떨어짐 내용의 이해가 쉬워 의사소통 용이
종류	VDM, Z, Petri-net, CSP등	FSM, Decision Table, ER모델링, State Chart(SADT) 등

㉕요구사항 확인(Requirement Validation, 요구사항 검증)

개발 자원을 요구사항에 할당하기 전에 요구사항 명세서가 정확하고 완전하게 작성되었는지를 검토하는 활동
요구사항 관리 도구를 이용하여 요구사항 정의 문서들에 대해 형상관리(SCM)을 수행한다.

*SCM: 소프트웨어 개발 단계 각 과정에서 만들어지는 프로그램, 프로그램을 설명하는 문서, 데이터 등을 통칭하여 형상이라고 하는데 이 형상을 관리하는 일련의 활동

2-3) 설계 단계

분석 단계에서 사용자의 요구사항을 토대로 요구 분석 명세서를 작성하는데 이때 작성하는 요구 분석 명세서에는 하드웨어나 소프트웨어의 종류는 무엇인지 같은 자세한 내용 없이 개념적이고 추상적인 성격이 강하다.

반면, 설계 단계에서는 고려하지 않았던 상세 내용을 충분히 반영하여 구현할 수 있는 수준으로 준비해야 한다. 분석 단계에서는 사용자의 요구를 what(무엇) 관점에서 바라보았다면, 설계 단계에서는 how(어떻게) 관점에서 생각한다.

소프트웨어 설계는 요구 분석 명세서와 설계 원리, 제약조건에 따라 상위 설계와 하위 설계로 나눌 수 있다.



상위 설계와 하위 설계

출처 : 네이버 지식백과(쉽게 배우는 소프트웨어 공학)

▲상위 설계

예비 설계(preliminary design)라고도 하며 집 짓기에서 **전체 골조(뼈대)를 세우는 것**과 유사하다.

- 1) 아키텍처(구조) 설계 : 시스템의 전체적인 구조 표현
- 2) 데이터 설계 : 시스템에 필요한 정보를 자료구조와 데이터베이스 설계에 반영
- 3) 시스템 분할 : 전체 시스템을 여러 개의 서브 시스템으로 분리
- 4) 인터페이스 정의 : 시스템의 구조와 서브 시스템들 사이의 인터페이스를 명확히 정의
- 5) 사용자 인터페이스 설계 : 사용자가 익숙하고 편리하게 사용할 수 있도록 사용자 인터페이스 설계

▲하위 설계

내부 구조를 상세히 나타내는 것과 유사하다.

각 모듈의 실제적인 내부를 알고리즘 형태로 표현하고 인터페이스에 대한 설명, 자료구조, 변수 등에 대한 상세한 정보를 작성한다.

◆설계의 원리

①분할과 정복

가장 세분화된 작은 시스템을 개발하고 하나씩 위로 올라가면서 완성시키는 방법으로 개발하는 것 하나의 일을 수행할 때 작은 단위로 나누고 각 작은 단위를 하나씩 처리하여 전체 일을 끝낸다는 의미

②모듈화(Modularity)

소프트웨어의 성능 향상, 시스템의 수정 및 재사용, 유지 관리 등이 용이하도록 시스템의 기능들을 모듈 단위로 나누는 것

③추상화(Abstraction)

문제의 전체적이고 포괄적인 개념을 설계한 후 차례로 세분화하여 구체화시켜 나가는 것

ex) 과정 추상화, 데이터 추상화, 제어 추상화

④단계적 분해(Stepwise Refinement)

문제를 상위의 중요 개념으로부터 하위의 개념으로 구체화시키는 분할 기법

Niklaus Wirth에 의해 제안된 하향식 설계 전략

⑤정보 은닉(Information Hiding)

모듈 내부에 포함된 절차와 자료들의 정보가 감추어져 다른 모듈이 접근하거나 변경하지 못하도록 하는 기법

◆소프트웨어 아키텍처

소프트웨어를 구성하는 요소들 간의 관계를 표현하는 시스템 구조 또는 구조체

간단한 소프트웨어라면 상관없지만 복잡하고 규모가 큰 소프트웨어를 개발하려면 전체적인 구조가 유기적으로 잘 구성되어야 한다. 또한 사용자가 만족할만한 품질 좋은 소프트웨어를 개발하려면 요구 분석, 설계 단계에서부터 품질 특성을 고려하여 개발해야 한다. 즉, 잘 정의된 전체적인 구조와 품질 좋은 소프트웨어를 만들려면 소프트웨어 아키텍처가 필요하다.

한 마디로 요약해서 소프트웨어의 뼈대를 만드는 것이다.

◆아키텍처 패턴

아키텍처를 설계할 때 참조할 수 있는 전형적인 해결방식 또는 예제

ex) 레이어 패턴, 클라이언트-서버 패턴, 파이프-필터 패턴, 모델-뷰-컨트롤러 패턴

㉠레이어 패턴(Layers pattern) : 시스템을 계층으로 구분하여 구성하는 고전적인 방법의 패턴

㉡클라이언트-서버 패턴(Client-Server Pattern) : 하나의 서버 컴포넌트와 다수의 클라이언트 컴포넌트로 구성되는 패턴(ex) OSI 참조 모델))

㉢파이프-필터 패턴(Pipe-Filter Pattern) : 데이터 스트림 절차의 각 단계를 필터로 캡슐화하여 파이프를 통해 전송하는 패턴(ex) UNIX의 셸(Shell))

㉣모델-뷰-컨트롤러 패턴(MVC: Model-View-Controller Pattern) : 서브 시스템을 모델, 뷰, 컨트롤러로 구조화하는 패턴

㉤마스터-슬레이브 패턴(Master-Slave Pattern) : 슬레이브 컴포넌트에서 처리된 결과물을 다시 돌려받는 방식으로 작업을 수행하는 패턴

㉥브로커 패턴(Broker Pattern) : 사용자가 원하는 서비스와 특성을 브로커 컴포넌트에 요청하면 브로커 컴포넌트가 요청에 맞는 컴포넌트와 사용자를 연결해주는 패턴

㉦피어-투-피어 패턴(Peer-To-Peer Pattern) : 피어(Peer)라 불리는 하나의 컴포넌트가 클라이언트가 될 수도, 서버가 될 수도 있는 패턴

㉧이벤트-버스 패턴(Event-Bus Pattern) : 소스가 특정 채널에 이벤트 메시지를 발행(Publish)하면, 해당 채널을 구독한 리스너들이 메시지를 받아 이벤트를 처리하는 패턴

㉨블랙보드 패턴(Blackboard Pattern) : 모든 컴포넌트들이 공유 데이터 저장소와 블랙보드 컴포넌트에 접근이 가능한 패턴

㉩인터프리터 패턴(Interpreter Pattern) : 프로그램 코드의 각 라인을 수행하는 방법을 지정하고, 기호마다 클래스를 갖도록 구성된 패턴

◆디자인 패턴

모듈 간의 관계 및 인터페이스를 설계할 때 참조할 수 있는 설계 과정에서 자주 발생하는 문제들에 대한 전형적인 해결방식 또는 예제를 의미

자주 사용하는 설계 형태를 정형화해서 이를 유형별로 설계 템플릿을 만들어둔 것

디자인 패턴을 한 번 만들어 놓으면 계속 재사용할 수 있고, 이를 변형하여 새로운 디자인 패턴을 만들 수도 있다. 많은 개발자들이 수많은 시행착오를 겪으며 만들어 놓은 좋은 설계 자료를 모아두고 새 프로젝트에 활용할 수 있는 것이다. 그리고 이 방법을 발전시켜 설계 자료를 유형별로 분류하면 개발기간을 줄이고 유지보수도 매우 쉬워질 것이다.

햄버거집으로 예를 들어보자.

빵, 치즈, 피클, 양상추, 토마토, 베이컨, 패티, 치킨 등등 다양한 버거 재료가 있는데 먹을 때마다 새로 정하기 힘들니까 치즈버거, 불고기버거처럼 패턴을 정해서 미리 만들어 놓는 것이다.

아키텍처 패턴에서 건물의 윤곽을 잡는 가이드라인을 제시했다면, **디자인 패턴**은 그보다 더 세밀한 부분의 건물의 각 방들을 인테리어 하는 과정이라고 보면 된다.

◆Gof 디자인 패턴

경험이 적은 사람들이 설계를 잘하기 위해서는 경험이 많은 전문가의 지식과 노하우를 전수받는 것이다. 어떻게 하면 너무 일반적이지도, 너무 구체적이지도 않은 형태로 소프트웨어 설계를 위한 지식이나 노하우를 공유하는 방법이 뭘까 해서 고안된 방법이 에릭 감마, 리처드 헬름, 랄프 존슨, 존 블리시데스가 제안한 Gof(Gang of Four)의 디자인 패턴이다.

디자인 패턴은 앞서 설명했듯이 여러 가지 문제에 대한 설계 사례로 분석하여 서로 비슷한 문제를 해결하기 위한 설계들을 분류하고 각 문제 유형별로 가장 적합한 설계를 일반화해 패턴으로 정립한 것을 말하는데 소프트웨어 설계에 대한 지식이나 노하우가 문제 유형별로 잘 구체화되어 있을 뿐 아니라, 동일한 문제 유형에 대해서는 그 해결 방법에 대한 지식이나 노하우가 패턴 형태로 충분히 일반화된 것이라고도 볼 수 있다. 단, 이 Gof의 디자인 패턴은 쉽게 재사용할 수 있도록 **객체지향** 개념에 따른 설계만을 패턴으로 지정하였다.

Gof 디자인 패턴 설명에 앞서 먼저 객체지향에 대한 개념부터 설명을 하겠다.

▶절차지향 언어(Procedural Programming)

순차적으로 처리되고 프로그램 전체가 유기적으로 연결되도록 만드는 방법이며 대표적인 예로 C언어가 있다. 컴퓨터의 처리구조와 유사해 실행 속도가 빠른 장점이 있지만 실행 순서가 정해져 있으므로 코드의 순서가 바뀌면 동일한 결과를 보장하기 어려워 유지보수가 어렵다.

(프로그램 전체가 유기적으로 연결되어 있으므로 일부분이 고장이 나도 전체를 수리해야 한다(매우 비효율적이다))

절차지향과 비슷한 예로는 자동차 제도가 있다. 자동차는 엔진, 차제, 핸들, 의자, 바퀴 순으로 차례대로 만들어져야 하고 순서도 틀리면 안된다고 한다.

이런 단점을 보완하기 위해 객체지향이 등장할 하게 된다.

▶객체지향 언어(Object-oriented Programming)

객체란 구체적인 사물, 혹은 사람이 될 수도 있고 추상적인 개념이 될 수도 있는 무언가인데 컴퓨터로 예를 들면 컴퓨터에 쓰이는 모니터, 키보드, 마우스, 본체, 스피커와 컴퓨터가 제공하는 기능(화면이 켜지고 다음 화면으로 넘어가고 소리가 나고 이런 모든 것)이다.

객체지향 언어란 소프트웨어의 각 요소들을 객체(Object)로 만든 후, 객체들을 조립해서 소프트웨어를 개발하는 기법이다.

쉽게 생각해서 컴퓨터로 예를 들면 컴퓨터의 모든 부품을 적절히 연결하고 조립해서 컴퓨터가 제대로 작동하도록 만드는 것이라고 볼 수 있다.

대표적인 예로는 자바가 있다.

▶객체지향의 구성요소 : 객체, 클래스, 메시지

㉠객체(Object) : 데이터와 이를 처리하기 위한 함수를 묶어 놓은 소프트웨어 모듈

데이터는 객체가 가지고 있는 정보로, 속성이나 상태 분류등을 말하고

함수는 객체가 수행하는 기능으로 객체가 갖는 데이터를 처리하는 알고리즘을 말한다.

프로그래밍에서는 하나의 특정 작업을 수행하기 위해 독립적으로 설계된 프로그램 코드의 집합을 함수라고 하는데 중복적인 코드의 작성을 최소화할 수 있다.

예를 들어 붕어빵으로 예를 들면 “붕어빵 틀에 반죽을 넣는다+팔을 넣는다”를 함수에 넣어보자.

함수 이름을 ‘붕어빵 만들기1’이라고 한다면 붕어빵을 2번 만든다고 할 때 일일이 “붕어빵 틀에 반죽을 넣는다+팔을 넣는다” 이렇게 2번 적어주지 않고 붕어빵 만들기1 2번을 적으면 원하는 대로 실행이 된다.

객체를 게임 캐릭터라고 생각해보자. 그 캐릭터라는 객체 안에는 캐릭터의 이름, 직업, 성격 등이 포함돼 있다.

```
1  캐릭터
2  {
3      이름 :
4      직업 :
5      성격 :
6  }
```

캐릭터를 여러 명 만든다고 생각을 해보자.

캐릭터가 몇 개 안된다면 그냥 이렇게 일일이 추가해도 되겠지만 코드도 길어지고 “이름, 직업, 성격”이라는 동일한 속성을 가지고 있고 “Andrew, 마법사, 밝음”이라는 데이터만 다를 뿐인데 계속 이렇게 일일이 입력하는 것은 매우 비효율적이다.

그리고 가령 내가 만약 이름, 직업, 성격 말고도 성별이라는 속성을 넣고 싶다면 이마저도 일일이 다 입력을 해야 한다. 이럴 때 사용하는 것이 바로 클래스이다.

```
1  캐릭터1
2  {
3      이름 : Andrew
4      직업 : 마법사
5      성격 : 밝음
6  }
7
8  캐릭터2
9  {
10     이름 : Jack
11     직업 : 기사
12     성격 : 용맹
13 }
```

*실제 코딩과 무관한 단순한 예시

㉠클래스(Class) : 공통된 속성과 연산을 갖는 객체의 집합

*패키지 : 자바에서 클래스와 인터페이스의 집합을 의미하며 관련 있는 클래스나 인터페이스를 함께 묶음으로써 파일을 효율적으로 관리할 수 있다.

```
1  class 캐릭터(이름, 직업, 성격)
2  {
3      이름=이름
4      직업=직업
5      성격=성격
6  }
7
8  캐릭터(Andrew, 마법사, 밝음)
9  캐릭터(Jack, 기사, 용맹)
```

*실제 코딩과 무관한 단순한 예시

클래스는 객체를 찍어내는 공장으로 객체를 봉어빵에 클래스를 봉어빵틀에 비유해도 무방하다.

이때 클래스에 속한 각각의 객체, 즉 클래스에서 생성된 객체(봉어빵틀에서 만들어진 봉어빵)을 인스턴스라고 부른다. 이렇게 봉어빵틀(클래스)에서 봉어빵(객체)을 만드는 과정을 인스턴스화라고 한다

클래스를 쓰는 이유는 함수와 데이터가 저장되는 변수(속성)까지 함께 묶어서 관리하기 위해서이다.

이때 클래스에서 포함되는 변수를 속성이라고 부르며, 클래스에 포함되는 함수를 메서드라고 부른다.

은행 업무를 클래스로 작성한다면 속성에는 계좌번호, 소유주, 잔액등/ 메서드에는 입금, 출금, 이체 등이 있을 것이다.

㉡메시지(Message) : 객체들 간의 상호작용에 사용되는 수단으로, 객체의 동작이나 연산을 일으키는 외부의 요구사항

▶객체지향의 특징 : 캡슐화, 상속, 추상화, 다형성 + 연관성

㉠캡슐화(Encapsulation) : 외부에서의 접근을 제한하기 위해 인터페이스를 제외한 세부 내용을 은닉하는 것
캡슐로 된 알약을 생각하면 되는데 특정 질환을 치료하기 위해 서로 다른 약들을 조합하여 캡슐에 담아놓는 것인데 캡슐 속 알갱이 하나하나가 어떤 기능을 하는지 외부에 비공개를 하고 알약을 사용하게 하는 것이다.

이렇게 한 모듈 내부에 포함된 절차와 자료들의 정보가 감추어져 다른 모듈이 접근하거나 변경하지 못하도록 하는 것을 정보은닉(Information Hiding)이라고 한다.

㉡상속(Inheritance) : 상위 클래스의 모든 속성과 연산을 하위 클래스가 물려받는 것

물려주는 클래스를 상위 클래스 또는 부모 클래스라고 하고 물려받는 클래스를 하위 클래스 또는 자식 클래스라고 한다.

말 그대로 부모님의 재산을 자식이 똑같이 쓸 수 있는 것을 생각하면 된다.

㉢추상화(Abstraction) : 객체의 공통적인 속성과 기능을 추출하여 정의하는 것

쉽게 예를 들면 바나나, 사과, 포도를 추상화하면 과일이 된다.

㉣다형성(Polymorphism) : 하나의 메시지에 대한 각각의 객체가 가지고 있는 고유한 방법으로 응답할 수 있는 능력

다형성의 원래 뜻은 어떤 객체의 속성이나 기능이 상황에 따라 여러 가지 형태를 가질 수 있는 성질이다. 그와 마찬가지로 객체지향에서의 다형성도 객체의 속성이나 기능이 그 맥락에 따라 다른 역할을 수행할 수 있는 것을 말한다.

예를 들어 ‘동물이 과일을 먹는다’라는 예제가 있다고 치자.

동물과 과일은 추상어이므로 동물은 상황, 맥락에 따라 코뿔소, 사자, 호랑이, 곰, 여우등이 될 수 있고 과일도 마찬가지로 바나나, 사과, 포도등으로 바뀔 수 있다.

이렇게 ‘동물’, ‘과일’ 같은 동일한 메서드 이름을 사용하지만 메서드에 대해 클래스마다 모두 다르게 구현되는 것(상황에 따라 곰, 여우/사과, 포도)이 바로 다형성이다.

이런 다형성과 관련된 개념으로 오버로딩과 오버라이딩이 있다.

*오버로딩(overloading) : 한 클래스에 이름이 동일한 메서드가 중복 정의되어 있는 경우(은행 클래스로 예를 들면 입금하기 메서드가 2개이상 있는 것)

자바에서 한 클래스 내에 이미 사용하려는 이름과 같은 이름을 가진 메서드가 있더라도 매개변수의 개수 또는 타입이 다르면 같은 이름을 사용해서 메서드를 정의할 수 있다

*오버라이딩(overriding) : 부모 클래스로부터 상속받은 메서드를 자식 클래스에서 재정의하는 것

상속받은 메서드를 그대로 사용할 수도 있지만, 자식 클래스에서 상황에 맞게 변경해야 하는 경우 오버라이딩 할 필요가 생긴다.

㉤연관성(Relationship) : 두 개 이상의 객체들이 상호 참조하는 관계

클래스들이 서로 관계를 맺고 이 관계를 통해 메시지를 주고받으며 기능을 제공한다.

종류	의미	특징
is member of	연관화(Association)	2개 이상의 객체가 상호 관련되어 있음을 의미
is instance of	분류화(Classification)	동일한 형의 특성을 갖는 객체들을 모아 구성하는 것
is part of	집단화(Aggregation)	관련 있는 객체들을 묶어 하나의 상위 객체를 구성하는 것
is a	일반화(Generalization)	공통적인 성질들로 추상화한 상위 객체를 구성하는 것
	특수화/상세화(Specialization)	상위 객체를 구체화하여 하위 객체를 구성하는 것

▶객체지향 설계 원칙(SOLID) : 변경이나 확장에 유연한 시스템을 설계하기 위해 지켜져야 할 원칙
 경우에 따라서는 원칙을 따르지 않을 때 더 빨리 개발하기도 하지만 이 경우 절약한 개발 시간보다 훨씬 많은 유지보수 시간이 필요할지도 모른다.

①단일 책임 원칙(SRP; single Responsibility Principle)

한 클래스에 단 하나의 책임(기능)만 가져야 한다는 원칙

②개방-폐쇄 원칙(OCP; Open Closed Principle)

클래스를 확장하는 것은 쉽게 변경은 어렵게 해야한다는 원칙

③리스코프 치환 원칙(LSP; Liskov Substitution Principle)

자식 클래스는 최소한 부모 클래스의 기능은 수행할 수 있어야 한다는 원칙

ex) 문서편집기 신버전에서 구버전에서 작성한 문서들을 열 수 있게 한다

④인터페이스 분리 원칙(ISP; Interface Segregation Principle)

자신이 사용하지 않는 인터페이스와 의존 관계를 맺거나 영향을 받지 않아야 한다는 원칙

뭔가를 변경할 때 전혀 관계없는 메서드의 변화에 영향을 주지 않기 위해서 필요하다.

⑤의존 역전 원칙(DIP; Dependency Inversion Principle)

의존 관계 성립 시 추상성이 높은 클래스와 의존 관계를 맺어야 한다는 원칙

ex) 사자와 의존 관계를 맺는게 아니라 동물과 의존 관계를 맺어야 한다는 의미

객체지향에 대한 설명은 여기까지 하고 다시 **Gof의 디자인 패턴**으로 돌아오자.

Gof의 디자인 패턴은 생성 패턴, 구조 패턴, 행위 패턴으로 구분된다.

㉔생성 패턴(Creational Pattern) : 클래스나 객체의 생성과 참조 과정을 정의하는 패턴

추상 팩토리 (Abstract Factory)	구체적인 클래스에 의존하지 않고, 서로 연관되거나 의존적인 객체들의 조합을 만드는 인터페이스를 제공하는 패턴 ex) 쏘나타와 아반테가 있다. 엔진, 차제, 핸들, 의자, 바퀴등의 자동차 구성품의 구체적인 설정값은 다른데 품목은 동일하므로 그 품목을 만드는 공장을 만들어 그 공장에서 구체적인 설정값만 받아서 쏘나타와 아반테를 둘 다 만들 수 있게 만들
빌더 (Builder)	복합 객체의 생성 과정과 표현 방법을 분리하여 동일한 생성 절차에서 서로 다른 표현 결과를 만들 수 있게 하는 패턴 ex) 곰인형 공장인데 a위치에서 곰 눈알과 곰 몸통을 한꺼번에 만드는게 아니라 a위치에서는 몸통만, b위치에서는 눈알만 해서 분업해서 하나의 제품을 만들
팩토리 메서드 (Factory Method)	객체 생성을 서브 클래스에서 처리하도록 분리하여 캡슐화한 패턴 가상 생성자(Virtual Constructor) 패턴이라고도 함 ex) 현대자동차에서 공장 클래스를 만들어 그 클래스에서 자동차를 만들
프로토타입 (Prototype)	원본 객체를 복제하는 방법으로 객체를 생성하는 패턴(클론)
싱글톤 (Singleton)	클래스에서 하나의 객체만 생성할 수 있고 그 하나의 객체를 생성하면 생성된 객체를 어디서든 참조할 수 있지만, 여러 프로세스가 동시에 참조할 수는 없는 패턴 ex) 붕어빵틀(클래스)에서 만든 붕어빵1(인스턴스)만 만들 수 있고 붕어빵틀을 또 사용하더라도 붕어빵2가 아니라 붕어빵1이 생성

⑧구조 패턴(Structural Pattern) : 구조가 복잡한 시스템을 개발하기 쉽도록 클래스나 객체들을 조합하여 더 큰 구조로 만드는 패턴

어댑터 (Adapter)	호환성이 없는 클래스들의 인터페이스를 다른 클래스가 이용할 수 있도록 변환해주는 패턴 ex) 해외여행 갈 때 돼지코 어댑터
브리지 (Bridge)	구현부에서 추상층을 분리하여, 서로가 독립적으로 확장할 수 있도록 구성한 패턴 ex) 모양과 색깔에 따라 다양하게 만들 수 있는 비누가 있다고 치자. 모양과 색깔을 분리해서 생각하는 것
컴포지트 (Composite)	여러 객체를 가진 복합 객체와 단일 객체를 구분 없이 다루고자 할 때 사용하는 패턴 ex) 마인드맵
데코레이터 (Decorator)	객체 간의 결합을 통해 능동적으로 기능들을 확장할 수 있는 패턴 ex) 햄버거(패티, 토마토, 피클, 양상추등을 결합)
퍼싸드 (Facade)	복잡한 서브 클래스들을 피해 더 상위에서 인터페이스를 구성함으로써 서브 클래스들의 기능을 간편하게 사용할 수 있도록 하는 패턴 ex) 은행ARS(입금, 출금, 상담원 연결등의 복잡한 서브클래스를 은행ARS라는 상위 인터페이스를 통해 간편하게 사용할 수 있음)
플라이웨이트 (Flyweight)	인스턴스가 필요할 때마다 매번 생성하는 것이 아니고 가능한 한 공유해서 사용함으로써 메모리를 절약하는 패턴
프록시 (Proxy)	접근이 어려운 객체와 여기에 연결하려는 객체 사이에서 인터페이스 역할을 수행하는 패턴 ex) 신용 카드는 은행 계좌의 프록시(대리)

⑨행위 패턴(Behavioral Pattern) : 클래스나 객체들이 서로 상호작용하는 방법이나 책임 분배 방법을 정의하는 패턴

책임 연쇄 (Chain of Responsibility)	요청을 처리할 수 있는 객체가 둘 이상 존재하여 한 객체가 처리하지 못하면 다음 객체로 넘어가는 형태의 패턴 ex) 그건 저희 부서 소관이 아니니까 다른 부서로 가세요
커맨드 (Command)	요청을 객체의 형태로 캡슐화하여 재이용하거나 취소할 수 있도록 요청에 필요한 정보를 저장하거나 로그에 남기는 패턴 ex) 배달 요청 사항
인터프리터 (Interpreter)	언어에 문법 표현을 정의하는 패턴
반복자 (Iterator)	자료구조와 같이 접근이 잦은 객체에 대해 동일한 인터페이스를 사용하도록 하는 패턴
중재자 (Mediator)	수많은 객체들 간의 복잡한 상호작용을 캡슐화하여 객체로 정의하는 패턴 ex) 공항에서 조종사들이 직접 통신하지 않고 관제탑(중재자)을 통해 통제를 받음
메멘토 (Memento)	특정 시점에서의 객체 내부 상태를 객체화함으로써 이후 요청에 따라 객체를 해당 시점의 상태로 복원할 수 있게 해주는 패턴
옵서버 (Observer)	한 객체의 상태가 변화하면 객체에 상속되어 있는 다른 객체들에게 변화된 상태를 전달하는 패턴 ex) 유튜브 채널 구독 설정을 해놓으면 일일이 그 채널에 찾아가지 않더라도 새 영상을 볼 수 있게 해줌
상태 (State)	객체의 상태에 따라 동일한 동작을 다르게 처리해야 할 때 사용하는 패턴 ex) 키보드에서 한/영기를 누르면 같은 키인데 다르게 처리됨
전략 (Strategy)	동일한 계열의 알고리즘들을 개별적으로 캡슐화하여 상호 교환할 수 있게 정의하는 패턴 ex) 집에 갈 때(동일한 경로) 버스를 탈지, 택시를 탈지 전략적으로 생각하는 것
템플릿 메서드 (Template Method)	상위 클래스에서 골격을 정의하고 하위 클래스에서 세부 처리를 구체화하는 구조의 패턴 ex) 대량 주택 건설(기본적인 틀은 동일하나 집집마다 일부 세부사항들은 조정 가능)
방문자 (Visitor)	각 클래스들의 데이터 구조에서 처리 기능을 분리하여 별도의 클래스로 구성하는 패턴 ex) 고객에 따라 고객 맞춤형으로 각각 다른 제품을 판매하는 보험설계사

◆모듈 설계

요구 분석을 마치면 구조적 방법에서는 DFD, 정보공학 방법에서는 ERD, 객체지향 방법에서는 유스케이스 다이어그램이 산출되고 이런 다이어그램이 중심이 된 요구 명세서가 작성된다.

그러면 설계 단계에서는 맨 먼저 전체 구조를 파악하여 표현하는 상위 설계 또는 아키텍처 설계를 한 후 하위 설계로 모듈 설계를 한다.

▶모듈화

어떤 큰 문제를 그대로 놓고 해결하는 것은 매우 어려운 일이므로 일반적으로 큰 문제를 작은 단위로 쪼개어 그것을 하나씩 해결한다. 소프트웨어 개발에서도 소프트웨어의 성능을 향상시키거나 시스템의 수정 및 재사용, 유지 관리 등을 위해 시스템의 기능들을 모듈 단위로 분해하는데 이를 모듈화라고 한다.

*모듈 : 모듈화를 통해 분리된 시스템의 각 기능으로, 서브루틴, 서브시스템, 소프트웨어 내의 프로그램, 작업 단위 등을 의미

▶모듈의 독립성

모듈이 다른 모듈과의 과도한 상호작용을 배제하고 하나의 기능만을 수행함으로써 이루어져야 한다.

모듈 간에 연관성이 높으면 관련된 모듈을 사용하기 위해 많은 지식이 필요하고 종속적인 관계로 복잡하게 연결되어 유지보수가 매우 어려워지므로 모듈을 분할하여 설계할 때 모듈 간의 관련성이 적게 설계하여 모듈 변경 시 다른 모듈에 영향을 최소화하고 유지보수를 쉽게 할 수 있도록 해야 한다.

이런 독립성을 측정하는 개념이 바로 응집도와 결합도인데 독립성을 높이려면 모듈의 결합도는 약하게, 응집도는 강하게, 모듈의 크기는 작게 만들어야 한다.

▶결합도(Coupling)

모듈 간의 상호 의존하는 정도 또는 두 모듈 사이의 연관 관계

결합도가 약할수록 품질이 높고, 강할수록 품질이 낮다.

결합도는 용돈을 주고받는 부모-자식 관계를 생각하면 쉽게 이해할 수 있다.

자식 입장에서는 용돈을 어디에 쓸 건지 꼬치꼬치 물어보는 부모(=결합도가 강함)보다 그냥 군말 없이 주는 부모(=결합도가 약함)가 좋을 것이다.

따라서 모듈에서 좋은 관계는 깊게 관여하지 않고 데이터(용돈)만 주고받는 관계이다.

내용 결합도	공통 결합도	외부 결합도	제어 결합도	스탬프 결합도	자료 결합도
결합도 강함	<----->				결합도 약함

종류	내용
내용 결합도 (Content Coupling)	한 모듈이 다른 모듈의 내부 기능 및 그 내부 자료를 직접 참조하거나 수정할 때의 결합도 ex) 옆집 이웃이랑 현관문 비밀번호를 공유하면서 교류하는 상황
공통(공유) 결합도 (Common Coupling)	공유되는 공통 데이터 영역을 여러 모듈이 사용할 때의 결합도 ex) 가족들끼리 수건 공유
외부 결합도 (External Coupling)	어떤 모듈에서 선언한 데이터(변수)를 외부의 다른 모듈에서 참조할 때의 결합도 ex) 반찬 만들 시간이 없을 때 반찬가게(외부)에서 반찬을 사서 먹는 상황
제어 결합도 (Control Coupling)	어떤 모듈이 다른 모듈 내부의 논리적인 흐름을 제어하기 위해 제어 신호나 제어 요소를 전달하는 결합도 ex) 너 편식하지 말랬지!!(아이를 제어하는 부모)
스탬프(검인) 결합도 (Stamp Coupling)	모듈 간의 인터페이스로 배열이나 레코드 등의 자료구조가 전달될 때의 결합도 ex) 검수용 스탬프(담당자 인적사항 대신 배열이나 레코드 같은 데이터 구조)
자료 결합도 (Data Coupling)	모듈 간의 인터페이스가 자료 요소로만 구성될 때의 결합도 ex) 군말하지 않고 용돈만 주고받는 부모-자식 관계

▶응집도(Cohesion)

모듈의 내부 요소들이 서로 관련되어 있는 정도

응집도가 강할수록 품질이 높고, 약할수록 품질이 낮다.

응집도는 책 정리를 예로 들 수 있다.

책을 주제에 따라 분류한 것은 응집도가 강하다고 할 수 있고 책을 무지성으로 분류한 것은 응집도가 약하다고 할 수 있다

기능적 응집도	순차적 응집도	교환적 응집도	절차적 응집도	시간적 응집도	논리적 응집도	우연적 응집도
응집도 강함	<----->					응집도 약함

종류	내용
기능적 응집도 (Functional Cohesion)	모듈 내부의 모든 기능 요소들이 단일 문제와 연관되어 수행될 경우의 응집도 ex) 반복적인 작업이 모여 있는 함수
순차적 응집도 (Sequential Cohesion)	모듈 내 하나의 활동으로부터 나온 출력 데이터를 그다음 활동의 입력 데이터로 사용할 경우의 응집도 ex) 1번 식(1+2)의 출력결과를 2번 식 변수a에 입력하는 경우(a+3)
교환(통신)적 응집도 (Communication Cohesion)	동일한 입력과 출력을 사용하여 서로 다른 기능을 수행하는 구성 요소들이 모였을 경우의 응집도 ex) 변수a를 2라고 똑같이 입력받지만 a+2, a+3, a+4 이 3개의 식이 다르게 계산 <순서는 중요하지 않아 순차적 응집보다 약하다>
절차적 응집도 (Procedural Cohesion)	모듈이 다수의 관련 기능을 가질 때 모듈 안의 구성 요소들이 그 기능을 순차적으로 수행할 경우의 응집도 ex) 하의 착용-상의 착용-양말 착용-외투 착용 <순서에 따라 수행되지만 한 요소의 출력이 다음 요소의 입력으로 사용되지 않으므로 순차적 응집보다 약하다>
시간적 응집도 (Temporal Cohesion)	특정 시간에 처리되는 몇 개의 기능을 모아 하나의 모듈로 작성할 경우의 응집도 ex) 출근 시간대(기상-화장실-씻기-식사-옷입기-출근) <순서와 무관하고 출력-입력 관계도 아니다>
논리적 응집도 (Logical Cohesion)	유사한 성격을 갖거나 특정 형태로 분류되는 처리 요소들로 하나의 모듈이 형성되는 경우의 응집도 ex) 수능영어, 편입영어, 토익을 영어 카테고리안에 묶음 <순서와 무관하고 출력-입력 관계도 아니다>
우연적 응집도 (Coincidental Cohesion)	모듈 내부의 각 구성 요소들이 서로 관련 없는 요소로만 구성된 경우의 응집도 ex) 아무렇게나 2줄을 선 후 완전 우연하게 1번째 줄은 a조, 2번째 줄은 b조 배정

2-4) 구현 단계

요구 분석후 생성되는 개체-관계 다이어그램(ERD), 클래스 다이어그램처럼 설계 과정에서 생성된 설계 사양서를 입력 데이터로 활용하여 코딩을 한다.

2-5) 테스트 단계

◆애플리케이션 테스트

애플리케이션에 잠재되어 있는 결함을 찾아내는 일련의 행위 또는 절차

원시 코드 속에 남아 있는 오류를 발견하는 것이기도 하지만 결함이 생기지 않도록 예방하는 과정이기도 하며 개발된 소프트웨어가 고객의 요구를 만족시키는지 확인하는 과정이다.

고객의 요구사항을 확인(Validation)하고 소프트웨어가 기능을 정확히 수행하는지 검증(Verification)한다.

*TMI : 대한민국에서는 모바일을 앱, PC를 프로그램이라고 부르지만 모바일과 PC는 둘 다 응용 프로그램(애플리케이션)이다.

◆애플리케이션 테스트의 기본 원리

완벽한 테스트 불가능	소프트웨어의 잠재적인 결함을 줄일 수 있지만 소프트웨어에 결함이 없다고 증명할 수는 없음
파레토 법칙 (Pareto Principle)	애플리케이션의 20%에 해당하는 코드에서 전체 결함의 80%가 발견된다는 법칙
살충제 패러독스 (Pesticide Paradox)	동일한 테스트 케이스로 동일한 테스트를 반복하면 더 이상 결함이 발견되지 않는 현상
테스팅의 정황(Context) 의존	소프트웨어의 특징, 테스트 환경, 테스터의 역량 등 정황에 따라 테스트 결과가 달라질 수 있으므로, 정황에 따라 테스트를 다르게 수행해야 함
오류-부재의 귀변 (Absence of Errors Fallacy)	소프트웨어의 결함을 모두 제거해도 사용자의 요구사항을 만족시키지 못하면 해당 소프트웨어는 품질이 높다고 말할 수 없는 것
테스트와 위험은 반비례	테스트를 많이 하면 할수록 미래에 발생할 위험을 줄일 수 있음
테스트의 점진적 확대	테스트는 작은 부분에서 시작하여 점점 확대하며 진행해야함
테스트의 별도 팀 수행	테스트는 개발자와 관계없는 별도의 팀에서 수행해야함

◆테스트 절차

- ①테스트 계획
- ②테스트 분석 및 디자인
- ③테스트 케이스 및 시나리오 작성
- ④테스트 수행
- ⑤테스트 결과 평가 및 결과서 작성
- ⑥결함 추적 및 관리

◆결함 관리 절차

- ①에러 발견
- ②에러 등록
- ③에러 분석
- ④결함 확정
- ⑤결함 할당
- ⑥결함 조치
- ⑦결함 조치 검토 및 승인

◆테스트의 분류

테스트는 그 종류가 너무 다양해 혼란을 방지하기 위해 크게 4가지로 분류한다.

ex) 프로그램 실행 여부에 따른 테스트, 테스트 기반에 따른 테스트, 시각에 따른 테스트, 목적에 따른 테스트

▶프로그램 실행 여부에 따른 테스트

정적테스트	프로그램을 실행하지 않고 명세서나 소스 코드를 대상으로 분석하는 테스트 ex) 동료 검토, 워크스루, 인스펙션, 코드 검사 등
동적 테스트	프로그램을 실행하여 오류를 찾는 테스트 ex) 블랙박스 테스트, 화이트박스 테스트

*동료 검토(peer review) : 소프트웨어 개발자의 동료들이 개발자 작업 내역을 검토하는 것

*워크스루(Walk Through) : 소프트웨어 개발자가 모집한 전문가들이 개발자 작업 내역을 검토하는 것

*인스펙션(Inspection) : 워크스루를 발전시킨 형태로, 소프트웨어 개발 단계에서 산출된 결과물의 품질을 평가하고 이를 개선하기 위한 방법을 제시

▶테스트 기반(Test Bases)에 따른 테스트

명세 기반 테스트	사용자의 요구사항에 대한 명세를 빠짐없이 테스트 케이스로 만들어 구현하고 있는지 확인하고 테스트 ex) 블랙박스 테스트
구조 기반 테스트	소프트웨어 내부의 논리 흐름에 따라 테스트 케이스를 작성하고 확인하는 테스트 ex) 화이트박스 테스트
경험 기반 테스트	유사 소프트웨어나 기술 등에 대한 테스트의 경험을 기반으로 수행하는 테스트

▶시각에 따른 테스트

검증(Verification) 테스트	개발자의 시각에서 제품의 생산 과정을 테스트하는 것 (제품이 명세서대로 완성됐는지를 테스트)
확인(Validation) 테스트	사용자의 시각에서 생산된 제품의 결과를 테스트하는 것 (사용자가 요구한 대로 제품이 완성됐는지, 제품이 정상적으로 동작하는지 테스트)

▶목적에 따른 테스트

회복(Recovery) 테스트	시스템에 여러 가지 결함을 주어 실패하도록 한 후 올바르게 복구되는지 확인하는 테스트
안전(Security) 테스트	시스템에 설치된 시스템 보호 도구가 불법적인 침입으로부터 시스템을 보호할 수 있는지를 확인하는 테스트
강도(Stress) 테스트	시스템에 과도한 정보량이나 빈도 등을 부과하여 과부하 시에도 소프트웨어가 정상적으로 실행되는지를 확인하는 테스트
성능(Performance) 테스트	소프트웨어의 실시간 성능이나 전체적인 효율성을 진단하는 테스트로, 소프트웨어의 응답 시간, 처리량 등을 테스트
구조(Structure) 테스트	소프트웨어 내부의 논리적인 경로, 소스 코드의 복잡도 등을 평가하는 테스트
회귀(Regression) 테스트	소프트웨어의 변경 또는 수정된 코드에 새로운 결함이 없음을 확인하는 테스트
병행(Parallel) 테스트	변경된 소프트웨어와 기존 소프트웨어에 동일한 데이터를 입력하여 결과를 비교하는 테스트

㉠ 화이트박스 테스트(White Box Test)

모듈의 원시 코드를 오픈시킨 상태에서 원시 코드의 논리적인 모든 경로를 테스트하여 테스트 케이스를 설계하는 방법으로 코드 기반 테스트라고도 한다.

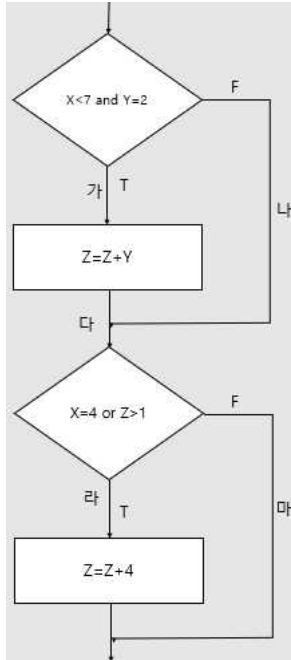
테스트에서 프로그램 코드의 가능한 경로를 모두 테스트하기는 힘들다. 그렇다면 프로그램 코드의 일부 경로만 정해 테스트를 해야하는데 어떤 대상을 테스트를 해야 할까.

그럴 때 필요한 것이 바로 검증 기준이다

검증 기준은 테스트 케이스들이 테스트에 얼마나 적절한지 판단하는 기준이다.

♠ 화이트박스 테스트 검증 기준

㉠문장 검증 기준(Statement Coverage) : 소스 코드의 모든 구문이 한 번 이상 수행되도록 테스트 케이스를 설계



왼쪽 그림은 제어 흐름(Control Flow)으로 표현한 그래프인데 T는 True(참)을 의미하고 F는 False(거짓)를 의미한다. 마름모는 입력받은 값 x, y에서 x는 7보다 작고 y=2가 맞는가? 라고 생각하면 된다.

and는 x<7과 Y=2가 둘 다 참이어야 참으로 반환하고 둘 중 하나만 참이면 거짓으로 반환한다.

or은 x<7 과 Y=2 둘 중 하나만 참이여도 참으로 반환한다.

왼쪽 그림에서 모든 문장을 거치는 루트는 가-다-라 루트뿐이므로 선택한 루트에 해당하는 테스트 데이터를 가지고 실행해본다.

맨 처음 입력값(테스트 데이터)를 X=4, Y=2, Z=0를 넣고 실행한다.

모든 문장을 통과하기 때문에(모두 참) 문장 검증 기준을 만족한다. 그런데 첫 번째 조건문에서 and를 실수로 or로 바꿨다고 해보자.

and를 or로 바꿔봤자 모든 문장이 통과하기 때문에 문장 검증 기준을 만족한다.

문장 검증 기준은 이러한 오류를 발견하지 못하므로 다른 검증 기준에 비해 약하고 최소한의 테스트라고 할 수 있다.

㉡분기 검증 기준(Branch Coverage) : 소스 코드의 모든 조건문이 한 번 이상 수행되도록 테스트 케이스를 설계
결정 검증 기준이라고도 하며 문장 검증 기준의 문제점을 해결할 수 있다.

테스트 케이스를 선정하는 기준은 원시 코드에 존재하는 조건문에 대해 T가 되는 경우와 F가 되는 경우가 최소한 한 번은 실행되게 입력 데이터를 테스트 케이스로 사용하는 것이다. 위의 예시같은 경우는 T가 되는 경우와 F가 되는 경우를 모두 거치게 되는 1개의 경로가 없으므로 2개의 테스트 케이스를 합친다.(가-다-라 루트와 나-마 루트를 같이 생각하는 것이다)

그런데 X=4 or Z>1이라는 조건문을 보자. Z>1이 어떤 식으로 바뀌어도 or이므로 X=4이기만 한다면 항상 참이다. 그래서 Z에 관한 식의 오류를 발견할 수 없다.

㉢조건 검증 기준(Condition Coverage) : 소스 코드의 모든 조건문에 대해 조건이 True인 경우와 False인 경우가 한 번 이상 수행되도록 테스트 케이스를 설계

분기 검증 기준의 문제점을 해결할 수 있다.

분기 검증 기준에서는 X=4 or Z>1이라는 조건문이 있으면 “X=4 or Z>1” 전체의 조건문만 판단하는데 조건 검증 기준에서는 “X=4”, “Z>1”이런 식으로 개별 조건식의 T와 F가 최소한 한 번은 수행할 수 있도록 하는 테스트 케이스를 설정한다.

㉣분기/조건 기준(Branch/ Condition Coverage) : 소스 코드의 모든 조건문과 각 조건문에 포함된 개별 조건식의 결과가 True인 경우와 False인 경우가 한 번 이상 수행되도록 테스트 케이스를 설계
즉, 개별 조건식을 모두 만족하면서 전체 조건식도 만족하는 테스트 케이스를 설정한다.

♠ 화이트박스 테스트 종류

(ㄱ) 기초 경로 검사(Base path Testing) : 메케이브(McCabe)가 만든 것으로 테스트 케이스 설계자가 절차적 설계의 논리적 복잡성을 측정할 수 있게 해주는 테스트 기법

원시 코드의 독립적인 경로가 최소한 한 번은 실행되는 테스트 케이스를 찾아 테스트를 수행하는 것이다.

(ㄴ) 제어 구조 검사(Control Structure Testing)

·조건 검사(Condition Testing) : 프로그램 모듈 내에 있는 논리적 조건을 테스트하는 테스트 케이스 설계 기법

·루프 검사(Loop Testing) : 프로그램 반복(Loop) 구조에 초점을 맞춰 실시하는 테스트 케이스 설계 기법

·데이터 흐름 검사(Data Flow Testing) : 프로그램에서 변수의 정의와 변수 사용의 위치에 초점을 맞춰 실시하는 테스트 케이스 설계 기법

㉠ 블랙박스 테스트(Black Box Test)

소프트웨어가 수행할 특정 기능을 알기 위해서 각 기능이 완전히 작동되는 것을 입증하는 테스트로 프로그램 내부의 구조나 알고리즘을 보지 않고 요구 분석 명세서나 설계 사양서에 테스트 케이스를 추출하여 테스트한다.

♠ 블랙박스 테스트 종류

㉡ 동치 분할 검사(Equivalence Partitioning Testing) 또는 동등 분할 기법

프로그램의 입력 조건에 타당한 입력 자료와 타당하지 않은 입력 자료의 개수를 균등하게 하여 테스트 케이스를 정하고, 해당 입력 자료에 맞는 결과가 출력되는지 확인하는 기법

ex) 수능 영어(90~100은 1등급, 80~89는 2등급 이런 식으로 동등 분할에서 대푯값을 이용해 테스트 케이스를 도출한다)

㉢ 경계값 분석(Boundary Value Analysis)

입력 조건의 중간값보다 경계값에서 오류가 발생 될 확률이 높다는 점을 이용하여 입력 조건의 경계값을 테스트 케이스로 선정하여 검사하는 기법

이상, 이하, 미만, 초과 같은 표현의 실수로 생기는 문제는 경계값에서 문제가 발생할 확률이 높으므로 경계값을 분석한다. 동등 분할 후 분할된 클래스의 경계값에 근거하여 테스트 케이스를 작성한다.

㉣ 원인-효과 그래프 검사(Cause-Effect Graphing Testing)

입력 데이터 간의 관계와 출력에 영향을 미치는 상황을 체계적으로 분석한 다음 효용성이 높은 테스트 케이스를 선정하여 검사하는 기법

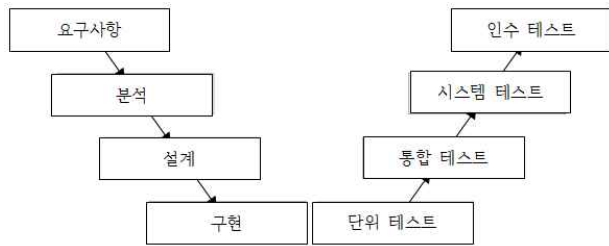
㉤ 오류 예측 검사(Error Guessing)

과거의 경험이나 확인자의 감각으로 테스트하는 기법

㉥ 비교 검사(Comparison Testing)

여러 버전의 프로그램에 동일한 테스트 자료를 제공하여 동일한 결과가 출력되는지 테스트하는 기법

◆개발 단계에 따른 애플리케이션 테스트



요구사항 정의, 분석, 설계, 구현 단계와 테스트 과정의 단위 테스트, 통합 테스트, 시스템 테스트, 인수 테스트가 V자 형태로 연결된다고 해서 개발 단계에 따른 테스트를 V모델이라고도 한다.

▶단위 테스트(Unit Test)

코딩 직후 소프트웨어 설계의 최소 단위인 모듈이나 컴포넌트에 초점을 맞춰 테스트

모듈 테스트라고도 하며 모듈 내부의 구조를 구체적으로 들여다볼 수 있는 화이트박스 테스트 같은 구조 기반 테스트를 주로 시행한다.

하나의 모듈을 테스트할 때 상위나 하위 모듈이 개발이 안된 경우도 있다. 이 때 상위나 하위 모듈이 개발될 때까지 기다릴 수 없으므로 가상의 상위나 하위 모듈을 만들어 사용해야 하는데 가상 상위 모듈을 테스트 드라이버(Driver)라 하고 가상 하위 모듈을 테스트 스텝(Stub)이라고 한다.

▶통합 테스트(Integration Test)

단위 테스트가 완료된 모듈들을 결합하여 하나의 시스템으로 완성시키는 과정에서의 테스트

단위 모듈이 개별적으로 존재하는게 아니라 여러 모듈이 유기적 관계를 맺고 있으므로 이러한 모듈들을 결합한 형태로 테스트를 수행해야 한다.

이처럼 단위 테스트가 끝난 모듈을 통합하는 과정에서 발생하는 오류 및 결함을 찾는 테스트 기법이다

통합 테스트는 모듈 통합을 한꺼번에 하는 비점진적 통합 방식과 단계적으로 통합하는 점진적 통합 방식이 있다.

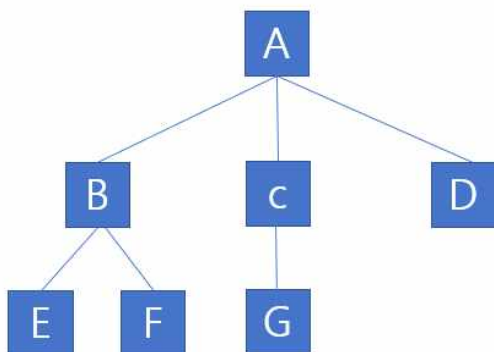
비점진적 통합 방식은 빅뱅 테스트를 예를 들 수 있는데 단위 테스트가 끝난 모듈을 한꺼번에 결합하여 수행하는 방식이다. 그러나 한꺼번에 통합하면 오류가 발생했을 때 어떤 모듈에서 오류가 존재하고 또 그 원인이 무엇인지 찾기 어렵기 때문에 완성된 모듈을 기존에 테스트된 모듈과 하나씩 통합하며 테스트를 한다.

점진적 통합 방식으로는 하향식 통합 테스트, 상향식 통합 테스트, 혼합식 통합 테스트가 있다.

@하향식 통합 테스트(Top Down Integration Test)

프로그램의 상위 모듈에서 하위 모듈 방향으로 통합하면서 테스트하는 기법

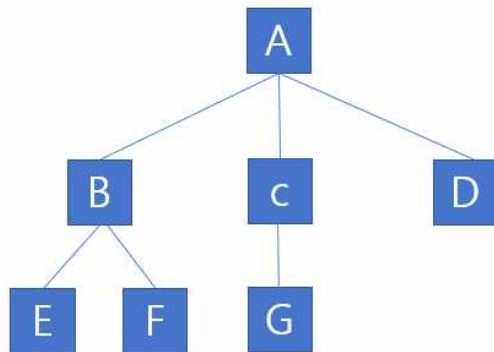
아래 그림에서 맨 상위 모듈 A를 모듈 B와 통합하여 테스트하고 그 다음으로 모듈 C를 먼저 할지 E를 먼저할지 결정해야 하는데 C를 먼저 선택하는 방식을 넓이 우선(Breadth first) 방식이라 하고 모듈 E를 먼저 선택하는 방식을 깊이 우선(Depth first) 방식이라 한다.



⑤상향식 통합 테스트(Bottom Up Integration Test)

프로그램의 하위 모듈에서 상위 모듈 방향으로 통합하면서 테스트하는 기법

아래 그림에서 모듈 E와 F를 모듈 B에 통합하여 테스트하고 그다음 모듈 G를 모듈 C에 통합하여 테스트하고 마지막으로 모듈 B, C, D를 모듈 A에 통합하여 테스트한다.



⑥혼합식 통합 테스트(샌드위치식 통합 테스트 방법)

하위 수준에서는 상향식 통합, 상위 수준에서는 하향식 통합을 사용하여 최적의 테스트를 지원하는 방식

*회귀 테스트(Regression Test)

통합 테스트로 인해 변경된 모듈이나 컴포넌트에 새로운 오류가 있는지 확인하는 테스트로 이미 테스트된 프로그램을 반복한다. 한 모듈의 수정이 다른 부분에 영향을 끼칠 수 있다고 생각하여 수정된 모듈뿐 아니라 관련된 모듈까지 문제가 없는지 테스트하는 것이다.

회귀 테스트는 2가지가 있는데 모든 테스트를 완료하여 사용자에게 전달하기 전 미처 발견하지 못한 오류를 찾아 수정한 후 다시 테스트하는 수정을 위한 회귀 테스트(Corrective regression test)와 사용 중에 일부 기능을 추가하여 새로운 버전을 만들고, 이 새 버전을 다시 테스트하는 점진적 회귀 테스트(progressive regression test)가 있다.

▶시스템 테스트(System Test)

개발된 소프트웨어가 해당 컴퓨터 시스템에서 완벽하게 수행되는가를 점검하는 테스트

시스템 테스트는 모듈이 모두 통합된 후 사용자의 요구사항들을 만족하는지 테스트하는 것이다.

즉, 개발자의 시각에서 확인하는 검증(Verification) 과정이다.

▶인수 테스트(Acceptance Test)

개발한 소프트웨어가 사용자의 요구사항을 충족하는지에 중점을 두고 테스트하는 방법이다.

즉, 사용자의 시각에서 확인하는 확인(Validation) 과정이다.

알파 테스트(Alpha Test)	개발자의 장소에서 사용자가 개발자 앞에서 행하는 테스트 기법 내부 필드 테스트라고도 하는데 개발 완료된 소프트웨어를 베타 테스트 전에 회사 내의 다른 직원에게 개발자 환경에서 사용해보도록 하고, 그들이 사용하는 것을 보면서 오류와 사용상의 문제점을 파악한다.
베타 테스트(Beta Test)	선정된 최종 사용자가 여러 명의 사용자 앞에서 행하는 테스트 기법 개발 완료되어 알파 테스트를 거친 소프트웨어를 특정 사용자에게 미리 사용해보도록 배포하는데, 베타 버전을 받은 사용자들이 사용자 입장에서 문제점이나 오류를 발견하면 개발자에게 알려주는 방식이다.

◆테스트 단계 기타 용어 정리

▶테스트 시나리오(Test Scenario) : 테스트 케이스를 적용하는 순서에 따라 여러 개의 테스트케이스를 묶은 집합

▶테스트 오라클(Test Oracle) : 테스트 결과가 올바른지 판단하기 위해 사전에 정의된 참 값을 대입하여 비교하는 기법

ex) 참 오라클, 샘플링 오라클, 추정 오라클, 일관성 검사 오라클

- 참(True) 오라클 : 모든 테스트 케이스의 입력값에 대해 기대하는 결과를 제공하는 오라클
- 샘플링(Sampling) 오라클 : 특정한 몇몇 테스트 케이스의 입력값들에 대해서만 기대하는 결과를 제공하는 오라클
- 추정(Heuristic) 오라클 : 특정 테스트 케이스의 입력값에 대해 기대하는 결과를 제공하고 나머지 입력값들에 대해서는 추정으로 처리하는 오라클
- 일관성 검사(Consistent) 오라클 : 애플리케이션에 변경이 있을 때, 테스트 케이스의 수행 전과 후의 결과값이 동일한지를 확인하는 오라클

▶테스트 하네스(Test Harness Tools)

테스트가 실행될 환경을 시뮬레이션 하여 컴포넌트 및 모듈이 정상적으로 테스트 되도록 하는 도구

▶테스트 하네스 구성요소

- 테스트 드라이버(Test Driver) : 테스트 대상의 하위 모듈을 호출하고, 파라미터를 전달하고, 모듈 테스트 수행 후의 결과를 도출하는 도구
- 테스트 스텝(Test Stub) : 제어 모듈이 호출하는 타 모듈의 기능을 단순히 수행하는 도구
- 테스트 슈트(Test Suites) : 테스트 대상 컴포넌트나 모듈, 시스템에 사용되는 테스트 케이스의 집합
- 테스트 케이스(Test Case) : 소프트웨어가 사용자의 요구사항을 정확하게 준수했는지 확인하기 위해 설계된 입력값, 실행 조건, 기대 결과 등으로 구성된 테스트 항목에 대한 명세서
- 테스트 스크립트(Test Script) : 자동화된 테스트 실행 절차에 대한 명세서
- Mock 오브젝트(Mock Object) : 사전에 사용자의 행위를 조건부로 입력해 두면, 그 상황에 맞는 예정된 행위를 수행하는 객체

*테스트 슈트와 테스트 시나리오의 차이 : 테스트 슈트가 단순한 묶음이라면 테스트 시나리오는 테스트 케이스의 동작 순서에 따른 묶음

2-6) 품질

◆품질

얼마나 좋은지, 나쁜지를 나타내는 정도를 나타내는 단어로 소프트웨어에는 요구 분석 명세서에 서술된 기능과 성능을 만족해야 한다는 것이 핵심이다.

개발자 입장에서는 프로그램에 결함이 없고 사용자의 요구사항을 정의한 요구 분석 명세서대로 만든 소프트웨어를 품질 좋은 소프트웨어라 할 수 있다.

소프트웨어가 좋은 품질이라는 목적을 달성하려면 품질을 고려한 계획을 세우고, 품질 요구사항에 대한 명세서가 작성되어야 하며, 단계별로 생산되는 산출물도 검사 항목에 따라 철저히 점검해야 한다.

또한 품질을 보증할 수 있는 프로세스를 마련해, 품질을 체크할 수 있는 과정을 명확히 하도록 해야 한다.

일반적으로 소프트웨어 품질 평가는 소프트웨어 제품 품질 특성 평가, 프로세스 품질 특성 평가로 나눌 수 있다.

제품 품질 특성 평가는 완성된 제품에 대한 평가이고 프로세스 품질 특성 평가는 소프트웨어 개발 과정의 각 단계마다 하는 평가이다.

♠소프트웨어 개발 표준

소프트웨어 개발 단계에서 수행하는 품질 관리에 사용되는 국제 표준

표준 프로세스는 음식의 레시피같은 것으로 기준과 목표, 방향을 제시해주기 때문에 몰라서 헤매는 시간을 줄여주고 생산성을 높인다.

ex) ISO/IEC 12207, CMMI, SPICE

▶ISO/IEC 12207

ISO(국제표준화기구)에서 만든 표준 소프트웨어 생명 주기 프로세스

▶CMMI(Capability Maturity Model Integration)

소프트웨어 개발 조직의 업무 능력 및 조직의 성숙도를 평가하는 모델

CMMI는 조직의 프로세스 개선을 위해 카네기멜론 대학의 소프트웨어 광학연구소(SEI)에서 개발되었는데 만약 CMMI와 같은 표준이 없다면 프로젝트 리더의 역량에 따라 결과가 달라진다.

●CMMI의 5단계(소프트웨어 프로세스 성숙도)

단계	특징
초기(Initial)	작업자 능력에 따라 성공 여부 결정
관리(Managed)	특정한 프로젝트 내의 프로세스 정의 및 수행
정의(Defined)	조직의 표준 프로세스를 활용하여 업무 수행
정량적 관리(Quantitatively)	프로젝트를 정량적으로 관리 및 통제
최적화(Optimizing)	프로세스 역량 향상을 위해 지속적인 프로세스 개선

▶SPICE(Software Process Improvement and Capability dEtermination)-정식명칭 : ISO/IEC 15504

소프트웨어 품질 및 생산성 향상을 위해 소프트웨어 프로세스를 평가 및 개선하는 국제 표준

◆ 애플리케이션 성능 개선

▶ 소스코드 최적화

나쁜 코드(Bad Code)를 배제하고 클린 코드(Clean Code)로 작성하는 것

- 클린 코드(Clean Code) : 누구나 쉽게 이해하고 수정 및 추가할 수 있는 단순 명료한 코드
- 나쁜 코드(Bad Code) : 프로그램의 로직(Logic)이 복잡하고 이해하기 어려운 코드

ex) 스파게티 코드, 외계인 코드

*스�파게티 코드 : 코드의 로직이 서로 복잡하게 얽혀 있는 코드

*외계인 코드 : 아주 오래되거나 참고문서 또는 개발자가 없어 유지보수 작업이 어려운 코드

▶ 클린 코드 작성 원칙

가독성, 단순성, 의존성 배제, 중복성 최소화, 추상화

▶ 소스 코드 품질 분석 도구

소스 코드의 코딩 스타일, 코드에 설정된 코딩 표준, 코드의 복잡도, 코드에 존재하는 메모리 누수 현상, 스레드 결함 등을 발견하기 위해 사용하는 분석 도구

▶ 소스 코드 품질 분석 도구의 분류

- 정적 분석 도구 : 작성한 소스 코드를 실행하지 않고 코딩 표준이나 코딩 스타일, 결함 등을 확인

ex) pmd, cppcheck, SonarQube, checkstyle, com, cobertura 등

- 동적 분석 도구 : 작성한 소스 코드를 실행하여 코드에 존재하는 메모리 누수, 스레드 결함 등을 분석

ex) Avalanche, Valgrind 등

*유지보수 단계는 별 내용이 없어서 생략

2-7) 소프트웨어 패키징

◆소프트웨어 패키징

모듈별로 생성한 실행 파일들을 묶어 배포용 설치 파일을 만드는 것
개발자가 아니라 사용자 중심으로 진행한다.

◆릴리즈 노트(Release Note)

소프트웨어 개발 과정에서 정리된 릴리즈 정보를 최종 사용자인 고객과 공유하기 위한 문서

◆형상 관리(SCM: Software Configuration Management)

개발과정에서 소프트웨어의 변경 사항을 관리하기 위해 개발된 일련의 활동
형상 관리는 소프트웨어 개발의 전 단계에 적용되는 활동이며, 유지보수 단계에서도 수행된다.

◆소프트웨어 버전 관리 도구

●공유 폴더 방식

버전 관리 자료가 지역 컴퓨터와 공유 폴더에 저장되어 관리 되는 방식
ex) SCCS, RCS, PVCS, QVCS 등

●클라이언트/ 서버 방식

버전 관리 자료가 서버에 저장되어 관리 되는 방식
ex) CVS, SVN(Subversion), CVSNT, Clear Case, CMBC, Perforce 등

●분산 저장소 방식

버전 관리 자료가 하나의 원격 저장소와 분산된 개발자 PC의 지역 저장소에 함께 저장되어 관리되는 방식
ex) Git, GNU arch, DVC, Bazaar, Mercurial, TeamWare, Bitkeeper, Plastic SCM 등

◆Subversion(서브버전, SVN)

CVS를 개선한 것이고 클라이언트/ 서버 구조이며 서버(저장소, Repository)에는 최신 버전의 파일들과 변경 사항이 관리 된다.

◆Git(깃)

리누스 토발즈(Linus Torvalds)가 2005년 리눅스 커널 개발에 사용할 관리 도구로 개발한 이후 주니오 하마노(Junio Hamano)에 의해 유지 보수되고 있으며 분산 버전 관리 시스템으로 2개의 저장소, 즉 지역 저장소와 원격 저장소가 존재한다.

버전 관리가 지역 저장소에서 진행되므로 버전 관리가 신속하게 처리되고 원격 저장소나 네트워크에 문제가 있어도 작업이 가능하다.

3) 화면 설계

화면 설계는 말그대로 우리가 휴대폰이나 컴퓨터의 화면이 어떻게 보이면 좋을지 설계하는 것이다.

설계라서 설계 단계에서 설명을 해야 되지만 앞에 있는 내용과 따로 떼어놔도 문제가 없을 것 같고 화면 설계라는 파트로 따로 정리하고 싶어서 분리를 했다.

◆사용자 인터페이스(UI, User Interface)

사용자와 시스템 간의 상호작용이 원활하게 이뤄지도록 도와주는 장치나 소프트웨어

◆UI 종류

- CLI(Command Line Interface) : 명령과 출력이 텍스트 형태로 이뤄지는 인터페이스
- GUI(Graphical User Interface) : 아이콘이나 메뉴를 마우스로 선택하여 작업을 수행하는 그래픽 환경의 인터페이스
- NUI(Natural User Interface) : 사용자의 말이나 행동으로 기기를 조작하는 인터페이스
- OUI(Organic User Interface) : 현실에 존재하는 모든 사물이 입출력장치로 변화할 수 있는 인터페이스

◆UI 기본원칙 4가지

- 직관성(Intuitiveness) : 누구나 쉽게 이해하고 사용 가능해야 함
- 유효성(Consistency) : 사용자의 목적을 정확하고 완벽하게 달성해야 함
- 학습성(Learnability) : 누구나 쉽게 배우고 익힐 수 있어야 함
- 유연성(Flexibility) : 사용자의 요구사항을 최대한 수용하고 실수를 최소화해야 함

◆UI 설계 도구

- 와이어프레임(Wireframe)

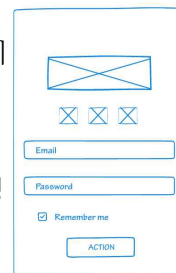
페이지에 대한 개략적인 레이아웃이나 UI 요소 등에 대한 뼈대를 설계하는 도구(쉽게 밑그림, 스케치라고 생각하면 될 듯)

- 목업(Mockup)

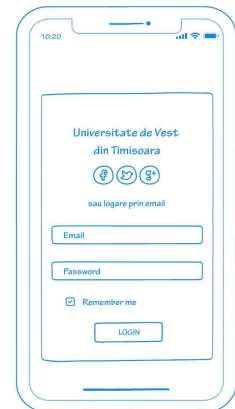
와이어프레임보다 좀 더 실제 화면과 유사하게 만든 정적인 형태의 모형

- 스토리보드(Story Board)

와이어프레임에 콘텐츠에 대한 설명, 페이지 간 이동 흐름 등을 추가한 문서



Low Fidelity Wireframe



High Fidelity Wireframe

- 프로토타입(Prototype)

와이어프레임이나 스토리보드 등에 움직임을 적용함으로써 실제 구현된 것처럼 테스트가 가능한 동적인 형태의 모형으로 손으로 직접 작성하는 페이퍼 프로토타입과 파워포인트 같은 디지털 프로토타입이 있다.

출처: 위키미디어

◆HCI(Human Computer Interaction or Interface)

사람이 시스템을 보다 편리하고 안전하게 사용할 수 있도록 연구하고 개발하는 학문

◆UX(User Experience)

사용자가 시스템이나 서비스를 이용하면서 느끼고 생각하게 되는 총체적인 경험

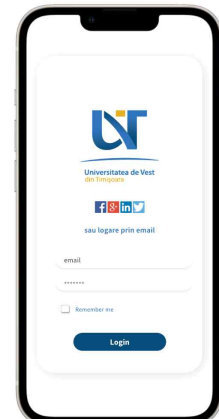
- 특징 : 주관성(Subjectivity), 정황성(Contextuality), 총체성(Holistic)

◆감성공학

제품이나 작업환경을 사용자의 감성에 알맞도록 설계 및 제작하는 기술



Mockup



Prototype

출처: 위키미디어

4) 기타 용어

◆소프트웨어 재사용(Software Reuse)

이미 개발되어 인정받은 소프트웨어를 다른 소프트웨어 개발이나 유지에 사용하는 것

ex) 합성 중심, 생성 중심

합성 중심 (Composition-Based)	전자 칩과 같은 소프트웨어 부품, 즉 블록을 만들어서 끼워 맞춰 소프트웨어를 완성시키는 방법으로, 블록 구성 방법이라고도 함
생성 중심 (Generation-Based)	추상화 형태로 써진 명세를 구체화하여 프로그램을 만드는 방법으로, 패턴 구성 방법이라고도 함

◆소프트웨어 재공학(Software Reengineering)

기존 시스템을 이용하여 보다 나은 시스템을 구축하고 새로운 기능을 추가하여 소프트웨어 성능을 향상시키는 것

◆CASE(Computer Aided Software Engineering)

소프트웨어 개발 과정에서 사용되는 요구 분석, 설계, 구현, 검사 및 디버깅 과정 전체 또는 일부를 컴퓨터와 전용 소프트웨어 도구를 사용하여 자동화하는 것

◆소프트웨어 개발 프레임워크

소프트웨어 개발에 공통적으로 사용되는 구성요소와 아키텍처를 일반화하여 손쉽게 구현할 수 있는 여러 가지 기능들을 제공해주는 반제품 형태의 소프트웨어 시스템

*반제품 : 완제품의 재료로 사용되기 위해 원료를 가공하여 만든 중간 제품

ex) 스프링 프레임워크, 전자정부 프레임워크, 닷넷 프레임워크

●소프트웨어 개발 프레임워크의 특성

특성	내용
모듈화(Modularity)	캡슐화를 통해 모듈화를 강화하고 설계 및 구현의 변경에 따른 영향을 최소화함으로써 소프트웨어의 품질을 향상
재사용성(Reusability)	재사용 가능한 모듈들을 제공함으로써 예산 절감, 생산성 향상, 품질 보증 가능
확장성(Extensibility)	다형성을 통한 인터페이스 확장이 가능하여 다양한 형태와 기능을 가진 애플리케이션 개발 가능
제어의 역흐름(Inversion of Control)	개발자가 관리하고 통제해야 하는 객체들의 제어를 프레임워크에 넘김으로써 생산성 향상

◆순환 복잡도(Cyclomatic Complexity)

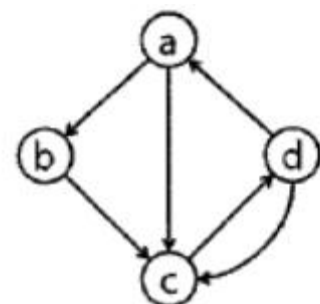
한 프로그램의 논리적인 복잡도를 측정하기 위한 소프트웨어의 척도

맥케이브 순환도(McCabe's Cyclomatic) 또는 맥케이브 복잡도 메트릭(McCabe's Complexity Metrics)라고도 한다.

예제1) 제어 흐름 그래프가 다음과 같을 때 McCabe의 cyclomatic 수는 얼마인가?

답 : 4

풀이 : 순환 복잡도=화살표 수(6) - 노드의 수(4) + 2 = 2



출처 : 2020년 8월 정보처리기사 필기

II. 데이터베이스(DB) 파트

◆데이터와 정보

데이터(Data)	현실 세계에서 단순히 관찰하거나 측정하여 수집한 사실이나 값
정보(Information)	의사 결정에 유용하게 활용할 수 있도록 데이터를 처리한 결과물

예를 들어 나무가 데이터라면 종이가 정보가 되는 것이다.

◆데이터베이스

특정 조직의 여러 사용자가 공유하여 사용할 수 있도록 통합해서 저장한 운영 데이터의 집합
실시간 접근이 가능하고 계속 변화하여 동시 공유가 가능하고 내용으로 참조가 가능하다.

◆DBMS(DataBase Management System; 데이터베이스 관리 시스템)

사용자의 요구에 따라 정보를 생성해주고, 데이터베이스를 관리해주는 소프트웨어
파일 시스템이 갖는 데이터의 종속성과 중복성의 문제를 해결하기 위해 제안되었다.

데이터의 종속성 (Data Dependency)	프로그램의 구조가 데이터의 구조에 영향을 받는 것을 의미하는데 데이터의 구조가 프로그램의 데이터 저장방식을 결정하고 반대로 프로그램의 데이터 저장방식에 따라 데이터의 저장방식이 바뀌는 것을 말한다. 데이터의 종속성 때문에 데이터의 구조가 변경되면 프로그램까지 같이 바뀌는 비용이 들기 때문에 프로그램 개발과 유지보수가 어려워진다.
데이터의 중복성 (Data Redundancy)	파일 시스템은 프로그램마다 데이터 종속성 등으로 인해 공유가 안되는 경우가 많아서 프로그램마다 같은 정보를 중복해서 저장하는 경우가 많다. 이는 저장공간의 낭비이기도 하지만 데이터를 관리하는 측면에서 같은 정보를 여러 곳에서 보관하면 수정 시에 모든 데이터를 수정해야 하는 문제가 발생한다.

▶DBMS의 필수 기능 3가지

정의(Definition)	데이터의 형(Type)과 구조에 대한 정의, 이용 방식, 제약조건 등을 명시
조작(Manipulation)	데이터 검색, 갱신, 삽입, 삭제 등을 위한 인터페이스 수단 제공
제어(Control)	데이터의 무결성, 보안, 권한 검사, 병행 제어를 제공하는 기능

◆데이터베이스 시스템(DBS; DataBase System)

데이터베이스에 데이터를 저장하고 저장된 데이터를 관리하여 조직에 필요한 정보를 생성해주는 시스템

*DB, DBMS, DBS : 데이터를 저장해두는 곳이 DB, DB를 관리하는 곳이 DBMS 마지막으로 DBS는 DB와 DBMS를 이용해 조직에 필요한 정보를 제공해주는 전체 시스템이다.

◆데이터베이스 구조

1) 스키마 : 데이터베이스의 구조와 제약조건을 정의한 것

아래가 스키마 예시인데 고객 번호는 정수로 이름은 최대 10자의 문자열로, 나이는 정수로, 주소는 최대 20자의 문자열만 허용하기로 했다는 내용이 담겨있다.

정의된 스키마에 따라 데이터베이스에 실제로 저장된 값은 인스턴스(Instance)라고 한다.

고객번호	이름	나이	주소
INT	CHAR(10)	INT	CHAR(20)

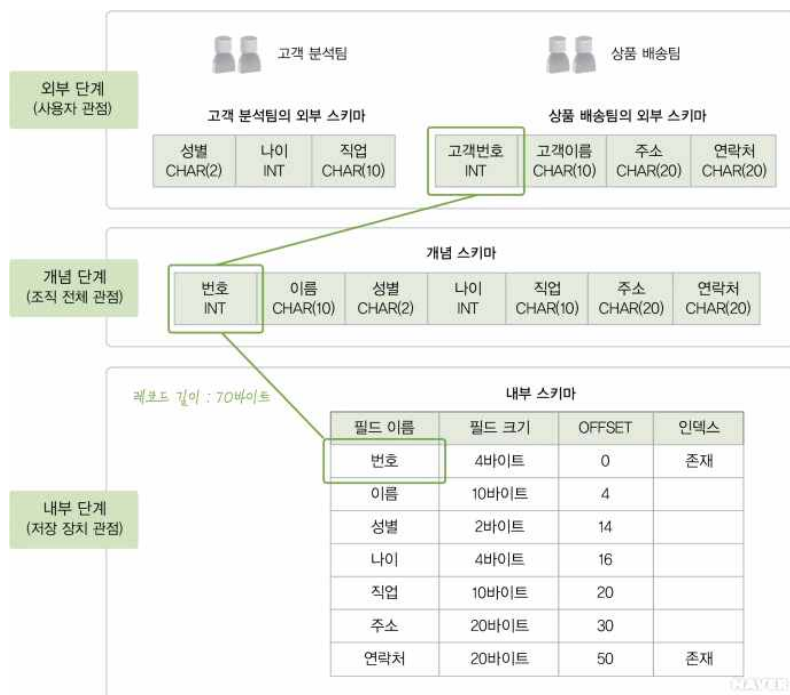
2) 3단계 데이터베이스 구조

하나의 데이터베이스를 세 단계로 나누어 이해하는 게 개별 사용자 관점에서 바라보는 외부 단계(external level), 조직 전체의 관점에서 바라보는 개념 단계(conceptual level), 물리적 저장 장치의 관점에서 바라보는 내부 단계(internal level)로 나눈다.

3단계를 아파트로 예를 들어 설명할 수 있다.

외부 단계는 집주인 관점인데 집주인이 매매한 아파트 호수만 보는 것이고 개념 단계는 관리인 관점인데 아파트 전체를 관리하기 위해 아파트 전체 호수를 보는 것이고 마지막으로 내부 단계는 건설 업체 관점으로 아파트가 어떤 구조, 어떤 방식으로 설계했는지까지 보는 것이다.

외부 스키마	외부 단계에서 사용자나 응용 프로그래머가 각 개인의 입장에서 필요로 하는 데이터베이스의 논리적 구조로 정의한 것 서브 스키마라고도 한다.
개념 스키마	데이터베이스의 전체적인 논리적 구조로 모든 사용자가 생각하는 데이터베이스의 모습을 하나로 합친 모습이다.
내부 스키마	물리적 저장장치의 입장에서 본 데이터베이스 구조



출처 : 내부 단계 (naver.com)

◆데이터의 독립성

데이터베이스를 3단계 구조로 나누고 단계별로 스키마를 유지하며 스키마 사이의 대응 관계를 정의하는 궁극적인 목적은 데이터의 독립성을 실현하기 위해서이다.

논리적 독립성	데이터의 논리적 구조를 변경시키더라도 응용 프로그램은 영향을 받지 않음
물리적 독립성	디스크를 추가/변경하더라도 응용 프로그램은 영향을 받지 않음

*데이터 사전(data dictionary)/시스템 카탈로그(system catalog) : 데이터베이스에 저장되는 데이터에 관한 정보를 저장하는 곳, 데이터 사전은 데이터에 관한 정보이므로 메타 데이터(meta data)라고도 한다.

◆데이터 언어

데이터 정의어(DDL)	스키마를 정의하거나 수정 또는 삭제하기 위해 사용
데이터 조작어(DML)	데이터의 삽입, 삭제, 수정, 검색 등의 처리를 요구하기 위해 사용
데이터 제어어(DCL)	동시 공유가 가능하면서도 무결성과 일관성을 유지하도록 내부적으로 필요한 규칙이나 기법을 정의하기 위해 사용

◆데이터 모델링

현실 세계에 존재하는 데이터를 컴퓨터 세계의 데이터베이스로 옮기는 변환 과정

▶개념적 모델링(conceptual modeling)

현실 세계에 존재하는 개체의 중요한 데이터를 추출하여 개념 세계로 옮기는 작업

ex) 원숭이를 예를 들면 엉덩이가 빨강다, 갈색이다 등등

▶논리적 모델링(logical modeling)

개념 세계의 데이터를 데이터베이스에 저장할 구조를 결정하고 이 구조로 표현하는 작업

ex) 엉덩이가 빨강다, 갈색이다 이런 것들을 표로 정리

◆데이터 모델

현실 세계의 정보들을 컴퓨터에 표현하기 위해서 단순화, 추상화하여 체계적으로 표현한 개념적 모델로 데이터 모델링을 쉽게할 수 있도록 도와준다.

종류로는 개념적 데이터 모델, 논리적 데이터 모델, 물리적 데이터 모델이 있는데 개념적 데이터 모델에는 대표적으로 E-R모델이 있고 논리적 데이터 모델로는 관계 데이터 모델이 대표적이다.

▶데이터 모델 표시요소 : 구조(Structure), 연산(Operation), 제약조건(Constraint)

구조	논리적으로 표현된 개체 타입들 간의 관계로서 데이터 구조 및 정적 성질 표현
연산	데이터베이스에 저장된 실제 데이터를 처리하는 작업에 대한 명세로서 데이터베이스를 조작하는 기본 도구
제약조건	데이터베이스에 저장될 수 있는 실제 데이터의 논리적인 제약조건

▶데이터 모델 구성요소 : 개체, 속성, 관계

1) 개체(Entity) : 현실 세계에서 조직을 운영하는 데 꼭 필요한 사람이나 사물과 같이 구별되는 모든 것

▶개체 인스턴스(Entity instance) : 개체를 구성하고 있는 속성이 실제 값을 가짐으로써 실체화된 개체

▶개체 세트 : 개체 인스턴스의 집합

2) 속성(attribute) : 개체가 가지고 있는 고유의 특성, 데이터베이스를 구성하는 가장 작은 논리적 단위
아래 예시에서 이름, 연락처가 속성이고 홍길동, 010-1234-1234 이런 데이터가 개체 인스턴스,

이름	연락처
홍길동	010-1234-1234
종진상	010-1111-2222
박민복	010-3333-4444

3) 관계(Relationship) : 개체와 개체가 맺고 있는 의미 있는 연관성

▶관계의 유형

일 대 일(1:1)	말 그대로 1:1 대응 관계(ex) 결혼)
일 대 다(1:N)	1:다 대응(ex) 일부다처제)
다 대 다(N:M)	다:다 대응(ex) 아내는 남편 여러 명, 남편도 아내 여러 명)

▶ 관계의 종류

종속 관계 (Dependent Relationship)	두 개체 사이의 주·종 관계를 표현한 것
중복 관계 (Redundant Relationship)	두 개체 사이에 2번 이상의 종속 관계가 발생하는 관계
재귀 관계 (Recursive Relationship)	개체가 자기 자신과 관계를 갖는 것 순환 관계(Recursive Relationship)라고도 함
배타 관계 (Exclusive Relationship)	개체의 속성이나 구분자를 기준으로 개체의 특성을 분할하는 관계

▶ 관계의 종속성

개체 B가 독자적으로 존재할 수 없고 다른 개체 A의 존재 여부에 의존적이라면, 개체 B가 개체 A에 종속되어 있다고 한다. 이는 개체 A가 존재해야 B가 존재할 수 있고 A가 삭제되면 개체 B도 함께 삭제되어야 함을 의미한다. 이러한 종속을 특별히 존재 종속(Existence dependence)라 한다. 이때 다른 개체의 존재 여부에 의존적인 개체 B를 약한 개체(weak entity)라 하고 다른 개체의 존재 여부를 결정하는 개체 A를 오너 개체(owner entity)라 한다.

예를 들어 학생과 학부모를 예로 들 수 있는데 학생이 없다면 학교에 학부모 데이터가 필요 없다. 따라서 학생이 오너 개체, 학부모가 약한 개체가 되는데 가령 아무개 학생의 학부모 이름이 홍길동인데 홍길동을 찾는다고 생각해보자.

학번	이름	전공
15234	아무개	음악
16236	박소지	영어
17432	김지비	물리

학생 개체

이름	비상연락처
홍길동	010-1234-1234
김장두	010-1633-2163
홍길동	010-3412-3567

학부모 개체

학부모 개체에서 학부모의 이름 홍길동만으로는 다른 학부모와 동명이인인 경우도 있으므로 찾기가 어렵다. 그래서 오너 개체인 학생 개체에 학번이라는 키 속성이 존재한다면 학생 개체를 식별하고 식별된 학생의 학부모 개체를 이름 속성으로 구별하면 된다. 즉, 학부모 개체의 키를 (학번, 이름)으로 구성한다. 이때 이름과 같이 약한 개체를 구별하는 역할을 하는 속성을 구별자(delimiter) 또는 부분키(Partial key)라고 한다.


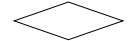



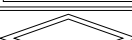
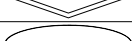

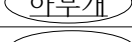
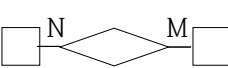
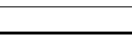
*키(Key) : 데이터베이스에서 조건에 만족하는 튜플을 찾거나 순서대로 정렬할 때 기준이 되는 속성

◆E-R 모델(개념적 데이터 모델)

피터 첸(Peter Chen)이 1976년에 제안한 것으로 현실 세계를 개체(entity)와 개체 간의 관계(relationship)를 이용해 개념적 구조로 표현하는 방법

현실 세계를 E-R모델을 이용해 개념적 모델링하여 그림으로 표현한 것을 E-R 다이어그램이라고 한다.

▶E-R 다이어그램

기호	기호 이름	의미
	사각형	개체
	마름모	관계
	타원	속성
	이중 타원	다중값 속성(복합 속성)
	이중 사각형	약한 개체
	이중 마름모	약한 개체가 오너 개체와 맺는 관계
	점선 타원	유도 속성
	밑줄 타원	기본키 속성(아무개는 그냥 예시)
	복수 타원	복합 속성
	관계	1:1, 1:N, N:M 등의 개체 간 관계 대응 수를 선 위에 기술
	선, 링크	개체 타입과 속성을 연결

*다중값 속성 : 속성이 값을 여러 개 가질 수 있는 경우

*복합 속성 : 의미를 분해할 수 있어 값이 여러 개의 의미를 포함하는 경우(ex) 주소를 xx시 xx동으로 분해)

*유도 속성 : 기존의 다른 속성의 값에서 유도되어 결정되는 속성(ex) 물건이 할인율에 의해 판매가격이 결정되므로 판매가격은 유도 속성)

*NULL 속성 : 아직 결정되지 않거나 모르는 값으로 공백이나 0과는 다르다.

*개체 A와 B 사이의 관계에서 개체 A의 모든 개체 인스턴스가 관계에 반드시 참여해야 된다면 개체 A가 관계에 '필수적 참여한다'라고 하며 필수적 참여 관계는 E-R 다이어그램에서 이중선으로 표현한다.

◆관계 데이터 모델(Relational Data Model)-논리적 데이터 모델

2차원적인 표를 이용해 데이터 상호 관계를 정의하는 DB구조

▶관계형 데이터베이스

2차원적인 표를 이용해서 데이터 상호 관계를 정의하는 데이터베이스

관계 데이터 모델에 따라 제작된 데이터베이스

1970년 IBM에 근무하던 코드(E.F.Codd)에 의해 처음 제안되었다.

▶관계형 데이터베이스 용어

열(속성, 애틀리뷰트)

고객아이디	고객이름	나이	등급	직업	적립금
CHAR(20)	CHAR(20)	INT	CHAR(10)	CHAR(10)	INT
apple	김현준	20	gold	학생	1000
banana	정소화	25	vip	간호사	2500
carrot	원유선	28	gold	교사	4500
orange	정지영	22	silver	학생	0

행(튜플)

도메인

출처 : [관계 데이터 모델의 기본 용어 \(naver.com\)](https://naver.com)

릴레이션 (Relation)	데이터들을 표 형태로 표현한 것
속성 (Attribute)	릴레이션의 열=파일에서 필드(field)에 대응하는 개념
차수(Degree)	속성의 수
튜플(Tuple)	릴레이션의 행=파일에서 레코드(record)에 대응하는 개념
카디널리티(Cardinality)	튜플의 수
도메인(Domain)	속성 하나가 가질 수 있는 모든 값의 집합(ex) 성별이면 남, 여)

▶릴레이션의 특성

튜플의 유일성	하나의 릴레이션에 동일한 튜플이 존재할 수 없다.
튜플의 무순서	하나의 릴레이션에 튜플 사이의 순서는 무의미하다
속성의 무순서	하나의 릴레이션에 속성 사이의 순서는 무의미하다.
속성의 원자성	속성 값으로 원자 값(분해 할 수 없는)만 사용할 수 있다.

◆키(Key)

데이터베이스에서 조건에 만족하는 튜플을 찾거나 순서대로 정렬할 때 기준이 되는 속성

▶키의 종류

후보키 (Candidate Key)	릴레이션을 구성하는 속성들 중에서 튜플을 유일하게 식별하기 위해 사용되는 속성들의 부분집합 기본키로 사용할 수 있는 속성으로 유일성과 최소성을 만족시켜야 한다.
기본키 (Primary Key)	후보키 중에서 특별히 선정된 주 키(Main Key), Null값을 가질 수 없다.
대체키 (Alternate Key)	후보키가 둘 이상일 때 기본키를 제외한 나머지 후보키, 보조키라고도 함
슈퍼키 (Super Key)	한 릴레이션 내에 있는 속성들의 집합으로 구성된 키 유일성은 만족하지만 최소성은 만족하지 못한다.
외래키 (Foreign Key)	다른 릴레이션의 기본키를 참조하는 속성 또는 속성들의 집합

아래 예제를 통해 키를 다시 한번 생각해보자.

학번	주민등록번호	성별	이름	입학년도
15234	030126-3234567	남	아무개	2015
16236	040516-2235627	여	박소지	2016
17432	050606-3304011	남	아무개	2017

1) 슈퍼키

슈퍼키는 각 행을 유일하게 식별할 수 있는 하나 또는 그 이상의 속성들의 집합이다.

학번과 주민등록번호 속성은 중복된 값이 존재하지 않기 때문에 슈퍼키가 가능하고

이름+입학년도를 생각해보자. 이름에 동명이인이 있어서 슈퍼키가 안되지만 이름과 입학년도를 함께 생각하면 중복된 값이 존재하지 않으므로 슈퍼키가 될 수 있다.

2) 후보키

후보키는 각 행을 유일하게 식별할 수 있는 최소한의 속성들의 집합이다.

학번과 주민등록번호가 후보키로 가능하다.

3) 기본키

후보키 중에서 선정된 키, 최소성과 유일성을 만족한다.

.학번과 주민등록번호 중 1개 속성이 기본키로 가능하다.

4) 대체키

후보키가 두 개 이상일 경우 그 중에서 어느 하나를 기본키로 지정하고 남은 후보키

만약 학번을 기본키로 설정했다면 대체키는 주민등록번호 속성이 된다.

ex) 다음 두 릴레이션(Relation)에서 외래키로 사용된 속성을 찾아 쓰시오.(단, 밑줄 친 속성은 기본키이다)

-필기 20년 6회 기출문제

과목(과목번호, 과목명)
수강(수강번호, 학번, 과목번호, 학기)

답 : 과목번호

◆관계 데이터 모델의 제약 조건

▶무결성(Integrity) : 데이터베이스에 저장된 데이터 값과 그것이 표현하는 현실 세계의 실제값이 일치하는 정확성

데이터베이스가 삽입·삭제·수정 연산으로 상태가 변하더라도 무결성 제약조건은 반드시 지켜져야 한다.

무결성은 권한이 있는 사용자의 잘못된 요구에 의해 데이터가 부정확해지지 않도록 보호하는 것이다.

▶무결성 제약조건

개체 무결성	기본키를 구성하는 모든 속성은 널 값을 가질 수 없다
참조 무결성	외래키는 참조할 수 없는 값을 가질 수 없다

◆관계 데이터 연산

연산은 원하는 데이터를 얻기 위해 릴레이션에 필요한 처리 요구를 수행하는 것으로 대표적으로 관계 대수와 관계 해석이 있다.

- 관계 대수 : 원하는 결과를 얻기 위해 데이터의 처리 과정을 순서대로 기술(절차 언어)
- 관계 해석 : 원하는 결과를 얻기 위해 처리를 원하는 데이터가 무엇인지만 기술(비절차 언어)

관계 대수와 관계 해석은 상용화된 관계 데이터베이스에서는 실제로 사용되지 않는 개념적 언어다. 하지만 새로운 데이터 언어가 제안되면 해당 데이터 언어의 유용성을 검증해야 하는데 검증의 기준 역할을 하는 것이 관계 대수와 관계 해석이다.

관계 대수에 속하는 대표적인 연산자 8개는 특성에 따라 일반 집합 연산자와 순수 관계 연산자로 분류할 수 있다.

▶순수 관계 연산자

SELECT	릴레이션에 존재하는 튜플 중에서 선택 조건을 만족하는 튜플의 부분집합을 구하여 새로운 릴레이션을 만드는 연산	σ
Project	주어진 릴레이션에서 속성 리스트에 제시된 속성값만 추출하여 새로운 릴레이션을 만드는 연산	π
Join	공통 속성을 중심으로 두 개의 릴레이션을 하나로 합쳐서 새로운 릴레이션을 만드는 연산	\bowtie
Division	릴레이션 A, B가 있을 때 릴레이션 B의 조건에 맞는 것들만 릴레이션 A에서 분리하여 프로젝션을 하는 연산	\div

▶일반 집합 연산자

합집합 (UNION)	합집합을 구하되, 중복되는 튜플은 제거되는 연산	\cup
교집합 (INTERSECTION)	교집합을 구하는 연산	\cap
차집합 (DIFFERENCE)	차집합을 구하는 연산	$-$
교차곱 (CARTESIAN PRODUCT)	튜플들의 순서쌍을 구하는 연산	\times

ex1) 아래 릴레이션에서 등급 속성을 π 한 결과 릴레이션을 나타내시오.

아이디	이름	나이	등급
a	김가	21	골드
b	김나	22	브론즈
c	김다	23	챌린저
d	김라	24	골드

답 :

등급
골드
브론즈
챌린저

ex2) 아래 릴레이션에서 등급 속성이 골드인 튜플을 σ 한 결과 릴레이션을 나타내시오.

아이디	이름	나이	등급
a	김가	21	골드
b	김나	22	브론즈
c	김다	23	챌린저
d	김라	24	골드

답 :

아이디	이름	나이	등급
a	김가	21	골드
d	김라	24	골드

ex3) 아래 2개의 릴레이션을 \bowtie 한 결과 릴레이션을 나타내시오.

아이디	이름	나이	등급
a	김가	21	골드
b	김나	22	브론즈
c	김다	23	챌린저
d	김라	24	골드

계정 생성일	아이디	현질금액
22.10.02	a	10만원
16.12.12	b	100만원
13.5.5	c	50만원

답 :

아이디	이름	나이	등급	계정 생성일	현질 금액
a	김가	21	골드	22.10.02	10만원
b	김나	22	브론즈	16.12.12	100만원
c	김다	23	챌린저	13.5.5	50만원

ex4) 아래 주문내역 릴레이션을 제품1, 제품2와 각각 ÷한 결과 릴레이션을 나타내시오.

주문고객	제품이름	제조업체
김가	과학책	케방
김나	수학책	케비
김라	만화책	케방
김가	만화책	케방
김나	만화책	케방

주문내역 릴레이션

제품이름
과학책
만화책

제품1 릴레이션

제품이름	제조업체
만화책	케방

제품2 릴레이션

㉔ 주문내역÷제품1한 결과 릴레이션

답 :

주문고객	제조업체
김가	케방

㉕ 주문내역÷제품2한 결과 릴레이션

답 :

주문고객
김라
김가
김나

ex5) 두 릴레이션을 각각 $R \cup S$, $R \cap S$, $R-S$ 했을 때 결과 릴레이션을 나타내시오

학번	이름	학번	이름
1234	김가	1234	김가
1235	김나	1237	김라
1236	김다	1238	김마

왼쪽 R 릴레이션 / 오른쪽 S 릴레이션

㉠ $R \cup S$ 의 경우

학번	이름
1234	김가
1235	김나
1236	김다
1237	김라
1238	김마

㉡ $R \cap S$ 의 경우

학번	이름
1234	김가

㉢ $R-S$ 의 경우

학번	이름
1235	김나
1236	김다

◆데이터베이스 설계

▶설계 순서

요구 조건 분석->개념적 설계->논리적 설계->물리적 설계-> 구현

①요구 조건 분석 : 구성원들이 데이터베이스를 사용하는 용도 파악, 다양한 요구사항 수집, 요구 조건 명세서 작성

②개념적 설계 : 요구사항을 개념적 데이터 모델을 이용해 표현(보통 E-R 모델 사용)

③논리적 설계 : 개념적 구조를 기반으로 논리적 구조 설계(일반적으로 관계 데이터 모델 사용)

④물리적 설계 : 논리적 구조를 기반으로 물리적 구조 설계(응답 시간을 최소화하고 저장공간 효율적 활용)

⑤구현 : DBMS에서 SQL로 작성한 명령문을 실행하여 데이터베이스를 실제로 생성

◆정규화(Normalization)

데이터베이스를 잘못 설계하면 불필요한 데이터 중복이 발생하여 데이터의 삽입·수정·삭제 연산을 수행할 때 부작용들이 발생할 수 있다. 이러한 부작용을 이상(anomaly) 현상이라 한다. 이상 현상을 제거하면서 데이터베이스를 올바르게 설계해 나가는 과정이 정규화다. 이상 현상의 종류부터 알아보도록 하자.

▶이상(Anomaly) 현상-삽입, 갱신, 삭제 이상

1) 삽입이상(Insertion Anomaly)

새 데이터를 삽입하기 위해 불필요한 데이터도 함께 삽입해야 하는 문제

고객아이디	이벤트번호	당첨여부	고객이름	등급
good12	1	Y	김씨	vip
bad55	2	N	이씨	gold
friend123	3	N	박씨	silver
seo41	4	N	나씨	gold

위와 같은 고객 아이디와 이벤트 번호가 기본키인 릴레이션에

고객 아이디가 melon254고 이름이 신씨고 등급이 silver인 사람을 삽입하려고 하는데 이 고객은 이벤트를 참여를 안해서 이벤트 번호 정보가 없다. 근데 기본키가 NULL값일 수는 없으므로 삽입이 불가능하게 된다.

2) 갱신 이상(Update Anomaly)

중복 튜플 중 일부만 변경하여 데이터가 불일치하게 되는 모순의 문제

고객아이디	이벤트번호	당첨여부	고객이름	등급
good12	1	Y	김씨	vip
good12	5	N	김씨	vip
good12	6	Y	김씨	vip
bad55	2	N	이씨	gold
friend123	3	N	박씨	silver
seo41	4	N	나씨	gold

위와 같이 고객 아이디, 이벤트 번호가 기본키인 릴레이션이 있는데

고객 아이디가 good12인 고객의 등급을 vip에서 vvip로 변경하고자 하는데

3개중에 실수로 일부만 vvip로 수정하게 되면 동일한 한 명의 고객의 등급 정보가 달라지게 되는 데이터 불일치가 발생하게 된다.

3) 삭제 이상(Deletion Anomaly)

튜플을 삭제하면 꼭 필요한 데이터까지 함께 삭제되는 데이터 손실의 문제

고객아이디	이벤트번호	당첨여부	고객이름	등급
good12	1	Y	김씨	vip
good12	5	N	김씨	vip
good12	6	Y	김씨	vip
bad55	2	N	이씨	gold
friend123	3	N	박씨	silver
seo41	4	N	나씨	gold

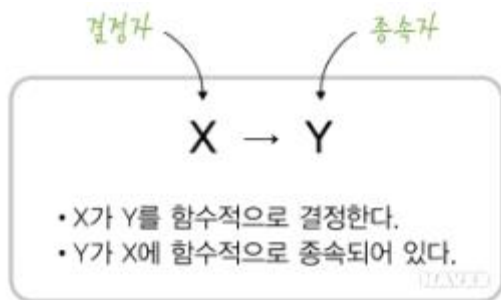
위와 같이 고객 아이디, 이벤트 번호가 기본키인 릴레이션이 있는데

아이디가 bad55인 고객이 이벤트 참여를 취소하여 이벤트 번호와 당첨 여부를 삭제하려고 할 때
당첨 여부와 관련 없음에도 불구하고 아이디가 bad55인 고객의 정보 전부 삭제된다.

▶ 정규화의 필요성

이상 현상을 발생하지 않도록 하기 위해서는 관련 있는 속성들만으로 릴레이션을 구성해야 하는데 그러므로 정규화가 필요하다. 정규화를 자세히 알아보기 전에 먼저 함수의 종속을 알아보자.

▶ 함수의 종속



[그림 9-4] 함수 종속성의 표현

출처 : [함수 종속 \(naver.com\)](https://naver.com)

하나의 릴레이션을 구성하는 속성들의 부분집합을 X와 Y라 할 때, 어느 시점에서든 릴레이션 내의 모든 튜플을 대상으로 한 X 값에 대한 Y 값이 항상 하나면 "X가 Y를 함수적으로 결정한다" 또는 "Y가 X에 함수적으로 종속되어 있다"고 한다. 함수 종속 관계는 $X \rightarrow Y$ 로 표현하고 X를 결정자, Y를 종속자라고 한다.

아래 릴레이션에서 고객 아이디가 고객 이름과 등급을 결정하므로

고객 아이디가 결정자, 고객 이름과 등급은 종속자임을 알 수 있다

고객아이디	고객이름	등급
good12	김씨	vip
bad55	이씨	gold
seo41	나씨	gold

아래 릴레이션에서

고객 아이디가 고객이름을 결정하고

고객 아이디+이벤트 번호가 당첨 여부를, 고객 이름을 결정한다.

고객 이름은 {고객 아이디, 이벤트 번호}라는 집합의 일부분인 {고객 아이디} 집합에서도 종속이므로

고객 이름은 {고객 아이디, 이벤트 번호}에 부분(Partial) 함수 종속되어 있다고 하고

당첨 여부는 {고객 아이디, 이벤트 번호}의 일부분이 아닌 속성 집합 전체에 종속돼있어 완전(full) 함수 종속이라고 한다.

고객아이디	이벤트번호	당첨여부	고객이름
good12	1	Y	김씨
good12	5	N	김씨
good12	6	Y	김씨
bad55	2	N	이씨
friend123	3	N	박씨
seo41	4	N	나씨

부분 함수 종속 관계가 성립하려면 결정자가 여러 개의 속성들로 구성되어야 한다.

▶ 정규화 순서

1) 제 1정규형 : 릴레이션에 속한 모든 속성의 도메인이 원자 값으로만 구성되어 있는 정규형

(아래 릴레이션은 원자값이 아니라 이벤트번호, 당첨여부에 값이 여러개 포함하므로 1정규형이 아니다)

고객아이디	이벤트번호	당첨여부
good12	1, 12, 13	Y, N, Y
good12	5	N
good12	6	Y

2) 제 2정규형 : 릴레이션이 제 1정규형에 속하고 기본키가 아닌 모든 속성이 기본키에 완전 함수 종속 아래와 같이 1정규형에 포함되는 릴레이션이 있는데

등급, 할인율은 {고객 아이디, 이벤트 번호}, {고객 아이디} 두 집합에 종속이라 부분 함수 종속이다

고객아이디	이벤트번호	당첨여부	등급	할인율
good12	1	Y	vip	20%
good12	5	N	vip	20%
good12	6	Y	vip	20%
bad55	2	N	gold	10%
friend123	3	N	silver	5%
seo41	4	N	gold	10%

제 2정규형으로 변환하면 아래와 같다.

고객아이디	이벤트번호	당첨여부
good12	1	Y
good12	5	N
good12	6	Y
bad55	2	N
friend123	3	N
seo41	4	N

고객아이디	등급	할인율
good12	vip	20%
bad55	gold	10%
friend123	silver	5%
seo41	gold	10%

3) 제 3정규형

릴레이션이 제2정규형에 속하고 기본키가 아닌 모든 속성이 기본키에 이행적 함수 종속이 되지 않아야 한다.

이행적(transitive) 함수 종속

세 개의 속성 집합 X, Y, Z에 대해 함수 종속 관계가

$X \rightarrow Y$ 와 $Y \rightarrow Z$ 가 존재하면 논리적으로 $X \rightarrow Z$ 가 성립하는데

이를 속성 집합 Z가 속성집합 X에 이행적으로 함수 종속되었다고 한다.

따라서 전 페이지에 제2정규형으로 변환했던 2개중에 아래의 릴레이션은 이행적 함수 종속이다

고객아이디->등급

등급->할인율

관계이므로 이행적 함수 종속(Transitive Functional Dependency)이다

고객아이디	등급	할인율
good12	vip	20%
bad55	gold	10%
friend123	silver	5%
seo41	gold	10%

제3정규형으로 변환하면 아래와 같다.

고객아이디	등급
good12	vip
bad55	gold
friend123	silver
seo41	gold

등급	할인율
vip	20%
gold	10%
silver	5%

4) 보이스/코드 정규형(BCNF)

릴레이션의 함수 종속 관계에서 모든 결정자가 후보키이면 보이스/코드 정규형에 속한다.

지금까지 본 예시들은 하나의 기본키, 하나의 후보키였는데 여러개의 후보키가 존재할 수도 있다.

후보키를 여러 개 가지고 있는 릴레이션에 발생할 수 있는 이상 현상을 해결하기 위해 제 3정규형보다 좀 더 엄격한 제약조건을 제시한 것이 보이스/코드 정규형으로 strong 3NF라고도 불린다.

*후보키 : 기본키로 사용할 수 있는 키(유일성과 최소성이 만족해야함)

*슈퍼키 : 하나 이상의 속성들의 집합으로 이루어진 키

아래는 강좌신청 릴레이션이다. 모든 속성이 원자 값으로만 구성되어 있어 제1정규형에 속하고 기본키가 아닌 속성인 담당강사 이름이 {고객 아이디, 인터넷 강좌} 전체에 종속이라 제2정규형에도 속하고 이행적 함수 종속도 아니라서 제 3규형에도 속하지만 담당 강사 이름 속성이 후보키가 아님에도 인터넷 강좌 속성을 결정하므로 보이스/코드 정규형에는 속하지 않는다.

고객아이디	인터넷강좌	담당강사이름
good12	영어	김씨
bad55	수학	박씨
friend123	영어	김씨
friend123	수학	곽씨
seo41	영어	오씨
seo41	수학	곽씨

강좌신청 릴레이션(단, 담당강사이름 중 동명이인이 없다는 전제)

변환하면 아래와 같다.

고객아이디	담당강사이름
good12	김씨
bad55	박씨
friend123	김씨
friend123	곽씨
seo41	오씨
seo41	곽씨

담당강사이름	인터넷강좌
김씨	영어
박씨	수학
곽씨	수학
오씨	영어

5) 제 4, 5정규형

제 4정규형은 릴레이션이 보이스/코드 정규형을 만족하면서 함수 종속이 아닌 다치 종속을 제거해야 만족할 수 있고 제 5정규형은 릴레이션이 제4정규형을 만족하면서 후보키를 통하지 않는 조인 종속을 제거해야 만족할 수 있다

실제로 데이터베이스를 설계할 때 모든 릴레이션이 무조건 제5정규형에 속하도록 분해해야 하는 것은 아니다. 오히려 제5정규형을 만족할 때까지 분해하는 것이 비효율적이고 바람직하지 않은 경우가 많다. 일반적으로는 제3정규형이나 보이스/코드 정규형에 속하도록 릴레이션을 분해하여 데이터 중복을 줄이고 이상 현상이 발생하는 문제를 해결한다.

▶반정규화(Denormalization) : 시스템의 성능을 향상하고 개발 및 운영의 편의성 등을 높이기 위해 정규화된 데이터 모델을 의도적으로 통합, 중복, 분리하여 정규화 원칙을 위배하는 행위, 비정규화라고도 한다.

ex) 테이블 통합, 테이블 분할, 중복 테이블 추가, 중복 속성 추가

●테이블 분할

수평 분할	레코드(Record)를 기준으로 테이블 분할
수직 분할	속성(Attribute)을 기준으로 테이블을 분할

◆트랜잭션(Transaction)

데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위 또는 한꺼번에 모두 수행되어야 할 일련의 연산들

트랜잭션의 모든 명령문이 완벽하게 처리되거나 하나도 처리되지 않아야 데이터베이스가 모순이 없는 일관된 상태를 유지할 수 있다. 트랜잭션은 데이터베이스에 장애가 발생했을 때 복구 작업을 수행하거나, 다수의 사용자가 동시에 사용할 수 있도록 제어 작업을 하는데 중요한 단위로 사용된다. 그러므로 데이터베이스의 무결성과 일관성을 보장하려면 작업을 수행하는 데 필요한 연산들을 하나의 트랜잭션으로 제대로 정의하고 관리해야 한다.

▶트랜잭션의 특성(ACID)

원자성 (Atomicity)	트랜잭션의 연산은 데이터베이스에 모두 반영되도록 완료(Commit)되든지 아니면 전혀 반영되지 않도록 복구(Rollback)되어야 함
일관성 (Consistency)	트랜잭션이 그 실행을 성공적으로 완료하면 언제나 일관성 있는 데이터베이스 상태로 변환함
독립성(격리성) (Isolation)	둘 이상의 트랜잭션이 동시에 병행 실행되는 경우 어느 하나의 트랜잭션 실행 중에 다른 트랜잭션의 연산이 끼어들 수 없음
영속성, 지속성 (Durability)	성공적으로 완료된 트랜잭션의 결과는 시스템이 고장나더라도 영구적으로 반영되어야 함

▶트랜잭션의 연산

Commit 연산	트랜잭션이 성공적으로 수행되었음을 선언(작업 완료)
Rollback 연산	트랜잭션이 수행을 실패했음을 선언(작업 취소)

*CRUD 분석(Create, Read, Update, Delete) : 프로세스와 테이블 간 CRUD 매트릭스를 만들어서 트랜잭션을 분석하는 것

◆회복

데이터베이스를 모순이 없는 일관된 상태로 유지하기 위해 데이터베이스 관리 시스템은 회복 기능을 제공한다. 데이터베이스가 조직의 중요한 데이터를 저장하고 있는 만큼 데이터베이스 관리 시스템의 회복 기능은 매우 중요한 기능이다. 데이터베이스 회복의 핵심 원리는 데이터 중복이다. 데이터를 별도의 장소에 미리 복사해두고, 장애로 문제가 발생했을 때 복사본을 이용해 원래의 상태로 복원하는 것이다.

▶회복을 위해 복사본을 만드는 방법

덤프(dump)	데이터베이스 전체를 다른 저장 장치에 주기적으로 복사하는 방법
로그(log)	데이터베이스에서 변경 연산인 실행될 때마다 데이터를 변경하기 이전 값과 변경한 이후의 값을 별도의 파일에 기록하는 방법

장애가 발생했을 때 덤프나 로그 방법으로 중복 저장한 데이터를 이용해 취할 수 있는 가장 기본적인 회복 방법은 redo나 undo 연산을 실행하는 것이다.

▶회복 연산

redo(재실행)	가장 최근에 저장한 DB복사본을 가져온 후 로그를 이용해 복사본이 만들어진 이후에 실행된 모든 변경 연산을 재실행하여 장애가 발생하기 직전의 DB 상태로 복구
undo(취소)	로그를 이용해 지금까지 실행된 모든 변경 연산을 취소하여 DB를 원래의 상태로 복구

▶로그 회복 기법

데이터를 변경한 연산 결과를 데이터베이스에 반영하는 시점에 따라 2가지로 구분된다.

즉시 갱신 기법 (Immediate Update)	트랜잭션이 데이터를 업데이트하면 트랜잭션이 부분 완료되기 전이라도 즉시 실제 데이터베이스에 반영하는 방법
연기 갱신 기법 (Deferred Update)	트랜잭션이 성공적으로 완료될 때까지 데이터베이스에 대한 실질적인 업데이트를 연기하는 방법

▶기타 회복 기법

그림자 페이지 대체 기법 (Shadow Paging)	업데이트 이전의 데이터베이스를 일정 크기의 페이지 단위로 구성하여 각 페이지마다 복사본인 그림자 페이지를 별도 보관해 놓고, 실제 페이지를 대상으로 업데이트 작업을 수행하다가 장애가 발생하여 트랜잭션 작업을 Rollback 시킬 때는 갱신 이후의 실제 페이지 부분을 그림자 페이지로 대체하여 회복시키는 기법
검사점 기법 (Check Point)	트랜잭션 실행 중 특정 단계에서 재실행할 수 있도록 업데이트 내용이나 시스템에 대한 상황 등에 관한 정보와 함께 검사점을 로그에 보관해 두고, 장애 발생시 트랜잭션 전체를 철회하지 않고 검사점부터 회복 작업을 수행하여 회복 시간을 절약하도록 하는 기법 (그냥 게임할 때 체크포인트 생각하면 된다)

◆병행 수행과 병행 제어

DBMS(데이터베이스 관리 시스템)는 여러 사용자가 DB를 동시에 공유할 수 있도록 여러 개의 트랜잭션이 동시에 수행되는 병행 수행(Concurrency)을 지원한다. 그런데 병행 수행되는 트랜잭션들이 서로 다른 데이터를 사용하여 연산을 실행하는 경우에는 괜찮지만 동시에 같은 데이터에 접근하여 변경 연산을 실행하려고 하면 문제가 발생할 수 있다. 그러므로 병행 수행을 하더라도 각 트랜잭션이 다른 트랜잭션의 방해받지 않고 정확한 수행 결과를 얻을 수 있도록 제어해야 하는데 이를 병행 제어(Concurrency Control) 또는 동시성 제어라고 한다.

▶병행 수행의 문제

갱신 분실(lost update)	하나의 트랜잭션이 수행한 데이터 변경 연산의 결과를 다른 트랜잭션이 덮어써 변경 연산이 무효화되는 것
모순성(Inconsistency)	하나의 트랜잭션이 여러 개의 데이터 변경 연산을 실행할 때 일관성 없는 상태의 데이터베이스에서 데이터를 가져와 연산을 실행함으로써 모순된 결과가 발생하는 것
연쇄 복귀 (Cascading rollback)	트랜잭션이 완료되기 전에 장애가 발생하여 rollback 연산을 수행하면 이 트랜잭션이 장애 발생 전에 변경한 데이터를 가져가 변경 연산을 실행한 또 다른 트랜잭션에도 rollback 연산을 연쇄적으로 실행해야 한다는 것이다.

▶병행 제어(Concurrency Control) 기법의 종류

로킹(Locking)	병행 수행되는 트랜잭션들이 동일한 데이터에 동시에 접근하지 못하도록 lock과 unlock이라는 두 개의 연산을 이용해 제어한다.
타임 스탬프 순서 (Time Stamp Ordering)	트랜잭션과 트랜잭션이 읽거나 갱신한 데이터에 대해 트랜잭션이 실행을 시작하기 전에 시간표를 부여하여 부여된 시간에 따라 트랜잭션 작업을 수행하는 기법

*로킹 단위(Locking granularity) : 병행제어에서 한꺼번에 로킹할 수 있는 객체의 크기

◆분산 데이터베이스 시스템(Distributed Database system)

데이터베이스 시스템을 한 곳에 설치하지 못하고 여러 곳에 분산 설치하여 운영하는 경우가 종종 있다. 물리적으로 분산된 데이터베이스 시스템을 네트워크로 연결해 사용자가 논리적으로는 하나의 중앙 집중식 데이터베이스 시스템처럼 사용할 수 있도록 한 것을 분산 데이터베이스 시스템이라고 한다.

▶분산 데이터베이스의 목표

분산 데이터베이스 시스템에서는 데이터베이스가 분산되어 있음을 사용자가 인식하지 못하는 것이 중요한데 이를 데이터 독립성이라고 하며 분산 데이터 독립성을 지원하기 위해서는 분산 투명성을 보장해야 한다.

▶분산 투명성(Distribution transparency)의 종류

위치 투명성 (Location Transparency)	데이터베이스가 지역적으로 분산되어 있지만, 사용자가 접근하려는 데이터의 실제 저장 위치를 알 필요 없이 데이터베이스의 논리적인 이름만으로 데이터에 접근할 수 있다.
중복 투명성 (Replication Transparency)	동일 데이터가 여러 곳에 중복되어 있더라도 사용자는 마치 하나의 데이터만 존재하는 것처럼 사용하고 시스템은 자동으로 여러 자료에 대한 작업을 수행한다.
병행 투명성 (Concurrency Transparency)	분산 데이터베이스와 관련된 다수의 트랜잭션들이 동시에 실행되더라도 그 트랜잭션의 결과는 영향을 받지 않는다.
장애 투명성 (Failure Transparency)	특정 지역에서 문제가 발생하더라도 전체 시스템이 작업을 계속 수행할 수 있다.
단편화 투명성 (Fragmentation Transparency)	단편화는 하나의 릴레이션을 더 작은 조각으로 나누고 각 조각을 별개의 릴레이션으로 처리하는 것인데 이처럼 단편화된 데이터를 여러 지역에 나누어 저장하지만 사용자는 데이터가 단편화되지 않은 것처럼 사용할 수 있다.

◆DB 기타 용어

인덱스(Index)	데이터 레코드를 빠르게 접근하기 위해 <키 값, 포인터> 쌍으로 구성되는 데이터 구조
뷰(View)	사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위해 하나 이상의 기본 테이블로부터 유도된 이름을 가지는 가상 테이블
클러스터 (Cluster)	동일한 성격의 데이터를 동일한 데이터 블록에 저장하는 물리적 저장 방법
클러스터링 (Clustering)	두 대 이상의 서버를 하나의 서버처럼 운영하는 기술
파티션 (Partition)	대용량 테이블이나 인덱스를 작은 논리적 단위인 파티션으로 나누는 것
RTO (Recovery Time Objective) 목표 복구 시간	비상사태 또는 업무 중단 시점으로부터 복구되어 가동될 때까지의 소요 시간
RPO (Recovery Point Objective) 목표 복구 시점	비상사태 또는 업무 중단 시점으로부터 데이터를 복구할 수 있는 기준점

▶ 파티션(Partition)

대용량 테이블이나 인덱스를 작은 논리적 단위인 파티션으로 나누는 것

범위 분할 (Range Partitioning)	지정한 열의 값을 기준으로 분할 ex) 일별, 월별 등
해시 분할 (Hash Partitioning)	해시 함수를 적용한 결과 값에 따라 데이터 분할
조합 분할 (Composite Partitioning)	범위 분할로 분할한 다음 해시 함수를 적용하여 다시 분할하는 방식

▶ 데이터베이스 이중화(Database Replication)

시스템 오류로 인한 데이터베이스 서비스 중단이나 물리적 손상 발생 시 이를 복구하기 위해 동일한 데이터베이스를 복제하여 관리하는 것

Eager 기법	트랜잭션 수행 중 데이터 변경이 발생하면 이중화된 모든 데이터베이스에 즉시 전달하여 변경 내용이 즉시 적용되도록 하는 기법
Lazy 기법	트랜잭션의 수행이 종료되면 변경 사실을 새로운 트랜잭션에 작성하여 각 데이터베이스에 전달되는 기법

▶ 접근 통제

데이터가 저장된 객체와 이를 사용하는 주체 사이의 정보 흐름을 제한하는 것이다.

DAC (Discretionary Access Control) 임의 접근 통제	데이터에 접근하는 사용자의 신원에 따라 접근 권한을 부여하는 방식
MAC (Mandatory Access Control) 강제 접근 통제	주체와 객체의 등급을 비교하여 접근 권한을 부여하는 방식
RBAC (Role based Access Control) 역할기반 접근 통제	사용자의 역할에 따라 접근 권한을 부여하는 방식

▶ 스토리지(Storage)

단일 디스크로 처리할 수 없는 대용량의 데이터를 저장하기 위해 서버와 저장장치를 연결하는 기술

DAS (Direct Attached Storage)	서버와 저장장치를 전용 케이블로 직접 연결하는 방식
NAS (Network Attached Storage)	서버와 저장장치를 네트워크를 통해 연결하는 방식
SAN (Storage Area Network)	DAS와 NAS를 혼합한 방식으로, 서버와 저장장치를 연결하는 전용 네트워크를 별도로 구성하는 방식

Ⅲ. 네트워크 파트

1. 네트워크(Network)

넓은 의미에서는 전화기, 팩스, 컴퓨터 등 지리적으로 떨어져 있는 장치들 간에 정보를 교환할 수 있도록 연결되어 있는 것이고

좁은 의미에서는 여러 컴퓨터들 사이에 정보를 주고받기 위해서 컴퓨터의 회선들이 망 형태로 연결되어 있는 것이다.

*인터넷(Internet) : 전 세계의 네트워크가 유기적으로 연결되어 동작하는 통합 네트워크

*호스트(Host) : 인터넷 네트워크에 접속되어 다른 인터넷 호스트와 통신 할 수 있는 컴퓨터

*표준화(Standardization) : 서로 다른 시스템이 상호 연동해 동작하려면 표준화라는 통일이 필요하다. 예를 들어 A4를 생각해 보면 A4가 표준화가 안돼 있다면 항상 특정 회사의 A4만 구매해야 하는 불편함이 생긴다.

2. 정보 전송

전화기, 컴퓨터, 팩스, 비디오 카메라, 스캐너 등으로 생성되는 다양한 형태의 정보(음성, 데이터, 화상 비디오 등)는 전기신호로 변환되어 송신된다.

예를 들면, 컴퓨터로 출력한 디지털 데이터를 전송하려면 아날로그 형태의 전기 신호로 바꾸어야 하고, 이 전기 신호는 전송선로의 영향을 덜 받도록 고주파수로 변환하는 변조 과정을 거친 후 송신되며 마지막으로 복조 과정을 거쳐 수신부에 디지털 신호로 입력된다.

*변조(Modulation) : 신호 정보를 전송매체의 채널 특성에 맞게끔 신호(정보)의 세기나 변위, 주파수, 위상 등을 적절한 파형 형태로 변환하는 것

*복조(Demodulation) : 변조되어 전송된 중에 손상된 파형을 원래의 정보신호 파형으로 복원하는 것

2-1 통신회선의 접속 방식

정보를 전송할 때 컴퓨터와 단말기를 연결하는데 연결하는 방식에는 4가지가 있다.


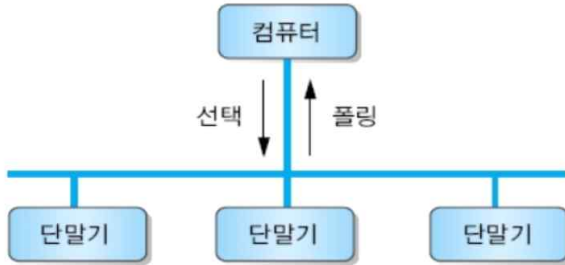
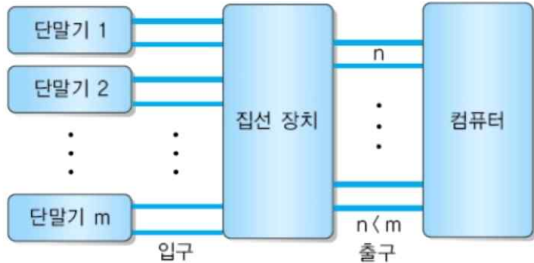
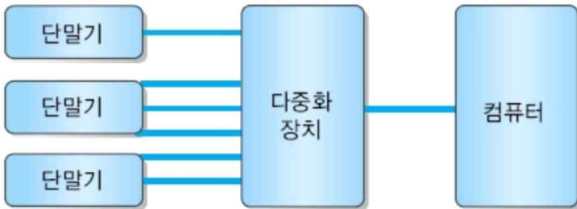
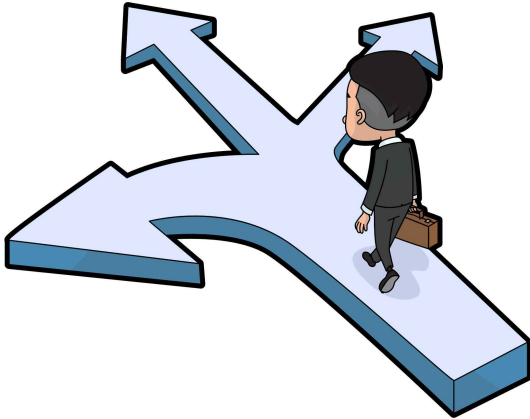
<p>점-대-점 회선 방식 (Point to Point)</p>	<p>-컴퓨터 시스템과 단말기를 전용회선으로 직접 연결하고 단말기를 여러 대 연결할 때도 1:1 연결함</p> <p>-응답 속도가 빨라 주로 고속 처리에 이용</p> <p>ex) 컴퓨터와 주변기기의 관계</p> 
<p>다지점 회선 방식 (Multipoint Line)</p>	<p>-컴퓨터 시스템에 연결된 전송회선 1개에 단말기를 여러대 연결</p> <p>ex) USB 허브를 통한 주변 장치 연결</p> 
<p>집선 회선 방식 (Line Concentration)</p>	<p>-일정한 지역 내에 있는 중심 부분의 집선 장치를 설치한 후 여기에 단말기를 여러 대 연결하는 방식</p> <p>-단말기에서 저속으로 전송되는 데이터를 컴퓨터에 고속으로 전송하는 역할</p> 
<p>회선 다중 방식 (Line Multiplexing)</p>	<p>일정한 지역에 있는 단말기 여러 대를 그 지역의 중심 부분에 설치된 다중화 장치에 연결하고 다중화 장치와 컴퓨터 사이는 대용량 회선으로 연결</p> 

사진 출처 : 제08절 정보 전송 방식 : 네이버 블로그 ([naver.com](https://blog.naver.com/dlqnf33))

2-2 통신회선의 교환 방식

컴퓨터 네트워크에는 전송 매체로 연결된 호스트들이 존재한다. 송신 호스트가 수신 호스트에 데이터를 전달(Transfer)하려면 전송과 교환 과정을 거쳐야 한다. 교환(Switching)은 전달 경로가 둘 이상일 때 라우터에서 데이터를 어느 방향으로 전달할지를 선택하는 기능으로, 다양한 기준에 따라 데이터를 올바른 경로로 전달할 수 있도록 해준다.



통신회선은 데이터를 교환하는 형태에 따라 비교환회선과 교환회선 방식으로 나뉜다.

비교환 회선 방식은 단말 장치끼리 직통회선을 연결하여 사용하고

교환 회선 방식은 교환기등을 이용하여 접속하며

교환회선 방식에는 회선 교환(Circuit Switching)방식과 축적 교환(Store and Forward Switching)이 있다.

회선 교환 (Circuit Switching)	-사용자가 직접 전화기의 번호판을 눌러 전화망을 이용해 상대방을 호출하고 연결 ex) 전화 교환망	
축적 교환 (Store and Forward Switching)	-교환기를 이용해서 정보를 메시지나 패킷 단위로 저장하고 전송 ex) 메시지 교환 방식, 패킷 교환 방식	
	메시지 교환 방식	정보를 전송하는 단위가 메시지이며, 길이가 매번 변한다.
	패킷 교환 방식	정보를 전송하는 단위가 패킷이며, 길이가 고정되고 규격화되어 있다.



출처 : [축적 교환 방식 네이버블로그](#)

교환 방식 중 패킷 교환 방식에 대해 조금 더 알아보자.

2-2-1 패킷 교환(Packet switching)

컴퓨터 네트워크와 통신의 방식 중 하나로 현재 가장 많은 사람들이 사용하는 통신 방식인데

데이터를 작은 조각인 패킷(packet)로 분할하여 전송하고, 수신측에서는 이러한 패킷들을 재조립하여 전체 데이터를 복원하는 방식

쉽게 생각해서 개미라는 데이터를 머리, 가슴, 배라는 패킷으로 분할한 다음 수신측에서 머리, 가슴, 배를 재조립해서 다시 개미를 만드는 것이라고 생각하면 된다.

가상회선 방식 (Virtual Circuit, VC)	가상회선을 만들어 송신 측과 수신 측간에 데이터를 전달하는 방식으로 연결 지향이며 전송이 끝난 후 가상회선은 종료된다.
데이터그램 방식 (Datagram)	정보를 전송하는 단위가 패킷이며, 길이가 고정되고 규격화되어 있다.

2-3 통신회선의 이용방식

단일 방식 (Simplex)	데이터를 한쪽 방향으로만 전송 가능 주로 단말기에서 컴퓨터 방향으로만 데이터 전송 ex) 라디오, TV 등
반이중 방식 (Half Duplex)	데이터를 양방향으로 모두 전송할 수 있으나 동시에 양방향으로는 전송 불가 ex) 팩스, 무전기 등
전이중 방식 (Full Duplex)	데이터를 동시에 양방향으로 전송할 수 있어 고속으로 처리 가능 ex) 전화 등

2-4 통신회선망의 구성 방식(구성 형태에 따른 분류)

여러 단말 장치가 유기적으로 결합된 통신회선망의 형태에 따라 트리형, 버스형, 성형, 링형, 망형 등이 있다.

2-5 데이터 전송 방식

데이터 전송하는 방식은 데이터를 보내는 방법에 따라 직렬 전송과 병렬 전송, 송수신 측 서로 간의 시간적 위치에 따라 동기식과 비동기식으로 나뉜다.

2-5-1 직렬전송과 병렬 전송

직렬 전송 (Serial Transmission)	-통신회선을 통하여 한 번에 한 비트씩 순서대로 전송하는 방식 -한 비트씩 전송하기 때문에 전송속도는 느리지만 통신회선 비용 저렴 -주로 원거리 통신에 사용 ex) 대부분의 데이터 통신(네트워크)
병렬 전송 (Parallel Transmission)	-여러 개의 전송로를 통하여 동시에 여러 비트를 전송하는 방식 -직렬 전송에 비해 전송 속도는 빠르나 통신회선을 구축하는데 많은 비용 필요 -근거리 통신에 사용 ex) 컴퓨터와 하드디스크 연결

*비트(bit) : 이진수를 뜻하는 'Binary Digit'의 약자로, 컴퓨터에서 CPU가 처리하는 데이터의 최소 단위 크기를 의미로, 컴퓨터는 2진수, 즉 0과 1로 모든 데이터를 처리하는데, 1비트는 0과 1을, 2비트는 00, 01, 10, 11 등 4개의 데이터를 처리할 수 있다.

흔히 컴퓨터를 할 때 자주 보는 32비트와 64비트가 있다. 64비트의 컴퓨터가 32비트의 컴퓨터보다 처리할 수 있는 데이터가 많고 처리 속도도 높다는 의미이다.

비동기식과 동기식을 알아보기 전에 컴퓨터에서 문자를 어떻게 처리할 수 있는지 먼저 알아보자.

☞ 어떻게 컴퓨터에서 숫자나 알파벳, 한글 같은 문자들을 볼 수 있을까?

컴퓨터는 0과 1의 이진수를 가지고 데이터를 처리하는데 65같은 10진수를 2진수로 변환을 하면 1000001이 된다. 0과 1밖에 모르는 컴퓨터를 위해 우리가 쓰는 문자를 A=65, B=66 이런식으로 숫자에 대응을 시키기 시작했고 문자를 숫자로 1대1 대응시키면 그 숫자를 2진수로 변환하면 되기 때문에 컴퓨터가 처리할 수 있었고 이렇게 만들어지는 코드를 아스키코드라고 부른다.

10진수 -> 2진수 변환

65

1000001

이렇게 사람이 쓰는 문자를 컴퓨터가 처리할 수 있도록 바이너리 신호로 바꿔주는 걸 문자 인코딩(Character Encoding)라고 한다.

아스키코드에는 알파벳과 숫자 등만 들어있었기 때문에 한글이나 다른 나라의 문자 체계를 표현할 수 없는 문제점이 있었고 그래서 각 나라마다 다른 문자셋이 있었으며 각 나라마다 다르니 불편해서 나오게 된 것이 바로 유니코드이다.

유니코드는 국제표준화기구(ISO)가 제정한 2바이트 국제 표준 문자부호 체계이며 세계 각국의 문자를 단일한 문자코드 체계로 통합하여, 변환 없이 데이터를 상호 교환 및 처리할 수 있도록 하기 위해 고안되었다.

*문자셋 : 1, 2, 3, 4, A, B, C, D처럼 문자들을 모아놓은 집합

2-5-2 비동기식 전송과 동기식 전송

데이터를 정확하게 송수신하려면 서로 간에 동기가 잘 맞아야 하는데 이때 동기화란 일정한 간격을 두고 일어나도록 시간의 간격을 조절하는 것을 말한다.

	비동기식 전송 (Asynchronous Transmission)	동기식 전송 (Synchronous Transmission)
정보 전송 형태	문자 단위	블록(프레임) 단위
정보 전송량	한번에 문자 한 개씩 송수신	송신 측과 수신 측 사이에 미리 정해진 숫자만큼 문자열을 한 묶음으로 만들어 한꺼번에 전송
클럭(Clock)	송수신 장치의 클럭 불일치 (타이밍이 달라도 상관없음) (데이터 전송 언제 시작할지 약속 불필요)	송수신 장치의 클럭 일치 (타이밍이 같아야함) (데이터 전송 언제 시작할지 미리 약속)
동기화 전송 구분	문자 단위로 동기화 시작비트와 정지비트가 존재하며 시작-정지 전송(Start-Stop)이라고도 함	데이터 묶음의 앞쪽에는 반드시 동기 문자가 와야하며 동기 문자는 송신 측과 수신 측이 서로 동기하는데 사용
중점적인 내용	데이터 중심 (택시라고 생각하면 됨) (택시 : 사람이 있을 때만 불규칙적으로)	회선 중심 (기차나 지하철 생각하면 됨) (기차 : 사람이 있든 없든 규칙적으로)
적합한 전송 상황	저속 전송에 적합	데이터가 많거나 고속 처리가 필요할 때 적합
휴지기간	문자를 연속해서 보낼 때 각 문자 사이에 일정하지 않은 휴지기간이 있을 수 있음	블록과 블록 사이(한 묶음으로 구성된 문자 사이)에 휴지기간이 없음
비용/회로 복잡도	저렴/회로 단순	비쌈/ 회로 복잡
전송 효율	문자마다 시작과 정지를 알리는 비트가 추가돼 전송 효율 떨어짐	휴지기간이 없어서 전송효율 높음
전송 방법	병렬형태로 작업 처리 작업이 종료되지 않은 상태이더라도 대기하지 않고 다음 작업 실행	직렬형태로 작업 처리 작업이 수행 중이면 다른 작업은 대기
예	편지, 이메일 등	채팅방, 화상 회의, 전화 등
종류		문자 동기 방식(ex) BSC 비트 동기 방식(ex) HDLC

*클럭 : 동작하는 순간을 제어하기 위한 시간 정보를 뜻하며 클럭이 필요한 이유는 오케스트라의 지휘자를 생각하면 되는데 오케스트라에 지휘자가 없다면 각각의 연주자들의 템포가 달라져 공연이 엉망이 되듯이 회로에서도 비슷한 현상이 일어난다.

*휴지기간 : 시스템이 정상적으로 작동하고 있지만 생산적으로 사용되지 않는 시간(데이터와 데이터 사이의 간격)

*프레임 : OSI계층에서 데이터링크 계층의 전송데이터를 프레임이라고 부르는데 일단 지금은 그냥 전송데이터라고 생각해도 된다.

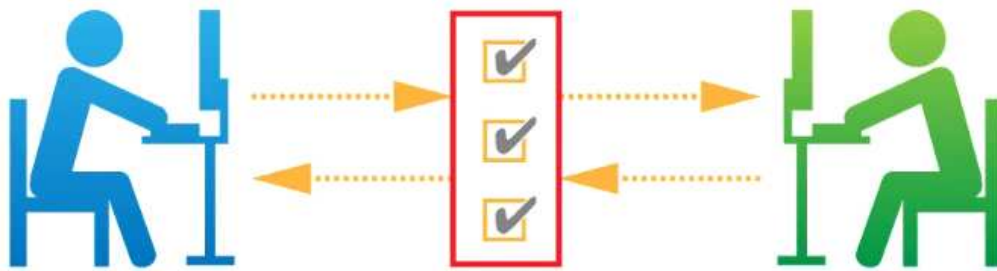
2-6 캐스팅 모드의 전송 방식

캐스팅 모드(Casting Mode)는 통신에 참여하는 송신자와 수신자의 수를 말한다.

유니캐스트 (Unicast)	1:1 데이터 통신
브로드캐스트 (Broadcast)	1:모두 방식 (1:모든 수신자) ex) 라디오, TV
멀티캐스트 (Multicast)	1: 다 방식 (1: 특정 수신자들) ex) 전자 우편
애니캐스트 (Anycast)	1: 1 방식 (1: 특정 수신자들 중 특정 수신자 1명) ex) 연결된 여러 프린터중 특정 1개 프린터

정보 전송은 이쯤에서 마무리하고 프로토콜에 대해 알아본다.

3. 프로토콜(Protocol) : 시스템이 데이터를 교환하기 위해 사용하는 통신 규칙



통신 프로토콜은 통신을 원하는 두 객체 간에 무엇을, 어떻게, 언제 통신할 것인지 서로 약속해 놓은 규정이다. 즉, 컴퓨터와 단말기, 컴퓨터 간에 서로 정보를 교환하기 위해 사전에 약속한 통신 규약이다.

통신 프로토콜은 여러 계층으로 나뉜 네트워크에서 똑같은 계층끼리 사용하는 표준화된 규약으로, 네트워크 기능을 효율적으로 발휘할 수 있도록 한다.

컴퓨터와 통신 관련 기술을 제조하는 회사마다 다른 프로토콜을 개발해 사용하다 보니 많은 혼란과 문제가 발생해 각기 다른 프로토콜을 하나로 통합하여 OSI 참조 모델이 등장하게 되었다.

▶프로토콜의 기본 요소

구문(Syntax)	전송하고자 하는 데이터의 형식, 부호화, 신호 레벨 등을 규정
의미(Semantics)	두 기기 간의 효율적이고 정확한 정보 전송을 위한 협조 사항과 오류 관리를 위한 제어 정보를 규정
시간(Timing)	두 기기 간의 통신 속도, 메시지의 순서 제어 등을 규정

▶프로토콜의 기능

단편화와 재결합, 캡슐화, 흐름제어, 오류제어, 동기화, 순서 제어, 주소 지정, 다중화, 경로 제어, 전송 서비스 등

프로토콜은 전화를 예로 들면 이해하기 쉽다. 전화는 송신자와 수신자의 합의를 통해 연결을 설정하고 양쪽이 동시에 말할 수 있는 양방향 통신 기능을 지원하지만 실제로 두 사람이 동시에 말을 하는 경우는 거의 없고 번갈아가며 대화를 하며 상대방이 한 말을 이해하지 못하면 다시 묻거나 해서 오류를 바로 잡기도 한다.

예를 들어 철수와 카톡을 하는데 필요한 약속을 철수카톡 프로토콜이라고 한다고 치자.

아래와 같이 철수와 카톡을 할 때는 욕을 하지 않고 영어를 쓰지 않아야 한다.

철수 카톡 프로토콜(약속)

1. 욕을 하지 않는다
2. 영어를 쓰지 않는다

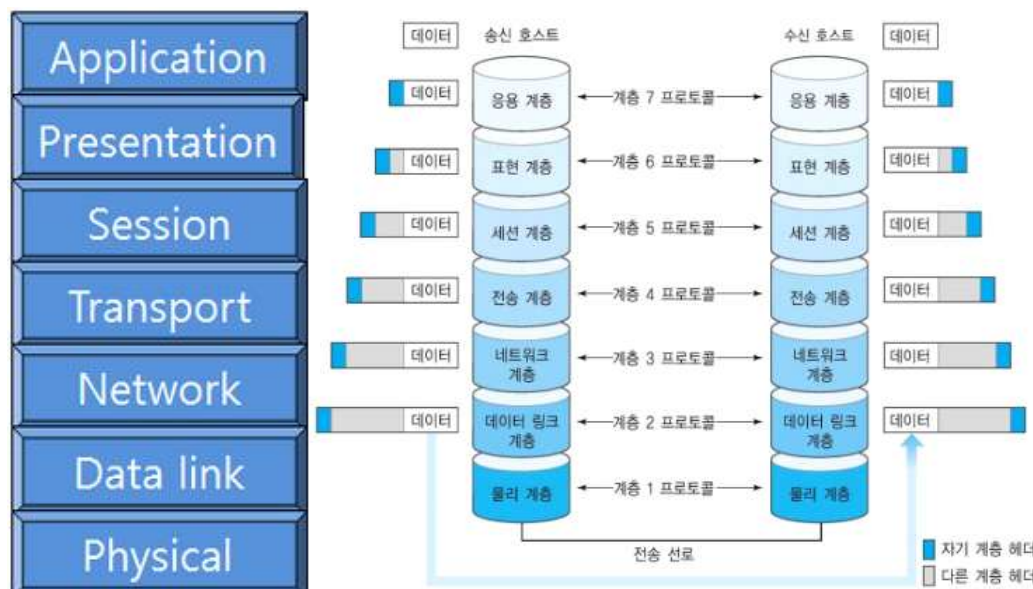
4. OSI 참조 모델(OSI 7계층)

특징이 다른 여러 호스트를 서로 연결해 통신하려면 연결방식을 표준화해야 하는데 국제 표준화 단체인 ISO(International Standard Organization)에서는 OSI(Open System Interconnection) 7계층 모델을 제안하였다. 쉽게 말해 서로 다른 컴퓨터들을 연결해 네트워크를 만들 때 이런 식으로 만들라고 하는 참고 자료이다. 또한 7계층으로 나눔으로써 특정한 곳에 이상이 생기면 다른 계층의 장비 및 소프트웨어를 확인할 필요 없이 이상이 생긴 단계의 장비만 고치면 되는 이점을 가지고 있다.

이 모델에 따르면 네트워크에 연결된 호스트는 6개 계층으로 모듈화된 통신 기능을 갖추어야 하며 이로써 데이터 송수신이 가능해진다.

일반 사용자는 OSI 7계층 맨 위에 있는 응용 계층을 통해 데이터의 송수신을 요청하며, 이 요청은 하위 계층으로 순차적으로 전달되어 맨 아래에 있는 물리 계층을 통해 상대 호스트에 전송된다. 그리고 요청이 각 계층으로 하달되는 과정에서 계층별로 담당하는 기능을 수행해 데이터를 안전하게 전달한다.

데이터를 수신한 호스트에서는 송신 호스트와 반대 과정으로 처리한다.



출처 : [네이버 지식백과] OSI 7계층 모델 (데이터 통신과 컴퓨터 네트워크)

쉽게 말해 데이터를 전해주는 사용자(송신자)가 데이터를 보내면 데이터가 7계층인 응용 계층을 시작으로 1계층인 물리 계층까지 이동한다. 그리고 그게 다시 1계층인 물리 계층을 시작으로 7계층 응용 계층까지 이동해서 자료를 받는 사용자(수신자)에게 데이터가 전해진다.

*1~3계층을 하위 계층, 4~7계층을 상위 계층이라 한다.

◆TCP/IP 모델 또는 인터넷 프로토콜 스위트(Internet Protocol Suite)

OSI 모델과 함께 자주 나오는 것이 TCP/ IP 모델이다.

인터넷에서 컴퓨터들이 서로 정보를 주고 받는데 쓰이는 프로토콜의 모음인데 TCP와 IP를 가장 많이 쓰이기 때문에 TCP/ IP(Transmission Control Protocol/Internet Protocol)라고도 불린다.

인터넷에 연결된 서로 다른 기종의 컴퓨터들이 데이터를 주고받을 수 있도록 하는 표준 프로토콜로 인터넷에 연결하고자 하는 호스트는 반드시 IP 프로토콜을 지원해야 한다.

OSI 모델은 7계층으로 나뉘어져 있는 것과 달리 TCP/IP 모델은 4계층으로 나뉘는데(5계층으로 나누기도 함) OSI 계층의 응용, 표현, 세션계층을 TCP/IP 모델에서는 응용 계층으로 볼 수 있고 데이터 링크, 물리 계층은 네트워크 액세스 계층으로 볼 수 있다.

TCP/ IP가 먼저 만들어졌고 국제표준인 OSI 7계층을 만들었을 때 이미 많은 곳에서 TCP/IP가 이미 사용되고 있었기에 현업에서는 TCP/IP 모델이 사용되고 있다.

▶TCP/IP 모델의 구조

OSI	TCP/IP	기능
응용, 표현, 세션 계층	응용 계층	응용 프로그램 간의 데이터 송·수신 제공 ex) SMTP, HTTP, FTP, DHCP, SNMP 등
전송 계층	전송 계층	호스트 간의 신뢰성 있는 통신 ex) TCP, UDP
네트워크 계층	인터넷 계층	데이터 전송을 위한 주소 지정, 경로 설정을 제공 ex) IP, ICMP, IGMP, ARP, RARP
데이터 링크, 물리 계층	네트워크 액세스 계층	실제 데이터(프레임)을 송·수신하는 역할

▶TCP/IP 모델의 특징

TCP	OSI 7계층의 전송 계층에 해당 신뢰성 있는 연결형 서비스 제공 패킷의 다중화, 순서 제어, 오류 제어, 흐름 제어 기능을 제공
IP	OSI 7계층의 네트워크 계층에 해당 데이터그램을 기반으로 하는 비연결형 서비스 제공 패킷의 분해/조립, 주소 지정, 경로 선택 기능 제공

TCP/IP는 다양한 기기와 앱에서 효율적으로 통신하고 데이터를 전송할 수 있도록 하는 방식이다.

TCP/ IP 모델은 이 정도로 하고 다시 OSI 계층으로 돌아가자.

▶OSI 7계층의 역할 및 기능

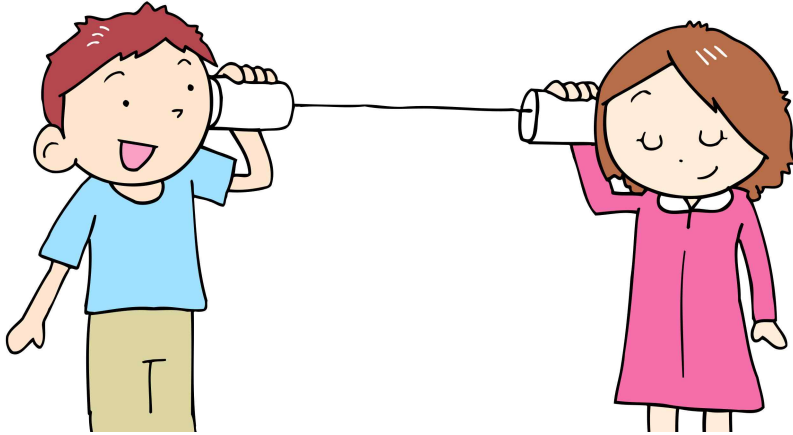
시작하기에 앞서 OSI 7계층은 상하구조 체계가 있 계층이 동작하기 위해선 아래 계층들이 잘 작동한다는 전제로 작동한다.

예를 들면 물리계층(1층)이 만약 영어 프리토킹 가능하다고 생각하면 데이터링크계층(2층)은 불어 프리토킹 가능하다고 나와 있지만(영어 프리토킹 가능하다는 전제는 깔고 가는 것이다)

① 물리 계층(Physical Layer)

나와 다른 사람이 소통을 하기 위해서는 전송매체가 필요하다.

송신자의 물리계층에서 0과 1로 이루어진 디지털 신호(데이터)를 전기신호로 변환해서 전송매체(케이블)을 통해 이동한 후 수신자의 물리계층에서 다시 0과 1로 이루어진 디지털 신호(데이터)로 변환된다.



물리(1)	<ul style="list-style-type: none">• 전송에 필요한 두 장치 간의 실제 접속과 절단 등 기계적, 전기적, 기능적, 절차적 특성에 대한 규칙을 정의• 받은 데이터를 전기신호로 변환하기 때문에 전송매체와 관련이 깊다 ex) 0과 1의 디지털 신호를 전기신호로 변환
-------	---

② 데이터 링크 계층(DataLink Layer)

데이터 링크 계층에서 MAC(Media Access Control address)라는 것이 나오는데 하드웨어 주소 혹은 물리적 주소라고 불리기도 하는데 컴퓨터간 데이터를 전송하기 위한 컴퓨터의 물리적 주소이다.

데이터 링크 계층은 직접 연결된 네트워크 장치 간에 전달이므로 엄청 먼 누군가에게 데이터를 보내려면 엄청나게 많은 사람을 거쳐야 할 것이다.

이렇게 나와 이웃 간의 통신을 보장해주는 것이 데이터 링크 계층이고 이런 통신에 필요한 것이 MAC 주소이다.

MAC주소는 주민등록번호와 비슷한 느낌으로 네트워크 계층에서 나오게 될 IP주소가 같더라도 MAC 주소로 해당 컴퓨터가 식별이 가능해지는 것이다.

다른 사용자에게 데이터를 전송할 때 전송하는 데이터의 IP주소뿐 아니라 MAC주소도 알아야하는데 IP주소를 통해 MAC주소를 알아오는 기능을 하는 프로토콜이 ARP이다.

그렇다면 MAC주소만 알면 되지 IP 주소를 알아야 하는 이유는 뭘까?

서울에서 김서방찾기, 모래에서 바늘찾기라는 속담을 아는가.

범위가 너무 넓으면 찾기 힘들다는 것이다.

IP 주소를 예를 들면 서울 동작구라고 한다면 MAC 주소를 서울 동작구 xx로 xx동 xx호 홍길동이라고 생각하면 된다.



데이터 링크(2)	<ul style="list-style-type: none">•직접 연결된 서로다른 네트워크 장치 간에 신뢰성 있고 효율적인 정보 전송을 할 수 있도록 시스템 간 연결 설정 유지 및 종료를 담당(흐름제어, 오류제어, 접근제어, 동기화)•네트워크 기기 간의 데이터 전송과 물리 주소(MAC)을 결정 <p>ex) 주요 프로토콜 : Ethernet(이더넷), HDLC, PPP 등</p>
-----------	--

③ 네트워크 계층(Network Layer) = TCP/IP 계층의 인터넷 계층

어디로 가야하는지 간략한 주소로 경로설정을 해서 가이드라인을 잡아주는 것이다.

네트워크(3)	<ul style="list-style-type: none"> ● 개방 시스템들 간의 네트워크 연결을 관리하는 기능과 데이터의 교환 및 중계 기능 *라우팅 : 주소(IP)를 정하고 경로 설정, 전달하는 것이 이루어짐 ● 한 네트워크에서 다른 네트워크로 데이터 전송 <p>ex) 주요 프로토콜 : IP, ARP, ICMP, NAT, RIP, BGP, OSPF 등</p>
---------	--

IP(Internet Protocol)	IP는 인터넷에 접속된 모든 호스트에게 할당된 IP주소를 이용하여 데이터가 최종 목적지까지 도달하기 위한 경로를 결정한다.
ICMP(Internet Control Message Protocol)	IP와 조합하여 통신 중에 발생하는 오류의 처리와 전송 경로 변경 등을 위한 제어 메시지를 관리
IGMP(Internet Group Management Protocol)	멀티캐스트를 지원하는 호스트나 라우터 사이에서 멀티캐스트 그룹 유지를 위해 사용
ARP(Address Resolution Protocol)	호스트의 IP 주소를 호스트와 연결된 네트워크 접속장치의 물리적 주소(MAC Address)로 바꿈
RARP(Reverse Address Resolution Protocol)	물리적 주소(MAC Address)를 IP 주소로 변환
RIP(Routing Information Protocol)	<ul style="list-style-type: none"> - 최소 Hop count를 파악하여 라우팅하는 프로토콜 - 거리와 방향으로 길을 찾아가는 Distance Vector 다이나믹 프로토콜 - 최단거리 즉, Hop count가 적은 경로를 택하여 라우팅하는 프로토콜로 Routing Table에 인접 라우터 정보를 저장하여 경로를 결정
NAT(Network Address Translation)	외부 네트워크에서 알려진 공인 IP 주소와 사설 IP 주소를 사용하는 내부 네트워크에서 IP 주소를 변환하는 것

▶라우팅

네트워크의 구성 형태에 대한 정보는 라우팅 테이블(Routing Table)이라는 기억 장소에 보관되며 이 정보를 이용해 패킷이 목적지까지 도달하기 위한 경로를 선택한다.

송수신 호스트 사이의 패킷 전달 경로를 선택하는 과정을 라우팅(Routing)이라 하고, 라우팅 테이블 정보는 네트워크 관리자나 네트워크 자신의 판단에 의해 계속 변경될 수 있다.

인터넷에는 수많은 호스트가 연결되므로 관리하는 라우팅 정보가 매우 많다. 그러다 보니 이를 적절히 관리하여 효과적으로 라우팅하는 작업이 생각보다 쉽지 않은데 그런 라우팅과 관련된 프로토콜을 알아보자.

네트워크 내부의 라우팅이나 네트워크 관리를 독자적으로 운영하는 것을 자율시스템(Autonomous System)이라고 하는데 자율시스템의 구성요소가 라우팅 정보를 저장·관리하며 이 정보를 사용하는 프로토콜을 인터넷 라우팅 프로토콜이라고 한다.

IGP (Interior Gateway Protocol, 내부 게이트웨이 프로토콜)	자율시스템 내의 라우팅에 사용되는 프로토콜 (자율시스템 내부에 위치한 게이트웨이를 내부 게이트웨이라고 함)
EGP (Exterior Gateway Protocol, 외부 게이트웨이 프로토콜)	자율시스템 간의 라우팅, 즉 게이트웨이 간의 라우팅에 사용되는 프로토콜 (자율시스템을 연결하는 게이트웨이를 외부 게이트웨이라고 함)

구분	프로토콜	설명
IGP	RIP(Routing Information Protocol)	현재 가장 널리 사용되는 라우팅 프로토콜로 거리 벡터 라우팅 프로토콜이라고도 불리며 라우터 간에 경로 정보를 교환하여 최단 경로를 계산한다.
	OSPF(Open Shortest Path First)	라우터 간에 경로 정보를 교환하고 각 라우터의 링크 상태 정보를 수집하여 네트워크의 최적 경로를 계산하며 RIP의 단점을 해결하여 새로운 기능을 지원하는 인터넷 프로토콜
EGP	BGP(Border Gateway Protocol)	자율시스템 간의 라우팅 프로토콜로 EGP의 단점을 보완

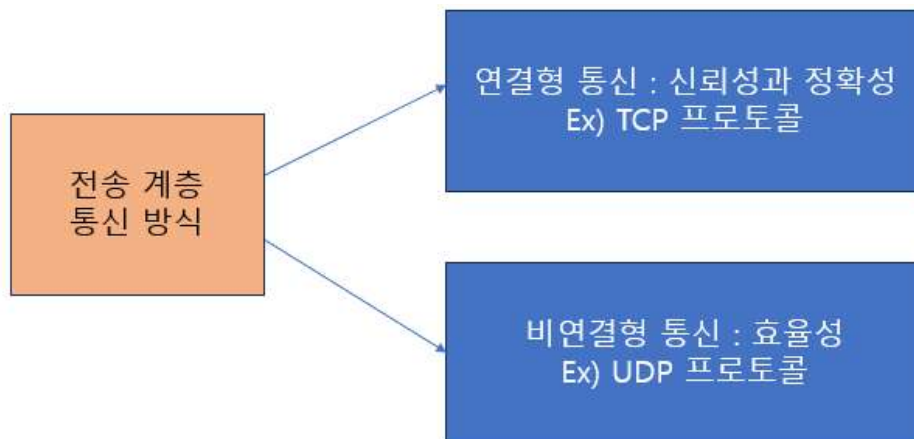
④ 전송 계층(Transport Layer)

데이터의 전달을 담당하는 계층으로 데이터가 전송되는 과정은 기본적으로 비신뢰성 환경인데 그래서 데이터 손실되거나 유실되며 에러가 발생하기도 한다. 목적지에 신뢰성 있게 보낼 수 있게 해준다.

전송(4)	<ul style="list-style-type: none"> ● 논리적 안정과 균일한 데이터 전송 서비스를 제공함으로써 종단시스템 간에 투명한 데이터를 전송 가능하게 함 ● 신뢰성있고 정확한 전달 ex) 주요 프로토콜 : TCP, UDP
-------	---

TCP(Transmission Control Protocol)	인터넷상에서 데이터를 메시지의 형태로 보내기 위해 IP와 함께 사용하는 프로토콜로 연결 지향방식, 높은 신뢰성, 흐름 제어 및 혼잡 제어 등의 특징을 가진다. ex) 파일 전송
UDP(User Datagram Protocol)	데이터를 데이터그램 단위로 처리하는 프로토콜로 비연결 방식, TCP보다 속도가 빠르고 낮은 신뢰성등의 특징을 가진다. ex) 실시간 서비스(Streaming)
RTP(Real-time Transport Protocol)	IP 네트워크 상에서 실시간 멀티미디어 데이터를 전송하기 위한 프로토콜로 주로 오디오, 비디오 및 데이터의 실시간 전송에 사용된다.

TCP는 연결형이고 신뢰성을 보장하니까 목적지에 잘 도착했는지 확인하는 등기 우편인 반면, UDP는 비연결형이고 효율성만 생각하니 목적지에 잘 도착했는지는 확인 안하는 일반 우편이다.



⑤ 세션 계층(Session Layer)

어떤 홈페이지에 로그인을 하고 일정기간이 지나면 자동으로 로그아웃이 되는 경우가 있다. 홈페이지 로그인 상태를 유지해주는 것처럼 송수신 간의 관련성을 유지하고 관리한다.

세션 계층의 중요한 기능으로 동기화라는 게 있는데 동기화는 게임할 때 saving point처럼 중간 저장같은 느낌이다. 동기화를 하면 동기화 하기 이전 존재하는 데이터는 모두 처리가 완료된 상태를 의미하며 송수신 중 오류가 발생하면 처음부터가 아닌 동기화 이후부터 재전송한다.

세션(5)	<ul style="list-style-type: none"> ● 송·수신 측 간의 관련성을 유지하고 대화 제어 담당, 연결을 생성, 유지, 종류를 관리하는 계층 ● 응용 프로그램들 간의 접속을 설정, 유지하고 끊어질 경우 데이터를 재전송하거나 연결을 복구 ex) 무전기처럼 일방적인 소통인지, 전화처럼 쌍방 소통인지 같은 것을 정함
-------	--

⑥ 표현 계층(Presentation Layer)

표현(6)	<ul style="list-style-type: none"> ● 응용 계층으로부터 받은 데이터를 세션 계층에 보내기 전에 통신에 적당한 형태로 변환하고 세션 계층에서 받은 데이터는 응용 계층에 맞게 변환하는 기능(코드 변환, 데이터 암호화, 데이터 압축, 구문 검색) <p>ex) 보내는 사람의 의도대로 자료를 확인하기 위해 공통 양식으로 표현하는 계층</p> <p>ex) 해당 데이터가 text확장자인지 jpg확장자인지 구분</p> <p>*암호화 : 암호가 되었다면 암호 해석해야 함을 알려줌</p> <p>*인코딩 : 영어로 되었다면 한국어가 아니라 영어로 읽어야 함을 알려줌</p> <p>*압축 : 압축해제가 필요함을 알려줌</p>
-------	---

⑦ 응용 계층(Application Layer)

응용(7)	<ul style="list-style-type: none"> ● 사용자가 OSI 환경에 접근할 수 있도록 서비스를 제공 ● 유저와 가장 가까운 층으로, 응용 프로세스(크롬이란 웹 클라이언트에 접속해)와 직접 관계해 일반적인 응용 서비스(HTTP 프로토콜 등)를 수행 <p>ex) 우리가 응용 프로그램, 앱을 사용하는 활동</p> <p>ex) 주요 프로토콜 : HTTP, SMTP, FTP, TELNET, SNMP, DHCP</p>
-------	---

SMTP(Simple Mail Transfer Protocol)	전자우편 전송
HTTP(HyperText Transfer Protocol)	웹 서버와 웹 브라우저 통신
FTP(File transfer Protocol)	파일 전송
TELNET	멀리 떨어져 있는 컴퓨터에 접속하여 자신의 컴퓨터처럼 사용할 수 있도록 해주는 서비스
DNS(Domain Name System)	도메인 이름을 IP 주소로 매핑(Mapping)하는 시스템 *매핑은 1:1 대응이라고 생각하면 됨
SNMP(Simple Network Management Protocol)	TCP/IP의 네트워크 관리 프로토콜
DHCP(Dynamic Host Configuration Protocol)	IP주소와 각종 TCP/IP 프로토콜의 기본 설정을 클라이언트에게 자동적으로 제공해주는 프로토콜

▶OSI 7계층의 역할 및 기능 요약

계층	특징
응용(7)	<ul style="list-style-type: none"> ●사용자가 OSI 환경에 접근할 수 있도록 서비스를 제공 ●유저와 가장 가까운 층으로, 응용 프로세스(크롬이란 웹 클라이언트에 접속해)와 직접 관계해 일반적인 응용 서비스(HTTP 프로토콜 등)를 수행 <p>ex) 우리가 응용 프로그램, 앱을 사용하는 활동</p> <p>ex) 주요 프로토콜 : HTTP, FTP, SMTP, TELNET, SNMP, DHCP</p>
표현(6)	<ul style="list-style-type: none"> ●응용 계층으로부터 받은 데이터를 세션 계층에 보내기 전에 통신에 적당한 형태로 변환하고 세션 계층에서 받은 데이터는 응용 계층에 맞게 변환하는 기능(코드 변환, 데이터 암호화, 데이터 압축, 구문 검색) <p>ex) 보내는 사람의 의도대로 자료를 확인하기 위해 공통 양식으로 표현하는 계층</p> <p>ex) 해당 데이터가 text확장자인지 jpg확장자인지 구분</p> <p>*암호화 : 암호가 되었다면 암호 해석해야 함을 알려줌</p> <p>*인코딩 : 영어로 되었다면 한국어가 아니라 영어로 읽어야 함을 알려줌</p> <p>*압축 : 압축해제가 필요함을 알려줌</p>
세션(5)	<ul style="list-style-type: none"> ●송·수신 측 간의 관련성을 유지하고 대화 제어 담당, 연결을 생성, 유지, 종료를 관리하는 계층 ●응용 프로그램들 간의 접속을 설정, 유지하고 끊어질 경우 데이터를 재전송하거나 연결을 복구 <p>ex) 무전기처럼 일방적인 소통인지, 전화처럼 쌍방 소통인지 같은 것을 정함</p>
전송(4)	<ul style="list-style-type: none"> ●논리적 안정과 균일한 데이터 전송 서비스를 제공함으로써 종단시스템 간에 투명한 데이터를 전송 가능하게 함 ●신뢰성있고 정확한 전달 <p>ex) 주요 프로토콜 : TCP, UDP</p>
네트워크(3)	<ul style="list-style-type: none"> ●개방 시스템들 간의 네트워크 연결을 관리하는 기능과 데이터의 교환 및 중계 기능 <p>*라우팅 : 주소(IP)를 정하고 경로 설정, 전달하는 것이 이루어짐</p> <ul style="list-style-type: none"> ●한 네트워크에서 다른 네트워크로 데이터 전송 <p>ex) 주요 프로토콜 : IP, ARP, ICMP, NAT, RIP, BGP, OSPF 등</p>
데이터 링크(2)	<ul style="list-style-type: none"> ●직접 연결된 서로다른 네트워크 장치 간에 신뢰성 있고 효율적인 정보 전송을 할 수 있도록 시스템 간 연결 설정 유지 및 종료를 담당(흐름제어, 오류제어, 접근제어, 동기화) ●네트워크 기기 간의 데이터 전송과 물리 주소(MAC)을 결정 <p>ex) 주요 프로토콜 : Ethernet(이더넷), HDLC, PPP 등</p>
물리(1)	<ul style="list-style-type: none"> ●전송에 필요한 두 장치 간의 실제 접속과 절단 등 기계적, 전기적, 기능적, 절차적 특성에 대한 규칙을 정의 ●받은 데이터를 전기신호로 변환하기 때문에 전송매체와 관련이 깊다 <p>ex) 0과 1의 디지털 신호를 전기신호로 변환</p>

전송 데이터는 송신 호스트의 응용 계층에서 시작해 하위 계층으로 순차적으로 전달되어, 최종적으로 물리 계층에서 수신 호스트에 전달되는데 데이터가 하위 계층으로 이동할 때는 각 계층의 프로토콜에서 정의한 헤더 정보가 추가되며(물리 계층 제외) 그렇게 각 계층에서 데이터를 전송할 때 필요한 정보(헤더)를 붙여 다음 계층에 보내는 과정을 캡슐화(Encapsulation)라고 한다.

수신 호스트에서는 데이터를 상위 계층으로 순차적으로 이동시켜 응용 계층에 도착하게 되는데 상위 계층으로 이동하며 순차적으로 추가됐던 헤더 정보를 제거하고 해석하는데 이 과정을 역캡슐화(Decapsulation)라고 한다.



출처 : nellholic108 velog

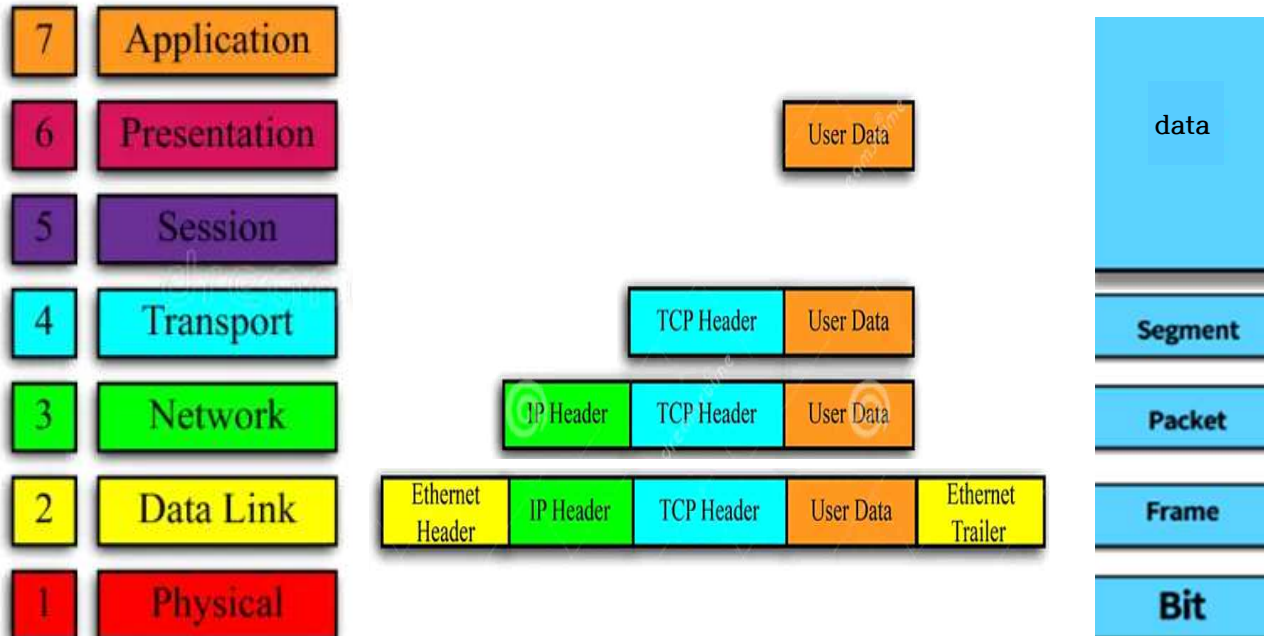
헤더는 각 계층의 필요한 정보를 담고 있다.

캡슐화 과정에서

User Data가 전송 계층으로 가면서 TCP 헤더가 붙으며 Segement가 되고

그 다음 네트워크 계층으로 가면서 IP 헤더가 붙으면서 Packet이 되고

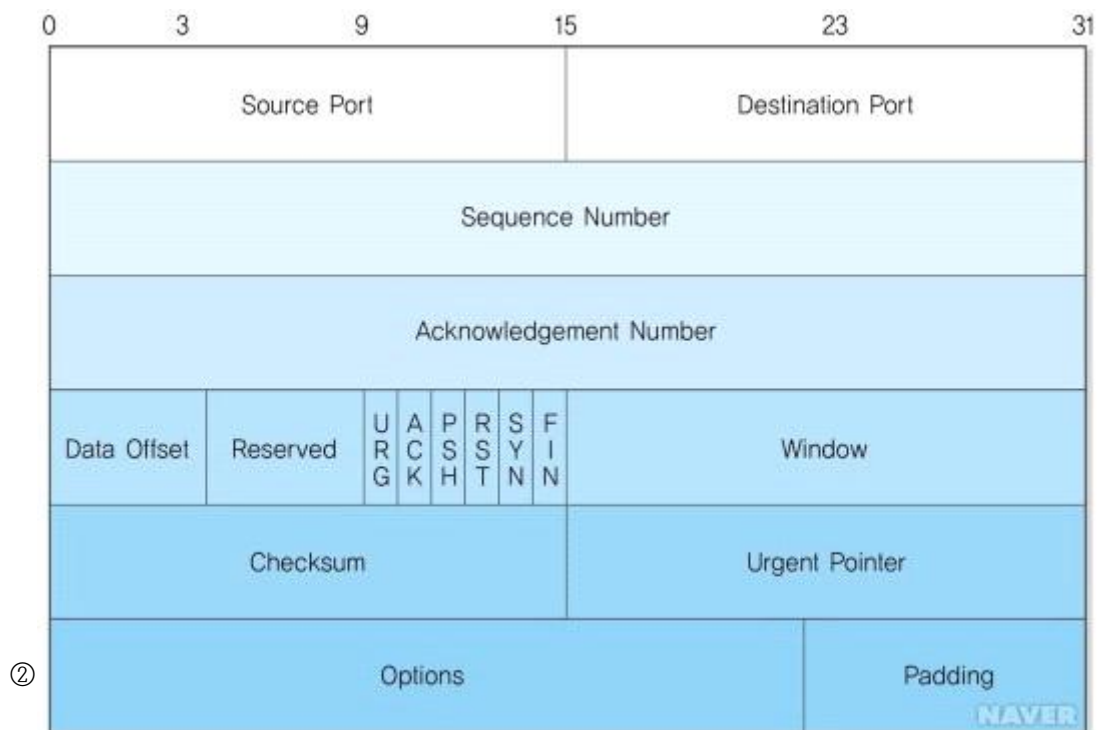
그 다음 데이터링크 계층으로 가면서 Ethernet Header와 Trailer가 붙으면서 Frame이 된다



그 중 TCP헤더와 IP 헤더에 대해서만 간략하게 알아보자.

① TCP 헤더의 구조

Source Port	출발지 포트 번호(16bit)
Destination Port	목적지 포트 번호(16bit)
Sequence Number	일련번호(32bit) : 송신자가 지정하는 순서 번호
Acknowledgement Number	확인 응답 번호(32bit) : 데이터를 정상적으로 수신했을 때 보내는 번호
Data Offset	헤더 길이(4bit)
Reserved	예약 영역(6bit)
URG/ACK/PSH/RST/SYN/FIN	TCP 플래그(6bit) : 각 비트가 활성화되면서 데이터 전송에 사용
Wiindow	윈도우 크기(16bit) : 한번에 전송할 수 있는 데이터의 양
Checksum	체크섬(16bit) : 오류검출 코드
Urgent Pointer	긴급 포인터(16bit) : 긴급 데이터 처리
Option	TCP 옵션



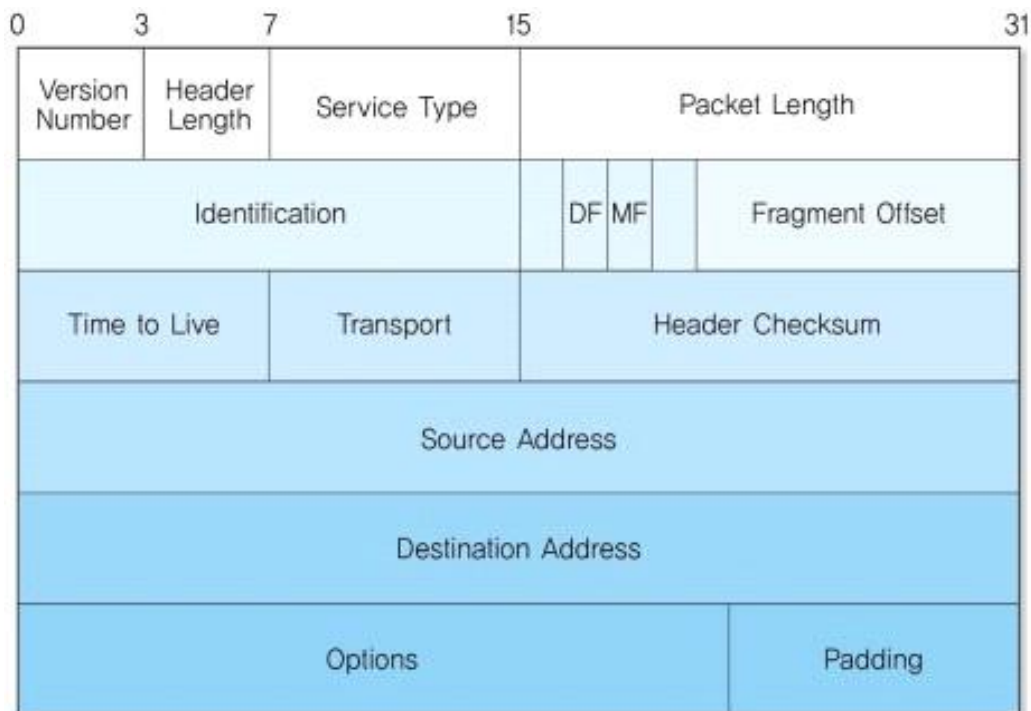
출처 : [TCP 헤더 \(naver.com\)](https://blog.naver.com/dlqnf33)

①-1 TCP 헤더의 플래그 비트

URG (Urgent)	Urgent Pointer 필드가 유효한지를 나타내며, 긴급 데이터를 전송하기 위해 사용
ACK (Acknowledgement)	Acknowledgment Number 필드가 유효한지를 나타냄
PSH (Push)	현재 세그먼트에 포함된 데이터를 상위 계층에 즉시 전달하도록 지시할 때 사용
RST (Reset)	연결의 리셋이나 유효하지 않은 세그먼트에 대한 응답용으로 사용
SYN (Synchroniz)	연결 설정 요구를 의미하는 플래그 비트므로 가상 회선 연결을 설정하는 과정에서 사용
FIN (Finish)	한쪽 프로세서에서 더는 전송할 데이터가 없어 연결을 종료하고 싶다는 의사 표시를 상대방에게 알리는 데 사용

② IP 헤더의 구조

Version Number	버전(4bit)
Header Length	헤더 길이(4bit)
Service Type	서비스 유형(8bit) : 패킷의 전송 우선순위 제공
Packet Length	전체 패킷 길이(16bit)
Identification(ID)	일련번호(16bit) : 전송하고자 하는 패킷을 식별하기 위해 부여하는 번호
DF/MF	플래그(3bit) : 비트값을 통해 단편화(분할)을 금지하거나 추가
Fragment Offset	조각의 위치(13bit) : 단편화(분할)된 패킷의 위치를 표현
Time to Live	TTL(8bit) : 통과 가능한 라우터의 남은 수
Transport	프로토콜 타입(8bit) : IP 패킷을 생성하도록 IP 프로토콜에게 데이터 전송을 요구한 전송 계층의 프로토콜을 가리킴
Header Checksum	헤더 체크섬(16bit) : 에러 발생 유무 검사
Source Address	출발지 IP 주소(32bit)
Destination Address	목적지 IP 주소(32bit)
Options	옵션



출처 : [IP 헤더 \(naver.com\)](https://blog.naver.com/dlqnf33)

5. 데이터링크 계층 심화

◆ 데이터 링크 계층

- 물리적으로 이웃하여 연결된 두 호스트 간의 신뢰성 있는 데이터 전송을 지원하며 물리 계층에서 발생하는 전송 오류를 감지하고, 복구하는 기능이 필요하다.
- 오류는 통신회선의 순간적인 절단 현상, 통신회선의 잡음, 장치의 기계적/구조적 원인, 전원 중단 등 전기적 원인 때문에 발생하는데 물리 계층에서는 데이터를 주고받기만 할 뿐 오류 검사는 불가능하여 오류를 검출, 수정, 처리하는 기능은 데이터 링크 계층에서 담당한다.

▶프레임

- 데이터 링크 계층에서는 전송 데이터를 프레임(Frame)이라는 작은 단위로 나누어 처리한다.
- 전송 프레임에는 상위 계층에서 보낸 전송 데이터에 오류 확인을 위한 체크섬(Checksum), 송수신 호스트의 주소, 기타 프로토콜에서 사용하는 제어 코드 같은 정보가 포함된다.

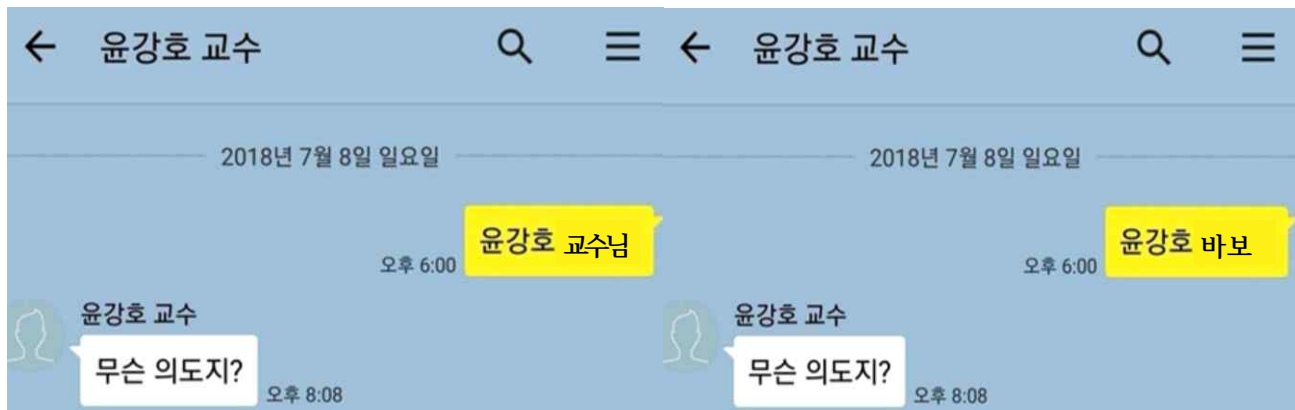
▶데이터 링크 계층의 주요 역할-오류 제어

- 오류 복구 기능을 수행하여 상위 계층인 네트워크 계층에 전송 오류가 발생하지 않는 논리적 전송 선로를 보장한다.
- 오류의 종류에는 데이터가 깨져서 도착하는 프레임 변형과 데이터가 목적지에 도착하지 못하는 프레임 분실이 있다.

① 프레임 변형 예시

(단, 이해를 돕기 위한 간단한 예시로 실제 프레임 변형과는 관계가 없다)

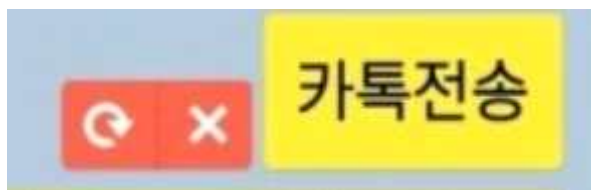
윤강호 교수님이라고 보냈는데 데이터가 깨져서 윤강호 바보라고 도착해버린 것이다.



② 프레임 분실 예시

(단, 이해를 돕기 위한 간단한 예시로 실제 프레임 분실과는 관계가 없다)

윤강호 교수님에게 카톡을 보냈는데 데이터가 목적지에 도착하지 못하는 것이다.



- 오류 복구 기능을 제공하려면 일차적으로 물리 매체를 이용한 전송 과정에서 오류가 발생했는지 감지할 수 있어야 하는데 수신 호스트의 응답 프레임으로 인지가 가능하며 컴퓨터 네트워크에서는 일반적으로 송신 호스트가 원래의 데이터를 재전송하는 기법을 사용한다.

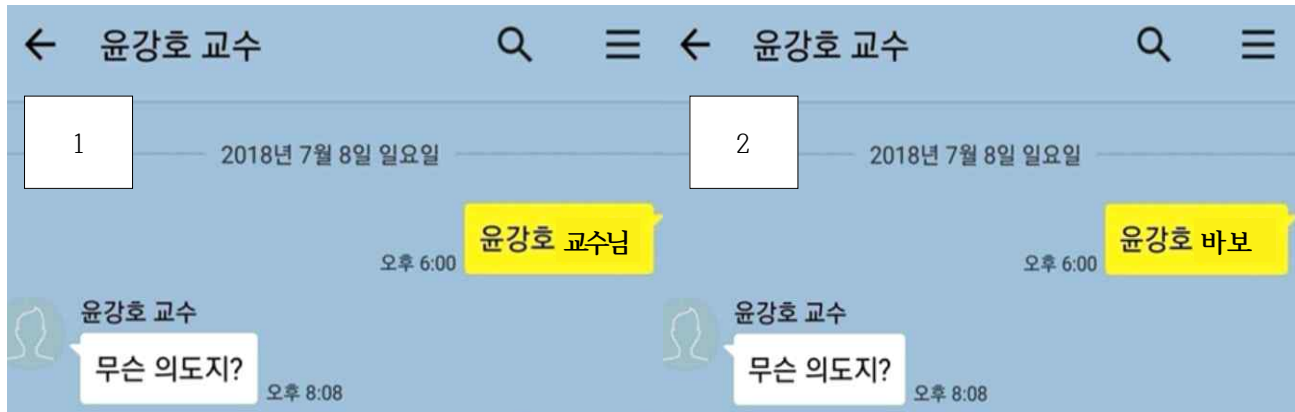
③ 오류 발생 여부 인지 방법

(ㄱ) 수신 호스트의 응답 프레임

송신 호스트가 전송한 데이터 프레임의 일부가 깨지는 프레임 변형 오류를 확인한 수신 호스트는 송신 호스트에게 응답 프레임을 전송해 원래의 데이터 프레임을 재전송하도록 요구할 수 있다.

수신 호스트가 전송하는 응답 프레임의 종류에는 데이터 프레임이 정상적으로 도착했을 때 회신하는 **긍정 응답 프레임**과 데이터 프레임이 깨졌을 때 회신하는 **부정 응답 프레임**이 있다.

송신 호스트의 재전송 기능은 수신 호스트의 부정 응답 프레임 회신에 의해 이루어진다.

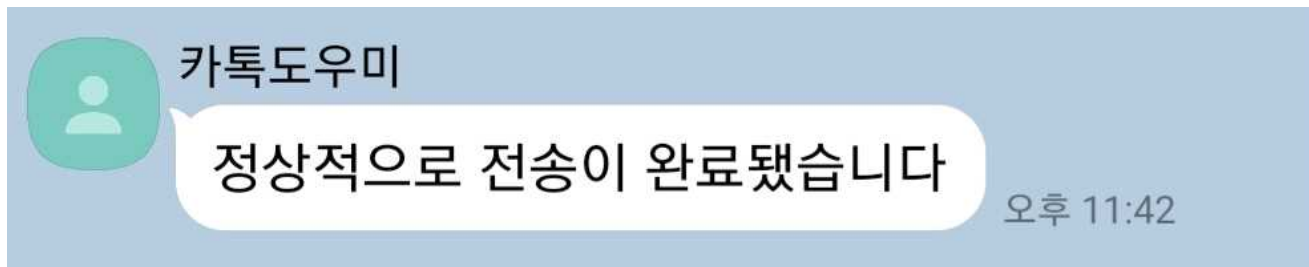


(ㄱ)-1 긍정 응답 프레임

1번의 경우 아래와 같이 카톡도우미가 정상적으로 전송이 완료됐다고 알려준다.

이렇게 보내는 긍정 응답을 ACK(ACKnowledgement)라고 한다.

(단, 이해를 돕기 위한 간단한 예시로 실제 프레임 변형과는 관계가 없다)

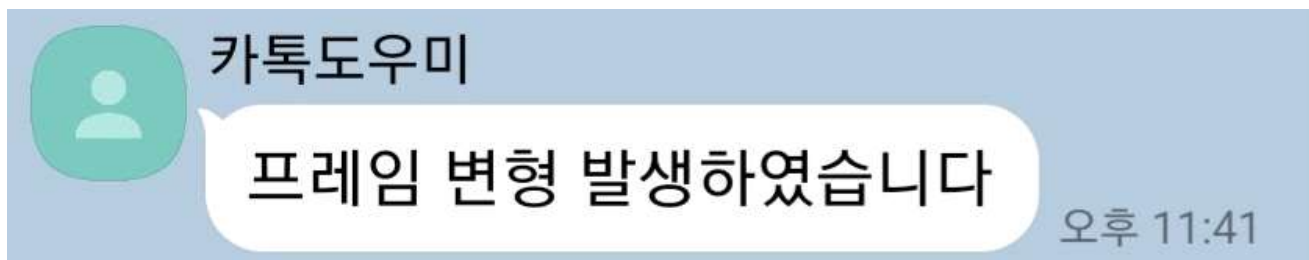


(ㄱ)-2 부정 응답 프레임

2번의 경우 아래와 같이 카톡도우미가 프레임 변형 발생했다고 알려주는데 이게 바로 부정 응답 프레임이다.

이렇게 보내는 부정 응답을 NAK(Negative Acknowledgement)라고 한다.

(단, 이해를 돕기 위한 간단한 예시로 실제 프레임 변형과는 관계가 없다)



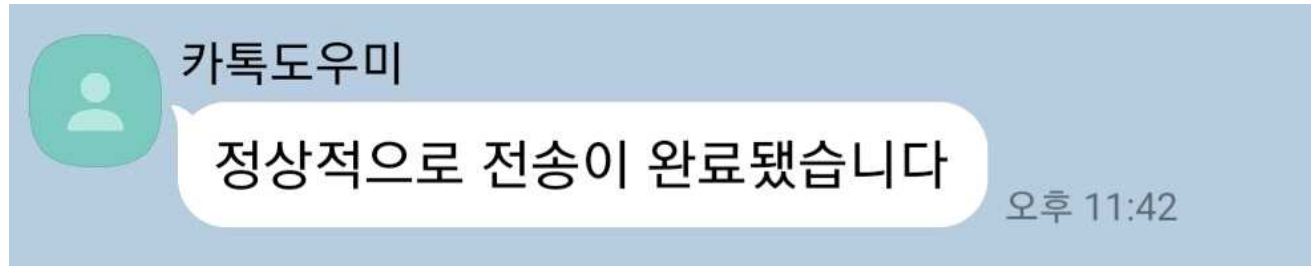
(ㄴ) 송신 호스트의 타이머 기능

송신 호스트가 전송한 데이터 프레임이 수신 호스트에게 도착하지 못하는 프레임 분실 오류가 발생하면 수신 호스트는 이 사실을 인지할 수 없으므로 송신 호스트쪽에서 오류를 해결해야 한다.

송신 호스트는 데이터 프레임을 전송한 후에 일정 시간 이내에 수신 호스트로부터 긍정 응답 프레임 회신이 없으면 타임아웃(Timeout) 기능을 동작시켜 데이터 프레임을 재전송한다.

프레임을 전송했는데 아래와 같이 카톡도우미의 긍정 응답 프레임이 없으면 타임아웃 기능이 동작돼 데이터 프레임을 재전송한다.

(단, 이해를 돕기 위한 간단한 예시로 실제 프레임 분실과는 관계가 없다)



데이터 링크 계층의 오류 복구 기능이 수행되는 과정에서 동일한 데이터 프레임이 수신 호스트에 중복해 도착할 수 있기 때문에 중복 데이터 처리는 프레임 내부에 각 프레임의 고유 번호인 순서 번호(Sequence Number)를 기록하여 해결한다.

간단하게 오류제어에 대해 알아보았는데 오류제어에 대해 좀 더 알아보자.

▶오류 제어2

오류는 다양한 방식으로 제어 가능하다.



1	오류 무시	영문 텍스트나 숫자가 포함되지 않은 간단한 문자 전송 등 그다지 중요하지 않은 데이터를 취급하는 데이터 통신에 사용된다.
2	반향(Echo) 검사	루프(Loop) 방식이라고도 하며 전송한 데이터와 수신한 데이터를 서로 비교하여 판단하는 방식이다.
3	검출 후 재전송 (ARQ; Automatic Repeat reQuest)	<ul style="list-style-type: none"> - 오류가 발생하면 수신 측은 송신 측에 오류가 발생한 사실을 알리고 오류가 발생한 프레임을 재전송할 것을 요구한다. - 후진 오류 수정(BEC; Backward Error Correction) 방식 또는 자동 반복 요청 방식이라고도 한다. -수신 측에서 오류를 검출하는 방식으로는 패리티 검사, 블록 합 검사, 순환 중복 검사(CRC) 등이 있다.
4	전진 오류 수정 (FEC; Forward Error Correction)	<ul style="list-style-type: none"> - ARQ 방식과 달리 수신 측에서 오류가 있음을 발견하면 해당 오류를 검출할 뿐만 아니라 오류 수정도 가능한 방식이다. - 자기 정정 방식이라고도 한다.

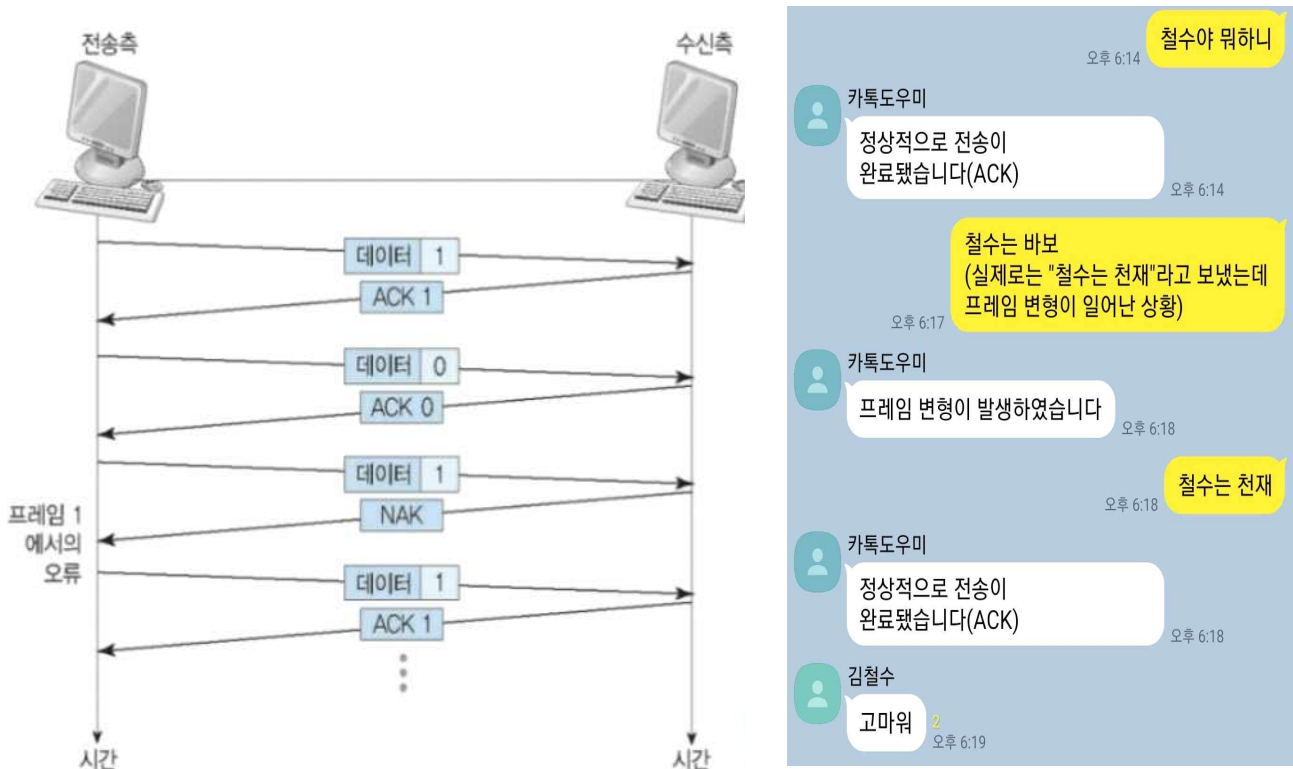
□ ARQ 방식의 종류

ARQ에는 크게 정지 대기 ARQ, 연속적 ARQ, 적응적 ARQ로 3가지로 나뉜다.

① 정지 대기 ARQ

1	정지 대기(Stop and Wait) ARQ	<p>형태가 가장 단순한 ARQ로 송신 측에서 하나의 블록을 전송하면 수신 측은 오류가 발생하였는지 점검한 후 ACK(긍정응답)나 NAK(부정응답)를 보내올 때까지 기다린다.</p> <p>수신 측에서 응답을 받아야 전송할 수 있는 방식이므로 다른 방식보다 전송 효율이 떨어진다.</p>
---	--------------------------	---

아래 카톡을 예로 ACK든 NAK든 수신측에 답변이 와야 그 다음 데이터를 전송할 수 있다.



카톡화면은 그냥 이해를 쉽게 하기 위해 만든 것일 뿐 실제 ARQ와 무관함
출처 : [ARQ\(Automatic Repeat Request\)란? \(velog.io\)](https://velog.io)

㉞ 연속적(Continuous) ARQ

2	연속적(Continuous) ARQ	정지대기 ARQ 방식에서 생기는 단점을 줄이기 위해 데이터 블록을 연속해서 보내는 방식을 이용한다. ex) Go-Back N ARQ, Selective-Repeat ARQ(선택적 ARQ)
---	---------------------	---

Go-Back N ARQ	오류가 난 지점부터 전송한 지점까지 모두 재전송 하는 기법
---------------	----------------------------------

Selective-Repeat ARQ (선택적 ARQ)	오류가 난 지점의 프레임만 재전송 하는 기법
-----------------------------------	--------------------------

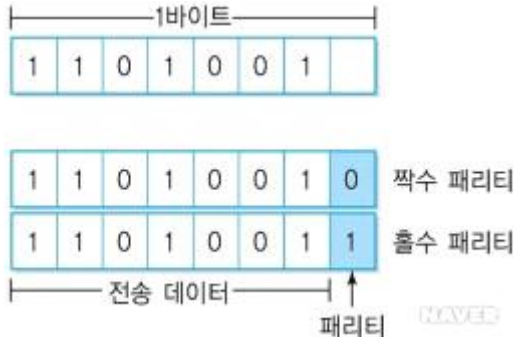
㉟ 적응적(Adaptive) ARQ

3	적응적(Adaptive) ARQ	전송 효율을 최대한 높이려고 데이터 블록의 길이를 동적으로 변경시켜 전송하는 방식이다.
---	-------------------	--

❑ 오류 검출 방식

컴퓨터 네트워크에서는 일반적으로 송신 호스트가 원래의 데이터를 재전송하는 기법을 사용하는데 재전송하기 위해서는 오류를 인지할 수 있어야 하며 전송 프레임에 오류 검출 코드를 넣어 수신 호스트가 전송 과정의 오류를 검출하도록 하는 것이다.

가장 간단한 오류 검출 코드 방법은 패리티 비트를 추가하는 것이고 컴퓨터 네트워크에서는 일반적으로 다항 코드 방식을 사용한다.

1	패리티 비트 검사 (Parity Bit Check)	<p>전송되는 문자마다 패리티 비트를 하나씩 추가해 짝수나 홀수 여부를 검사하는 방법이다.</p> <p>예를 들어 아래와 같이 1101001이라는 데이터가 있다고 하면 맨 뒤 비어있는 칸에 0 또는 1의 패리티 비트를 추가해준다. 총 8칸에 있는 0과 1중 홀수 패리티는 1의 개수를 홀수 개로 유지하고 짝수 패리티는 1의 개수를 짝수 개로 유지한다.</p>  <p>출처 : 오류 검출 (naver.com)</p> <p>데이터 전송 과정에서 오류가 발생하면 1의 개수가 바뀌게 돼 이 변경된 사실을 통해 오류를 검출한다. 단, 오류 비트가 짝수 개 발생하면 오류 검출이 어렵다.</p>
2	블록 합 검사 (Block Sum Check)	<p>짝수개 비트 오류를 검출할 수 없는 패리티 비트 검사를 개선한 방법이다.</p> <p>데이터 블록의 수평과 수직에 각각 패리티 비트를 추가하여 다수의 비트 오류를 검출한다. 단, 오버헤드가 심한 단점이 있다.</p>
3	순환 중복 검사 (CRC; Cyclic Redundancy Check)	<p>집단 오류를 검출하기 위해 다항식 코드를 사용하여 오류를 검사하는 방식이다.</p> <p>각 문자마다 부가 비트를 붙일 필요는 없으나 프레임의 실제 내용으로 계산하는 프레임 검사 순서(FCS; Frame Check Sequence)를 프레임의 끝에 추가하여 전송한다.</p> <p>FCS를 BCS(Block Check Sequence)라고도 한다.</p>

❑ 오류 정정 방식

일반적으로는 오류 검출 후 재전송방식이지만 프레임에 오류 복구 코드를 넣어 수신 호스트가 오류 검출과 복구 기능을 모두 수행하도록 하는 방법도 있다.

해밍 부호 검사(Hamming Code Check)는 1비트 오류를 검출하고 복구하는 기능을 한다.

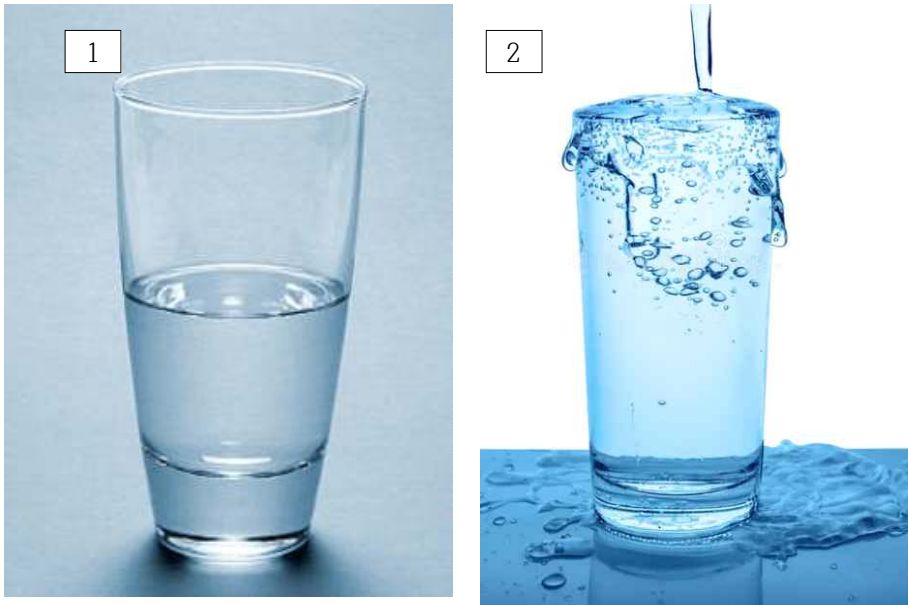
▶데이터 링크 계층의 주요 역할-흐름 제어

오류 제어와 함께 데이터 링크 계층에서 제공하는 주요 기능은 전송 데이터의 속도 조절이다.

송신 호스트는 수신 호스트가 감당할 수 있을 정도의 전송 속도를 유지하면서 데이터 프레임을 전송해야 하는데, 이러한 기능을 흐름 제어(Flow Control)라 한다.

송신 측이 수신 측보다 느릴 때는 문제 없지만 빠르면 수신 측의 한정된 버퍼가 데이터를 처리하지 못하거나 아예 버리는 상황이 발생한다.

대충 생각하면 아래와 같다. 버퍼가 1번과 같은 경우에는 수신 측에서 충분히 데이터를 처리할 여유가 있지만 2번의 경우 데이터를 처리하기 이전에 물이 넘쳐서 데이터가 버려진다.



이와 같이 흐름 제어 기능을 제공하지 않으면 수신 호스트는 자신에게 도착한 데이터 프레임을 내부 버퍼에 보관할 여유를 갖지 못한다. 따라서 전송 매체를 통해 올바르게 도착한 데이터가 분실되는 결과를 초래할 수 있다.

□흐름 제어 방식 종류

1	정지 대기 방식 (Stop-and-Wait)	데이터를 전송할 때 송신 측에서는 한번에 프레임 1개만 전송 가능하며 수신 측에서는 다음 프레임을 맞게 전송하였는지 결정하여 송신 측에 통보한다. 흐름 제어 방식 중 가장 간단하며 프레임을 몇 개의 큰 단위로 전송할 때 효율적이다.
2	슬라이딩 윈도우 방식 (Sliding Window)	데이터를 전송할 때 송신 측에서는 한번에 윈도우 크기만큼 프레임을 연속해서 전송할 수 있으며 수신 측에서는 적절한 간격으로 이 윈도우 크기의 개수만큼 크기를 조절하여 송신 측에 통보한다. 흐름 제어 방식 중에서 가장 대표적인 것이다. 여기서 말하는 윈도우는 메모리 버퍼의 일정 영역을 뜻한다

▶슬라이딩 윈도우 프로토콜(Sliding Window Protocol)

두 호스트 간의 데이터 전송을 위한 일반적인 통신 프로토콜로, 오류 제어와 흐름 제어 기능을 함께 지원한다.

1	정보 프레임 (Information Frame)	상위 계층(세션 계층, 표현 계층, 응용 계층)이 전송을 요구한 데이터를 수신 호스트에 전송하는 용도로 사용하며 데이터와 프레임의 순서 번호, 송수신 호스트의 주소 정보도 포함한다.
2	긍정 응답 프레임 (Positive Acknowledgement)	정보 프레임을 수신한 호스트는 프레임의 내용이 깨졌는지 확인해야 하는데 프레임에 문제가 없으면 송신 호스트에게 해당 프레임을 올바르게 수신했다는 의미로 ACK 프레임을 회신한다
3	부정 응답 프레임 (Negative Acknowledgement)	전송 과정에서 프레임 변형 오류가 발생하면 수신 호스트는 송신 호스트에게 NAK 프레임을 회신한다. 송신 호스트는 오류를 인지하고 정보 프레임을 다시 전송한다.

*자세한 사항은 위에 오류제어 파트를 참고하자

송신 호스트는 송신한 정보 프레임을 자신의 내부 버퍼에 유지해야 하며, 이를 송신 윈도우라고 한다.

수신 호스트는 수신한 정보 프레임을 보관하기 위해 내부 버퍼인 수신 윈도우를 유지할 수 있다.

윈도우(윈도)는 그냥 임시 보관함 정도로 생각하면 될 듯하다.

▶HDLC 프로토콜(High level Data Link Control)

HDLC는 컴퓨터 데이터 통신에 적합한 전송제어방식이자 대표적인 데이터링크 프로토콜 중 하나로 데이터링크 프로토콜은 문자 지향 방식과 비트 지향 방식으로 분류되는데 HDLC는 비트 지향 방식이다.

정보 전송 파트에서 비동기식 전송과 동기식 전송의 차이로 잠깐 나왔던 그 개념인데 HDLC는 동기식 전송이고 동기식 전송 중에 비트 동기 방식이다.

동기식 전송 (Synchronous Transmission)	정보 전송 형태	블록(프레임) 단위
	정보 전송량	송신 측과 수신 측 사이에 미리 정해진 숫자만큼 문자열을 한 묶음으로 만들어 한꺼번에 전송
	클럭(Clock)	송수신 장치의 클럭 일치 (타이밍이 같아야함) (데이터 전송 언제 시작할지 미리 약속)
	동기화 전송 구분	데이터 묶음의 앞쪽에는 반드시 동기 문자가 와야하며 동기 문자는 송신 측과 수신 측이 서로 동기하는데 사용
	중점적인 내용	회선 중심 (기차나 지하철 생각하면 됨) (기차 : 사람이 있든 없든 규칙적으로)
	적합한 전송 상황	데이터가 많거나 고속 처리가 필요할 때 적합
	휴지기간	블록과 블록 사이(한 묶음으로 구성한 문자 사이)에 휴지기간이 없음
	비용/회로 복잡도	비쌈/ 회로 복잡
	전송 효율	휴지기간이 없어서 전송효율 높음
	전송 방법	직렬형태로 작업 처리 작업이 수행 중이면 다른 작업은 대기

□ 문자 동기 방식과 비트 동기 방식의 차이

문자 동기	<p>전송되는 데이터의 블록 앞에 특정 동기 문자인 SYN(00010110)을 붙여 동기를 맞추고 실제 데이터 블록의 앞에는 STX(0010000), 뒤에는 ETX(0011000)를 추가하여 전송 데이터의 시작과 끝을 나타낸다.</p> <p>ex) BSC</p> <p>SYN : 동기문자 STX : 시작문자 ETX : 종료문자</p> <p>DLE : 제어문자 앞에 추가적으로 삽입(데이터 투명성)</p> <p>*바이트 스티핑(Byte stuffing) : 데이터에 제어문자와 동일한 문자가 포함될 경우, 수신측은 프레임의 데이터를 잘 못 해석할 수 있어 제어문자 앞에 DLE문자를 추가적으로 삽입한다.</p>
<div data-bbox="255 822 1260 992"> </div> <p>문자 동기 방식</p> <p>사진 출처 : 동기식 전송 vs 비동기식 전송 : 네이버 블로그 (naver.com)</p>	
비트 동기	<p>전송 단위를 일련의 비트 묶음으로 보고 비트 블록의 처음과 끝을 표시하는 특별한 비트인 플래그 비트(01111110)를 추가해 전송</p> <p>대표적인 비트 동기 방식이 HDLC</p>
<div data-bbox="255 1350 1260 1520"> </div> <p>비트 동기 방식</p> <p>사진 출처 : 동기식 전송 vs 비동기식 전송 : 네이버 블로그 (naver.com)</p>	

□ HDLC 좀 더 알아보기

① 현재는 주소(Address)부분이 8비트밖에 되지 않아 잘 사용되지는 않고 있지만 ISO에 의해 1974년에 제정·공표되어 세계적으로 널리 사용된 표준이자 고속 데이터 전송에 적합한 비트 지향형 프로토콜로 다음과 같은 많은 데이터 링크 제어 프로토콜들의 전신이다.

- LAP-B (Link-Access Procedure Balanced) : 패킷교환망에서 사용
- LLC (Logical Link Control) : LAN에서 사용
- LAP-D (Link-Access Procedure, D Channel) : ISDN에서 사용
- PPP (Point-to-Point Protocol) : 인터넷에서 많이 사용

사진 출처 : [HDLC \(ktword.co.kr\)](http://hdlc.ktword.co.kr)

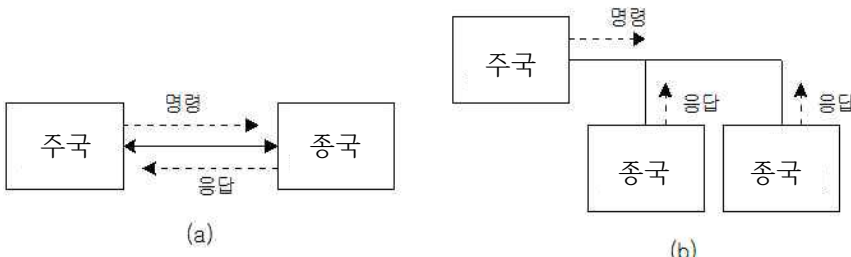
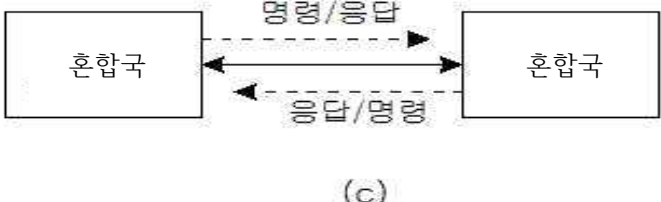
BSC (Binary Synchronous Control)	프레임에 전송 제어 문자를 삽입하여 전송을 제어하는 문자 위주의 프로토콜
LAP-B (Link Access Procedure, Balanced)	HDLC 프로토콜로부터 X.25 패킷교환을 위해 개발된 점대점 데이터링크 접속용 ITU-T 프로토콜 표준
LAP-D (Link-Access Procedure D Channel)	ISDN에서 D 채널 상의 데이터링크의 설정, 유지, 해제 기능과 데이터 프레임 배열, 에러검출 기능 등을 수행하는 비트 중심 프로토콜
PPP (Point-to-Point Protocol)	두 대의 컴퓨터가 직렬 인터페이스를 이용하여 통신을 할 때 필요한 프로토콜로서, 특히 전화회선을 통해 서버에 연결하는 PC에서 자주 사용
LLC (Logical Link Control)	두 장비간에 링크를 설정하고 프레임을 송수신하는 방식과 상위 레이어 프로토콜의 종류를 알리는 역할

② HDLC의 데이터 송수신

HDLC 프로토콜은 컴퓨터가 일대일 혹은 일대다로 연결된 환경에서 데이터의 송수신 기능을 제공하는데 데이터 통신을 위해 연결된 호스트는 일반적으로 주국과 종국으로 구분되며 이들의 기능을 모두 지닌 혼합국도 있다.

- * 주국(Primary station) : 링크 제어를 책임지는 컴퓨터로써 데이터 전송에서 항상 명령을 수행
- * 종국(Secondary station) : 주국의 통제하에서 동작하는 컴퓨터, 명령에 대해서 응답을 수행
- * 혼합국(Combined station) : 명령과 응답을 모두 처리 가능
- * 명령(Command) : 주국에서 종국으로 전달하는 메시지
- * 응답(Response) : 종국에서 주국으로 회신하는 메시지

HDLC 프로토콜의 링크 구성은 불균형 구성과 균형 구성으로 나뉜다.

<p>불균형 구성 (Unbalanced Configuration)</p>	<ul style="list-style-type: none"> • 주국과 종국으로 구성 • 1 : 多 구성 (주국과 1 이상의 종국과의 통신)  <p>(a) (b)</p> <p>출처 : [정보통신개론] 흐름제어, HDLC 프로토콜 : 네이버 블로그 (naver.com)</p>
<p>균형 구성 (Balanced Configuration)</p>	<ul style="list-style-type: none"> • 혼합국으로 구성 • 1 : 1 구성 (2개가 서로 대등한 쌍으로 통신)  <p>(c)</p> <p>출처 : [정보통신개론] 흐름제어, HDLC 프로토콜 : 네이버 블로그 (naver.com)</p>

③HDLC 전송모드

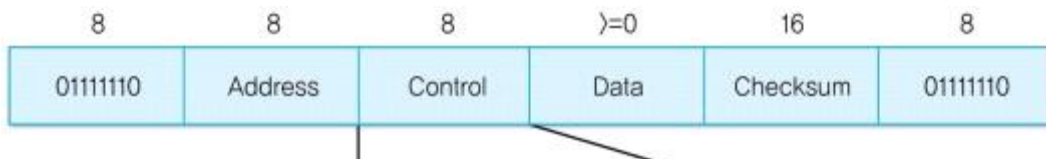
HDLC 프로토콜은 데이터 통신을 하기 전에 연결 설정과정을 거치는데 처음 연결 설정을 요청하는 경우는 3가지 전송모드 중 하나로 데이터 통신을 한다.

정규 응답 모드(NRM) (Normal Response Mode)	불균형 모드를 의미하기 때문에 호스트 하나는 주국으로 동작하고, 다른 하나는 종국으로 동작한다. 종국에서 데이터를 전송하려면 반드시 주국의 허락을 받아야 한다.
비동기 균형 모드(ABM) (Asynchronous Balanced Mode)	두 개의 호스트가 동일한 능력을 갖는 혼합국으로 동작하며, 양쪽에서 명령과 응답을 모두 전송할 수 있다.
비동기 응답 모드(ARM) (Asynchronous Response Mode)	불균형 모드이나 종국이 주국의 허락 없이도 데이터를 전송할 수 있는 권한을 갖는다.

④ HDLC 프레임 구조



사진 출처 : [HDLC \(ktword.co.kr\)](http://ktword.co.kr)



출처 : [프레임 구조 \(naver.com\)](http://naver.com)

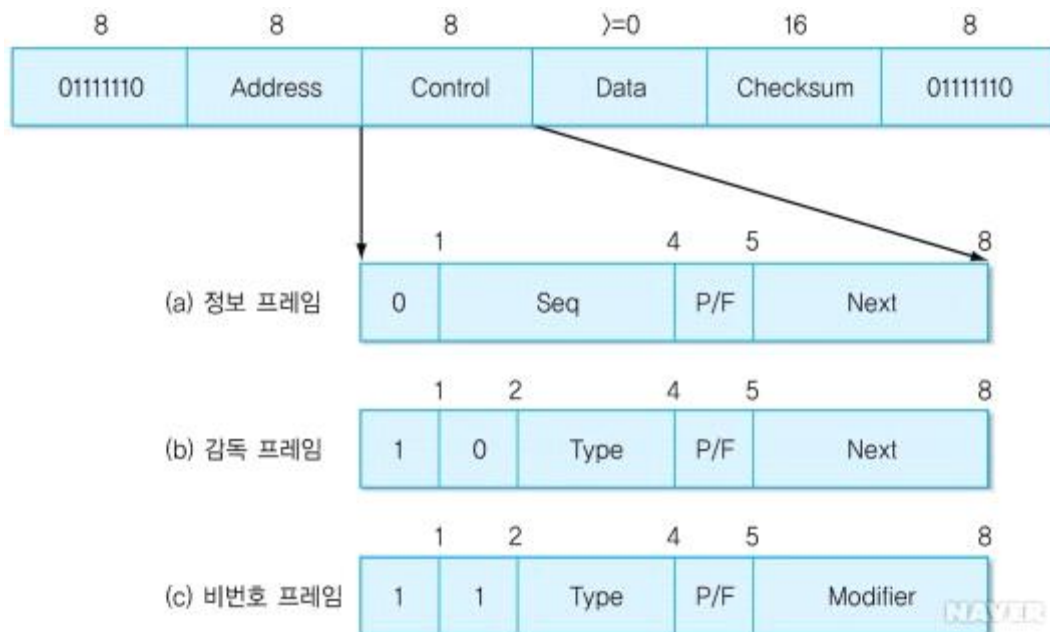
플래그 (Flag)	- 프레임의 시작과 끝을 나타내는 식별자(01111110)
주소 (Address)	- 일대다로 연결된 환경에서 특정 호스트를 구분하여 지칭하는 목적으로 사용 - 주국에서 정보 프레임을 전송할 때는 목적지인 종국 주소를 가짐 - 종국에서 전송할 때는 송신 호스트인 종국의 주소를 가짐 - 1:1 환경에서는 명령과 응답을 구분하는 용도로 사용
제어 필드 (Control Field)	- 프레임의 종류를 구분
Checksum (FCS; Frame Check Sequence)	-CRC 오류 검출 용으로 사용

⑤ HDLC 프레임의 종류

프레임의 종류에는 정보 프레임, 감독 프레임, 비번호 프레임이 있다.

프레임은 Control 필드(제어 필드) 값에 의해 구분된다.

④정보 프레임

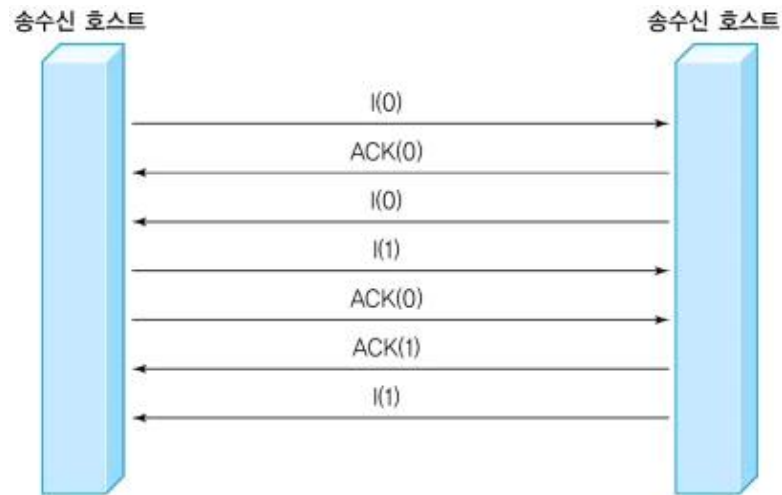


출처 : [프레임 구조 \(naver.com\)](https://blog.naver.com/dlqnf33)

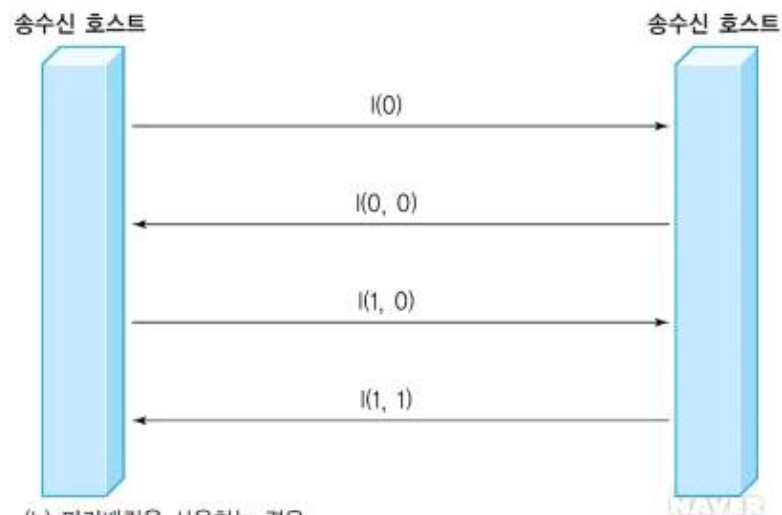
<p>정보 프레임 (Information Frame) (I-frame)</p>	<p>- 사용자 데이터(Payload) 및 일부 제어 정보의 전달에 쓰임</p> <p>① Seq : 프레임은 흐름제어를 할 때 슬라이딩 윈도우 프로토콜을 사용하는데 흐름제어시 정보프레임의 송신용 순서 번호로 사용</p> <p>② Next : 피기뱅킹을 이용한 응답기능으로 사용</p> <p>③ P/F : 주국컴퓨터가 다수의 종국 컴퓨터를 제어하기 위해 사용</p>
---	---

*피기배킹(Piggybacking) : 양방향 전송 기능을 갖는 채널에서는 고정된 송수신 호스트의 구분 없이, 양방향으로 동시에 정보 프레임과 응답 프레임을 교차하여 전송할 수 있는데
적당히 정보 프레임의 구조를 재정의하면 정보 프레임을 전송하면서 응답 까지 함께 보낼 수 있는 것

아래 그림에서 피기배킹을 사용하지 않는 경우는 ACK(응답 프레임) 따로 I(정보프레임) 따로 보내는데 밑에 피기배킹을 사용한 경우는 ACK와 I를 같이 보내는 것을 볼 수 있다.



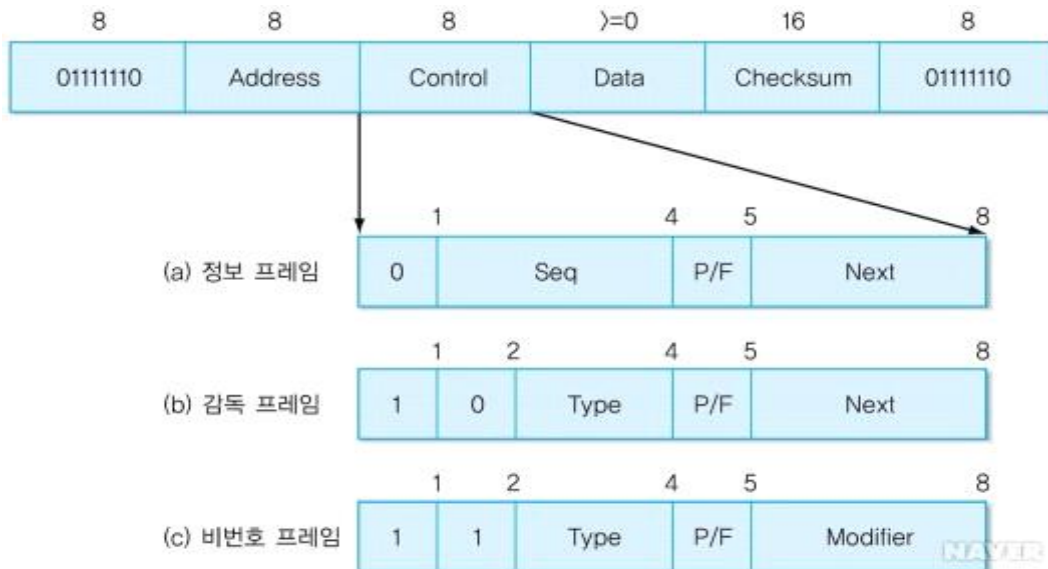
(a) 피기배킹을 사용하지 않는 경우



(b) 피기배킹을 사용하는 경우

출처 : [피기배킹 \(naver.com\)](https://blog.naver.com/dlqnf33)

㉔ 감독 프레임



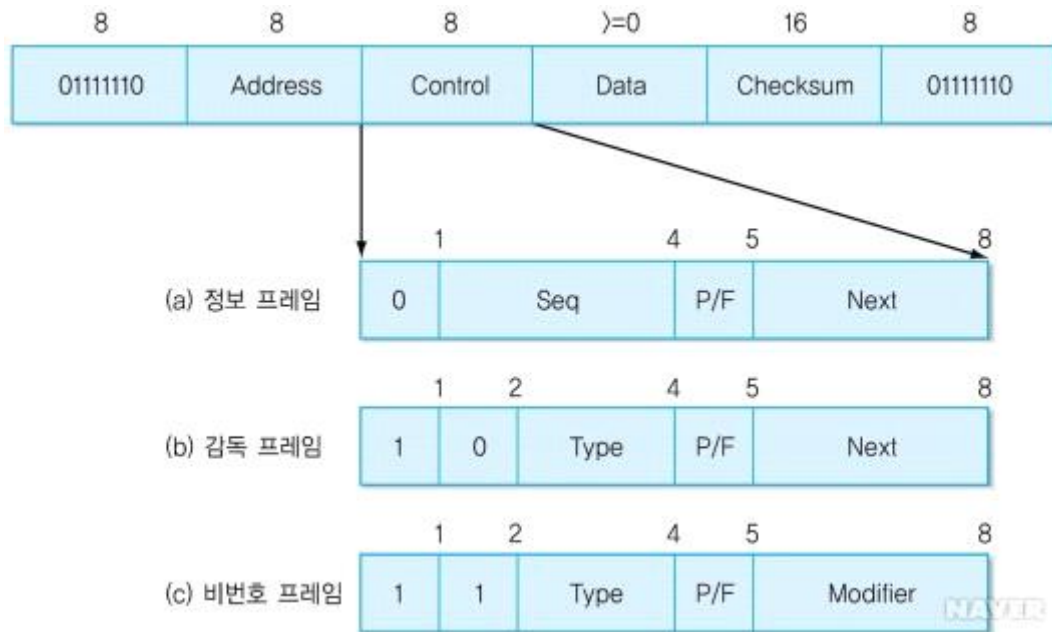
출처 : [프레임 구조 \(naver.com\)](https://blog.naver.com/dlqnf33)

(a) Type 0	1	0	00	P/F	Next	RR : Receive Ready
(b) Type 1	1	0	01	P/F	Next	REJ : Reject
(c) Type 2	1	0	10	P/F	Next	RNR : Receive Not Ready
(d) Type 3	1	0	11	P/F	Next	SREJ : Selective Reject

감독 프레임/출처 : [프레임 종류 \(naver.com\)](https://blog.naver.com/dlqnf33)

감독 프레임 (Supervisor Frame) (S-frame)	<ul style="list-style-type: none"> - 확인응답, 데이터 링크의 흐름제어 및 오류제어 용도로 쓰임 -Type 0 : RR로 정의된 긍정 응답 프레임, 다음에 수신을 기대하는 프레임 번호를 Next 필드에 표시 -Type 1 : REJ로 정의된 부정 응답 프레임으로 Next 필드에는 재전송되어야 하는 프레임의 번호를 표시 -Type 2 : RNR로 정의된 응답 프레임으로 흐름 제어 기능까지 제공하며 Next 필드에 표시한 순서 번호를 갖는 정보 프레임의 바로 앞 번호까지 제대로 수신되었다는 긍정 응답 기능과 함께, 송신 호스트에게 송신을 중지하도록 요구 -Type 3 : SREJ로 정의된 프레임으로, 선택적 재전송 방식에서 부정 응답 기능을 지원
---	--

© 비번호 프레임



출처 : [프레임 구조 \(naver.com\)](https://blog.naver.com/dlqnf33)

비번호 프레임 (Unnumbered Frame) (U-frame)	- 링크 자체의 관리 용으로 많이 쓰이지만, 비연결형 데이터 전송용으로도 쓰임
--	---

비번호 프레임은 Type과 Modifier 필드를 합해 총 5비트로 다음과 같은 프레임을 정의한다.
대충 리모컨으로 모드 조작해서 연결시키고 연결끊고 하는 거라고 생각하면 될 듯하다.

SABM(Set ABM)	- 비동기 균형(Asynchronous Balanced) 모드의 연결 설정을 요구
SNRM(Set NRM)	- 정규 응답(Normal Response) 모드의 연결 설정을 요구
SARM(Set ARM)	- 비동기 응답(Asynchronous Response) 모드의 연결 설정을 요구
DISC(DISConnect)	- 연결 설정 해제를 요구
RSET(ReSET)	- 비정상적인 프로토콜의 동작에 따른 리셋 기능을 수행
FRMR(FRaMe Reject)	- 비정상적인 프레임의 수신을 거부
UA(Unnumbered ACK)	- 비번호 프레임에 대한 응답 기능을 수행

6. 네트워크 계층 심화(서브네팅)

인터넷은 IP 프로토콜을 지원하는 전 세계의 모든 네트워크가 연결된 시스템을 의미하며, 라우터라는 중개 장비를 사용해 네트워크를 연결한다.

◆IP주소(Internet Protocol Address)

네트워크 계층의 기능을 수행하는 IP 프로토콜이 호스트를 구분하려고 사용하는 주소 체계로 임의의 호스트를 인터넷에 연결하려면 반드시 IP주소를 할당받아야 한다.

ex) 192.168.1.20

IP주소의 기억을 쉽게 하기 위해 도메인 이름이라 부르는 별칭을 사용한다.

▶도메인 네임(Domain Name)

숫자로 된 IP 주소를 사람이 이해하기 쉬운 문자 형태로 표현한 것

ex) assembly.go.kr

*도메인 이름은 사용자의 편의를 위해 만들어진 것일뿐, 실제 원하는 사이트에 접속하기 위해서는 IP주소가 필요하다.

*문자로 된 도메인 네임을 컴퓨터가 이해할 수 있는 IP주소로 변환하는 역할을 하는 시스템을 DNS(Domain Name System)라고 하며 이런 역할을 하는 서버를 DNS서버라 한다.

▶IPv6(Internet Protocol Version 6)

인터넷의 급격한 팽창으로 부여할 수 있는 주소도 곧 고갈될 것이라 예상돼, IP주소의 확장을 위한 IP가 등장하게 되었는데 그게 바로 IPv6이다.

현재의 IP 프로토콜은 IPv6과 구분하기 위해 IPv4로 표현하며 IPv6은 128비트의 긴 주소를 사용하여 IPv4의 주소 부족 문제를 해결할 수 있다.

♠IPv6의 구성

유니캐스트(Unicast)	1 대 1 통신
멀티캐스트(Multicast)	1 대 다 통신
애니캐스트(Anycast)	1대 1통신(가장 가까이 있는 수신자와)

▶IP 주소의 관리

일단 IP는 32비트의 이진수로 구성되는데 보통 8비트씩 4개의 필드로 나누어 십진수로 표현하고 각각 0~255까지 숫자를 나타낼 수 있다.



IP주소는 임의로 할당되는 것이 아니라 어떤 규칙에 따라 인접한 숫자를 그룹으로 묶어 관리되는데 그 방법이 바로 IP 주소의 네트워크 ID와 호스트 ID를 구분하여 사용하는 것이다.

IP를 네트워크 ID와 호스트 ID로 나눌 수 있는데 네트워크 ID가 같다는 의미는 같은 네트워크 상에 있다는 의미로, 같은 네트워크 상에 있다면 서로 자유롭게 통신이 가능하다. 예를 들어 192.168.1.20과 192.168.1.21은 같은 네트워크 상에 있다는 뜻이다.

*네트워크 ID: 네트워크를 구분하여 주는 ID

*호스트 ID : 해당 네트워크에 속한 사용자에게 부여하는 고유번호

네트워크 주소를 제외하고 나머지 호스트 ID로 해당 네트워크에서 몇 개의 IP가 만들어질 수 있는지 결정한다.

IP주소는 소속 네트워크의 규모와 특성에 따라 A, B, C, D, E 클래스로 나뉜다.

A클래스는 국가나 대형 통신망에서 사용되고, B클래스는 중대형 통신망, C클래스는 소규모 통신망에 사용되며 D클래스는 멀티캐스트용(1대 다 통신), E클래스는 실험적 주소로 사용된다.

Network ID and Host ID

Class A 0xxx xxxx.xxxx xxxx.xxxx xxxx.xxxx xxxx
Network ID Host ID

Class B 10xx xxxx.xxxx xxxx.xxxx xxxx.xxxx xxxx
Network ID Host ID

Class C 110x xxxx.xxxx xxxx.xxxx xxxx.xxxx xxxx
Network ID Host ID

Class D 1110 xxxx.xxxx xxxx.xxxx xxxx.xxxx xxxx
Network ID

Class E 1111 0xxx.xxxx xxxx.xxxx xxxx.xxxx xxxx
Network ID

Class A: 1.0.0.0 to 127.255.255.255

Class B: 128.0.0.0 to 191.255.255.255

Class C: 192.0.0.0 to 223.255.255.255

Class D: 224.0.0.0 to 239.255.255.255

Class E: 240.0.0.0 to 255.255.255.255

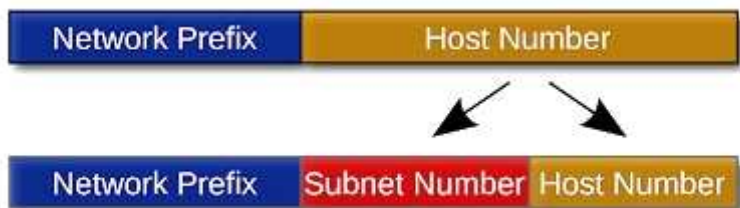
출처 : [What is "network ID" and "host ID" in IP Addresses? - GeeksforGeeks](https://www.geeksforgeeks.org/what-is-network-id-and-host-id-in-ip-addresses/)

앞서 말했듯이 IP주소에서 네트워크 ID가 같으면 같은 네트워크 상에 있어서 자유롭게 통신이 가능하고 호스트 ID의 경우의 수가 A클래스가 가장 많다. A클래스 네트워크 하나에 연결될 수 있는 호스트 컴퓨터 수가 16,777,214개로 현실적으로 이 같은 엄청난 수의 컴퓨터를 연결하여 운영할 수 있는 조직은 없으며 결과적으로 IP주소 공간을 낭비하게 되고 중대 규모의 조직에서는 B 클래스를 쓰는데 B 클래스의 IP는 오히려 부족 문제가 발생하게 되었다. 따라서 이런 IP주소 부족 문제를 해결하기 위해 서브넷이라는 개념이 등장하게 되었다.

▶서브넷/서브넷 마스크/서브네팅

- 서브넷 : IP주소에서 네트워크 영역을 부분적으로 나눈 부분 네트워크
- 서브넷 마스크 : 서브넷을 만들기 위해 IP 주소체계의 Network ID와 Host ID를 분리하는 역할을 하는 비트
- 서브네팅: 할당된 네트워크 주소를 다시 여러 개의 작은 네트워크로 나누어 사용하는 것
- 기본 서브넷 마스크 : 별개의 서브넷 마스크를 생성하지 않아도 기본적으로 적용되어 있는 서브넷 마스크

(A클래스 : 255.0.0.0, B클래스 : 255.255.0.0, C클래스 : 255.255.255.0)



출처 : <https://just-my-blog.tistory.com/22>

서브네팅을 쉽게 예를 들면 먼저 성별을 여자(네트워크 ID)와 남자(Host ID)로 분리하고 남자를 여자로 성전환한 남자와 성전환하지 않은 남자로 분리한 다음 여자와 여자로 성전환한 남자를 묶어서 생각한다고 생각하면 된다.

서브네팡팅에 대해 자세히 알아보기 전에 간단하게 10진수를 2진수로 바꾸는 방법을 알아보자.

8개의 비트를 앞에서부터 2의 7제곱, 2의 6제곱, 2의 5제곱, 2의 4제곱, 2의 3제곱, 2의 2제곱, 2의 1제곱, 2의 0제곱으로 나타낼 수 있고 139.128.19.132를 이진수로 바꾸면 10001011.01111111.00010011.10000100으로 바꿀 수 있다.

10001011 ₂							
2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰
128	64	32	16	8	4	2	1
139		128		19		132	
128+8+3		64+32+16+8+4+2+1		16+2+1		128+4	
10001011		01111111		00010011		10000100	

아래와 같은 방법으로도 이진수 변환이 가능하다.

10진수로 주어진 수를 2로 나눠서 나머지 값을 아래수부터 배열한 것

ex) (35)₁₀ → (100011)₂:

2		35	
2		17	... 1
2		8	... 1
2		4	... 0
2		2	... 0
1		1	... 0

구체적으로 서브네팡팅에 대해 알아보자.

192.168.32.0 /24라고 적혀있다면 192.168.32.0은 IP이고 /24는 서브네팡 마스크의 bit수를 의미한다.

/24는 1이 24개 있다는 의미이므로 11111111.11111111.11111111.0이 되고 이것을 십진수로 바꾸면 255.255.255.0이 된다.

/24는 서브네팡 마스크의 bit수를 의미하며
x 1개가 1비트이므로 24개비트라고 하면
1을 24개 채워주면 된다

xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
11111111	11111111	11111111	00000000
255	255	255	0

/24의 경우 할당 가능 네트워크는 1개이고 할당 가능한 호스트 수는 $2^8 - 2$ 로 254개이다.

할당 가능 네트워크는 앞서 말했듯이 네트워크 ID가 같으면 자유롭게 통신이 가능한데 그런 네트워크가 1개밖에 없다는 뜻이고 가능한 호스트 수는 호스트ID가 0과 1, 2가지 종류로 8자리 있으니 경우의 수가 2의 8제곱이 되는데 이때 2가지의 경우는 제외하고 계산을 해서 254이다.

제외되는 2가지 경우는 네트워크 주소와 브로드캐스트 주소인데 이 2개는 호스트를 할당할 수 없다.

*네트워크주소 : 일반적으로 하나의 네트워크를 통칭하기 위해 사용하는 주소로 일반적인 네트워크 주소의 표현은 해당 네트워크의 맨 첫번째 IP 주소를 이용해 표시.

*브로드캐스트 주소 : 특정 네트워크에 속하는 모든 호스트들이 듣게되는 주소로 특정 네트워크의 맨 마지막 주소

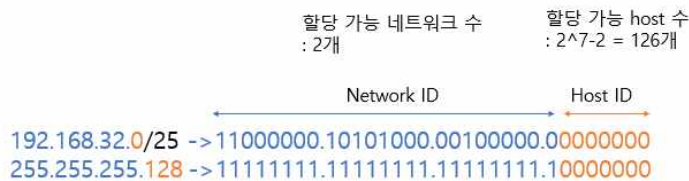
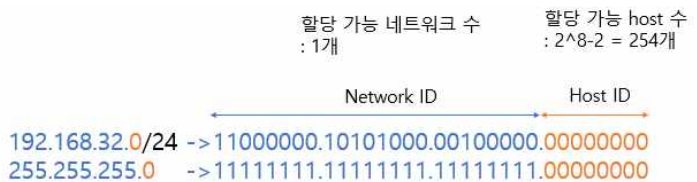
이때 192.168.32.0 /24를 192.168.32.0/25로 바꿔보자. 이때 서브넷 마스크에서는 무조건 왼쪽부터 차례대로 채워져야 한다. 예를 들면 /26이면 호스트ID가 00110000 이렇게 안되고 무조건 11000000

아래 그림과 같이 서브넷마스크 비트수가 25가 되면 원래 호스트ID 8자리 중 맨 앞자리가 네트워크 ID로 바뀌고 남은 7자리가 호스트ID가 된다.

따라서 할당 가능 네트워크는 2개가 되고 할당 가능 호스트 수는 $2^7 - 2$ 로 126개가 된다.

할당 가능 네트워크가 2개가 된다는 의미는 서브네팅하기 전에는 하나의 네트워크였는데 서브네팅 후에는 다른 네트워크가 되었다는 의미이다. (ex) 원래는 A부서만 쓰는 네트워크인데 서브네팅해서 A부서와 B부서가 쓰는 네트워크로 바뀜)

할당 가능 네트워크가 2개가 되는 이유는 0xxxxxxx, 1xxxxxxx 2가지 경우가 있고 이 2가지가 다른 네트워크가 됐기 때문이다.



출처 : <https://code-lab1.tistory.com/34>

좀 더 잘 이해를 하기 위해 네트워크 주소를 구하는 문제를 함께 풀어보자.

ex) IP주소가 139.127.19.132이고 서브넷 마스크 255.255.255.192일 때 아래의 답을 작성하시오
(2022년 2회 실기 기출문제)

- (1) 괄호 안에 들어갈 네트워크 주소 (139.127.19. ())
(2) 해당 네트워크 주소와 브로드캐스트 주소를 제외한 호스트 개수

(1) : 128

(2) : 62

네트워크 주소를 구하는 방법은 IP주소와 서브넷 마스크를 각각 이진수로 변환 후 and연산을 해서 덧셈을 해주면 된다.
and 연산은 둘 다 1인 것만 반환한다.

10001011.01111111.00010011.10000000이 나와서 139.128.19.128이 된다.

서브넷 마스크가 11000000이므로 바뀔 수 있는 호스트 경우는 뒤에 0의 개수이다. 0이 6개이므로 $2^6 - 2 = 62$

128	64	32	16	8	4	2	1
139	128	19	132				
128+8+3	64+32+16+8+4+2+1	16+2+1	128+4				
10001011	01111111	00010011	10000100				
255	255	255	192				
128+64+32+16+8+4+2+1	128+64+32+16+8+4+2+1	128+64+32+16+8+4+2+1	128+64				
11111111	11111111	11111111	11000000				
이진수로 변환한 것을 덧셈(1있는 자리만 반환)하면 네트워크 ID가 나옴							
10001011	01111111	00010011	10000100				
11111111	11111111	11111111	11000000				
10001011	01111111	00010011	10000000				
139	128	19	128				

식을 표로 정리하면 다음과 같다.

서브넷 마스크 bit 수	서브넷 마스크	네트워크 수(서브넷 개수)	호스트 수
/25	255.255.255.128	2	128
/26	255.255.255.192	4	64
/27	255.255.255.224	8	32
/28	255.255.255.240	16	16
/29	255.255.255.248	32	8
/30	255.255.255.252	64	4
/31	255.255.255.254	128	2
/32	255.255.255.255	256	1

ex) 첫번째 네트워크 주소가 192.168.1.0/26일때 FLSM 3개로 분할했을때 두번째 네트워크 브로드캐스트 IP를 10진수로 변환한 값을 작성하시오. (2022년 3회 정보처리기사 실기)

답 : 192.168.1.127

기본 서브넷 마스크가 /24인데 문제에서는 /26인 것을 보니 11111111.11111111.11111111.11000000인 것을 알 수 있다. 앞 문제에서 /25일 경우에는 0xxxxxxx, 1xxxxxxx 이렇게 네트워크 수(서브넷 수)가 2가지인 것을 알 수 있었는데 /26일 경우에는 00xxxxxx, 01xxxxxx, 10xxxxxx, 11xxxxxx 이렇게 네트워크 수(서브넷 수)가 4가지인 것을 알 수 있다. 문제에서는 동일한 크기로 3개로 분할 했을 때 두 번째 네트워크의 브로드캐스트(마지막 IP)를 구하는 문제이므로 01111111인 것을 알 수 있다. 01111111은 127이므로 192.168.1.127이다. (맨 마지막 IP가 111111인 이유는 십진수가 1->2->3 되는 것처럼 이진수도 00->01->10->11 이런 식으로 커지기 때문에 111111이 제일 큰 수이기 때문이다, 같은 방식으로 네트워크 주소는 000000이다.)

*FLSM(Fixed Length Subnet Mask) : 한 대역을 동일한 크기로 나누는 것

ex) 192.168.1.0 /24 네트워크를 각 서브넷당 55개의 Host를 할당할 수 있도록 서브네팅 할 때 서브넷 마스크와 서브넷의 개수를 구하시오.

서브넷 마스크 : 255.255.255.192

서브넷의 개수(네트워크 개수) : 4개

서브네팅을 할 때

서브넷 마스크 비트수가 /25라고 생각하면 바뀔 수 있는 호스트 경우는 1xxxxxxx로 7개이므로 $2^7 - 2 = 126$

서브넷 마스크 비트수가 /26라고 생각하면 바뀔 수 있는 호스트 경우는 11xxxxxx로 6개이므로 $2^6 - 2 = 62$

서브넷 마스크 비트수가 /27라고 생각하면 바뀔 수 있는 호스트 경우는 111xxxxx로 5개이므로 $2^5 - 2 = 30$

서브넷당 55개의 Host를 할당하기 가장 적절한 경우는 비트수가 /26인 경우이므로 서브넷마스크는 255.255.255.192이고 서브넷의 경우가 00, 01, 10, 11 이렇게 4가지가 있으므로 4개이다.

IV. 웹 파트

◆WWW(World Wide Web)

인터넷에 연결된 컴퓨터를 통해 사람들이 정보를 공유할 수 있는 전 세계적인 정보 공간으로 간단히 웹이라고 부르는 경우가 많다.

◆웹의 주요 기술

- ▶HTTP(Hypertext Transfer Protocol) : 웹 브라우저와 웹 서버간의 통신에 사용되는 프로토콜
- ▶MIME(Multipurpose Internet Mail Extensions) : 이메일에서 텍스트, 이미지, 비디오, 오디오 등 다양한 유형의 파일을 보낼 수 있게 해주는 표준화된 방식
- ▶URL(Uniform Resource Locator) : 인터넷 상에서 특정 웹 페이지의 위치를 나타내는 데 사용되는 문자열
- ▶HTML(Hypertext Markup Language) : 웹 페이지를 작성하는데 사용되는 마크업 언어

◆마크업 언어(Markup Language)

태그 등을 이용하여 문서나 데이터의 구조를 명시하는 언어의 한 가지

문서가 화면에 표시되는 형식을 나타나거나 데이터의 논리적인 구조를 명시하기 위한 규칙들을 정의한 언어의 일종
마크업 언어는 아날로그적인 기록매체와 유사한 역할을 한다고 볼 수 있는데 한번 작성해놓으면 언제나 동일한 모습을 보여준다. 책을 하나의 프로그램으로 생각한다면 사람이 책에 볼펜으로 밑줄을 긋거나 필기를 하는 것도 마크업의 일종이다. 완벽하진 않지만 비유하자면 마크업언어가 스케치(디자인)이라면 마크업언어+프로그래밍언어는 애니메이션(움직임)이라고 볼 수 있다.

◆웹 서비스

네트워크 상에서 서로 다른 종류의 컴퓨터들 간에 상호작용을 하기 위한 소프트웨어 시스템이다.

웹 서비스는 서비스 지향적 분산 컴퓨팅 기술의 일종으로 SOAP, WSDL, UDDI등의 주요 표준 기술로 이루어진다.
웹 서비스의 모든 메시지에는 주로 XML이 사용된다.

●SOAP(Simple Object Access Protocol)

XML과 HTTP등을 기본으로 하여 다른 컴퓨터에 있는 데이터나 서비스를 호출하기 위한 통신규약(Protocol)
HTTP, HTTPS, SMTP등을 사용하여 XML 기반의 메시지를 컴퓨터 네트워크상에서 교환하는 형태의 프로토콜로써 웹 서비스의 기본적인 메시지 수단

●UDDI(Universal Description Discovery and Integration)

개방형 표준과 비독점적 기술을 기반으로 개발된 전역 비즈니스 레지스터리
이 레지스터리를 이용하여 다양한 웹서비스를 사용자는 쉽게 검색하여 사용할 수 있다.

●WSDL(Web Services Description Language)

웹서비스에서 제공하는 기능들을 외부에서 이용할 수 있도록 그 사용 방법을 알려주는 인터페이스 언어로 XML 기반으로 작성된다.

◆XML : 특수한 목적을 갖는 마크업 언어를 만드는 데 사용되는 다목적 마크업 언어

웹 브라우저간 HTML문법이 호환되지 않는 문제와 SGML의 복잡함을 해결하기 위해 개발되었다.

◆AJAX(Asynchronous Javascript and XML) : 기존의 HTML 관련 기술과는 달리 사용자 클릭 후 페이지 전체를 새로 고치지 않고 필요한 부분만 바로 고치는 형태로 빠른 응답을 제공하는 기술

◆JSON(JavaScript Object Notation) : 자바스크립트에서의 객체 표기 방법을 기반으로 한 데이터 형식으로 각 데이터(토큰)을 속성과 값의 쌍으로 표현한다.

V. 보안 파트

◆보안의 3대 요소

기밀성 (Confidentiality)	시스템 내의 정보와 자원은 인가된 사용자에게만 접근이 허용
무결성 (Integrity)	시스템 내의 정보는 오직 인가된 사용자만 수정할 수 있음
가용성 (Availability)	인가받은 사용자는 시스템 내의 정보와 자원을 언제라도 사용할 수 있음

◆AAA

Authentication (인증)	자신의 신원을 시스템에 증명하는 과정
Authorization (권한부여)	인증이 수행된 객체에 대해 이미 설정된 권한 구성에서 객체가 요구하는 접근 권한 부여의 적절성을 확인하는 과정
Accounting (계정관리)	접근에 성공한 객체에 대한 로그를 남기는 활동

◆네트워크 보안-서비스 거부 공격(DOS; Denial of Service)

1) 취약점 공격형 : 공격 대상이 특정 형태의 오류가 있는 네트워크 패킷의 처리 로직에 문제 있을 때 그것을 이용하여 공격 대상의 오동작을 유발시키는 공격

Boink, Bonk, TearDrop 공격	UDP, TCP 패킷의 시퀀스 넘버를 조작하여 공격 시스템에 과부하를 일으킨다.
LAND Attack (Local Area Network Denial Attack)	패킷을 전송할 때 송신 IP 주소와 수신 IP 주소를 모두 공격 대상의 IP 주소로 하여 공격 대상에게 전송하는 것으로, 자신에 대해 무한히 응답하게 한다.

2) 자원 고갈 공격형 : 네트워크 대역폭이나 시스템의 CPU 세션 등의 자원을 소모시키는 공격

Ping of Death	패킷의 크기를 인터넷 프로토콜 허용 범위 이상으로 전송하여 공격 대상의 네트워크를 마비시킨다.
SYN Flooding	SYN 패킷만 보내어 각 서버의 동시 가용 사용자 수를 점유하여 다른 사용자가 서버를 사용할 수 없게 한다.
Smurf(Smurfing)	IP나 ICMP의 특성을 악용하여 엄청난 양의 데이터를 한 사이트에 집중적으로 보냄으로써 네트워크를 불능 상태로 만든다.
Mail Bomb	스팸을 이용한 대량 메일을 전송한다.
UDP flooding	대량의 UDP 패킷을 만들어 임의의 포트 번호로 전송하여 응답 메시지(ICMP Destination Unreachable)를 생성하게 하여 지속해서 자원을 고갈시키는 공격

3) 분산 서비스 거부(DDoS; Distributed Denial of Service) 공격 : 공격자가 한 지점에서 서비스 거부 공격을 수행하는 형태를 넘어 광범위한 네트워크를 이용하여 다수의 공격 지점에서 동시에 한 곳을 공격하도록 하는 형태의 분산 서비스 거부 공격

◆네트워크 보안-네트워크 침해 공격

APT(지능형 지속 위협) (Advanced Persistent Threats)	다양한 IT 기술과 방식들을 이용해 조직적으로 특정 기업이나 조직 네트워크에 침투해 활동 거점을 마련한 뒤 때를 기다리면서 보안을 무력화시키고 정보를 수집한 다음 외부로 빼돌리는 행태의 공격
무작위 대입 공격 (Brute Force Attack)	암호화된 문서의 암호키를 찾아내기 위해 적용 가능한 모든 값을 대입하여 공격하는 방식
큐싱(Qshing)	QR코드(Quick Response Code)를 통해 악성 앱의 다운로드를 유도하거나 악성 프로그램을 설치하도록 하는 금융사기 기법
SQL 삽입(Injection) 공격	전문 스캐너 프로그램 혹은 봇넷 등을 이용해 웹사이트를 무차별적으로 공격하는 과정에서 취약한 사이트가 발견되면 데이터베이스 등의 데이터를 조작하는 일련의 공격방식
크로스 사이트 스크립팅 (XSS, Cross Site Scripting)	웹페이지의 내용을 사용자 브라우저에 표현하기 위해 사용되는 스크립트의 취약점을 악용한 해킹 기법 사용자가 특정 링크를 클릭하면 악성 스크립트가 실행되어 페이지가 깨지거나, 사용자의 컴퓨터에 있는 개인정보, 내부 자료 등이 해커에게 전달
스니핑(Sniffing)	네트워크 상에서 전송되는 데이터를 가로채서 엿보는 것
스푸핑(Spoofing)	자신이 누군가 다른 사람이나 기계인 것처럼 위장하여 정보를 송수신하는 것으로 IP주소를 변조한 후 시스템에 접근함으로써 나중에 IP주소에 대한 추적을 피하는 해킹 기법의 일종
세션 하이재킹 (Session Hijacking)	인증된 사용자의 세션을 탈취하여 사용자의 권한을 도용하는 공격 기법
워터링 홀 공격(Watering Hole Attack)	타깃 삼은 특정 집단이 주로 방문하는 웹 사이트를 감염시키고, 피해 대상이 그 웹사이트를 방문할 때까지 기다리는 공격 방법
이블 트윈(Evil Twin)	와이파이(WiFi) 무선 네트워크에서 공격자가 가짜 AP(Access Point)를 구축하고 강한 신호를 보내어 사용자가 가짜 AP에 접속하게 함으로써 사용자 정보를 중간에서 가로채는 기법

◆네트워크 보안-사회 공학

인간 상호 작용의 깊은 신뢰를 바탕으로 사람들을 속여 정상 보안 절차를 깨뜨리고 정보를 얻는 비기술적인 침입 수단

스미싱 (Smishing)	문자 메시지(SMS)를 이용해 사용자의 개인 신용 정보를 빼내는 수법
스피어 피싱 (Spear Phishing)	특정 대상을 선정한 후 그 대상에게 일반적인 이메일로 위장한 메일을 지속적으로 발송하여, 발송 메일의 본문 링크나 첨부된 파일을 클릭하도록 유도해 사용자의 개인정보를 탈취하는 수법
파밍 (Pharming)	악성코드에 감염된 PC를 조작해, 이용자가 인터넷 '즐거찾기' 또는 포털사이트를 통해 금융회사 홈페이지에 접속하여도 피싱(가짜)사이트로 유도되어 금융정보를 탈취하여 유출된 정보로 예금 인출하는 방식
웨일링 (Whaling)	스피어 피싱의 한 종류로 CEO, CFO 등 기업 내 고위 경영진, 정치인, 연예인 등을 타깃으로 한 공격

*포렌식 : 컴퓨터 관련 조사/수사를 지원하며, 디지털 데이터가 법적 효력을 갖도록 하는 과학적·논리적 절차와 방법을 연구하는 학문

◆네트워크 보안-기타 용어

좀비(zombie) PC	악성코드에 감염되어 다른 프로그램이나 컴퓨터를 조종하도록 만들어진 컴퓨터로, C&C(Command & Control) 서버의 제어를 받아 주로 DDoS공격 등에 이용됨
C&C 서버	해커가 원격지에서 감염된 좀비 PC에 명령을 내리고 악성코드를 제어하기 위한 용도로 사용하는 서버
봇넷(Botnet)	악성 프로그램에 감염되어 악의적인 의도로 사용될 수 있는 다수의 컴퓨터들이 네트워크로 연결된 형태
웜(Worm)	네트워크를 통해 연속적으로 자신을 복제하여 시스템의 부하를 높임으로써 결국 시스템을 다운시키는 바이러스의 일종으로, 분산 서비스 거부 공격, 버퍼 오버플로 공격, 슬래머 등이 웜 공격의 한 형태
제로 데이 공격 (Zero Day Attack)	보안 취약점이 발견되었을 때 발견된 취약점의 존재 자체가 널리 공표되기도 전에 해당 취약점을 통하여 이루어지는 보안 공격으로, 공격의 신속성을 의미
키로거 공격 (Key Logger Attack)	컴퓨터 사용자의 키보드 움직임을 탐지해 ID, 패스워드, 계좌번호, 카드번호 등과 같은 개인의 중요한 정보를 몰래 빼가는 해킹 공격
랜섬웨어 (Ransomware)	인터넷 사용자의 컴퓨터에 잠입해 내부 문서나 파일 등을 암호화해 사용자가 열지 못하게 하는 프로그램으로, 암호 해독용 프로그램의 전달을 조건으로 사용자에게 돈을 요구하기도 함
백도어(Back Door, Trap Door)	시스템 설계자가 서비스 기술자나 유지 보수 프로그래머의 액세스 편의를 위해 시스템 보안을 제거하여 만들어놓은 비밀 통로로, 컴퓨터 범죄에 악용되기도 함
트로이 목마 (Trojan Horse)	정상적인 기능을 하는 프로그램으로 위장하여 프로그램 내에 숨어 있다가 해당 프로그램이 동작할 때 활성화되어 부작용을 일으키는 것으로, 자기 복제 능력은 없음
크린덴셜 스테핑 (Credential stuffing)	공격자가 이미 확보한 크린덴셜(로그인 정보 등 개인 신상과 관련해 암호화한 정보)을 다른 계정들에 마구 대입하는(stuffing) 방식으로 이용자 정보를 침해하는 행위
아일랜드 호핑 (Island Hopping)	공격자들이 한 조직을 공격해 해당 조직의 고객이나 파트너사, 그 외 협력사 등의 네트워크들을 차례차례 공략하는 것
공급망 공격 (Supply Chain Attack)	소프트웨어 공급망에 침투하여 악성코드를 배포하는 공격으로, SW 빌드 및 배포 과정에 악성코드를 삽입하여 선의의 소프트웨어를 통해 이용자들을 공격
Drive-by Download	사용자가 의도치 않은 상황에서 사용자도 모르게 악성코드가 다운로드 되는 침해유형

메시지를 전달할 때 발신자와 수신자만 이해할 수 있게 해주는 암호는 초기에는 군사와 정치적인 목적으로만 사용되다가 최근에는 일상생활에도 널리 쓰이고 있다. 암호의 방식이 모두 같은 것은 아니며 근대에 접어들면서 암호 알고리즘이 표준화되고 있지만 아직도 다양한 암호화 알고리즘이 만들어지고 사용된 후 사라진다. 평문을 암호문으로 바꾸는 것을 암호화(Encryption)이라고 하고 암호문을 평문으로 바꾸는 것을 복호화(Decryption)이라고 한다. 이제 암호 알고리즘에 대해 알아보자.

◆암호 알고리즘

패스워드, 주민번호, 은행 계좌와 같은 중요 정보를 보호하기 위해 평문을 암호화된 문자로 만드는 절차 또는 방법

▶ 암호 방식 분류



1) 개인키 암호화(Private Key Encryption) 기법 : 동일한 키로 데이터를 암호화하고 복호화하는 암호화 기법으로 대칭 암호 기법 또는 단일키 암호화 기법이라고 한다.

▶개인키 암호화 기법의 종류

스트림 암호화 방식	평문과 동일한 길이의 스트림을 생성하여 비트 단위로 암호화 하는 방식(LFSR, RC4)
블록 암호화 방식	한 번에 하나의 데이터 블록을 암호화 하는 방식(DES, SEED, AES, ARIA)

2) 공개키 암호화(Public Key Encryption) 기법 : 데이터를 암호화할 때 사용하는 공개키(Public Key)는 사용자에게 공개하고 복호화할 때의 비밀키(Secret Key)는 관리자가 비밀리에 관리하는 암호화 기법 비대칭 암호 기법이라고도 하며 대표적으로는 RSA(Rivest Shamir Adleman)기법이 있다.

3) 양방향 알고리즘의 종류

SEED	전자상거래, 금융, 무선통신 등에서 전송되는 개인정보와 같은 중요한 정보를 보호하기 위해, 1999년 2월 한국인터넷진흥원(KISA)과 국내 암호전문가들이 순수 국내기술로 개발한 128비트 블록의 암호 알고리즘
ARIA (Academy, Research Institute, Agency)	전자정부 구현 등으로 다양한 환경에 적합한 암호화 알고리즘이 필요함에 따라 국가보안기술연구소(NSRI) 주도로 학계, 국가정보원 등의 암호전문가들이 힘을 모아 2004년에 개발한 국가 블록 암호화 알고리즘
DES (Data Encryption Standard)	1975년 미국 NBS(현 NIST)에서 발표한 개인 암호화 알고리즘 DES를 3번 적용하여 보안을 더욱 강화한 3DES(Triple DES)도 있음
AES (Advanced Encryption Standard)	2001년 미국 표준 기술 연구소(NIST)에서 발표한 개인키 암호화 알고리즘 DES의 암호화 강도가 점점 약해지면서 새롭게 개발된 알고리즘
RSA (Rivest Shamir Adleman)	1978년 MIT에 라이베스트(Rivest), 샤미르(Shamir), 애들먼(Adelman)에 의해 제안된 공개키 암호화 알고리즘 큰 숫자를 소인수분해 하기 어렵다는 것에 기반하여 만들어짐
IDEA (International Data Encryption Algorithm)	국제 데이터 암호화 알고리즘은 스위스에서 1990년 Xuejia Lai, James Messey가 만든 PES를 개량하여, 1991년에 제작된 블록 암호 알고리즘
Skipjack	미 국가안보국(NSA, National Security Agency)에서 개발한 Clipper 칩에 내장된 블록 알고리즘

4) 해시(Hash) : 임의의 길이의 입력 데이터나 메시지를 고정된 길이의 값이나 키로 변환하는 것
해시 알고리즘을 해시 함수라고 부르며, 해시 함수로 변환된 값이나 키를 해시값 또는 해시키라고 부른다.
데이터의 암호화, 무결성 검증을 위해 사용될 뿐만 아니라 정보보호의 다양한 분야에서 활용된다.

▶해시 함수의 종류

SHA시리즈	1993년 미국 국가안보국(NSA)이 설계, 미국 국립표준 기술 연구소(NIST)에 의해 발표됨, 초기 개발된 SHA-0 이후 SHA-1이 발표되었고, 다시 SHA-2라고 불리는 SHA-224, SHA-256, SHA-384, SHA-512가 발표됨
MD5(Message Digest algorithm 5)	1991년 R.Rivest가 MD4를 대체하기 위해 고안한 암호화 해시 함수, 블록 크기는 512비트이며, 키 길이는 128비트
N-NASH	1989년 일본의 전신전화주식회사(NTT)에서 발표한 암호화 해시 함수, 블록 크기와 키 길이가 모두 128비트
SNEFRU	1990년 R.C.Merkle가 발표한 해시 함수, 32비트의 프로세서에서 구현을 용이하게 할 목적으로 개발됨

◆보안 솔루션

접근 통제, 침입 차단 및 탐지 등을 수행하여 외부로부터의 불법적인 침입을 막는 기술 및 시스템

SSO (Single Sign On)	1회 사용자 인증으로 다수의 애플리케이션 및 웹사이트에 대한 사용자 로그인을 허용하는 인증 솔루션
방화벽 (Firewall)	내부의 네트워크와 인터넷 간의 전송되는 정보를 선별하여 수용, 거부, 수정하는 기능을 가진 침입 차단 시스템
침입 탐지 시스템(IDS) (Intrusion Detection System)	컴퓨터 시스템의 비정상적인 사용, 오용, 남용 등을 실시간으로 탐지하는 시스템
침입 방지 시스템(IPS) (Intrusion Prevention System)	비정상적인 트래픽을 능동적으로 차단하고 격리하는 등의 방어 조치를 취하는 보안 솔루션
데이터 유출 방지(DLP) (Data Leakage/Loss Prevention)	내부 정보의 외부 유출을 방지하는 보안 솔루션
VPN(가상 사설 통신망) (Virtual Private Network)	공중 네트워크와 암호화 기술을 이용하여 사용자가 마치 자신의 전용 회선을 사용하는 것처럼 해주는 보안 솔루션
NAC (Network Access Control)	네트워크에 접속하는 내부 PC의 MAC 주소를 IP 관리 시스템에 등록한 후 일관된 보안 관리 기능을 제공하는 보안 솔루션
ESM (Enterprise Security Management)	다양한 장비에서 발생하는 로그 및 보안 이벤트를 통합하여 관리하는 보안 솔루션
SIEM (Security Information and Event Management)	조직에서 비즈니스에 문제를 일으키기 전에 보안 위협을 탐지, 분석 및 대응하도록 도와주는 솔루션
SOAR (Security Orchestration Automation and Response)	보안오케스트레이션이라고 불리며 SIEM을 포함, 보안 관제의 단점을 극복했으며 여러 보안장비에서 데이터를 수집하여 분석하는 단순 업무를 자동화시켜주는 솔루션

VI. 운영체제 파트

◆운영체제(OS: Operation System)

컴퓨터 시스템의 자원들을 효율적으로 관리하며, 사용자가 컴퓨터를 편리하고 효과적으로 사용할 수 있도록 환경을 제공하는 여러 프로그램의 모임

ex) Windows, UNIX, LINUX 등

*Windows : 1990년대 마이크로소프트가 개발한 운영체제

*UNIX : 1960년대 AT&T 벨(Bell) 연구소, MIT, General Electric이 공동 개발한 운영체제

커널(Kernel)	하드웨어를 보호하고, 프로그램과 하드웨어 간의 인터페이스 역할을 담당 UNIX의 핵심적인 부분
셸(Shell)	사용자의 명령어를 인식하여 프로그램을 호출하고 명령을 수행하는 명령어 해석기 시스템과 사용자 간의 인터페이스를 담당

*LINUX : 1991년 리누스 토발즈(Linus Torvalds)가 UNIX를 기반으로 개발한 운영체제

*MacOS : 1980년대 애플이 UNIX를 기반으로 개발한 운영체제

*Android : 구글(Google)에서 개발한 리눅스 커널 기반의 개방형 모바일 운영체제

*iOS : 애플에서 개발한 유닉스 기반의 모바일 운영체제

◆운영체제의 목적 4가지

처리 능력 향상, 사용 가능성 향상, 신뢰도 향상, 반환 시간 단축

처리 능력(Throughput)	일정 시간 내에 시스템이 처리하는 일의 양
반환 시간(Turn Around Time)	시스템에 작업을 의뢰한 시간부터 완료될 때까지 걸린 시간
사용 가능성(Availability)	시스템을 사용할 필요가 있을 때 즉시 사용 가능한 정도
신뢰도(Reliability)	시스템이 주어진 문제를 정확하게 해결하는 정도

◆운영체제의 기능

- 1) 사용자가 복잡한 하드웨어를 몰라도 쉽게 이용할 수 있도록 가상적인 컴퓨터 환경을 제공한다.
- 2) 컴퓨터 자원을 관리한다.

◆기억장치의 종류 4가지

*기억장치 : 데이터, 프로그램, 연산의 중간 결과 등을 일시적 또는 영구적으로 저장하는 장치

●주기억장치 : 컴퓨터가 실행될 때 사용되는 기억장치로 전원이 끊어져도 내용이 보존되는 롬(ROM)과 전원이 꺼지면 모든 내용이 지워지는 램(RAM)으로 나눌 수 있다.

●보조기억장치 : 컴퓨터의 중앙처리장치가 아닌 외부에서 프로그램이나 데이터를 보관하기 위한 기억장치로 대표적으로 HDD(하드디스크)가 있다.

●캐시 메모리 : 속도가 빠른 장치와 느린 장치에서 속도 차이에 따른 병목 현상을 줄이기 위한 메모리

●레지스터 : 중앙처리장치 내부에 존재하는 기억장치

◆운영체제의 기능-주기억장치 관리

보조기억장치의 프로그램이나 데이터를 주기억장치에 적재시키는 시기, 적재 위치 등을 지정하여 한정된 주기억장치의 공간을 효율적으로 사용하기 위해 관리를 한다.

ex) 반입(Fetch) 전략, 배치(Placement) 전략, 교체(Replacement) 전략

▶반입(Fetch) 전략

보조기억장치에 보관 중인 프로그램이나 데이터를 언제 주기억장치로 적재할 것인지 결정하는 전략

요구 반입 (Demand Fetch)	실행 중인 프로그램이 특정 프로그램이나 데이터 등의 참조를 요구할 때 적재하는 방법
예상 반입 (Anticipatory Fetch)	실행 중인 프로그램에 의해 참조될 프로그램이나 데이터를 미리 예상하여 적재하는 방법

▶배치(Placement) 전략

새로 반입되는 프로그램이나 데이터를 주기억장치의 어디에 위치 시킬 것인지 결정하는 전략

최초 적합 (First Fit)	프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 첫 번째 분할 영역에 배치시키는 방법
최적 적합 (Best Fit)	프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화를 가장 작게 남기는 분할 영역에 배치시키는 방법
최악 적합 (Worst Fit)	프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화를 가장 많이 남기는 분할 영역에 배치시키는 방법

*단편화 : 주기억장치의 분할된 영역에 프로그램이나 데이터를 할당할 경우, 분할된 영역이 프로그램이나 데이터보다 작거나 커서 생기는 빈 기억 공간

책 정리로 예를 들면 최초 적합은 그냥 들어갈 수 있는 첫 번째 영역에 책을 꽂는 것이고 최적 적합은 적당한 빈 공간을 찾아본 후 적당한 곳에 책을 꽂는 것이고 최악 적합은 엄청 넓은 공간에 매우 작은 책을 꽂는 느낌이다.

ex) 기억장치 상태가 다음 표와 같다. 최초, 최적, 최악 적합의 경우에 10k의 프로그램이 할당받게 되는 번호를 각각 구하시오.

영역 번호	영역 크기	상태
1	2k	공백
2	12k	사용중
3	16k	공백
4	14k	공백
5	20k	공백

- 1) First Fit : 3
- 2) Best Fit : 4
- 3) Worst Fit : 5

최초, 최적, 최악 적합의 이런 방식은 구현이 간단하다는 장점이 있으나, 할당 후에 빈 공간이 발생하고 각각의 영역에 빈 공간을 모두 모으면 그 프로그램을 실행할 수 있음에도 불구하고 분할된 영역보다 큰 프로그램을 실행할 수 없다는 단점이 있다. 최근에는 이 방식을 지원하는 시스템은 거의 없고 대부분의 시스템은 가상 메모리(또는 가상 기억장치)방식을 지원한다.

◆가상 기억장치(Virtual Memory)=가상 메모리

보조기억장치의 일부를 주기억장치처럼 사용하는 것으로 용량이 작은 주기억장치를 마치 큰 용량을 가진 것처럼 사용하는 기법

실행될 프로그램이 주기억장치보다 크거나 여러 개인 경우에는 주기억장치 공간의 부족으로 인해 프로그램이 제대로 실행되지 못할 수 있는데 그래서 당장 실행에 필요한 부분만 주기억장치에 저장하고 나머지는 보조기억장치에 두고 동작하도록 한다.

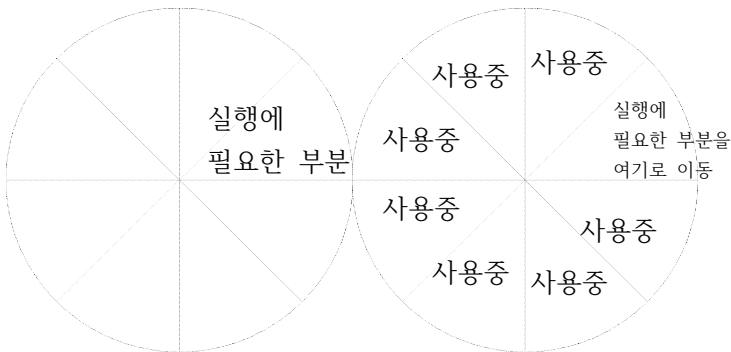
가상기억장치의 일반적인 구현 방법 : 페이징 기법, 세그먼테이션 기법

1) 페이징(Paging) 기법

프로그램과 주기억장치의 영역을 동일한 크기로 나눈 후 나뉜 프로그램을 동일하게 나뉜 주기억장치의 영역에 적재시켜 실행하는 기법

*프로그램을 일정한 크기로 나눈 단위를 페이지라고 하고 페이지 크기로 일정하게 나누어진 주기억장치의 단위를 페이지 프레임이라고 한다.

아래 그림처럼 주기억장치와 가상기억장치에 보관되어있는 프로그램을 동일한 크기로 나누고 가상기억장치에 보관되어 있는 파일 중에 당장 실행에 필요한 부분을 주기억장치의 빈 공간에 적재시킨다.



왼쪽 : 가상기억장치/오른쪽 : 주기억장치

2) 세그멘테이션(Segmentation) 기법

가상기억장치에 보관되어 있는 프로그램을 다양한 크기의 논리적인 단위로 나눈 후 주기억장치에 적재시켜 실행시키는 방법

*프로그램을 배열이나 함수 등과 같은 논리적인 크기로 나눈 단위를 세그먼트라고 하며, 각 세그먼트는 고유한 이름과 크기를 갖는다.

*페이징과 세그멘테이션의 가장 큰 차이점은 페이지는 크기가 동일하지만 세그먼트는 크기가 일정하지 않다는 것이다.

▶교체(Replacement) 전략

주기억장치의 모든 영역이 이미 사용중인 상태에서 새로운 프로그램이나 데이터를 주기억장치에 배치하려고 할 때, 이미 사용되고 있는 영역 중에서 어느 영역을 교체할 것인지를 결정하는 전략

ex) FIFO, OPT, LRU, LFU, NUR, SCR등

◆페이지 교체 알고리즘

페이지 부재가 발생하면 가상기억장치에서 필요한 페이지를 찾아 주기억장치에 적재해야 하는데 이때 주기억장치의 모든 페이지 프레임이 사용중이면 어떤 페이지 프레임을 선택하여 교체할 것인지를 결정해야 한다.

*페이지 부재(Page Fault) : CPU가 액세스한 가상 페이지가 주기억장치에 없는 경우

1) OPT(Optimal replacement, 최적 교체)

앞으로 가장 오랫동안 사용하지 않을 페이지를 교체하는 기법, 벨레이디(Belady)가 제안

페이지 부재 횟수가 가장 적게 발생하는 가장 효율적인 알고리즘

2) FIFO(First In First Out)

각 페이지가 주기억장치에 적재될 때마다 그때의 시간을 기억시켜 가장 먼저 들어오고 가장 오래 있었던 페이지를 교체하는 기법

ex) 4개의 페이지를 수용할 수 있는 주기억장치가 있으며, 초기에는 모두 비어있다고 가정한다. 다음의 순서로 페이지 참조가 발생할 때, FIFO 페이지 교체 알고리즘을 사용할 경우 페이지 결함의 발생 횟수를 구하시오.

*페이지 부재(Page Fault) : CPU가 액세스한 가상 페이지가 주기억장치에 없는 경우

페이지 참조 순서 : 1, 2, 3, 1, 2, 4, 5, 1

출처 : 2020년 9월 필기 기출문제

답 : 6

풀이 :

4개의 페이지고 원래는 비어있으니까 아래 같은 상황이다.

아래 표로 순서대로 표현하면

페이지 참조 순서 1부터 일단 참조 시도한다.

비어있었으므로 페이지 결함이 발생하고 1을 넣는다.

그다음 2를 참조 시도한다. 마찬가지로 페이지 결함이 발생한다. 2가 없었으므로 2를 넣는다.

그다음 3을 참조 시도한다. 마찬가지로 페이지 결함이 발생한다. 3도 없었으므로 3을 넣는다.

그다음 1을 참조한다. 1은 이미 있으므로 결함이 발생하지 않는다.

그다음 2를 참조한다. 2도 이미 있으므로 결함이 발생하지 않는다.

그다음 4를 참조 시도한다. 마찬가지로 페이지 결함이 발생한다. 4도 없었으므로 4를 넣는다.

그다음 5를 참조 시도한다. 마찬가지로 페이지 결함이 발생한다. 근데 공간이 없으므로 가장 먼저 들어왔었던 1을 버리고 5로 교체한다.

그다음 1을 참조 시도한다. 마찬가지로 페이지 결함이 발생한다. 근데 또 공간이 없으므로 가장 먼저 들어왔었던 2를 버리고 1로 교체한다.

	1	1	1	1	1	1	5	5
페이지		2	2	2	2	2	2	1
프레임			3	3	3	3	3	3
						4	4	4
부재 발생	●	●	●			●	●	●

3) LRU(Least Recently Used)

최근에 가장 오랫동안 사용하지 않은 페이지를 교체하는 기법

ex) 4개의 페이지를 수용할 수 있는 주기억장치가 있으며, 초기에는 모두 비어있다고 가정한다. 다음의 순서로 페이지 참조가 발생할 때, LRU 페이지 교체 알고리즘을 사용할 경우 페이지 결함의 발생 횟수를 구하시오.

페이지 참조 순서 : 1, 2, 3, 1, 2, 4, 5, 3

답 : 6

풀이 :

4개의 페이지고 원래는 비어있으니까 아래 같은 상황이다.

아래 표로 순서대로 표현하면

페이지 참조 순서 1부터 일단 참조 시도한다.

비어있었으므로 페이지 결함이 발생하고 1을 넣는다.

그다음 2를 참조 시도한다. 마찬가지로 페이지 결함이 발생한다. 2가 없었으므로 2를 넣는다.

그다음 3을 참조 시도한다. 마찬가지로 페이지 결함이 발생한다. 3도 없었으므로 3을 넣는다.

그다음 1을 참조한다. 1은 이미 있으므로 결함이 발생하지 않는다.

그다음 2를 참조한다. 2도 이미 있으므로 결함이 발생하지 않는다.

그다음 4를 참조 시도한다. 마찬가지로 페이지 결함이 발생한다. 4도 없었으므로 4를 넣는다.

그다음 5를 참조 시도한다. 마찬가지로 페이지 결함이 발생한다. 근데 공간이 없으므로 가장 오랫동안 사용되지 않았던 3을 버리고 5로 교체한다.

그다음 3을 참조 시도한다. 마찬가지로 페이지 결함이 발생한다. 근데 또 공간이 없으므로 가장 오랫동안 사용되지 않았던 1을 버리고 3으로 교체한다.

	1	1	1	1	1	1	1	3
페이지		2	2	2	2	2	2	2
프레임			3	3	3	3	5	5
						4	4	4
부재 발생	●	●	●			●	●	●

4) LFU(Least Frequently Used)

사용 빈도가 가장 적은 페이지를 교체하는 기법

5) NUR(Not Used Recently)

LRU와 비슷한 알고리즘으로, 최근에 사용하지 않은 페이지를 교체하는 기법

최근의 사용 여부를 확인하기 위해 각 페이지마다 두 개의 비트가 사용된다.(참조 비트, 변형 비트)

LRU는 상대적으로 가장 오래된 걸 교체하는 것이고 NUR은 절대적으로 최근에 사용하지 않는 것을 교체

6) SCR(Second Chance Replacement, 2차 기회 교체)

가장 오랫동안 주기억장치에 있던 페이지 중 자주 사용되는 페이지의 교체를 방지하기 위한 기법

●가상기억장치 기타 용어

*Locality(국부성, 지역성, 구역성, 국소성) : 프로세스가 실행되는 동안 주기억장치를 참조할 때 일부 페이지만 집중적으로 참조하는 성질이 있다는 이론

*워킹 셋(Working Set) : 프로세스가 일정 시간 동안 자주 참조하는 페이지의 집합, 데닝(Denning)이 제안

*스래싱(Thrashing) : 프로세스의 처리 시간보다 페이지 교체에 소요되는 시간이 더 많아지는 현상

◆운영체제의 기능-프로세스 관리

▶프로세스(Process)

일반적으로 프로세서에 의해 처리되는 사용자 프로그램, 즉 실행 중인 프로그램을 의미

*프로세서 : 컴퓨터 분야에서 무엇인가를 처리·가공하는 기능을 가진 하드웨어, 소프트웨어로 대표적으로 컴퓨터의 뇌라고 불리는 CPU가 있다.

*CPU(Central Processing Unit) : 중앙처리장치로 컴퓨터의 정중앙에서 모든 데이터를 처리하는 장치

우리는 컴퓨터에서 음악을 들으면서 게임을 하는 것을 당연하게 생각하지만 사실 컴퓨터의 세상에서 여러 개의 프로세스가 동시에 실행되는 것은 놀라운 일이다. 하나의 CPU 즉 프로세서는 한순간에 하나의 프로세스만 실행할 수 있기 때문이다. 프로세스가 동시에 여러 개 실행될 수 있는 이유는 운영체제가 엄청나게 빠르게 CPU가 실행할 프로세스를 교체하기 때문이다. 프로세스에 대한 정보는 PCB에 저장된다.

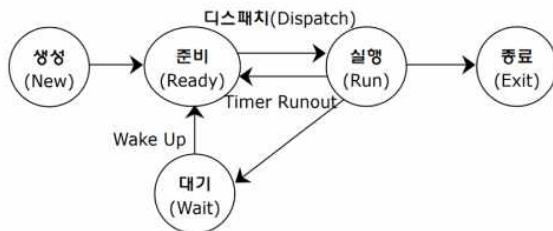
▶PCB(Process Control Block, 프로세스 제어 블록)

운영체제가 프로세스에 대한 중요한 정보를 저장해 놓은 곳

운영체제는 프로세스들의 실행 사이에 프로세스를 교체하고 재시작할 때 오류가 발생하지 않도록 관리해야 한다. 이를 위해 운영체제는 프로세스의 상태를 실행, 준비, 대기 상태로 분류하고 프로세스들을 상태전이를 통해 체계적으로 관리한다.

▶프로세스 상태 전이

프로세스가 시스템 내에 존재하는 동안 프로세스의 상태가 변하는 것



출처 : IT위키

준비(Ready)	프로세스가 프로세서를 할당받기 위해 기다리고 있는 상태
실행(Run)	준비상태 큐에 있는 프로세스가 프로세서를 할당받아 실행되는 상태
대기(Wait)	프로세스에 입·출력 처리가 필요하면 현재 실행 중인 프로세스가 중단되고, 입·출력 블록(Block)
종료(Exit)	프로세스의 실행이 끝나고 할당이 해제된 상태
Dispatch	준비상태에서 대기하고 있는 프로세스 중 하나가 프로세서를 할당받아 실행 상태로 전이되는 과정
Wake up	입·출력 작업이 완료되어 프로세스가 대기 상태에서 준비상태로 전이 되는 과정

*스레드(Thread) : 시스템의 여러 자원을 할당받아 실행하는 프로그램 단위(프로세스의 일부 특성을 갖고 있기 때문에 경량 프로세스라고도 한다)

▶스케줄링(Scheduling)

운영체제가 여러 개의 프로세스가 CPU에서 실행되는 순서를 결정하는 것

CPU는 한번에 하나의 프로세스만 실행시킬 수 있기 때문에 프로그램의 순서를 정해야 한다.

스케줄링에서 우선순위가 높으면 먼저 실행될 수 있는데 이를 스케줄링 우선순위라 한다.

장기 스케줄링	어떤 프로세스가 시스템의 자원을 차지할 수 있도록 할 것인가를 결정하여 준비상태 큐로 보내는 작업
중기 스케줄링	어떤 프로세스들이 CPU를 할당받을 것인지 결정하는 작업
단기 스케줄링	프로세스가 실행되기 위해 CPU를 할당받는 시기와 특정 프로세스를 지정하는 작업

●비선점(Non-Preemptive) 스케줄링

이미 할당된 CPU를 다른 프로세스가 강제로 빼앗아 사용할 수 없는 스케줄링 기법

ex) FCFS, SJF, 우선순위, HRN, 기한부 등

●선점(Preemptive) 스케줄링

우선순위가 높은 다른 프로세스가 CPU를 강제로 빼앗아 사용할 수 있는 스케줄링 기법

ex) RR(Round Robin), SRT, 선점 우선순위, 다단계 큐, 다단계 피드백 큐 등

◆주요 스케줄링 알고리즘

▶FCFS(First Come First Service, 선입 선출)=FIFO(First In First out)

준비상태 큐에 도착한 순서에 따라 차례로 CPU를 할당하는 기법

▶SJF(Shortest Job First, 단기 작업 우선)

준비상태 큐에서 기다리고 있는 프로세스들 중에서 실행 시간이 가장 짧은 프로세스에게 먼저 CPU를 할당하는 기법

ex) SJF(Shorest Job First) 스케줄링에서 다음과 같은 프로세스가 차례로 큐에 도착하였을 때, 평균 반환 시간과 평균 대기 시간을 계산하시오. (출처 : 20년 9월 필기 문제)

프로세스 번호	실행시간
p1	7
p2	8
p3	4
p4	3

평균 반환 시간 : 11.5

평균 대기 시간 : 6

풀이 : SJF는 실행 시간이 가장 짧은 프로세스부터 실행시키게 해주는데 실행 순서는 시간 순서니까 p4, p3, p1, p2가 된다. 예를 들어 p4가 실행되는 동안 그 다음 실행될 p3이 기다리고 있을텐데 p4가 실행 시간이 3이므로 p3은 3만큼 대기 해줘야한다. 따라서 p3의 대기 시간이 3이 되고 다른 프로세스도 마찬가지로 생각해주면 된다.

식으로 쓰면 대기 시간=앞 프로세스의 반환 시간, 반환 시간= 대기 시간+ 실행 시간

프로세스 번호	실행 시간	대기 시간	반환 시간
p4	3	0	3
p3	4	3	7
p1	7	7	14
p2	8	14	22

반환 시간과 대기 시간은 일반적인 평균을 구하는 방법과 동일하게 다 더한 후 개수만큼 나눠주면 된다.

평균 반환 시간 : $(3+7+14+22)/4$, 평균 대기 시간 : $(0+3+7+14)/4$

▶HRN(Highest Response-ratio Next)

대기 시간과 서비스(실행) 시간을 이용하는 기법으로 실행 시간이 긴 프로세스에 불리한 SJF 기법을 보완하기 위한 것이다.

ex) HRN 비선점형 스케줄링의 우선순위를 구하는 계산식을 쓰시오.(출처 : 20년 5월 기출문제)

답 :
$$\frac{\text{대기시간} + \text{서비스시간}}{\text{서비스시간}}$$

ex) HRN 방식으로 스케줄링을 할 경우, 입력된 작업이 다음과 같을 때 처리되는 작업 순서를 나열하시오.
(20년 8월 필기 문제)

작업	대기 시간	서비스(실행) 시간
A	5	20
B	40	20
C	15	45
D	20	2

답 : D->B->C->A

풀이 :
$$\frac{\text{대기시간} + \text{서비스시간}}{\text{서비스시간}}$$
로 계산을 해서 숫자가 가장 높은 것부터 낮은 순으로 나열하면 된다.

▶RR(Round Robin)

시분할 시스템을 위해 설계된 선점형 스케줄링의 하나로서 프로세스들 사이에 우선순위를 두지 않고 순서대로 시간 단위로 CPU를 할당하는 방식

RR은 FCFS와 유사하지만 프로세스마다 CPU를 사용할 수 있는 최대시간이 있다. 프로세스는 자기에게 주어진 시간 동안만 작업할 수 있다.

*시분할 시스템(Time Sharing System) : 여러 명의 사용자가 사용하는 시스템에서 컴퓨터가 사용자들의 프로그램을 번갈아가며 처리해줌으로써 각 사용자들에게 독립된 컴퓨터를 사용하는 느낌을 주는 것으로, 라운드 로빈(Round Robin)이라고도 한다

우선순위 스케줄링의 단점은 순위가 낮은 프로세스들은 무작정 대기를 해야 돼서 CPU를 계속 기다리고만 있는 상태가 되는 것인데 그것을 보완하기 위해 나온게 바로 라운드 로빈 기법이다.

라운드 로빈 스케줄링은 프로세스가 하나 끝날 때까지 CPU를 가지고 있는게 아니라, 할당된 시간만큼 돌아가며 처리하는 방식이다. 이렇게 하면 우선순위가 낮은 프로세스도 공정하게 실행될 수 있다.

라운드 로빈은 라운드 로빈 DSN, 리눅스 등을 포함한 대부분의 시스템에서 사용하고 있는 방식이다.

ex) 준비상태 큐에 프로세스 A, B, C가 차례로 도착하였다. 라운드 로빈(Round Robin)으로 스케줄링할 때 타임 슬라이스를 4초로 한다면 평균 반환 시간은? 17초

(18년 8월 필기 문제)

프로세스	A	B	C
실행시간(초)	17	4	5

RR은 프로세스가 하나 끝날 때까지 CPU를 가지고 있는게 아니라, 할당된 시간만큼 돌아가며 처리하는 방식이다. 시간제한이 있고 그 시간제한 동안 프로세스를 실행하고 시간이 초과되면 맨 뒤 순서로 가서 다시 실행되길 기다린다.

먼저 A를 실행하는데 시간제한이 4이므로 4만큼 실행되고 나머지 13은 C 뒤로 이동된다
B는 실행 시간이 4이므로 실행이 종료되고 그다음 C로 넘어간다.

프로세스	실행시간	시간할당	대기시간	반환시간	남은시간
A	17	4	0	4	13
B	4	4	4	8	0
C	5	4	8	12	1
A	13	4	12	16	9
C	1	4	16	17	0
A	9	4	17	21	5
A	5	4	21	25	1
A	1	4	25	26	0

반환 시간 = 프로세스가 끝난 시간-도착 시간

대기 시간 = 프로세스가 끝난 시간-도착 시간-실행 시간

이 문제에서 도착 시간은 없으므로 전부 0이고

-A가 최종 종료된 반환시간은 26

-B가 최종 종료된 반환시간은 8

-C가 최종 종료된 반환시간은 17

*평균 반환 시간= (26+8+17)/3=17

ex) 다음을 라운드 로빈으로 스케줄링하는데 시간 할당량은 3이다. 다음 프로세스들의 평균 반환시간과 평균 대기시간을 구하시오.

프로세스	도착시간	실행시간
P1	0	8
P2	1	5
P3	2	2

평균 반환 시간 : 11

평균 대기 시간 : 6

프로세스	도착시간	실행시간	시간할당	대기시간	반환시간	남은시간
P1	0	8	3	0	3	5
P2	1	5	3	3	6	2
P3	2	2	3	6	8	0
P1		5	3	8	11	2
P2		2	3	11	13	0
P1		2	3	13	15	0

▲반환시간 = 프로세스가 끝난 시간- 도착시간

-P1 반환시간 =15-0=15

-P2 반환시간 =13-1=12

-P3 반환시간 =8-2=6

*평균 반환시간 = (15+12+6)/3=11

▲대기시간 = 프로세스가 끝난 시간-도착 시간-실행시간

-P1 대기시간 = 15-0-8=7

-P2 대기시간 = 13-1-5=7

-P3 대기시간 = 8-2-2=4

*평균 대기시간=(7+7+4)/3=6

▶SRT(Shortest Remaining Time)

SJF방식을 선점 스케줄링 방식으로 변경한 기법

최단 잔여시간을 우선으로 하는 스케줄링이다

진행 중인 프로세스가 있어도, 최단 잔여시간인 프로세스를 위해 sleep 시키고 짧은 프로세스를 먼저 할당한다.

ex) 다음을 SRT로 스케줄링하는데, 다음 프로세스들의 평균 반환시간과 평균 대기시간을 구하시오.

프로세스	도착시간	실행시간
P1	0	8
P2	1	4
P3	2	9
P4	3	5

먼저 P1밖에 없으므로 P1을 실행한다.

그런데 1초후에 P2가 들어왔는데 P1보다 짧기 때문에 P1을 중단시키고 P2를 실행한다.

P2가 실행이 완료된 후 P3, P4는 이미 그 전에 들어왔고 P4가 더 짧으므로 바로 P4가 실행된다.

이 이후는 그냥 짧은 순으로 진행된다

▲반환시간 = 프로세스가 끝난 시간- 도착시간

-P1 반환시간 =17-0=17

-P2 반환시간 =5-1=4

-P3 반환시간 =26-2=24

-P4 반환시간=10-3=7

*평균 반환시간 = (17+4+24+7)/4=13

▲대기시간 = 프로세스가 끝난 시간-도착 시간-실행시간

-P1 대기시간 = 17-0-8=9

-P2 대기시간 = 5-1-4=0

-P3 대기시간 = 26-2-9=15

-P4 대기시간 = 10-3-5=2

*평균 대기시간=(9+0+15+2)/4=6.5

프로세스	도착시간	실행시간	대기시간	반환시간	남은시간
P1	0	8	0	1	7
P2	1	4	1	5	0
P4	3	5	5	10	0
P1		7	10	17	0
P3	2	9	17	26	0

◆교착상태(Dead Lock)

둘 이상의 프로세스들이 서로 남이 가진 자원을 요구하면서 양쪽 모두 작업 수행을 못하고 대기 상태로 놓이는 상태

▶교착상태 발생 조건

4가지 조건 중 하나도 충족되지 않으면 교착상태가 발생하지 않는다.

상호 배제 (Mutual Exclusion)	한 번에 한 개의 프로세스만이 공유 자원을 사용할 수 있어야함
점유와 대기 (Hold and Wait)	최소한 하나의 자원을 가지고 있으면서 다른 프로세스에 할당되어 있는 자원을 추가로 요구하기 위해 대기하는 프로세스가 있어야함
비선점 (Non-preemption)	다른 프로세스에 할당된 자원은 사용이 끝날 때까지 강제로 빼앗을 수 없어야 함
환형 대기 (Circular Wait)	공유 자원과 공유 자원을 사용하기 위해 대기하는 프로세스들이 원형으로 구성되어 있어 자신에게 할당된 자원을 가지면서 앞이나 뒤에 있는 프로세스의 자원을 요구해야 함

▶교착상태 해결 방법

예방 기법 (Prevention)	교착상태가 발생하지 않도록 사전에 시스템을 제어
회피 기법 (Avoidance)	교착상태가 발생할 가능성을 배제하지 않고 교착상태가 발생하면 적절히 피함 (은행원 알고리즘(Banker's Algorithm)이 사용됨)
발견 기법 (Detection)	시스템에 교착상태가 발생했는지 점검하여 교착상태에 있는 프로세스와 자원을 발견하는 것
회복 기법 (Recovery)	교착상태를 일으킨 프로세스를 종료하거나 교착상태의 프로세스에 할당된 자원을 선점하여 프로세스나 자원을 회복하는 것

*은행원 알고리즘 : E. J. Dijkstra가 제안한 것으로 은행에서 모든 고객의 요구가 충족되도록 현금을 할당하는 데서 유래한 기법

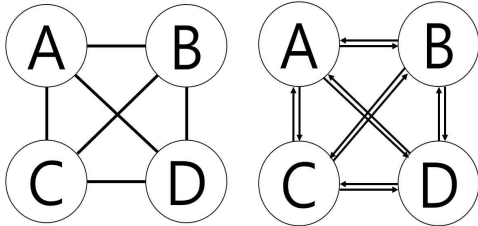
VII. 자료구조와 정렬 파트

좋은 프로그램을 작성하기 위해서는 적절한 데이터 구조를 선택하는 것이 중요하다.

자료구조는 기억장치의 공간 내에 저장하는 방법과 자료 간의 관계, 처리 방법 등을 연구 분석하는 것을 말한다.

배열(Array)	크기와 형(Type)이 동일한 자료들이 순서대로 나열된 자료의 집합
연속 리스트 (Contiguous List)	배열과 같이 연속되는 기억장소에 저장되는 자료 구조
연결 리스트 (Linked List)	각 데이터들을 포인터로 연결하여 관리하는 구조
스택(Stack)	리스트의 한쪽 끝으로만 자료의 삽입, 삭제 작업이 이루어지는 자료 구조
큐(Queue)	리스트의 한쪽에서는 삽입 작업이 이루어지고 다른 한쪽에서는 삭제 작업이 이루어지는 자료구조
그래프(Graph)	정점과 간선의 두 집합으로 이루어지는 자료구조

ex) 정점이 4개인 경우 무방향 그래프와 방향 그래프의 최대 간선 수를 구하시오.



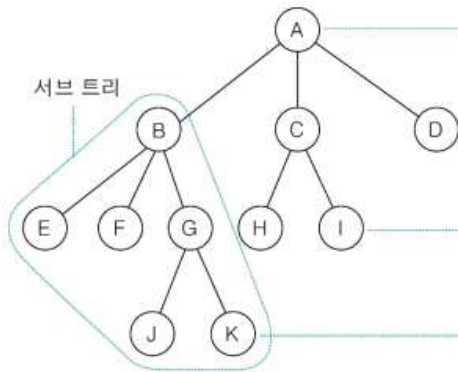
무방향 그래프 최대 간선 수 공식 : $n(n-1)/2$
 방향 그래프 최대 간선 수 공식 : $n(n-1)$
 n은 정점의 개수

무방향 그래프 최대 간선 수 공식 : $4(4-1)/2=6$

방향 그래프 최대 간선 수 공식 : $4(4-1)=12$

▶트리

정점과 선분을 이용하여 사이클 이루지 않도록 구성한 그래프의 특수한 형태



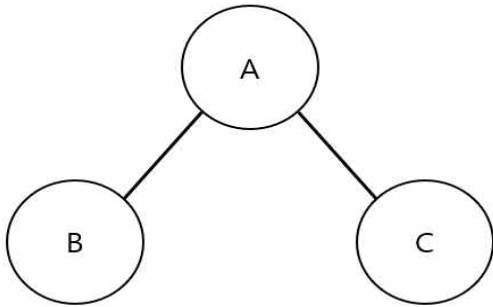
- 1) 근노드 : 트리의 맨 위에 있는 노드 ex) A
- 2) 디그리(Degree, 차수) : 각 노드에서 뻗어나온 가지의 수 ex) A의 경우는 3, C의 경우는 2, D의 경우는 0
- 3) 단말 노드(Terminal Node) : 자식이 하나도 없는 노드 ex) D
- 4) 비단말 노드(Non-Terminal Node) : 자식이 하나라도 있는 노드
- 5) 조상 노드(Ancestors Node) : 임의의 노드에서 근 노드에 이르는 경로상에 있는 노드들 ex) K의 조상 노드는 G, B, A
- 6) 자식 노드(Son Node) : 어떤 노드에 연결된 다음 레벨의 노드들
- 7) 부모 노드(Parent Node) : 어떤 노드에 연결된 이전 레벨의 노드들
- 8) 형제 노드(Brother Node) : 동일한 부모를 갖는 노드들
- 9) 트리의 디그리 : 노드들의 디그리 중에서 가장 많은 수

출처 : 트리 (naver.com)

▶이진 트리의 순회(Traversal)

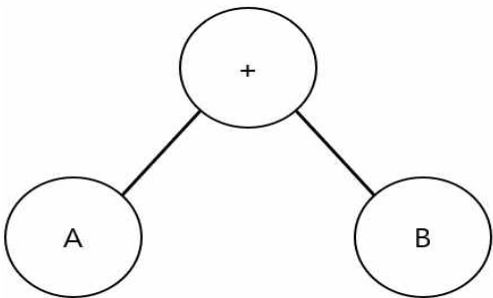
모든 노드를 특정한 순서대로 한번씩 방문하는 것

*이진 트리 : 차수가 2 이하인 노드들로 구성된 트리



- Preorder 순회법(전위 순회) : Root->Left->Right
- Inorder 순회법(중위 순회) : Left->Root->Right
- Postorder 순회법(후위 순회) : Left->Right->Root

▶수식의 표기법

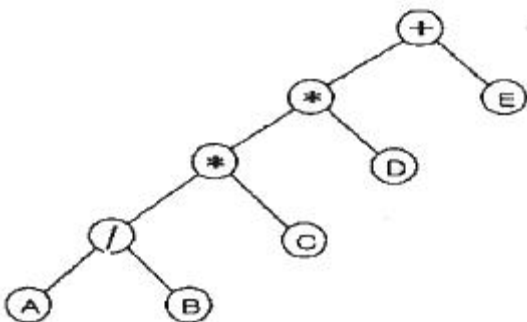


- 전위 표기법(Prefix) : 연산자->Left->Right
- 중위 표기법(Infix) : Left->연산자->Right
- 후위 표기법(PostFix) : Left->Right->연산자

ex) 다음 Postfix 연산식에 대한 연산 결과를 쓰시오. (20년 9월 필기 기출문제)

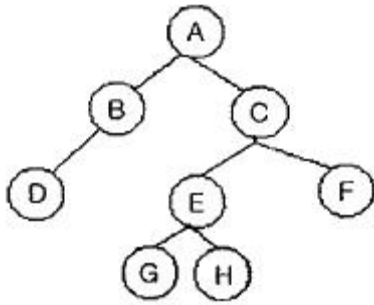
3 4 * 5 6 * +

ex) 다음 트리를 전위 순회했을 때 방문한 노드를 순서대로 쓰시오. (20년 6월 필기 기출문제)



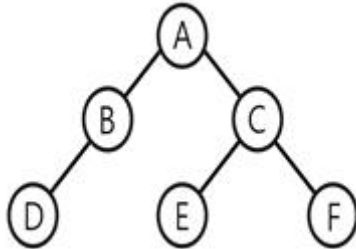
답 : +**/ABCDE

ex) 다음 트리를 Preorder 운행법으로 운행할 경우 가장 먼저 탐색 되는 노드를 쓰시오. (20년 8월 필기 기출문제)



답 : A

ex) 다음 트리로 Inorder 운행을 수행했을 때 방문한 노드를 순서대로 쓰시오. (20년 9월 필기 기출문제)



답 : DBAECF

◆정렬

삽입 정렬 (Insertion Sort)	두 번째 자료부터 시작하여 그 앞(왼쪽)의 자료들과 비교하여 삽입할 위치를 지정한 후 자료를 뒤로 옮기고 지정한 자리에 자료를 삽입하여 정렬하는 방식
선택 정렬 (Selection Sort)	최솟값을 찾아 첫 번째 레코드 위치에 놓고, 나머지 중에서 다시 최솟값을 찾아 두 번째 레코드 위치에 놓는 방식을 반복하여 정렬하는 방식
버블 정렬 (Bubble Sort)	파일에서 인접한 두 개의 레코드 키값을 비교하여 그 크기에 따라 레코드 위치를 서로 교환하는 방식

ex) 다음 자료에 대하여 선택 정렬을 이용하여 오름차순으로 정렬하고자 한다. 3회전 후의 결과를 쓰시오. (20년 8월 필기 기출문제)

37, 14, 17, 40, 35

답 : 14, 17, 35, 40, 37

풀이

처음 : 37 14 17 40 35

1차 : 37 14 17 40 35 -> 14 37 17 40 35

-> 14 37 17 40 35 -> 14 37 17 40 35

1칸과 2칸 비교해서 14가 37보다 작으니 1칸으로 이동하고 17, 40, 35도 마찬가지로 비교하는데 전부 14보다 크므로 정렬 종료

2차 : 14 37 17 40 35 -> 14 17 37 40 35 -> 14 17 37 40 35

2칸과 3칸 비교해서 17이 37보다 작으니 2칸으로 이동하고 40, 35도 마찬가지로 비교하는데 전부 17보다 크므로 정렬 종료

3차 : 14 17 37 40 35 -> 14 17 37 40 35 -> 14 17 35 40 37

3칸과 4칸 비교하고 3칸과 5칸 비교했더니 37보다 35가 더 작으므로 35는 3칸으로 이동하고 정렬 종료

ex) 다음 자료에 대하여 삽입 정렬을 이용하여 오름차순으로 정렬하고자 한다. 3회전 후를 쓰시오. (20년 9월 필기 기출문제 변형)

8	3	4	9	7
---	---	---	---	---

답 : 3, 4, 8, 9, 7

풀이

처음 :

8	3	4	9	7
---	---	---	---	---

1차 :

8	3	4	9	7
---	---	---	---	---

 ->

3	8	4	9	7
---	---	---	---	---

먼저 2칸을 기준으로 2칸을 1칸과 비교해서 3이 더 작으니 3을 1칸으로 이동

2차 :

3	8	4	9	7
---	---	---	---	---

 ->

3	8	4	9	7
---	---	---	---	---

 ->

3	4	8	9	7
---	---	---	---	---

그다음 3칸을 기준으로 3칸을 1칸과 비교하고 그다음 3칸을 2칸과 비교했더니 4가 더 작아서 2칸으로 이동

3차 :

3	4	8	9	7
---	---	---	---	---

 ->

3	4	8	9	7
---	---	---	---	---

 ->

3	4	8	9	7
---	---	---	---	---

4칸을 1칸과 비교하고 그다음 4칸을 2칸과 비교하고 또 4칸을 3칸과 비교

ex) 다음 자료에 대하여 버블 정렬을 이용하여 오름차순으로 정렬하고자 한다. 1회전 후의 결과를 쓰시오.

37	14	17	40	35
----	----	----	----	----

답 : 14, 17, 37, 35, 40

풀이

처음 :

37	14	17	40	35
----	----	----	----	----

1차 :

37	14	17	40	35
----	----	----	----	----

 ->

14	37	17	40	35
----	----	----	----	----

 ->

14	17	37	40	35
----	----	----	----	----

->

14	17	37	40	35
----	----	----	----	----

 ->

14	17	37	35	40
----	----	----	----	----

1칸, 2칸 비교/ 2칸, 3칸 비교/ 3칸, 4칸 비교/ 4칸, 5칸 비교

2차 :

14	17	37	35	40
----	----	----	----	----

 ->

14	17	37	35	40
----	----	----	----	----

 ->

14	17	37	35	40
----	----	----	----	----

->

14	17	35	37	40
----	----	----	----	----

 ->

14	17	36	37	40
----	----	----	----	----

★참고문헌, 사이트

I. 소프트웨어 공학 파트

- 1) [소프트웨어 - 나무위키 \(namu.wiki\)](https://namu.wiki)
- 2) <https://terms.naver.com/list.naver?cid=58528&categoryId=58528&so=st4.asc>
- 3) 2021 시나공 정보처리기사 실기
- 4) [절차지향언어 vs 객체지향언어 : 네이버 블로그 \(naver.com\)](https://naver.com)
- 5) [\(32\) 객체지향 프로그래밍? 문과도 이해썬가능. 10분컷. - YouTube](https://www.youtube.com/watch?v=...)
- 6) [코딩의 시작, TCP School](https://www.tcp-school.com)
- 7) [객체 지향 프로그래밍의 4가지 특징 | 추상화, 상속, 다형성, 캡슐화 - \(codestates.com\)](https://codestates.com)
- 8) [\[Java\]오버로딩 & 오버라이딩\(Overloading & Overriding\) \(tistory.com\)](https://tistory.com)
- 9) [싱글턴 패턴 - 위키백과, 우리 모두의 백과사전 \(wikipedia.org\)](https://wikipedia.org)
- 10) [\[GOF\] ♡ 추상 팩토리\(Abstract Factory\) 패턴 - 완벽 마스터하기 \(tistory.com\)](https://tistory.com)
- 11) [생성 디자인 패턴 \(refactoring.guru\)](https://refactoring.guru)
- 12) [브리지 \(refactoring.guru\)](https://refactoring.guru)
- 13) [13 데코레이터 패턴 \(Decorator Pattern\) \(tistory.com\)](https://tistory.com)
- 14) [퍼사드 \(refactoring.guru\)](https://refactoring.guru)
- 15) [09 퍼사드 패턴 \(Facade Pattern\) \(tistory.com\)](https://tistory.com)
- 16) [프록시 \(refactoring.guru\)](https://refactoring.guru)
- 17) [책임 연쇄 \(refactoring.guru\)](https://refactoring.guru)
- 18) [커맨드 \(refactoring.guru\)](https://refactoring.guru)
- 19) [애플리케이션 - 나무위키 \(namu.wiki\)](https://namu.wiki)
- 20) [테스트 설계기법_동적테스트_명세기반\(블랙박스\)_동등분할, 경계값분석, 의사결정테이블, 상태전이, 유즈케이스, 분류트리, 페어와이즈, 클래스 테스트 기법 \(tistory.com\)](https://tistory.com)
- 21) [File:Wireframe mockup prototype.png - Wikimedia Commons](https://commons.wikimedia.org/wiki/File:Wireframe_mockup_prototype.png)

II. 데이터베이스(DB) 파트

- 1) 2021 시나공 정보처리기사 실기
- 2) [데이터베이스 개론 : 네이버 지식백과 \(naver.com\)](https://naver.com)
- 3) [데이터의 종속성 \(Data Dependency\) : 네이버 블로그 \(naver.com\)](https://naver.com)
- 4) [\[SQL\] 키\(슈퍼키,대체키,후보키,기본키,외래키\) \(tistory.com\)](https://tistory.com)

Ⅲ. 네트워크 파트

- 1) 2021 시나공 정보처리기사 실기
- 2) [데이터 통신과 컴퓨터 네트워크 : 네이버 지식백과 \(naver.com\)](#)
- 3) [네트워크 초등 소프트웨어 용어 사전](#)
- 4) [네트워크 OSI 7 계층 \(velog.io\)](#)
- 5) <https://just-my-blog.tistory.com/22>
- 6) [\[네트워크\] 서브넷, 서브넷마스크, 서브넷팅이란? | 서브넷팅 예제 \(tistory.com\)](#)
- 7) [인터넷 프로토콜 스위트 - 위키백과, 우리 모두의 백과사전 \(wikipedia.org\)](#)
- 8) [TCP/IP란 무엇이며 어떤 원리로 작동하나요? | NordVPN](#)
- 9) [개발자를 꿈꾸는 프로그래머 :: DHCP란? \(tistory.com\)](#)
- 10) [\[TCP/UDP\] TCP와 UDP의 특징과 차이 - MangKyu's Diary \(tistory.com\)](#)
- 11) [라우팅 - 위키백과, 우리 모두의 백과사전 \(wikipedia.org\)](#)
- 12) NEW 정보통신개론-고응남
- 13) [회선 교환 방식과 축적 교환 방식\(패킷 교환 방식, 메시지 교환 방식\) : 네이버 블로그 \(naver.com\)](#)
- 14) [\[네트워크\] OSI 7 Layers - 개념 정리 및 각 계층 정리 - SH's Devlog \(tistory.com\)](#)
- 15) [네트워크 | OSI 7 계층 그림과 함께 이해하기 \(velog.io\)](#)
- 16) [\[네트워크 OSI 7계층\] 1계층 물리계층에 대해 살펴보자. Physical Layer 피지컬계층! \(tistory.com\)](#)
- 17) [\[네트워크 관리\] 통신회선과 다중화 전송 - 퍼플의 개발 일지 \(tistory.com\)](#)
- 18) [제08절 정보 전송 방식 : 네이버 블로그 \(naver.com\)](#)
- 19) [ITWorld 용어풀이 | 32비트, 64비트 - ITWorld Korea](#)
- 20) [\[볼륨주\] 문자 인코딩, 유니코드, UTF-8이 뭔가요? - YouTube](#)
- 21) [유니코드 한자시스템 \(aks.ac.kr\)](#)
- 22) [동기식 전송과 비동기식 전송 :: 진리 블로그 \(tistory.com\)](#)
- 23) [클럭 - 나무위키 \(namu.wiki\)](#)
- 24) [Clk \(ktword.co.kr\)](#)
- 25) [네트워크관리사 필기 완벽대비 9강 - 동기식과 비동기식 전송 - YouTube](#)
- 26) [동기식 비동기식 전송 3분 정리 \(tistory.com\)](#)
- 27) [동기식 전송 vs 비동기식 전송 : 네이버 블로그 \(naver.com\)](#)
- 28) [HDLC \(ktword.co.kr\)](#)
- 29) [\[정보통신개론\] 흐름제어, HDLC 프로토콜 : 네이버 블로그 \(naver.com\)](#)
- 30) [HDLC 프로토콜\(High-Level Data Link Control \) \(tistory.com\)](#)
- 31) [\[컴퓨터네트워크\] 데이터 링크 제어 프로토콜\(HDLC & SDLC & BSC & PPP\) \(itnovice1.blogspot.com\)](#)
- 32) [ARQ\(Automatic Repeat Request\)란? \(velog.io\)](#)
- 33) [\[데이터 링크 프로토콜\] 동기 데이터 링크 프로토콜 - 반나무_뿌리 \(tistory.com\)](#)

IV. 웹 파트

- 1) [Web Service란? \(easylaw.go.kr\)](http://easylaw.go.kr)
- 2) [마크업 언어 - 위키백과, 우리 모두의 백과사전 \(wikipedia.org\)](http://wikipedia.org)
- 3) [마크업 언어 - 나무위키 \(namu.wiki\)](http://namu.wiki)
- 4) 2021 시나공 정보처리기사 실기
- 5) NEW 정보통신개론-고응남
- 6) [월드 와이드 웹 - 위키백과, 우리 모두의 백과사전 \(wikipedia.org\)](http://wikipedia.org)
- 7) [WWW \(ktword.co.kr\)](http://ktword.co.kr)
- 8) [웹 애플리케이션 서버 - 위키백과, 우리 모두의 백과사전 \(wikipedia.org\)](http://wikipedia.org)

V. 보안 파트

- 1) 2021 시나공 정보처리기사 실기
- 2) [정보 보안 개론 : 네이버 지식백과 \(naver.com\)](http://naver.com)
- 3) [파밍\(Pharming\) < 전자금융범죄 \(easylaw.go.kr\)](http://easylaw.go.kr)
- 4) [사회공학적 해킹 : 네이버 블로그 \(naver.com\)](http://naver.com)
- 5) [\[카드뉴스\] 왕초보 용어 풀이, 웨일링 공격이란? \(boannews.com\)](http://boannews.com)
- 6) [SSO란 무엇인가요? - Single Sign-On 설명 - AWS \(amazon.com\)](http://amazon.com)
- 7) [SIEM이란? | Microsoft Security](http://Microsoft Security)
- 8) [ESM / SIEM / SOAR 각각의 의미?? \(tistory.com\)](http://tistory.com)
- 9) [국제 데이터 암호화 알고리즘 - 위키백과, 우리 모두의 백과사전 \(wikipedia.org\)](http://wikipedia.org)

VI. 운영체제 파트

- 1) [주기억장치-초등 소프트웨어 용어사전](#)
- 2) [주기억장치-소프트웨어 어휘다지기-중등](#)
- 3) [보조기억장치-두산백과](#)
- 4) [안경잡이개발자 :: 컴퓨터의 기억장치란? \(tistory.com\)](http://tistory.com)
- 5) [컴퓨터 개론 : 네이버 지식백과 \(naver.com\)](http://naver.com)
- 6) [\[운영체제\] 프로세스가 뭐지? - 멍멍멍 \(tistory.com\)](http://tistory.com)
- 7) 2021 시나공 정보처리기사 실기
- 8) [라운드 로빈 - 해시넷 \(hash.kr\)](http://hash.kr)
- 9) [<운영체제> 시분할 시스템이란 \(tistory.com\)](http://tistory.com)
- 10) [CPU 스케줄러 전략 \(FCFS, SJF, HRN, SRT, Round Robin, Multi-Level Queue\) \(tistory.com\)](http://tistory.com)
- 11) [교착 상태 \(naver.com\)](http://naver.com)
- 12) [라운드 로빈 \(tistory.com\)](http://tistory.com)

VII. 자료구조와 정렬 파트

- 1) 2021 시나공 정보처리기사 실기
- 2) [컴퓨터 개론 : 네이버 지식백과 \(naver.com\)](http://naver.com)