

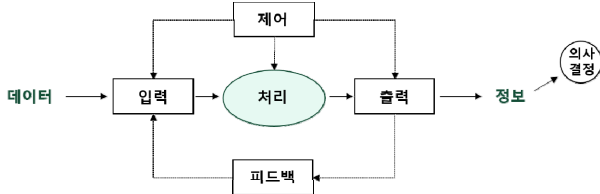
1과목

소프트웨어 설계

▶ 시스템(System)

어떤 목적을 위해 하나 이상의 상호 관련된 요소의 유기적 결합체

- **특성** : 종합성, 목적성, 자동성, 제어성
- **구성 요소** : 입력(Input), 처리(Process), 출력(Output), 제어(Control), 피드백(Feedback)



▶ 소프트웨어 공학

신뢰도가 높은 소프트웨어를 만들기 위한 방법으로, 도구와 절차들을 체계화한 학문

- **일반적인 소프트웨어 개발 과정** : 계획 → 요구분석 → 설계 → 구현(개발) → 테스트(시험) → 유지보수
- **소프트웨어 위기** : 시스템의 대규모화에 따라 소프트웨어의 신뢰성 저하, 개발 시간 지연, 인력 부족, 인건비 상승으로 인한 개발비의 증대, 계획의 지연 등의 현상이 현저하여 개발 계획의 수행을 매우 어렵게 만드는 상황
 - 하드웨어 성능 발달로 인해 소프트웨어 개발 속도가 하드웨어 개발 속도를 따라가지 못해, 사용자들의 요구사항을 감당할 수 없는 문제가 발생한 것

▶ 일반적인 소프트웨어 개발 과정

계획 → 요구분석 → 설계 → 구현(개발) → 테스트(시험) → 유지보수

계획	비용, 기간 등 프로젝트를 수행하는 데 필요한 것에 대해 계획하는 단계
요구분석	사용자의 요구사항을 분석하고 시스템으로 구현 가능한지 판단하는 단계
구현	프로그램 언어를 선정하고, 설계 명세서를 컴퓨터가 이해할 수 있도록 표현하는 단계
테스트	요구사항에 맞게 작동하는지 테스트하는 단계
유지보수	버전 업데이트 및 새로운 기능 추가 등을 유지보수하는 단계

▶ 소프트웨어 생명 주기(Software Life Cycle)

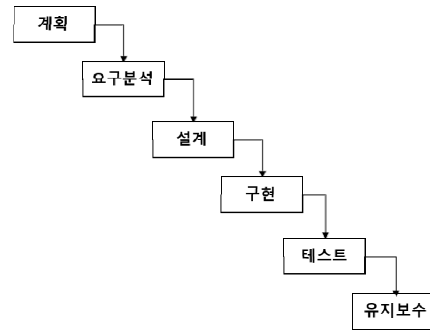
소프트웨어 제품을 계획할 때부터 시작하여 운용 및 유지보수에 이르기까지 변화의 전 과정

▶ 폭포수 모델(Waterfall Model)

가장 오래된 모델로 순차적으로 한 단계, 한 단계를 진

행해 나가는 모델

- 선형 순차적 모델, 고전적 생명주기 모형



▶ 프로토타입 모델(Prototype Model)

사용자의 요구사항에 따라 프로토타입(시제품)을 신속히 개발하여 제공한 후, 사용자의 피드백을 통해 개선하고 보완해가는 모델

- 폭포수 모델의 단점을 보완한 모델



▶ 나선형 모델(Spiral Model)

시스템 개발 시 위험을 최소화하기 위해 점진적으로 완벽한 시스템으로 개발해 나가는 모델

- **4가지 주요 활동** : 목표 설정 → 위험 분석 → 개발과 검증 → 고객 평가



▶ RAD(Rapid Application Development) 모델

사용자의 적극적인 참여와 강력한 소프트웨어 개발 도구를 이용하여 매우 짧은 주기(60~90일)로 개발을 진행하는 순차적 모델

▶ 애자일 모델(Agile Model)

소프트웨어 개발 과정에서 지속적으로 발생하는 변경에 유연하고 기민하게 대응하여 생산성과 품질 향상을 목표로 하는 협력적인 모델

• 애자일 선언문(Agile Manifesto)

- 공정과 도구보다 → 개인과 상호작용을
- 포괄적인 문서보다 → 작동하는 소프트웨어를
- 계약 협상보다 → 고객과의 협력을
- 계획에 따르기보다 → 변화에 대응하기를 가치있게 여긴다.

• **종류**

- 스크럼(Scrum)
- 익스트림 프로그래밍(XP; eXtreme Programing)
- 린(Lean) 소프트웨어 개발 방법론
- 칸반(Kanban)
- 적응형 소프트웨어 개발 방법론(ASD; Adaptive Software Development)
- 기능 주도 개발 방법론(FDD; Feature Driven Development, 기능 기반, 기능 중심, 특징 주도)
- 동적 시스템 개발 방법론(DSDM; Dynamic Systems Development Method)
- DAD(Disciplined Agile Delivery)
- 크리스털 패밀리(Crystal Family)
- 애자일 UP(AUP; Agile Unified Process)

▶ **스크럼(Scrum)**

매일 정해진 시간에 정해진 장소에서 짧은 시간의 개발을 하는 팀을 위한 프로젝트 관리 중심의 방법론

- 30일 단위의 짧은 개발 기간으로 분리하여, 반복적으로 수행하는 스프린트를 중심으로 진행

▶ **익스트림 프로그래밍**

(XP; eXtreme Programming)

고객과 함께 2주 정도의 반복 개발로, 고객 만족을 강조하고 테스트와 우선 개발을 특징으로 하는 방법론

- **XP의 5가지 핵심 가치** : 의사소통(Communication), 단순성(Simplicity), 피드백(Feedback), 용기(Courage), 존중(Respect)
- **XP의 실천 방법(Practice)**
 - 페어 프로그래밍(Pair Programming, 짝 프로그래밍)
 - 단체 소유권(Collective Ownership)
 - 지속적인 통합(Continuous Integration)
 - 전체 팀(Whole Team)
 - 고객 테스트(Customer Tests)
 - 소규모 릴리즈(Small Releases)
 - 테스트 주도 기법(Test-Driven Development)
 - 디자인 개선(Design Improvement) 또는 리팩토링(Refactoring)

▶ **린(Lean) 소프트웨어 개발 방법론**

자동차 제조사 도요타의 생산 방식인 '낭비를 발견하고 제거하여 고객에게 가치를 빠르게 제공하자'를 소프트웨어 개발에 적용한 방법론

▶ **요구 공학(Requirements Engineering)**

요구사항의 획득, 분석, 명세, 검증 및 변경 관리 등에 대한 제반 활동과 원칙

• **요구 공학의 프로세스**

- 요구사항 개발 : 요구사항 도출 → 요구사항 분석 → 요구사항 명세 → 요구사항 확인(검증)
- 요구사항 관리 : 요구사항 협상, 요구사항 기준선, 요구사항 변경 관리, 요구사항 확인

▶ **요구사항 분류**

• **기술하는 내용에 따른 분류**

- 기능적 요구사항 : 시스템이 수행해야 하는 작업에 관한 요구사항
- 비기능적 요구사항 : 소프트웨어 기능들에 대한 조건과 제약에 관한 요구사항

• **관점에 따른 분류**

- 사용자 요구사항 : 사용자 관점의 요구사항
- 시스템 요구사항 : 개발자 관점의 요구사항

▶ **요구사항 개발**

- **요구사항 도출(Requirement Elicitation, 요구사항 수집)** : 요구사항이 어디에 있고 어떻게 수집할 것인가와 관련되어 있는 단계
 - 기법 : 인터뷰, 설문 조사, 브레인 스토밍, 워크숍, 유스케이스, 프로토타이핑 등
- **요구사항 분석(Requirement Analysis)** : 요구사항 간에 상충되는 것을 해결하고, 소프트웨어의 범위를 파악하며, 소프트웨어가 환경과 어떻게 상호 작용하는지 이해하는 단계
 - 기법 : 자료 사전, 자료 흐름도, 소단위 명세서, 개체 관계도 등
- **요구사항 명세(Requirement Specification)** : 체계적으로 검토, 평가, 승인될 수 있는 문서를 작성하는 단계
 - 기법 : 비정형 명세 기법, 정형 명세 기법
- **요구사항 확인(Requirement Validation)** : 분석가가 요구사항을 이해했는지 확인(Validation)이 필요하고, 요구사항 문서가 회사의 표준에 적합하고 이해가 가능하며, 일관성 있고 완전한지 검증(Verification)하는 단계

▶ **정형 기술 검토**

(FTR; Formal Technical Review)

동료 검토 (Peer Review)	<ul style="list-style-type: none"> 요구사항 명세서 작성자가 명세서 내용을 직접 설명하고 동료들이 이를 들으면 서 결함을 발견하는 형태의 검토 방법
워크스루 (Walkthrough)	<ul style="list-style-type: none"> 비공식적 검토 과정으로 검토 회의 전에 요구사항 명세서를 미리 배포하여 사전 검토한 후에 짧은 검토 회의를 통해 결함을 발견하는 검토 방법

	<ul style="list-style-type: none"> 오류 검출에 초점을 두고 해결책은 나중에 미룸
인스펙션 (Inspection)	<ul style="list-style-type: none"> 요구사항 명세서 작성자를 제외한 다른 검토 전문가들이 요구사항 명세서를 확인하면서 결함을 발견하는 형태의 검토 방법 결함 발생에 대한 해결책을 제시

▶ 모델링(Modeling)

복잡한 현실 세계의 현상을 특정한 목적에 맞추어 일정한 형식으로 이해하기 쉽게 표현하는 일

- **모델링 언어** : 요구사항 정의·분석·설계의 결과물을 다양한 다이어그램으로 표현하는 표기법
- **모델링 언어 종류** : 자료 사전, 자료 흐름도, 소단위 명세서, 개체 관계도, UML 등

▶ 자료 사전(DD; Data Dictionary, 데이터 사전)

시스템과 관련된 모든 자료의 명세와 자료 속성을 파악할 수 있도록 조직화한 것

- **자료 사전의 기호와 의미**





기호	의미
=	정의
{ }	반복
+	연결
* *	주석, 설명
[]	선택
()	생략

▶ 자료 흐름도(DFD; Data Flow Diagram)

시스템과 관련된 모든 자료의 명세와 자료 속성을 파악할 수 있도록 조직화한 것

- 자료 흐름 그래프, 버블 차트

- **자료 흐름도의 기호와 의미**

기호	의미
	단말(External Entity, Terminator)
	처리(Process, 프로세스)
	자료 저장소(Data Store)
	자료의 흐름(Data Flow)

▶ 소단위 명세서(Mini-Spec)

세분화된 자료 흐름도에서 최하위 단계 버블(프로세스)의 처리 절차를 기술한 것

- 프로세스 명세서

▶ CASE(Computer Aided Software Engineering)

소프트웨어 개발 과정 일부 또는 전체를 자동화하기 위한 도구

- **분류** : 상위(Upper) CASE, 하위(Lower) CASE, 통합(Integrate) CASE
- **CASE의 원천 기술**
 - 구조적 기법
 - 프로토타이핑 기술
 - 자동 프로그래밍 기술
 - 정보 저장소 기술
 - 분산처리 기술
- **대표적인 요구사항 분석 CASE 종류** : SADT, SREM, PSL/PSA, TAGS

▶ HIPO(Hierarchy Input Process Output)

시스템 분석, 설계, 문서화에 사용되는 하향식 소프트웨어 개발을 위한 도구

- **HIPO Chart의 종류**

- 가시적 도표(Visual Table of Contents, 도식 목차)
- 총체적 다이어그램(Overview Diagram, 총괄 도표, 개요 도표)
- 세부적 다이어그램(Detail Diagram, 상세 도표)

▶ N-S Chart(Nassi-Shneiderman Chart)

순서도와는 달리 논리 기술에 중점을 두고 상자 도형을 이용한 도형식 설계 도구




- 박스 다이어그램

▶ UML(Unified Modeling Language, 통합 모델링 언어)

객체 지향 소프트웨어 개발 과정에서 산출물을 명세화, 시각화, 문서화할 때 사용되는 모델링 기술과 방법론을 통합하여 만든 표준화된 범용 모델링 언어

- **구성 요소** : 사물(Things), 관계(Relationship), 다이어그램(Diagram)

▶ 관계(Relationship)

연관 관계 (Association)		<ul style="list-style-type: none"> 클래스(사물)들이 개념상 서로 연결되었음을 나타낸다. 한 클래스가 다른 클래스에서 제공하는기능을 사용할 때를 나타낸다.
직접 연관 관계 (Directed Association)		<ul style="list-style-type: none"> 클래스들이 개념상 서로 강하게 연결되 었음을 나타낸다. 한 클래스가 다른 클래스에서 제공하는 기능을 사용할 때를 나타낸다.
집합 관계, 집약 관계		<ul style="list-style-type: none"> 클래스들 사이의 전체 (Whole)또는 부분 (Part)

(Aggregation)		<p>같은 관계를 나타낸다.</p> <ul style="list-style-type: none"> 전체 객체의 라이프 타임과 부분 객체의 라이프 타임은 독립적이다. 즉 전체 객체가 없어져도 부분 객체는 없어지지 않는다.
합성 관계, 복합 관계, 포함 관계 (Composition)	—◆	<ul style="list-style-type: none"> 클래스들 사이의 전체 (Whole) 또는 부분 (Part) 같은 관계를 나타낸다. 전체 객체의 라이프 타임과 부분 객체의 라이프 타임은 의존적이다. 즉 전체 객체가 없어지면 부분 객체도 없어진다.
의존 관계 (Dependency)	→	<ul style="list-style-type: none"> 연관 관계와 같이 한 클래스가 다른 클래스에서 제공하는 기능을 사용할 때를 나타낸다. 차이점은 두 클래스의 관계가 한 메소드를 실행하는 동안 매우 짧은 시간만 유지되며, 클래스가 변경되면 다른 클래스에 영향을 준다.
일반화 관계, 상속 관계 (Generalization)	→	<ul style="list-style-type: none"> 한 클래스(Parent)가 다른 클래스(Child)를 포함하는 상위 개념 관계임을 나타낸다. 일반화된 객체와 좀 더 특수화된 객체 사이의 관계를 의미한다.
구현 관계, 실체화 관계 (Realization, Interface Realization)	→	<ul style="list-style-type: none"> 책임들의 집합인 인터페이스와 이 책임들을 실제로 실현한 클래스들 사이의 관계를 나타낸다. 한 객체가 다른 객체에게 동작(Operation)을 수행하도록 지정하는 의미적 관계이다.

▶ 다이어그램(Diagram)

• 구조 다이어그램(Structural UML Diagrams)

정적인 부분, 시스템의 구조를 나타내기 위한 다이어그램

- 클래스 다이어그램(Class Diagram)
- 객체 다이어그램(Object Diagram)
- 패키지 다이어그램(Package Diagram)
- 컴포넌트 다이어그램(Component Diagram)
- 복합 구조 다이어그램(Composite Structure Diagram)
- 배치 다이어그램(Deployment Diagram)
- 프로파일 다이어그램(Profile Diagram)

• 행위 다이어그램(Behavioral UML Diagrams)

동적인 부분, 시스템의 행위를 나타내기 위한 다이어

그램

- 유스케이스 다이어그램(Usecase Diagram)
- 활동 다이어그램(Activity Diagram)
- 상태 다이어그램(State Diagram)
- 시퀀스(순차) 다이어그램(Sequence Diagram)
- 통신 다이어그램(Communication Diagram)
- 타이밍 다이어그램(Timing Diagram)
- 상호작용 개요 다이어그램(Interaction Overview Diagram)

▶ UML을 이용한 모델링

• 기능 모델링 : 시스템이 제공할 기능을 표현

- 사용자 관점의 요구 기능
- 종류 : 유스케이스 다이어그램, 활동 다이어그램 등

• 정적 모델링 : 시스템 내부 구성 요소를 표현

- 요구 기능 구현을 위한 개발자 관점
- 종류 : 클래스 다이어그램, 패키지 다이어그램 등

• 동적 모델링 : 시스템 내부 구성 요소의 상태가 시간의 흐름에 따라 변화하는 과정과 변화하는 과정에서 발생하는 상호 작용을 표현

- 종류 : 상태 다이어그램, 시퀀스 다이어그램, 통신 다이어그램

▶ 유스케이스 다이어그램(UseCase Diagram)

사용자 관점에서 시스템이 제공하는 기능 및 그와 관련한 외부 요소를 표현한 다이어그램

- **구성 요소** : 시스템(System), 액터(Actor), 유스케이스(Usecase), 관계(Relationship)
- **관계** : 연관(association), 포함(include), 확장(extend), 일반화(generalization)

▶ 클래스 다이어그램(Class Diagram)

시스템을 구성하는 클래스와 클래스가 가지는 속성, 메소드, 클래스 사이의 관계를 표현한 다이어그램

- **구성 요소** : 클래스(Class), 다중성(Multiplicity), 접근 제한자(Access Modifier, 접근 제어자), 관계(Relationship)

▶ 시퀀스 다이어그램(Sequence Diagram)

상호 작용하는 시스템이나 시스템 내부 객체 간에 주고받는 메시지를 시간의 흐름에 따라 표현한 다이어그램

- **구성 요소** : 액터(Actor), 객체(Object), 메시지(Message), 생명선(Lifeline), 활성화 상자(Activation Box, 실행), 프레임(Frame)

▶ 패키지 다이어그램(Package Diagram)

요소들을 그룹화한 패키지들의 관계를 계층적 구조로 표현한 다이어그램

- **구성 요소** : 패키지(Package), 의존 관계(import,

access)

▶ UI(User Interface, 사용자 인터페이스)

사용자와 컴퓨터 상호 간의 소통을 원활하게 도와주는 장치 또는 소프트웨어

• 설계 원칙

직관성	누구나 쉽게 이해하고 사용할 수 있어야 한다.
유효성	사용자의 목적을 정확하게 달성해야 한다.
학습성	누구나 쉽게 배우고 익힐 수 있어야 한다.
유연성	사용자의 요구사항을 최대한 수용하며, 오류를 최소화해야 한다.

• 종류

- CLI(Command Line Interface)
- GUI(Graphical UI)
- NUI(Natural UI)
- VUI(Voice UI)
- OUI(Organic UI)

▶ UX(User eXperience,

사용자 경험, 사용자 인식 반응 경험)

사용자가 어떤 시스템, 제품, 서비스를 직·간접적으로 이용하면서 느끼고 생각하게 되는 지각, 반응, 행동 등의 총체적 경험

▶ 감성공학(Human Sensibility Ergonomics)

인체의 특징과 감성을 제품 설계에 최대한 반영시키는 기술

- HCI 설계에 인간의 특성과 감성을 반영

• **HCI(Human Computer Interaction or Interface)** : 인간과 컴퓨터 간의 상호 작용에 관한 연구

▶ UI 설계 도구

와이어프레임(Wireframe)	<ul style="list-style-type: none"> • 화면 단위의 레이아웃을 설계하는 작업 • UI 요소 등에 대한 뼈대
목업(Mockup)	<ul style="list-style-type: none"> • 실물과 흡사한 정적인 형태의 모형
스토리보드(Storyboard)	<ul style="list-style-type: none"> • 디자이너와 개발자의 의사소통을 위한 도구 • 콘텐츠의 설명 및 페이지 간의 이동 흐름 등을 시각화한 문서 • 서비스를 위한 대부분의 정보가 수록
프로토타입(Prototype)	<ul style="list-style-type: none"> • 정적인 화면에 동적 효과를 적용함으로써, 시뮬레이션이 가능한 동적 모형 • 전체적인 기능을 간략한 형태로 구현한 시제품

▶ UI 요소

텍스트 박스(Text Box, Input Box)	사용자로부터 텍스트를 입력 받을 수 있는 상자
라디오 버튼(Radio Button)	하나만 선택 가능한 버튼
체크 박스(Check Box)	한 개 또는 여러 개의 선택이 동시에 가능한 버튼
콤보 박스(Combo Box)	선택 목록을 만드는 상자
토글 버튼(Toggle Button)	하나의 설정 값으로부터 다른 값으로 전환하는 버튼

▶ 객체 지향(Object-Oriented)

- 현실 세계의 실체(Entity, 개체)를 속성과 메소드가 결합된 독립적인 형태의 객체(Object)로 표현하는 개념
- 객체들이 메시지를 통해 상호 작용함으로써 전체 시스템이 운영되는 개념

• 구성 요소

객체(Object)	<ul style="list-style-type: none"> • 속성과 그 속성에 관련되는 메소드를 모두 포함한 개념 • 클래스의 인스턴스(Instance)
클래스(Class)	<ul style="list-style-type: none"> • 공통된 속성과 메소드를 갖는 객체의 집합 • 유사한 객체들을 묶어 공통된 특성을 표현한 데이터를 추상화하여 모델링한 것
메시지(Message)	<ul style="list-style-type: none"> • 객체 간에 상호작용을 하는 데 사용되는 수단

• 특징

캡슐화(Encapsulation)	<ul style="list-style-type: none"> • 속성과 메소드를 하나로 묶어 객체로 구성하는 것 • 실제 구현 내용 일부를 외부에 감추어 은닉하는 것
정보은폐(Information Hiding, 정보은닉)	<ul style="list-style-type: none"> • 다른 객체로부터 자신의 자료를 숨기고 자신의 연산만을 통하여 접근을 허용하는 것
추상화(Abstraction)	<ul style="list-style-type: none"> • 복잡한 문제의 본질을 이해하기 위해 세부 사항은 배제하고 중요한 부분을 중심으로 간략화하는 것 • 데이터의 공통되는 속성이나 메소드를 묶어서 추출하는 것 • 유형 : 과정(Procedure) 추상화, 자료(Data) 추상화, 제어(Control) 추상화
상속(Inheritance)	<ul style="list-style-type: none"> • 상위 클래스의 속성과 메소드를 하위 클래스가 물려받는 것
다형성(Polymorphism)	<ul style="list-style-type: none"> • 한 메시지가 객체에 따라 다른 방법으로 응답할 수 있는 것

▶ 객체 지향 설계 원칙(S.O.L.I.D)

• 단일 책임 원칙

(SRP; Single-Responsibility Principle)

- 클래스를 변경해야 하는 이유는 오직 하나여야 한다.
- 클래스는 한 가지 기능만 갖도록 설계한다.

• 개방 폐쇄의 원칙

(OCP; Open-Closed Principle)

- 확장에는 열려 있어야 하고, 변경에는 닫혀있어야 한다.
- 클래스를 확장은 쉽게, 변경은 어렵게 설계한다.

• 리스코프 교체의 원칙

(LSP; Liskov Substitution Principle)

- 기반 클래스(상위 클래스)는 파생 클래스(하위 클래스)로 대체할 수 있어야한다.
- 상위 타입 객체를 하위 타입 객체로 치환해도 상위 타입을 사용하는 프로그램은 정상적으로 동작해야 한다.

• 인터페이스 분리의 원칙

(ISP; Interface Segregation Principle)

- 하나의 일반적인 인터페이스보다는 구체적인 여러 개의 인터페이스가 낫다.
- 클라이언트는 자신이 사용하지 않는 메소드와 의존 관계를 맺거나 영향을 받으면 안 된다.

• 의존 관계 역전의 원칙

(DIP; Dependency Inversion Principle)

- 하나의 일반적인 인터페이스보다는 구체적인 여러 개의 인터페이스가 낫다.
- 클라이언트는 자신이 사용하지 않는 메소드와 의존 관계를 맺거나 영향을 받으면 안 된다.

▶ 객체 지향 분석 방법론

부치(Booch) 기법	자료 흐름도(DFD)를 사용해서 객체를 분해하고, 객체 간의 인터페이스를 찾아 이것들을 에이다(Ada) 프로그램으로 변환시키는 기법
코드(Coad)와 요돈(Yourdon) 기법	개체 관계도(ERD)를 사용하여 개체의 활동들을 데이터 모델링 하는 데 초점을 둔 기법
워프-브록(Wirfs-Brock) 기법	분석과 설계 간 구분이 없고, 고객 명세서를 평가하여 설계 작업까지 연속적으로 수행하는 기법
제이콥슨(Jacobson) 기법	유스케이스를 사용하여 분석하는 기법
럼바우(Rumbaugh)의 분석 기법	그래픽 표기법을 이용하여 소프트웨어 구성 요소를 모델링한 기법

▶ 럼바우(Rumbaugh)의 분석 기법

그래픽 표기법을 이용하여 소프트웨어 구성 요소를 모델링한 기법

- 객체 모델링 기법(OMT)

• 분석 활동 순서 : 객체 모델링 → 동적 모델링 → 기능 모델링

객체 모델링 (Object Modeling)	<ul style="list-style-type: none"> • 정보 모델링(Information Modeling) • 시스템에서 요구되는 객체를 찾아내어 객체들의 특성을 규명 • 객체 다이어그램 이용
동적 모델링 (Dynamic Modeling)	<ul style="list-style-type: none"> • 객체들의 제어 흐름, 상호 반응, 연산 순서를 나타내주는 과정 • 상태 다이어그램(STD) 이용한
기능 모델링 (Functional Modeling)	<ul style="list-style-type: none"> • 각 객체에서 수행되는 동작들을 기술 • 자료 흐름도(DFD) 이용

▶ 소프트웨어 설계

• 종류

- 상위 설계 : 아키텍처(구조) 설계, 데이터 설계, 인터페이스 정의, 사용자 인터페이스(UI) 설계
- 하위 설계 : 모듈 설계, 자료구조 서설계, 알고리즘 설계

• 원리

- 분할과 정복(Divide and Conquer)
- 추상화(Abstraction)
- 단계적 분해(Stepwise Refinement)
- 모듈화(Modularization)
- 정보 은닉(Information Hiding)

▶ 소프트웨어 아키텍처(Software Architecture)

소프트웨어의 골격이 되는 기본 구조로, 복잡한 개발을 체계적으로 접근하기 위한 밑그림

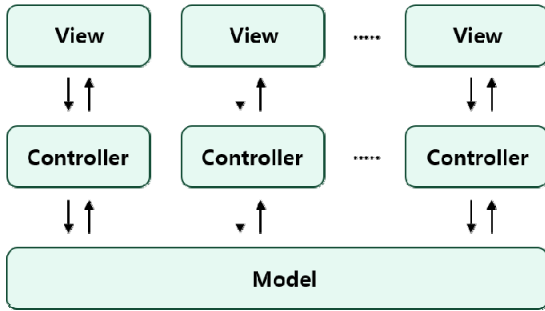
• 품질 속성

시스템 측면	성능, 보안, 가용성, 기능성, 사용성, 변경 용이성, 확장성 등
비즈니스 측면	시장 적시성, 비용과 혜택, 예상 시스템 수명 등
아키텍처 측면	개념적 무결성, 정확성, 구축 가능성 등

• 패턴(스타일) 종류 : MVC 구조, 클라이언트/서버 구조, 저장소 구조(데이터 중심형 모델), 계층 구조, 데이터 흐름 구조(파이프 필터 구조), 마스터-슬레이브 구조

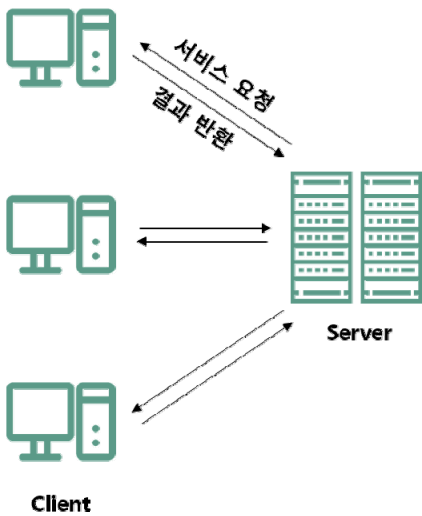
▶ MVC 구조(Model, View, Controller)

구현하려는 전체 어플리케이션을 모델(Model), 뷰(View), 컨트롤러(Controller)로 구분하고, 사용자 인터페이스와 비즈니스 로직을 서로 분리하여 개발하는 방법



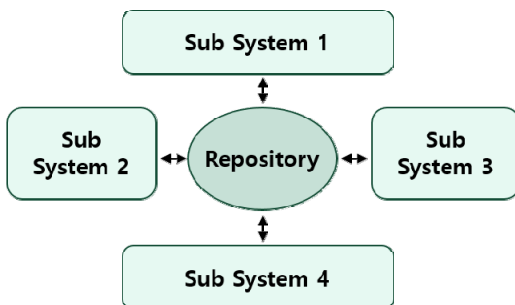
▶ 클라이언트/서버 구조(Client/Server)

네트워크를 이용하는 분산 시스템 형태의 모델로, 정보를 요청하는 클라이언트와 정보를 제공하는 서버를 분할하여 사용하는 방법



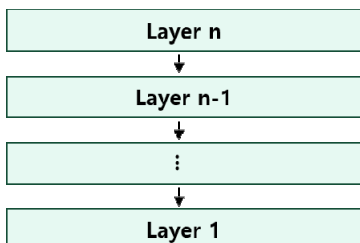
▶ 저장소 구조(Repository, 데이터 중심형 모델)

공동으로 활용하는 데이터를 저장소에 보관하고, 모든 서브 시스템이 저장된 공유 데이터에 접근하여 저장, 검색, 변경하는 역할을 하는 방법



▶ 계층 구조(Layering)

시스템을 계층으로 구분하여, 계층 하나를 서브 시스템으로 생각하는 고전적인 방법



▶ 데이터 흐름 구조

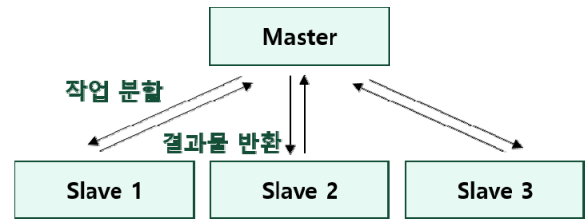
(Pipe Filter, 파이프 필터 구조)

필터에 해당되는 서브 시스템이 하나의 데이터를 입력으로 받아 처리한 후, 그 결과를 다음 서브 시스템으로 넘겨주는 과정을 반복하는 방법



▶ 마스터-슬레이브 구조(Master-Slave)

마스터는 슬레이브로 작업을 분할하고, 슬레이브에서 처리된 결과물을 마스터가 다시 돌려받는 방법



▶ 디자인 패턴(Design Pattern)

어떤 분야에서 반복적으로 나타나는 문제점들에 대한 전문가들의 경험을 정리하여 해결 방안을 제시한 패턴
- 유사한 문제를 해결하기 위해 설계들을 분류하고 각 문제 유형별로 가장 적합한 설계를 일반화하여 체계적으로 정리해 놓은 것

- **구성 요소** : 패턴의 이름과 구분, 문제 및 배경, 해법, 사례, 결과, 샘플 코드

▶ GoF(Gang of Four) 디자인 패턴

에릭 감마(Erich Gamma), 리처드 헬름(Richard Helm), 랄프 존슨(Ralph Johnson), 존 블리시데스(John Vlissides)가 제안한 디자인 패턴

- 생성 패턴, 구조 패턴, 행위 패턴으로 구분하여 패턴을 설명

- **생성 패턴(Creational Pattern)** : 객체의 생성과 과정을 캡슐화하여 객체가 변경되어도 프로그램의 구조에 영향을 크게 받지 않도록 유연성을 더해주는 패턴

- 팩토리 메소드(Factory Method)
- 추상 팩토리(Abstract Factory)
- 빌더(Builder)
- 프로토타입(Prototype)
- 싱글톤(Singleton)

- **구조 패턴(Structural Pattern)** : 새로운 기능을 구현하기 위해 객체를 구성하는 방식에 초점을 두어, 클래스나 객체들을 조합하여 더 큰 구조로 만들 수 있게 해주는 패턴

- 어댑터(Adapter)
- 브리지(Bridge)
- 컴포지트(Composite)

- 데코레이터(Decorator)
- 퍼사드(Facade)
- 플라이웨이트(Flyweight)
- 프록시(Proxy)
- **행위 패턴(Behavior Pattern)** : 하나의 객체로 수행할 수 없는 작업을 여러 객체로 분배하여, 결합도를 최소화하도록 클래스나 객체들이 상호 작용하는 방법이나 책임 분배 방법을 정의하는 패턴
 - 인터프리터(Interpreter)
 - 템플릿 메소드(Template Method)
 - 책임 연쇄(Chain of Responsibility)
 - 커맨드(Command)
 - 이터레이터(Iterator)
 - 미디에이터(Mediator)
 - 메멘토(Memento)
 - 옵저버(Observer)
 - 스테이트(State)
 - 스트레티지(Stratgy)
 - 비지터(Visitor)

▶ 모듈과 모듈화

- **모듈(Module)** : 소프트웨어 구조를 이루는 기본적인 단위
 - 작업 단위(단위 모듈), 소프트웨어 내 프로그램, 부시스템, 서브루틴 등
- **모듈화(Modularity)** : 소프트웨어 기능들을 모듈 단위로 분해하는 것
 - 모듈화 측정 척도 : 응집도(Cohesion), 결합도(Coupling)

▶ 응집도(Cohesion)

- 모듈 안의 요소들이 서로 관련되어 있는 정도
- 독립적인 모듈이 되기 위해서는 응집도가 강해야 함

• 종류

기능적 응집도 (Functional Cohesion)	단일 기능의 요소로 하나의 모듈을 구성한 경우의 응집도
순차적 응집도 (Sequential Cohesion)	요소1의 출력을 요소2의 입력으로 사용하는 두 요소가 하나의 모듈을 구성한 경우
교환(통신)적 응집도 (Communication Cohesion)	동일한 입력과 출력을 사용하여 서로 다른 기능을 수행하는 구성 요소들이 모여 하나의 모듈을 구성한 경우의 응집도
절차적 응집도 (Procedural Cohesion)	모듈이 다수의 관련 기능을 가질 때 모듈 안의 구성 요소들이 그 기능을 순차적으로 수행할 경우의 응집도
시간적 응집도 (Temporal Cohesion)	연관된 기능보다는 특정 시간에 처리되어야 하는 활동들을 한 모듈에서 처리하는 경우의 응집도

논리적 응집도 (Logical Cohesion)	유사한 성격을 갖거나 특정 형태로 분류되는 처리 요소들이 한 모듈에서 처리되는 경우의 응집도
우연적 응집도 (Coincidental Cohesion)	모듈 내부의 각 구성 요소들이 연관이 없을 경우의 응집도

▶ 결합도(Coupling)

모듈 간의 상호 의존도

- 독립적인 모듈이 되기 위해서는 결합도가 약해야 함

• 종류

자료 결합도 (Data Coupling)	모듈 간의 인터페이스로 전달되는 매개변수를 통해서만 모듈 간의 상호 작용이 일어나는 경우의 결합도
스탬프 결합도 (Stamp Coupling)	모듈 간의 인터페이스로 배열이나 객체, 자료 구조 등이 전달되는 경우의 결합도
제어 결합도 (Control Coupling)	단순 처리할 대상인 값만 전달되는 게 아니라 어떻게 처리를 해야 한다는 제어 요소(Flag, 플래그)가 전달되는 경우의 결합도
외부 결합도 (External Coupling)	다른 모듈 내부에 있는 데이터(변수)를 외부의 다른 모듈에서 참조하는 경우의 결합도
공통 결합도 (Common Coupling)	파라미터가 아닌 모듈 밖에 선언되어 있는 공통 데이터 영역을 참조 및 갱신하는 식으로 상호 작용하는 경우의 결합도
내용 결합도 (Content Coupling)	다른 모듈 내부에 있는 기능이나 자료를 외부의 다른 모듈에서 직접 참조하거나 갱신하는 경우의 결합도

▶ 공유도와 제어도

- **공유도(Fan-In, 팬인)** : 자신을 사용하는 모듈의 수
 - 상위 모듈의 개수
 - Fan-In이 높으면 재사용성 우수
- **제어도(Fan-Out, 팬아웃)** : 자신이 호출하는 모듈의 수
 - 하위 모듈의 개수
 - Fan-Out이 낮으면 재사용성 우수

▶ 공통 모듈

정보시스템 구축 시 자주 사용하는 기능들로서, 재사용이 가능하게 패키지로 제공하는 독립된 모듈

• 명세 기법

정확성 (Correctness)	시스템 구현 시 해당 기능이 필요하다는 것을 알 수 있도록 정확히 작성한다.
명확성 (Clarity)	해당 기능을 이해할 때 중의적으로 해석되지 않도록 명확하게 작성한다.

완전성 (Completeness)	시스템 구현을 위해 필요한 모든 것을 기술한다.
일관성 (Consistency)	공통 기능 간 상호 충돌이 발생하지 않도록 작성한다.
추적성 (Traceability)	기능에 대한 요구사항의 출처, 관련 시스템 등의 관계를 파악할 수 있도록 작성한다.

- **재사용 범위에 따른 분류** : 함수와 객체 재사용, 컴포넌트 재사용, 애플리케이션 재사용

▶ **협약에 의한 설계(DBC; Design By Contract)**

소프트웨어 컴포넌트를 설계할 때 클래스에 대한 여러 가정을 공유하도록 명세한 것

- 계약 프로그래밍, 클래스 상세 설계

- **세 가지 타입** : 선행 조건(Precondition), 결과 조건(Postcondition, 후행 조건), 불변 조건(Invariant, 불변식)

▶ **코드(Code)**

컴퓨터에서 자료 처리를 쉽게 하고자 사용하는 기호

• **기능**

- 3대 기능 : 배열, 분류, 식별
- 기타 기능 : 표준화, 암호화, 확장성, 연상성(표의성), 단순화

• **코드의 종류**

순차 코드 (Sequence Code)	코드화 대상 항목을 어떤 일정한 배열로 일련번호를 부여하는 방법
블록 코드 (Block Code, 구분 코드)	공통성이 있는 것끼리 블록으로 구분하고, 각 블록 내에서 일련번호를 부여하는 방법
그룹 분류 코드 (Group Classification Code)	일정 기준에 따라 대분류, 중분류, 소분류 등으로 구분하여 일련번호를 부여하는 방법
10진 코드 (Decimal Code, 도서 분류 코드)	코드화 대상 항목을 0~9까지 10진 분할하고, 다시 그 각각에 대하여 10진 분할하는 방법
표의 숫자 코드 (Significant Digit Code)	코드화 대상 항목의 물리적 수치 등을 나타내는 문자, 숫자 혹은 기호를 그대로 코드로 사용하여 일련번호를 부여하는 방법
연상 코드 (Mnemonic Code)	코드화 대상의 명칭이나 약호를 코드의 일부에 넣어서 대상을 외우기 쉽도록 일련번호를 부여하는 방법

• **코드 오류의 종류**

필사 오류(Transcription Error, 오자 오류)	입력 시 임의의 한 자리를 잘못 기록한 경우
전위 오류 (Transposition Error)	입력 시 좌우 자리를 바꾸어 기록한 경우
생략 오류 (Omission Error)	입력 시 한 자리를 빼놓고 기록한 경우

추가 오류 (Addition Error)	입력 시 한 자리를 추가로 기록한 경우
이중 전위 오류 (Double Transposition Error)	전위 오류가 중복 발생한 경우
임의 오류 (Random Error)	오류가 두 가지 이상 결합하여 발생한 경우

▶ **직접 연계 방식**

중간 매개체 없이 송신 시스템과 수신 시스템을 직접 연계하는 방식

- **종류** : DB Link, API, 화면 링크, DB Connection Pool, JDBC

▶ **간접 연계 방식**

송신 시스템과 수신 시스템 사이에 중간 매개체를 활용하여 연계하는 방식

- **종류** : EAI, ESB, 웹 서비스, 소켓

▶ **EAI(Enterprise Application Integration,**

기업 애플리케이션 통합 솔루션)

기업에서 운영되는 서로 다른 애플리케이션 및 플랫폼 간의 정보 전달, 연계, 통합 등 상호 연동을 가능하게 해주는 솔루션

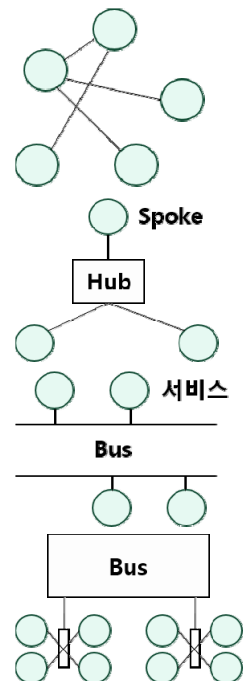
• **구축 유형**

- Point-to-Point(PPP) : 두 대의 컴퓨터가 직렬 인터페이스를 이용하여 통신할 때 사용하는 방식

- Hub & Spoke : 단일 접점인 허브 시스템을 통해 데이터를 전송하는 중앙 집중형 방식

- Message Bus(ESB 방식) : 애플리케이션 사이에 미들웨어를 두어 처리하는 방식

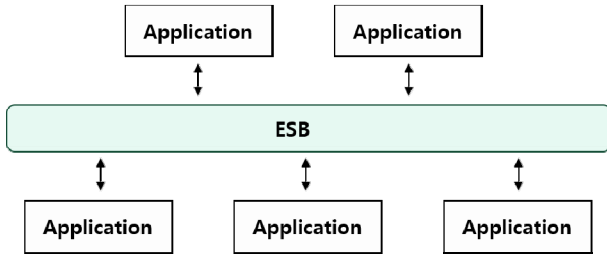
- Hybrid : 그룹 내에서는 Hub & Spoke 방식으로 연결하고, 그룹 간에는 Message Bus 방식으로 사용하는 방식



▶ **ESB(Enterprise Service Bus)**

애플리케이션 간 연계, 데이터 변환, 웹 서비스 지원, 자원 연결 및 통합 등 표준 기반의 인터페이스를 제공하는 솔루션

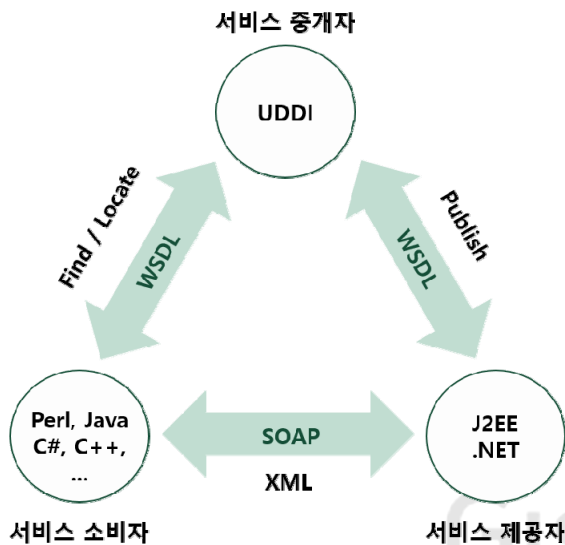
- EAI에 표준화와 분산화를 결합한 것
- 서비스 중심의 통합을 지향



▶ 웹 서비스(Web Service)

네트워크상에서 서로 다른 종류의 컴퓨터들 간에 상호 작용을 하기 위한 소프트웨어 시스템

- 웹 서비스는 UDDI, SOAP, WSDL 프로토콜을 이용하여 시스템 간의 연계 제공



- **UDDI(Universal Description, Discovery and Integration)** : 웹 서비스에 대한 정보를 등록하고 검색하기 위한 저장소
 - 공용 등록부 서비스
- **SOAP(Simple Object Access Protocol)** : XML 기반의 메시지를 교환하기 위한 통신 규약
 - 기본적으로 HTTP 기반에서 동작
- **WSDL(Web Service Description Language)** : 웹 서비스와 관련된 포맷이나 프로토콜 등을 표준적인 방법으로 기술하고 게시하기 위한 언어 또는 파일
 - XML로 작성되었으며, UDDI의 기초가 되는 언어

▶ 소켓(Socket)

네트워크 통신에서 데이터를 송·수신할 수 있는 통신 접속점

- 소켓을 생성하여 포트를 할당하고, 클라이언트의 요청을 연결하여 통신
- 네트워크 프로그램의 기반 기술

▶ 미들웨어(Middleware)

운영체제(Server)와 응용 프로그램(Client) 사이에 위치하는 컴퓨터 소프트웨어

- 응용 프로그램에 운영체제가 제공하는 서비스를 추가

및 확장하여 제공하는 역할

• 종류

RPC(Remote Procedure Call, 원격 프로시저 호출)	응용 프로그램의 프로시저를 사용하여 원격 프로시저를 마치 로컬 프로시저처럼 호출하는 방식의 미들웨어
MOM(Message Oriented Middleware, 메시지 지향 미들웨어)	이기종 시스템 간의 통신을 비동기 방식으로 지원하는 메시지 기반 미들웨어
TP-Monitor (Transaction Processing Monitor, 트랜잭션 처리 모니터)	최소 처리 단위인 트랜잭션을 감시하여 일관성 있게 보관 및 유지하고, 트랜잭션의 완벽한 처리를 보장하기 위한 역할을 하는 트랜잭션 관리 미들웨어
ORB(Object Request Broker, 객체 요청 브로커)	객체 지향 미들웨어로, CORBA 표준 스펙을 구현한 미들웨어
WAS(Web Application Server, 웹 애플리케이션 서버)	사용자 또는 사용자의 요청에 따라 결과 값이 변하는 동적 콘텐츠를 처리하기 위한 웹 환경을 구현하는데 사용되는 미들웨어

2과목

소프트웨어 개발

▶ 선형 검색(순차 검색, Full Table Scan)

모든 레코드를 대상으로 순차적으로 검색

▶ 이진 검색(이분 검색, Binary Search)

중간 값을 비교하여 검색

▶ 인덱스(Index) 검색

검색의 기준이 되는 칼럼을 뽑아 인덱스로 지정하여 검색

▶ 해싱(Hashing) 검색

해싱 함수를 사용하여 검색

• 해싱 관련 용어

해싱 함수	해시 테이블의 주소를 생성해 내는 함수
해시 테이블	해싱 함수에 의하여 참조되는 테이블
버킷(Bucket)	하나의 주소를 갖는 파일의 한 구역
슬롯(Slot)	n개의 슬롯이 모여 하나의 버킷을 형성
충돌(Collision)	서로 다른 2개 이상의 레코드가 같은 주소를 갖는 현상
시노님(Synonym)	같은 주소를 갖는 레코드의 집합
오버플로(Overflow)	버킷 내 기억 공간이 없는 현상

• 해싱 함수 종류

제산법(Division)	키를 임의의 양의 정수로 나눈 나머지를 그 키의 레코드 주소로 결정하는 방법
폴딩(Folding)	키를 여러 부분으로 나누고, 나누어진 각 부분의 값을 모두 더하거나 보수(XOR)를 취한 결과 값을 레코드 주소로 결정하는 방법
계수 분석(Digit Analysis, 숫자 분석)	키값을 구성하는 숫자의 분포를 파악하여 균등한 분포의 숫자를 선택하여 레코드 주소를 결정하는 방법
제곱법(Mid-Square)	키값을 제곱한 값의 중간 부분 값을 선택하여 레코드 주소로 결정하는 방법
기수 변환(Radix Transformation)	주어진 키값을 다른 진법으로 변환하여 얻은 결과 값을 레코드 주소로 결정하는 방법

▶ 자료 구조(Data Structure)

자료를 저장하는 논리적인 방법

- **선형 구조(Linear Structure)** : 일정한 순서에 의해 데이터를 순차적으로 하나씩 나열시킨 구조
 - 종류 : 배열(Array), 선형 리스트(Linear List), 스택(Stack), 큐(Queue), 덱(Deque)
- **비선형 구조(Non-Linear Structure)** : 하나의 데이터 뒤에 여러 개의 데이터가 존재할 수 있는 구조
 - 종류 : 트리(Tree), 그래프(Graph)

▶ 배열(Array)

동일한 크기와 형식(Type)으로 구성된 연속적인 기억 공간을 가지는 자료 구조

1	2	3	4
a[0]	a[1]	a[2]	a[3]

▶ 선형 리스트(Linear List)

일정한 순서에 의해 나열된 자료 구조

- **연속 리스트(Contiguous List)** : 배열과 같이 빈 공간 없이 연속되는 기억 공간에 저장되는 자료 구조

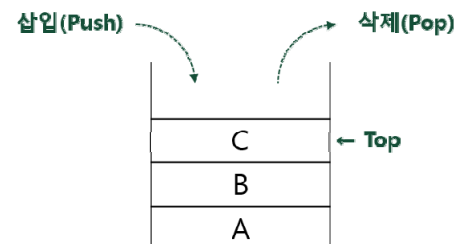
1	가
2	나
3	다
4	라

- **연결 리스트(Linked List)** : 데이터의 저장 순서는 상관없지만, 데이터 항목의 순서에 따라 각 노드에 포인터를 두어 서로 연결시키는 자료 구조

	data	link	
1	가	5	5 기사 100
11	나	16	16 나비 110
21	다		
31	라		
			110 농사

▶ 스택(Stack)

리스트 한쪽에서만 삽입과 삭제가 이루어지는 후입선출(LIFO; Last In First Out) 형식의 자료 구조



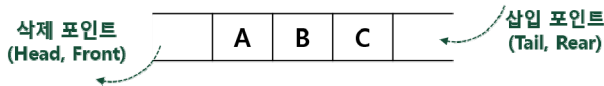
• 스택(Stack)의 응용 분야

- 부 프로그램 호출 시 복귀 주소를 저장할 때(함수 호출의 순서 제어)

- 인터럽트가 발생하여 복귀 주소를 저장할 때
- 후위 표기법(Post-fix expression)으로 표현된 수식을 연산할 때
- 0-주소 명령어의 자료 저장소
- 재귀(Recursive) 프로그램의 순서 제어
- 컴파일러를 이용한 언어 번역
- 깊이 우선 탐색(DFS; Depth First Search)

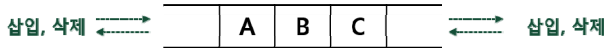
▶ 큐(Queue)

한쪽에는 반드시 삽입, 또 다른 한쪽에는 반드시 삭제가 이루어지는 선입선출(FIFO; First In First Out) 형식의 자료 구조



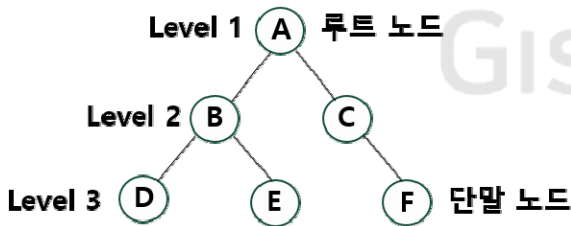
▶ 덱(Deque)

스택과 큐의 장점을 조합한 구조로 양쪽 모두 삽입, 삭제가 가능한 자료 구조



▶ 트리(Tree)

정점(Node, 노드)과 가지(Branch, 링크, 간선)를 이용한 사이클(Cycle)이 이루어지지 않는 자료 구조



• 트리 관련 용어

노드(Node)	트리를 구성하는 기본 원소 • 예 A, B, C, D, E, F
가지(Branch, 링크, 간선)	노드와 노드 간의 연결선
차수(Degree)	특정 노드의 자식 수 • 예 B의 차수 : 2
트리의 차수	트리의 모든 노드 중에 가장 높은 차수 • 예 2
깊이(Depth)	루트에서 특정 노드 사이의 가지 개수 • 예 F의 깊이 : 2
높이(Height)	트리가 가지는 최대 레벨 • 예 3
루트 노드 (Root Node)	트리 구조상 가장 최상위 노드 • 예 A
단말 노드 (Terminal node) = 리프 노드 (Leaf Node)	자식이 없는 노드 • 예 D, E, F

자식 노드 (Son Node)	특정 노드의 하위 노드 • 예 B의 자식 노드 : D, E
부모 노드 (Parent Node)	특정 노드의 상위 노드 • 예 C의 부모 노드 : A
형제 노드 (Brother Node, Sibling)	동일한 부모를 가지는 노드 • 예 D의 형제 노드 : E

▶ 이진 트리(Binary Tree)

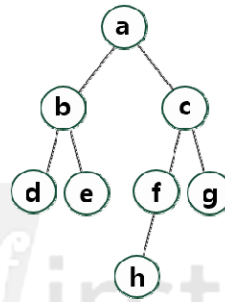
각각의 노드가 최대 두 개의 자식 노드를 가지는 트리

- 종류 : 정(Full) 이진 트리, 포화(Perfect) 이진 트리, 완전(Complete) 이진 트리, 편향(Skewed) 이진 트리

▶ 이진 트리 운행법

- Preorder(전위) : Root → Left → Right
- Inorder(중위) : Left → Root → Right
- Postorder(후위) : Left → Right → Root

예 이진 트리 운행법

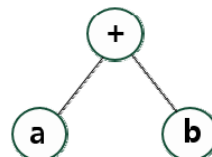


- Preorder(전위) : a, b, d, e, c, f, h, g
- Inorder(중위) : d, b, e, a, h, f, c, g
- Postorder(후위) : d, e, b, h, f, g, c, a

▶ 수식 표기법 변환

- PreFix(전위 표기법) : 연산자 → Left 피연산자 → Right 피연산자
- InFix(중위 표기법) : Left 피연산자 → 연산자 → Right 피연산자
- PostFix(후위 표기법) : Left 피연산자 → Right 피연산자 → 연산자

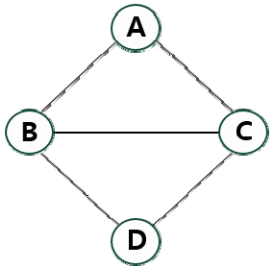
예 수식 표기법 변환



- PreFix(전위 표기법) : + a b
- InFix(중위 표기법) : a + b
- PostFix(후위 표기법) : a b +

▶ 그래프(Graph)

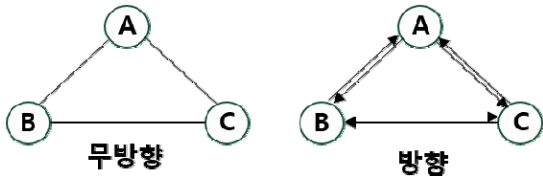
정점(V; Vertex)과 간선(E; Edge)의 두 집합으로 이루어진 사이클(Cycle)이 있는 자료 구조



• 최대 간선 수

- 정점이 n 개인 무방향 그래프에서 최대 간선 수 : $n(n-1)/2$ 개
- 정점이 n 개인 방향 그래프에서 최대 간선 수 : $n(n-1)$ 개

예 최대 간선 수

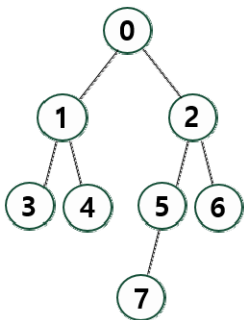


- 무방향 그래프 : $3(3-1)/2 = 3$ 개
- 방향 그래프 : $3(3-1) = 6$ 개

▶ 그래프 탐색

- 깊이 우선 탐색(DFS; Depth First Search) : 노드의 자식들을 우선으로 탐색하는 방법
- 너비 우선 탐색(BFS; Breadth First Search) : 노드의 인접한 모든 정점들을 우선으로 탐색하는 방법

예 그래프 탐색



- DFS 탐색 순서 : $0 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 7 \rightarrow 6$
- BFS 탐색 순서 : $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$

▶ 알고리즘(Algorithm)

주어진 문제를 풀기 위한 절차나 방법

- 조건 : 입력, 출력, 명확성, 유한성, 효과성
- 설계 기법

분할 정복법 (Divide and Conquer)	방대한 문제를 조금씩 나뉘가면서 쉽게 풀 수 있는 문제 단위로 나눈 다음, 그것들을 다시 합쳐서 해결하는 알고리즘 설계 기법
--------------------------------	---

동적 계획법 (DP; Dynamic Programming)	특정 범위까지의 값을 구하기 위해서 그것과 다른 범위까지의 값을 이용하여 효율적으로 값을 구하는 알고리즘 설계 기법
그리디 알고리즘 (Greedy Algorithm, 욕심쟁이 알고리즘)	'매 선택에서 지금 이 순간 당장 최적인 답을 선택하여 적합한 결과를 도출하자'라는 모토를 가지는 알고리즘 설계 기법
백트래킹 (Backtracking)	모든 경우의 수를 전부 고려하는 해결 방법으로, 상태 공간을 트리로 나타낼 수 있을 때 적합한 알고리즘 설계 기법

▶ 알고리즘 효율성 평가

알고리즘의 복잡한 정도를 나타내는 척도인 시간 복잡도와 공간 복잡도로 알고리즘의 효율성을 평가

- 시간 복잡도(Time Complexity) : 프로그램을 실행시켜 완료하는 데 걸리는 시간을 계산한 것
 - 얼마나 많은 시간이 필요한가?
- 공간 복잡도(Space Complexity) : 기억 공간의 소요량을 계산한 것이다.
 - 얼마나 많은 메모리 공간이 필요한가?
- Big-O 표기법(Big O Notation)

$O(n)$	입력 자료를 차례로 하나씩 모두 처리하는 선형 복잡도 • 예 순차 검색
$O(n^2)$	제곱형 복잡도 • 예 선택 정렬, 버블 정렬, 삽입 정렬
$O(1)$	상수형 복잡도 • 예 해시 함수
$O(\log_2 n)$	로그형 복잡도 • 예 이진 검색
$O(n \log_2 n)$	선형 로그형 복잡도 • 예 퀵 정렬, 힙 정렬, 병합(합병) 정렬

▶ 정렬 알고리즘

선택 정렬 (Selection Sort)	자료 배열 중에 최솟값(또는 최댓값)을 찾아 그 값을 첫 번째 위치에 놓고, 첫 번째 위치를 제외한 나머지 자료 배열 중에서 최솟값을 찾아 두 번째 위치에 놓는 과정을 반복하는 정렬 방법
버블 정렬 (Bubble Sort)	자료 배열 중 인접한 두 요소를 비교하여 교체하는 정렬 방법
삽입 정렬 (Insertion Sort)	자료 배열의 모든 요소를 앞에서부터 차례대로 이미 정렬된 배열 부분과 비교하여, 자신의 위치를 찾아 삽입하는 정렬 방법
퀵 정렬 (Quick Sort)	기준점(Pivot)을 기준으로 좌우를 비교하여 정렬하는 방법
2-Way 병합 정렬 (합병 정렬)	자료 배열을 균등한 크기로 분할하고 분할된 부분 자료 배열을 정

Merge Sort)	렬한 다음, 두 개의 정렬된 부분 자료 배열을 병합하는 과정을 반복하여 정렬하는 방법
힙 정렬 (Heap Sort)	완전 이진 트리의 일종으로 자료 배열로 힙(Heap)을 구성하고, 가장 큰 값을 갖는 루트 노드를 제거하는 과정을 반복하여 정렬하는 방법

▶ 소프트웨어 테스트

구현된 응용 애플리케이션이나 시스템이 사용자가 요구하는 기능의 동작과 성능, 사용성, 안정성 등을 만족하는지 확인하고 소프트웨어의 결함을 찾아내는 활동

- **확인(Validation)테스트** : 고객의 요구사항에 맞게 구현되었는지 확인하는 것
 - 사용자 입장, 결과 중요
- **검증(Verification)테스트** : 설계 명세서에 맞게 만들어졌는지 점검하는 것
 - 개발자 입장, 생산 과정 중요

▶ 소프트웨어 테스트 원리

- 테스트는 결함의 존재를 밝히는 활동이다.
- 완벽한 테스트는 불가능하다.
- 테스트는 개발 초기에 시작해야 한다.
- 결함 집중(Defect Clustering) : 애플리케이션 결함의 대부분은 소수의 특정한 모듈에 집중되어 존재한다.
- 파레토(Pareto)의 법칙 : 전체 결함의 80%는 소프트웨어 제품의 전체 기능 중 20%에 집중되어 있다.
- 살충제 패러독스(Pesticide Paradox) : 동일한 테스트 케이스(Test Case)로 반복 실행하면 결함을 발견할 수 없으므로, 주기적으로 테스트 케이스를 리뷰하고 개선해야 한다.
- 테스트는 정황(Context, 맥락, 관계)에 의존한다.
- 오류-부재의 궤변(Absence of Errors Fallacy) : 소프트웨어 결함을 모두 제거해도 사용자의 요구사항을 만족시키지 못하면 해당 소프트웨어는 품질이 높다고 할 수 없다.

▶ 소프트웨어 테스트 유형

• 테스트 목적에 따른 분류

회복(Recovery) 테스트	시스템에 고의로 실패를 유도하고 시스템이 정상적으로 복구하는지 확인하는 테스트
안전(Security) 테스트	불법적인 소프트웨어가 접근하여 시스템을 파괴하지 못하도록 소스 코드 내의 보안 결함을 미리 점검하는 테스트
강도(Stress, 부하) 테스트	시스템에 과다 정보량을 부과하여 과부하 시에도 시스템이 정상적으로 작동되는지를 검증하는 테스트

성능(Performance) 테스트	사용자의 이벤트에 시스템이 응답하는 시간, 특정 시간 내에 처리하는 업무량, 사용자 요구에 시스템이 반응하는 속도 등을 확인하는 테스트
구조(Structure) 테스트	시스템의 내부 논리 경로, 소스 코드의 복잡도를 평가하는 테스트
회귀(Regression) 테스트	변경 또는 수정된 코드에 대하여 새로운 결함 발견 여부를 평가하는 테스트
병행(Parallel) 테스트	변경된 시스템과 기존 시스템에 동일한 데이터를 입력 후 결과를 비교하는 테스트

• 테스트 종류에 따른 분류

명세 기반 테스트	주어진 명세를 빠짐없이 테스트 케이스로 구현하고 있는지 확인하는 테스트 <ul style="list-style-type: none"> • 종류 : 동등 분할, 경계 값 분석 등
구조 기반 테스트	소프트웨어 내부 논리 흐름에 따라 테스트 케이스를 작성하고 확인하는 테스트 <ul style="list-style-type: none"> • 종류 : 구문 기반, 조건 기반 등
경험 기반 테스트	유사 소프트웨어나 유사 기술 평가에서 테스트의 경험을 토대로 한, 직관과 기술 능력을 기반으로 수행하는 테스트 <ul style="list-style-type: none"> • 종류 : 오류 예측 검사 등

• 프로그램 실행 여부에 따른 분류

정적 테스트 (Static Test)	프로그램 실행 없이 소스 코드의 구조를 분석하여 논리적으로 검증하는 테스트 <ul style="list-style-type: none"> • 종류 : 코드 검사, 워크스루, 인스펙션 등
동적 테스트 (Dynamic Test)	프로그램의 실행을 요구하는 테스트 <ul style="list-style-type: none"> • 종류 : 블랙박스 테스트, 화이트박스 테스트 등

▶ 블랙박스 테스트(Black Box Test)

사용자의 요구사항 명세서를 보면서 구현된 기능을 테스트

- 성능, 부정확한 기능, 인터페이스 오류 발견

• 종류

동치 분할 검사 (Equivalence Partitioning Testing)	입력 자료에 초점을 맞춰 테스트 케이스를 만들고 검사하는 기법
경계 값 분석 (Boundary Value Analysis)	입력 조건의 중간 값보다 경계 값에서 오류가 발생할 확률이 높으므로 입력 조건의 경계 값으로 테스트하는 기법

원인-효과 그래프 검사 (Cause-Effect Graphing Testing)	입력 자료 간의 관계와 출력에 영향을 미치는 상황을 체계적으로 분석한 후 효용성이 높은 테스트 케이스를 선정해서 검사하는 기법
오류 예측 검사 (Fault Based Testing)	과거 경험이나 테스터의 감각으로 테스트하는 기법
비교 검사 (Comparison Testing)	여러 버전의 프로그램에 동일한 자료를 제공해 동일한 결과가 출력되는지 검사하는 기법

▶ 화이트박스 테스트(White Box Test)

프로그램의 수행 경로 구조, 루프 등 내부 로직을 보면서 테스트

- 논리 구조상의 오류 발견

• 종류

기초 경로 검사 (Basic Path Testing)	<ul style="list-style-type: none"> 프로그램의 제어 구조를 기반으로 논리적 복잡성을 측정할 수 있게 해주는 테스트 기법 맥케이브(McCabe) 제안
제어 구조 검사	<ul style="list-style-type: none"> 조건 검사(Condition Testing) : 논리적 조건을 테스트하는 기법 루프 검사(Loop Testing) : 반복(Loop) 구조를 중심으로 테스트하는 기법 데이터 흐름 검사(Data Flow Testing) : 변수 정의, 변수 사용 위치에 초점을 맞춰 테스트하는 기법

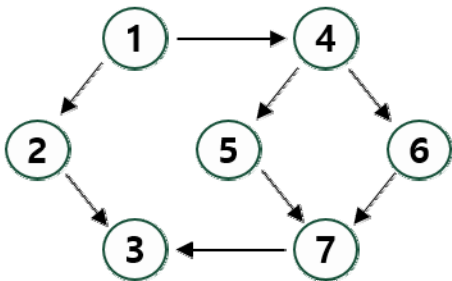
▶ 맥케이브의 순환 복잡도

(Cyclomatic Complexity)

원시 코드의 복잡도를 정량적으로 평가하는 방법

• 공식 : 복잡도 = 영역 수(폐구간) + 1

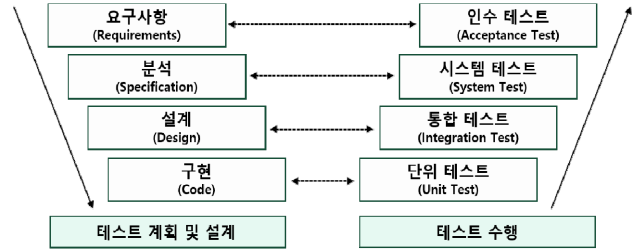
예 맥케이브의 순환 복잡도



- 복잡도 = 영역수 + 1
= 2 + 1
= 3

▶ 소프트웨어 생명 주기의 V 모델

애플리케이션 테스트와 소프트웨어 개발 단계를 연결하여 표현한 모델



단위 테스트 (Unit Test)	<ul style="list-style-type: none"> 구현된 모듈의 기능 수행 여부를 판정 내부에 존재하는 논리적 오류를 검출할 수 있는 방안을 파악
통합 테스트 (Integration Test)	<ul style="list-style-type: none"> 모듈 간의 인터페이스 연계를 검증하고 오류를 확인 모듈 간의 상호 작용 및 연계 동작 여부를 판정하는 방안을 파악 종류 : 비점증적인 방식(빅뱅 방식), 점증적 통합 방식(하향식 통합 테스트, 상향식 통합 테스트, 혼합식 통합 테스트)
시스템 테스트 (System Test)	<ul style="list-style-type: none"> 전체 시스템이 정상적으로 작동하는지 판정 기능 명세를 확인하는 방안을 파악
인수 테스트 (Acceptance Test)	<ul style="list-style-type: none"> 사용자가 요구분석 명세서에 명시된 사항을 모두 충족하는지 판정 시스템이 예상대로 동작하고 있는지를 판정하는 방안을 파악 종류 : 사용자 인수 테스트, 운영상의 인수 테스트, 계약 인수 테스트, 규정 인수 테스트, 알파 검사, 베타 검사

▶ 알파 검사(Alpha Test)

개발자의 장소에서 사용자가 시험하고 개발자는 뒤에서 결과를 지켜보는 검사

▶ 베타 검사(Beta Test)

실 업무를 가지고 사용자가 직접 시험하는 검사

▶ 하향식 통합 테스트

(Top Down Integration Test)

메인 제어 모듈로부터 아래 방향으로 제어의 경로를 따라 이동하여 하향식으로 통합하면서 테스트를 진행하는 방식

- 방식 : 깊이-우선 방식, 너비-우선 방식
- 스텝(Stub) : 모듈 간에 통합 테스트를 하기 위해 일시적으로 제공되는 시험용 모듈

- 상위 모듈은 있지만 하위 모듈이 없는 경우 하위

모듈을 대체

▶ 상향식 통합 테스트

(Bottom Up Integration Test)

최하위 레벨의 모듈 또는 컴포넌트로부터 위쪽 방향으로 제어의 경로를 따라 이동하면서 구축과 테스트를 진행하는 방식

- **드라이버(Driver)** : 하위 모듈은 있으나 상위 모듈이 없는 경우 하위 모듈 구동하기 위한 제어 프로그램
 - 테스트 대상 하위 모듈 호출, 파라미터 전달, 모듈 테스트 수행 후 결과 도출 등에 사용

▶ 테스트 케이스(Test Case)

입력 값, 실행 조건, 기대 결과로 구성된 테스트 항목의 명세서

- 명세 기반 테스트의 설계 산출물

▶ 테스트 오라클(Test Oracle)

테스트의 결과가 참인지 거짓인지를 판단하기 위해 사전에 정의된 참 값을 입력하여 비교하는 기법 및 활동

• 유형

참(True) 오라클	모든 입력 값에 대해 기대하는 결과를 생성함으로써 발생한 오류를 모두 검출하는 오라클
샘플링(Sampling) 오라클	특정한 몇 개의 입력 값에 대해서만 기대하는 결과를 제공하는 오라클
휴리스틱(Heuristic) 오라클	특정 입력 값에 대해 올바른 결과를 제공하고, 나머지 값들에 대해서는 휴리스틱(추정)으로 처리하는 오라클
일관성 검사(Consistent) 오라클	애플리케이션 변경이 있을 때, 수행 전과 후의 결과 값이 동일한지 확인하는 오라클

▶ 테스트 시나리오(Test Scenario)

- 테스트 수행을 위한 여러 테스트 케이스의 집합
- 테스트 케이스의 동작 순서를 기술한 문서
- 테스트를 위한 절차를 명세한 문서

▶ 나쁜 코드(Bad Code)

다른 개발자가 로직(Logic)을 이해하기 어렵게 작성된 코드

- 처리 로직이 서로 얽혀 있는 스파게티 코드
- 변수나 메소드에 대한 이름 정의를 알 수 없는 코드
- 동일한 처리 로직이 중복되게 작성된 코드

▶ 외계인 코드(Alien Code)

아주 오래되거나 참고 문서 또는 개발자가 없어 유지보수 작업이 어려운 코드

▶ 클린 코드(Clean Code)

잘 작성되어 가독성이 높고, 단순하며, 의존성을 줄이고, 중복을 최소화하여 깔끔하게 잘 정리된 코드

- **작성 원칙** : 가독성, 단순성, 의존성, 중복성, 추상화

▶ 정적 분석 도구

작성된 소스 코드를 실행시키지 않고, 코드 자체만으로 코딩 표준 준수 여부, 코딩 스타일 적정 여부, 잔존 결함 발견 여부를 확인하는 코드 분석 도구

- **종류** : PMD, Cppcheck, SonarQube, Checkstyle, CC,M Cobertura

▶ 동적 분석 도구

애플리케이션을 실행하여 코드에 존재하는 메모리 누수 현상을 발견하고, 발생한 스레드의 결함 등을 분석하기 위한 도구

- **종류** : Avalanche, Valgrind

▶ 리팩토링(Refactoring)

코드의 외부 행위는 바꾸지 않고 내부 구조를 개선하여 소프트웨어 시스템을 변경하는 프로세스

- 기능을 추가해서는 안 되고, 코드의 성능과 구조에만 신경을 씀
- 이미 존재하는 코드의 설계를 안전하게 향상시키는 기술
- 소프트웨어를 보다 이해하기 쉽고 수정하기 쉽게 개선하기 과정

▶ 재공학(Re-Engineering)

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

- 기존 소프트웨어를 수정 및 보완하여 재구축

▶ 역공학(Reverse-Engineering)

소프트웨어를 분석하여 소프트웨어 개발 과정과 데이터 처리 과정을 설명하는 분석 및 설계 정보를 재발견하거나 다시 만들어내는 작업

- 현재 프로그램으로부터 데이터, 아키텍처, 절차에 관한 분석 및 설계 정보를 추출

▶ 재사용(Re-Use)

목표 시스템의 개발 시간 및 비용 절감을 위하여 검증된 기능을 파악하고 재구성하여 시스템에 응용하기 위한 최적화 작업

- 이미 개발된 소프트웨어를 다른 소프트웨어의 개발이나 유지에 이용

- **형태**

- 편의적 재사용 : 프로젝트를 시작할 때 재사용 가능한 컴포넌트가 있는지 찾아보고 재사용한다.
- 계획적 재사용 : 컴포넌트를 후에 재사용이 가능하도록 전략적으로 설계해 나간다.

• 방법

합성 중심 (Composition-Based)	<ul style="list-style-type: none"> • 블록(모듈)을 만든 후 끼워 맞춰 소프트웨어를 완성하는 방법 • 블록 구성 방법
생성 중심 (Generation-Based)	<ul style="list-style-type: none"> • 추상화 형태로 쓰인 명세를 구체화하여 프로그램을 만드는 방법 • 패턴 구성 방법

▶ 제품 소프트웨어 패키징

개발이 완료된 제품 소프트웨어를 고객에게 전달하기 위한 형태로 패키징하고, 설치와 사용에 필요한 내용을 포함하는 매뉴얼을 작성하며, 제품 소프트웨어에 대한 패치 개발과 업그레이드를 위해 버전 관리를 수행하는 능력

- 고객(사용자) 편의성 중심

▶ 릴리즈 노트(Release Note)

최종 사용자인 고객과 릴리즈 정보를 공유하는 문서

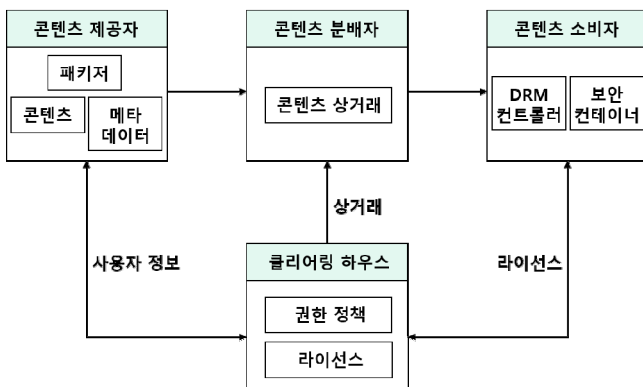
- **작성 항목** : Header, 개요, 목적, 이슈 요약, 재현 항목, 수정/개선 내용, 사용자 영향도, SW 지원 영향도, 노트, 면책 조항, 연락 정보

▶ 디지털 저작권 관리

(DRM; Digital Rights Management)

웹을 통해 유통되는 각종 디지털 콘텐츠의 안전 분배와 불법 복제 방지를 위한 보호 방식

• 구성 요소



콘텐츠 제공자 (Contents Provider)	콘텐츠를 제공하는 저작권자 • 패키지(Packager) : 콘텐츠를 메타 데이터와 함께 배포 가능한 단위로 묶는 기능
콘텐츠 분배자 (Contents Distributor)	쇼핑몰 등으로써 암호화된 콘텐츠 제공

클리어링 하우스 (Clearing House)	키 관리 및 라이선스 발급 관리
콘텐츠 소비자 (Contents Customer)	콘텐츠를 구매하는 주체 • DRM 컨트롤러(DRM Controller) : 배포된 콘텐츠의 이용 권한 통제 • 보안 컨테이너(Security Container) : 원본을 안전하게 유통하기 위한 전자적 보안 장치

• 기술 요소

- 암호화(Encryption)
- 키 관리(Key Management)
- 암호화 파일 생성(Packager)
- 식별 기술(Identification)
- 저작권 표현(Right Expression)
- 정책 관리(Policy Management)
- 크랙 방지(Tamper Resistance)
- 인증(Authentication)

▶ 제품 소프트웨어 설치 매뉴얼

제품 소프트웨어 개발 단계부터 적용한 기준이나 패키징 이후 설치의 주요 내용 등을 문서로 기록한 것

- **작성 항목** : 제품 소프트웨어 개요, 설치 관련 파일, 설치 아이콘, 프로그램 삭제, 관련 추가 정보

▶ 제품 소프트웨어 사용자 매뉴얼

사용에 필요한 절차 및 환경 등 전체 내용을 포함하는 매뉴얼을 작성하고, 제품 기능 및 고객 지원까지를 포함하여 문서로 기록한 것

- **작성 4단계** : 작성 지침 정의 → 사용자 매뉴얼 구성 요소 정의 → 구성 요소별 내용 작성 → 사용자 매뉴얼 검토

▶ 소프트웨어 품질(Software Quality)

사용자의 소프트웨어 요구사항을 충족하기 위한 능력에 영향을 미치는 소프트웨어 제품의 모든 특성과 속성

• 측정 항목

정확성 (Correctness)	사용자의 요구 기능을 충족시키는 정도
신뢰성 (Reliability)	정확하고 일관된 결과를 얻기 위해 요구되는 기능을 오류 없이 수행하는 정도
효율성 (Efficiency)	요구되는 기능을 수행하기 위한 필요한 자원의 소요 정도
무결성 (Integrity)	허용되지 않는 사용이나 자료의 변경을 제어하는 정도
사용 용이성 (Usability)	사용에 필요한 노력을 최소화하고 쉽게 사용할 수 있는 정도
유지보수성 (Maintainability)	변경 및 오류의 교정에 대한 노력을 최소화하는 정도

유연성 (Flexibility)	소프트웨어를 얼마만큼 쉽게 수정할 수 있는가 하는 정도
시험 용이성 (Testability)	의도대로 기능이 수행되는 것을 보장하기 위해 프로그램을 시험(Test)할 수 있는 정도
이식성 (Portability)	다양한 하드웨어 환경에서도 운용할 수 있도록 쉽게 수정할 수 있는 정도
재사용성 (Reusability)	전체나 일부 소프트웨어를 다른 응용 목적으로 사용할 수 있는가 하는 정도
상호 운용성 (Interoperability)	다른 소프트웨어와 정보를 교환할 수 있는 정도

▶ ISO/IEC 9126

품질 특성 및 측정 기준을 제시한 국제 표준
- 호환성과 보안성 강화를 위해 ISO/IEC 25010으로 대체됨

• 품질 요구사항

품질 요구사항	상세 품질 요구사항
기능성 (Functionality)	적절성, 정밀성, 상호 운용성, 보안성, 준수성
신뢰성 (Reliability)	성숙성, 고장 허용성, 회복성, 준수성
사용성 (Usability)	이해성, 학습성, 운용성, 친밀성, 준수성
효율성 (Efficiency)	시간 효율성, 자원 활용성, 준수성
유지 보수성 (Maintainability)	분석성, 변경성, 안정성, 시험성, 준수성
이식성 (Portability)	적용성, 설치성, 대체성, 공존성, 준수성

▶ ISO/IEC 14598

소프트웨어 제품 평가 프로세스의 개요와 평가에 대한 안내 지침 및 요구사항을 제공

▶ ISO/IEC 12119

소프트웨어 패키지에 대한 품질 요구사항 및 시험 사항을 규정한 국제 표준

▶ ISO/IEC 25000

ISO/IEC 9126, 14598, 12119 등의 여러 표준 문서를 통합하여 재구성하여 만든 표준 문서

▶ ISO/IEC 9000

품질 경영과 품질 보증에 관한 국제 규격

▶ ISO/IEC 12207

소프트웨어 개발 생명 주기 표준을 제공

• 프로세스

기본 생명 주기 프로세스	획득(계약 준비), 공급(계약), 개발(SW 구현), 운영, 유지보수 프로세스
지원 생명 주기 프로세스	품질 보증, 검증, 확인, 활동 검토, 감사, 문서화, 형상 관리, 문제 해결 프로세스
조직 생명 주기 프로세스	관리, 기반 구조, 개선, 훈련(교육) 프로세스

▶ ISO/IEC 15504(SPICE)

소프트웨어 프로세스 평가를 위한 표준

• **프로세스** : 5개의 프로세스 범주로 구분된 40개 프로세스로 구성

- 고객-공급(Customer-Supplier) 프로세스
- 공학(Engineering) 프로세스
- 지원(Support) 프로세스
- 관리(Management) 프로세스
- 조직(Organization) 프로세스

• **수행 능력** : 프로세스의 수행 능력을 6단계로 구분

- Level 0. 불완전(Incomplete) 단계
- Level 1. 수행(Performed) 단계
- Level 2. 관리(Managed) 단계
- Level 3. 확립(Established) 단계
- Level 4. 예측(Predictable) 단계
- Level 5. 최적화(Optimizing) 단계

▶ ISO/IEC 15288

프로세스 및 생명 주기 단계를 포함하는 시스템 엔지니어링 표준

▶ CMMI(능력 성숙도 통합 모델)

소프트웨어 개발 조직의 업무 능력 및 조직의 성숙도를 평가하는 모델

- 시스템과 소프트웨어 영역을 하나의 프로세스 개선 톨로 통합

• 5단계

- 1. 초기(Initial) 단계
- 2. 관리(Managed) 단계
- 3. 정의(Defined) 단계
- 4. 정량적 관리(Quantitatively Managed)
- 5. 최적화(Optimizing) 단계

▶ CMM(능력 성숙도 모델)

소프트웨어 기술을 지원하는 조직의 응용 프로그램을 개선하기 위한 절차

- 소프트웨어 개발 모델에 한정된 영역

• 5단계

- 1. 초기(Initial) 단계

- 2. 반복(Repeatable) 단계
- 3. 정의(Defined) 단계
- 4. 관리(Managed) 단계
- 5. 최적화(Optimizing) 단계

▶ 형상 관리

(SCM; Software Configuration Management)

소프트웨어의 개발 과정에서 발생하는 산출물의 변경사항을 버전관리하기 위한 활동

• 형상 항목

- 소프트웨어 공학 기반 표준과 절차(방법론, WBS, 개발 표준)
- 소프트웨어 프로젝트 계획서
- 소프트웨어 요구사항 명세서
- 소프트웨어 아키텍처, 실행 가능한 프로토타입
- 소프트웨어 화면, 프로그램 설계서
- 데이터베이스 기술서(스키마, 파일 구조, 초기 내용)
- 소스 코드 목록 및 소스 코드
- 실행 프로그램
- 테스트 계획, 절차, 결과
- 시스템 사용 및 운영과 설치에 필요한 매뉴얼
- 유지 보수 문서(변경 요청서, 변경 처리 보고서 등)

- **절차** : 형상 식별 → 변경 제어 → 형상 상태 보고 → 형상 감사

형상 식별	형상 관리 대상을 식별하여 이름과 관리 번호를 부여하고, 계층 구조로 구분하여 수정 및 추적이 쉽게 하고, 베이스라인의 기준을 정하는 작업
변경 제어 (= 형상 통제)	식별된 형상 항목의 변경 요구를 검토 및 승인하여 적절히 통제함으로써 현재의 베이스라인에 잘 반영될 수 있도록 조정하는 작업
형상 상태 보고(기록)	형상의 식별, 통제, 감사 작업의 결과를 기록 및 관리하고 보고서를 작성하는 작업
형상 감사	베이스라인의 무결성을 평가하기 위하여 확인·검증 과정을 통해 공식적으로 승인하는 작업

▶ 버전 관리(Version Control, Revision Control)

소프트웨어 개발과 관련하여 코드와 라이브러리, 관련 문서 등 시간의 변화에 따른 변경을 관리하는 활동

• 주요 용어

저장소 (Repository)	파일의 현재 버전과 변경 이력 정보를 저장하는 저장소
가져오기 (Import)	버전 관리가 되지 않은 저장소에 파일을 처음으로 복사
체크아웃 (Check-out)	프로그램 수정을 위해 저장소 파일을 받음
체크인 (Check-in)	프로그램 수정 후 저장소에 새로운 버전으로 갱신

커밋 (Commit)	파일 갱신 완료 • 체크인 시 이전 갱신 사항이 있는 경우 충돌(Conflict)을 알리고, Diff 도구 이용하여 수정
동기화 (Update)	자신의 작업 공간을 저장소의 최신 버전으로 동기화

▶ 버전 관리 방식

- **공유 폴더 방식** : 개발 완료 파일을 약속된 위치의 공유 폴더에 복사하는 방식
- 종류 : RCS, SCCS 등
- **클라이언트/서버 방식** : 중앙에 버전 관리 시스템이 항상 동작하여 관리하는 방식
- 종류 : CVS, 서브버전(SVN), 클리어 케이스(Clear Case) 등
- **분산 저장소 방식** : 로컬 저장소와 원격 저장소로 분산된 구조로 파일을 원격 저장소와 개발자의 로컬 저장소에 함께 저장하여 관리하는 방식
- 종류 : 비트키퍼(Bitkeeper), 깃(Git) 등

▶ JSON(JavaScript Object Notation, 제이슨)

속성-값의 쌍으로 이루어진 데이터 객체를 전달하기 위해 사용하는 개방형 표준 포맷

▶ AJAX(비동기식 자바스크립트 XML,

Asynchronous JavaScript and XML)

클라이언트와 서버 간에 비동기적으로 XML 데이터를 주고받는 기술

- 전체 페이지를 새로 고치지 않고 페이지의 일부만을 위한 데이터를 로드하는 기법
- 이용자가 웹 페이지와 자유롭게 상호 작용할 수 있도록 하는 기술

▶ XML(eXtensible Markup Language)

웹 브라우저 간에 HTML 문법이 호환되지 않는 문제와 SGML의 복잡함을 해결하기 위해 개발된 다목적 마크업 언어

▶ 인터페이스 구현 검증 도구

xUnit	java(Junit), C++(Cppunit), .Net(Nunit) 등 다양한 언어를 지원하는 단위 테스트 프레임워크
STAF	서비스 호출, 컴포넌트 재사용 등 다양한 환경을 지원하는 테스트 프레임워크
FitNesse	웹 기반 테스트 케이스 설계/실행/결과 확인 등을 지원하는 테스트 프레임워크
NTAF	STAF와 FitNesse를 통합한 Naver 테스트 자동화 프레임워크
Selenium	다양한 브라우저 지원 및 개발 언어를 지원하는 웹 애플리케이션 테스트 프레임워크

watir	Ruby 프로그래밍 언어 기반 웹 애플리케이션 테스트 프레임워크
-------	-------------------------------------

▶ 인터페이스 보안

영역	설명
네트워크	인터페이스 송·수신 간 중간자에 의한 데이터 탈취, 위·변조를 막기 위해 네트워크 트래픽에 대한 암호화를 적용 <ul style="list-style-type: none"> • 보안 기능 : IPsec AH 적용, IKE 프로토콜 적용, IPsec ESP 적용, IPsec Transport mode 적용, SSL의 서버 인증 모두 운영, S-HTTP를 적용하여 메시지 암호화, 서버/클라이언트 상호 인증 필요 등
애플리케이션	애플리케이션 구현 코드 상에서 보안 취약점을 보완하는 방향으로 애플리케이션 보안 기능을 적용
데이터베이스	데이터베이스의 접근 권한 및 SQL, 프로시저, 트리거 등 데이터베이스 동작 객체의 보안 취약점을 보완하기 위해 보안 기능을 적용

▶ IPsec(Internet Protocol Security,

인터넷 보안 프로토콜)

망 계층(Network Layer)인 인터넷 프로토콜에서 보안성을 제공해 주는 표준화된 기술

- **IKE(Internet Key Exchange, 인터넷 표준 암호키 교환 프로토콜, 키 관리 프로토콜)** : 생성한 암호키를 상대방에게 안전하게 송신하기 위한 방법
- **AH(Authentication Header, 인증 헤더)** : 출발지 인증, 데이터 무결성은 제공하지만, 기밀성은 제공하지 않음
- **ESP(Encapsulating Security Payload, 보안 페이로드 캡슐화)** : 기밀성, 출발지 인증, 데이터 무결성 등을 제공

3과목

데이터베이스 구축

▶ 스키마(Schema)

데이터베이스의 구조와 제약조건에 대한 명세를 기술한 것

- 시간에 따라 불변인 특성을 가짐
- 데이터 사전에 저장
- 메타 데이터(Meta Data)

• 종류

외부 (External) 스키마	<ul style="list-style-type: none"> • 사용자나 응용 프로그래머가 보는 관점 • 서브 스키마, 사용자 뷰
개념 (Conceptual) 스키마	<ul style="list-style-type: none"> • 데이터베이스의 전체적인 논리적 구조 • 전체적인 뷰, 범기관적, 총괄적 입장
내부 (Internal) 스키마	<ul style="list-style-type: none"> • 데이터베이스의 전체적인 물리적 구조 • 실제 데이터를 저장

- **데이터 사전(Data Dictionary)** : 시스템 자신이 필요로 하는 여러 객체(Object)에 관한 정보를 포함하고 있는 시스템 데이터베이스
 - 시스템 카탈로그(System Catalog)

▶ 인스턴스(Instance)

정의된 스키마에 따라 데이터베이스에 실제로 저장된 값

- 시간에 따라 동적으로 변화

▶ 데이터베이스 설계 순서

요구조건 분석 → 개념적 설계 → 논리적 설계 → 물리적 설계 → 구현 → 운영 → 감시 및 개선

- **요구조건 분석** : 요구조건 명세서 작성
- **개념적 설계** : 개체 타입과 이들 간의 관계 타입을 이용해 현실 세계를 개념적으로 표현
 - ER 다이어그램 작성
 - DBMS에 독립적인 개념 스키마 모델링
 - 트랜잭션 모델링
- **논리적 설계** : 목표 DBMS에 맞추어 논리적 모델로 설계
 - 트랜잭션 인터페이스 설계
 - 스키마 평가 및 정제
 - 목표 DBMS에 맞는 스키마 설계
 - 정규화 과정 수행
- **물리적 설계** : 저장 레코드 양식의 설계 및 물리적 구조 데이터를 표현
 - 설계 시 고려사항 : 응답시간, 저장 공간의 효율성, 트랜잭션의 처리량
 - 접근 경로 설계

- 레코드 집종의 분석 및 설계
- 트랜잭션의 세부 사항 설계

- **구현** : 목표 DBMS의 DDL로 스키마 작성, 응용 프로그램을 위한 트랜잭션 작성

▶ 데이터 모델링

데이터베이스를 구축하기 위한 분석 및 설계의 과정

• 데이터 모델의 종류

개념적 데이터 모델	현실 세계를 추상적으로 표현
논리적 데이터 모델	개념적 데이터 모델을 컴퓨터가 이해할 수 있도록 표현
물리적 데이터 모델	저장 레코드 양식의 설계 및 물리적 구조 데이터 표현

• 데이터 모델에 표시할 요소

구조 (Data Structure)	개체 간의 관계
연산 (Operation)	데이터를 처리하는 방법
제약조건 (Constraint)	실제 데이터의 논리적인 제약조건


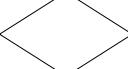

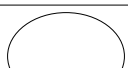

▶ ER 모델(Entity-Relationship Model,

ER Diagram, 개체관계도)

개체와 개체 간의 관계를 도식화한 개체관계도

- 1976년 P.Chen이 제안

• 구성 요소

요소	기호
개체	
관계	
속성	
기본키 속성	
연결, 링크	

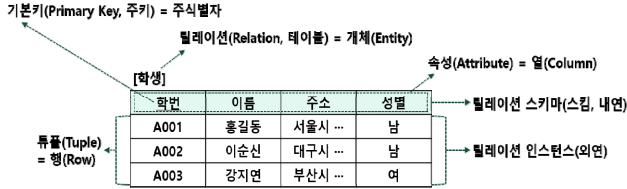
- 개체(Entity) : 데이터베이스에 표현하려고 하는 현실 세계의 대상체
- 속성(Attribute) : 개체의 성질, 분류, 식별, 수량, 상태 등
- 관계(Relationship) : 두 개체 간에 의미 있는 연결

▶ 논리적 데이터 모델의 종류

- **관계형 데이터 모델**
 - 구조 : 표, 테이블, 릴레이션
 - 관계 표현 : 키, 1:1, 1:N, N:M
- **계층형 데이터 모델**

- 구조 : 트리
- 관계 표현 : 부모-자식 관계, N:M 직접 표현 불가
- **네트워크형 데이터 모델**
 - 구조 : 그래프, 망
 - 관계 표현 : 오너-멤버 관계, N:M 직접 표현 불가

▶ 관계형 데이터베이스의 릴레이션 구조



릴레이션 (Relation)	데이터들을 표(Table) 형태로 표현한 것 • 예 [학생] 릴레이션, [학생] 테이블
튜플 (Tuple)	속성의 모임으로 구성된 릴레이션을 구성하는 각 행
속성 (Attribute)	개체의 성질, 분류, 식별, 수량, 상태 등을 나타낸 것
도메인 (Domain)	한 속성에 나타날 수 있는 값들의 범위 • 예 성별 속성의 도메인은 '남' 또는 '여'로, 그 외의 값은 입력될 수 없다.
카디널리티 (Cardinality)	튜플들의 수 • 예 [학생] 릴레이션의 카디널리티는 3이다.
차수(Degree, 디그리)	속성들의 수 • 예 [학생] 릴레이션의 차수는 4이다.
널(Null)	'해당 없음' 등의 이유로 정보 부재를 나타내기 위해 사용하는 특수한 데이터 값

▶ 키(Key)

릴레이션에서 튜플을 유일하게 구별하는 속성 또는 속성들의 집합

- 식별자(Identifier)
- **특성** : 유일성(Uniqueness), 최소성(Minimality)
- **종류**

슈퍼키 (Super Key)	<ul style="list-style-type: none"> • 한 릴레이션 내에 있는 속성 또는 속성들의 집합 • 유일성은 만족하지만, 최소성은 만족하지 못함
후보키 (Candidate Key)	<ul style="list-style-type: none"> • 한 릴레이션 내에 있는 모든 튜플을 유일하게 식별할 수 있는 속성 또는 속성들의 집합 • 유일성과 최소성을 모두 만족함
기본키 (Primary Key)	<ul style="list-style-type: none"> • 후보키 중에서 특별히 선정된 키 • 중복이 되어서는 안되며, 널(Null) 값을 가질 수 없음
대체키 (Alternate Key)	<ul style="list-style-type: none"> • 후보키 중에서 기본키를 제외한 후보키

	<ul style="list-style-type: none"> • 후보키가 둘 이상인 경우 그 중 하나를 기본키로 지정하면, 나머지 후보키들은 대체키가 됨
외래키 (Foreign Key)	<ul style="list-style-type: none"> • 다른 릴레이션의 기본키를 참조하는 속성 또는 속성들의 집합 • 참조 릴레이션의 기본키와 동일 • 릴레이션들 간의 관계를 나타내기 위해 사용

▶ 무결성(Integrity)

데이터의 정확성과 일관성을 유지하고 보증하는 것

• 종류

참조 무결성 (Referential Integrity)	릴레이션은 참조할 수 없는 외래키 값을 가질 수 없다.
개체 무결성 (Entity Integrity)	한 릴레이션의 기본키를 구성하는 속성값은 널(Null) 값이나 중복 값을 가질 수 없다.
도메인 무결성 (Domain Integrity)	각 속성 값은 반드시 정의된 도메인에 속한 값이어야 한다.

• 규정

도메인 무결성 규정 (Domain Integrity Rules)	'각 속성 값은 반드시 정의된 도메인에 속한 값이어야 한다.'라는 도메인 무결성에 관한 규정
릴레이션 무결성 규정 (Relation Integrity Rules)	한 튜플이 릴레이션에 삽입될 수 있는가, 한 릴레이션과 또 다른 릴레이션의 튜플 간의 관계는 적절한가에 대한 규정

▶ 이상(Anomaly)

릴레이션에서 일부 속성들의 종속으로 인해 데이터의 중복이 발생하여 테이블 조작 시 불일치가 발생하는 것

• 종류

삽입 이상 (Insertion Anomaly)	불필요한 정보를 함께 저장하지 않고는 어떤 정보를 저장하는 것이 불가능한 현상
삭제 이상 (Deletion Anomaly)	유용한 정보를 함께 삭제하지 않고는 어떤 정보를 삭제하는 것이 불가능한 현상
갱신 이상 (Update Anomaly)	반복된 데이터 중에 일부만 수정하면 데이터의 불일치가 발생하는 현상

▶ 정규화(Normalization)

이상 현상 발생 가능성을 줄이기 위한 무손실 분해 행위

- 데이터의 중복을 방지하기 위함

• **과정** : 1NF → 2NF → 3NF → BCNF → 4NF → 5NF

▶ 제1정규형(1NF)

반복되는 속성을 제거하여 모든 속성이 원자 도메인만으로 되어 있는 정규형

▶ 제2정규형(2NF)

제1정규형이고, 부분 함수적 종속을 제거하여 완전 함수적 종속을 만족하는 정규형

- **함수적 종속** : 어떤 릴레이션 R에서 X와 Y를 각각 R의 속성 집합의 부분 집합이라고 할 경우, 속성 X의 값 각각에 대하여 시간에 관계없이 항상 속성 Y의 값이 오직 하나만 연관되어 있을 때, Y는 X에 함수적 종속이라 한다.
- 표기법 : $X \rightarrow Y$

▶ 제3정규형(3NF)

제2정규형이고, 이행적 함수 종속 관계를 제거하여 비이행적 함수 종속 관계를 만족하는 정규형

- **이행적 함수 종속** : $X \rightarrow Y$ 이고, $Y \rightarrow Z$ 인 경우, $X \rightarrow Z$ 는 이행적 함수 종속 관계이다.

▶ BCNF(Boyce/Codd Normal Form)

제3정규형이고, 결정자가 후보키가 아닌 함수적 종속을 제거하여 모든 결정자가 후보키인 정규형

▶ 제4정규형(4NF)

BCNF이고, 다치 종속을 제거한 정규형

▶ 제5정규형(5NF)

제4정규형이고, 조인 종속성을 이용한 정규형

▶ 반정규화(비정규화, 역정규화, Denormalization)

정규화된 데이터 모델을 시스템 성능 향상, 개발 과정의 편의성, 운영의 단순화를 목적으로 수행되는 의도적인 정규화 원칙 위배 행위

- 데이터 모델을 중복(추가), 통합(병합), 분리(분할)하는 과정
- **유형** : 테이블 분할, 중복 테이블 생성, 중복 속성 생성, 테이블 통합, 테이블 제거

▶ 인덱스(Index)

검색을 빠르게 하기 위해 <키 값, 포인터> 쌍으로 구성된 보조적인 데이터 구조

- **종류** : B 트리 인덱스, 비트맵 인덱스, 비트맵 조인 인덱스, 함수 기반 인덱스, 도메인 인덱스

▶ 뷰(View)

사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위해 하나 이상의 기본 테이블로부터 유도된 가상

테이블

▶ 파티션(Partition)

대용량 테이블이나 인덱스를 관리하기 쉬운 논리적 단위인 파티션이라는 작은 단위로 분할하는 것

- **종류** : 범위 분할(Range Partitioning), 해시 분할(Hash Partitioning), 조합 분할(Composite Partitioning), 라운드 로빈 분할(Round Robin Partitioning), 목록 분할(List Partitioning)

▶ 클러스터(Cluster)

데이터 저장 시 동일한 성격의 데이터를 동일한 데이터 블록에 저장하는 물리적 저장 기법

▶ 분산 데이터베이스

물리적으로 분산된 데이터베이스를 논리적인 하나의 데이터베이스로 인식하는 기법

• 4대 목표

- 위치 투명성(Location Transparency)
- 중복(복제) 투명성(Replication Transparency)
- 병행 투명성(Concurrency Transparency)
- 장애 투명성(Failure Transparency)

▶ 데이터베이스 이중화(Database Replication)

데이터베이스 중단이나 물리적 손상 발생 시 이를 복구하기 위해 동일한 데이터베이스를 복제하여 관리하는 것

▶ 스토리지(Storage)

대용량 데이터를 저장할 수 있는 저장 장치

• 종류

DAS (Direct Attached Storage)	• 서버와 저장 장치를 직접 연결하는 방법
NAS (Network Attached Storage)	• 서버와 저장 장치를 네트워크로 연결하는 방법
SAN (Storage Area Network)	<ul style="list-style-type: none"> • 서버와 저장 장치를 연결하는 전용 네트워크를 별도로 구성하는 방법 • 파이버 채널 스위치(FCS) 이용

▶ 데이터 웨어하우스(DW; Data Warehouse)

사용자의 의사 결정에 도움을 주기 위해 시스템에서 추출/변환/통합되고 요약된 주제 중심적인 데이터베이스

- 쌓여있는 데이터를 빅 데이터 분석으로 활용하는 것

▶ 데이터 마트(Data Mart)

데이터의 한 부분으로서, 특정 사용자가 관심을 갖는 데이터들을 담은 비교적 작은 규모의 데이터 웨어하우스

스

- 데이터 웨어하우스에 있는 일부 데이터를 가지고 특정 사용자를 대상으로 함

▶ 데이터 마이닝(Data Mining)

많은 데이터 가운데 숨겨져 있는 유용한 상관관계를 발견하여, 미래에 실행 가능한 정보를 추출해 내고 의사 결정에 이용하는 과정

- 대량의 데이터 집합으로부터 의미 있는 패턴이나 규칙을 발견하는 것

- 도구 : OLAP(On-Line Analytical Processing)

▶ OLAP(On-Line Analytical Processing,

온라인 분석 처리, 올랩)

사용자가 다양한 각도에서 직접 대화식으로 정보를 분석하는 과정으로, 데이터 웨어하우스를 다차원적으로 분석하고 시각화하는 과정

- 데이터 웨어하우스의 데이터를 전략적인 정보로 변환시키는 역할

- 연산

Drill Up(Roll-Up)	상세한 작은 범위에서 요약된 큰 범위로 단계적 접근하는 기능
Drill Down	요약된 큰 범위에서 상세한 작은 범위로 단계적 접근하는 기능
Drill Across	다른 큐브의 데이터에 접근하는 기능
Drill Through	OLAP에서 데이터 웨어하우스 또는 OTLP에 존재하는 상세 데이터에 접근하는 기능
Pivoting	보고서의 열과 행, 페이지의 차원들을 바꿔서 볼 수 있는 기능
Slicing	특정 관점으로 큐브를 잘라서 볼 수 있는 기능
Dicing	Slicing 기법에서 더 세분화하는 기능

▶ CRUD 매트릭스

(Create Read Update Delete Matrix)

업무 프로세스와 데이터 간 상관 분석표

- 구분 : 생성(Create), 이용(Read), 수정(Update, 갱신), 삭제>Delete)

예) CRUD 매트릭스

프로세스 \ 개체	거래처
거래처 등록	C
거래처 정보 변경	R, U

▶ 접근 통제(Access Control, 접근 제어)

사용자가 특정 자원에 접근할 때 접근을 요구하는 사용

자에 대한 식별과 보안 정책에 근거하여 접근을 승인하거나 거부하는 것

- 종류

임의 접근 통제(DAC; Discretionary Access Control, 신분 기반 정책)	자원(객체)에 접근하는 사용자(주체)의 신원에 따라 접근 권한을 부여하는 방식
강제 접근 통제(MAC; Mandatory Access Control, 규칙 기반 정책)	사용자와 자원 모두 보안 레벨(등급, 수준)을 부여받아 서로의 레벨을 비교하여 접근 권한을 부여하는 방식
역할 기반 접근 통제(RBAC; Role-Based Access Control)	사용자에게 할당된 역할(Role)에 기반하여 접근 권한을 부여하는 방식

▶ DAC 보안 모델

- 접근 제어 행렬(Access Control Matrix, 접근 통제 행렬) : 객체에 대한 접근 권한을 행렬로 표시한 기법
- 자격 목록(Capability List, Capability Tickets) : 접근 제어 행렬에 있는 주체를 중심으로 자격 목록을 구성한 것
- 접근 제어 목록(ACL; Access-Control List) : 접근 제어 행렬에 있는 객체를 중심으로 접근 목록을 구성한 것

▶ MAC 보안 모델

벨-라파둘라 모델(BLP; Bell-LaPadula Confidentiality Model)	<ul style="list-style-type: none"> • 기밀성을 강조한 최초의 수학적 모델 • 군사용 보안 정책
비바 무결성 모델(Biba Integrity Model)	<ul style="list-style-type: none"> • 무결성을 위한 최초의 상업적 모델 • 비인가자에 의한 데이터 변형 방지만 취급
클락-윌슨 무결성 모델(CWM; Clark-Wilson integrity Model)	<ul style="list-style-type: none"> • 주체가 직접 객체에 접근할 수 없고 프로그램을 통해서만 객체에 접근할 수 있는 무결성 중심의 상업적 모델
만리장성 모델(CWM; Chinese Wall Model, Brewer-Nash Model)	<ul style="list-style-type: none"> • 주체의 이전 동작에 따라 변화할 수 있는 접근 통제를 제공하는 모델

▶ 데이터 정의어

(DDL; Data Definition Language)

데이터의 형태, 구조, 데이터베이스의 저장에 관한 내용을 정의 및 변경하는 데이터 언어

• 종류

CREATE	도메인, 테이블, 뷰, 인덱스 정의
ALTER	테이블 구조 변경
DROP	도메인, 테이블, 뷰, 인덱스 삭제

▶ 데이터 조작어

(DML: Data Manipulation Language)

사용자의 요구에 따라 삽입, 수정, 삭제, 검색 등을 지원하는 데이터 언어

• 종류

INSERT	튜플 삽입
UPDATE	튜플 수정(갱신)
DELETE	튜플 삭제
SELECT	튜플 검색

▶ 데이터 제어어

(DCL: Data Control Language)

무결성 유지, 보안, 권한, 병행 수행 제어, 회복 등 정확성과 안정성을 유지하는 데이터 언어

• 종류

GRANT	데이터베이스 사용자에게 권한 부여
REVOKE	데이터베이스 사용자의 권한 취소
COMMIT	트랜잭션이 성공했을 경우 그 결과를 데이터베이스에 적용하여 작업을 완료 시킴
ROLLBACK	트랜잭션의 실패로 작업을 취소하고, 이전 상태로 되돌림
SAVEPOINT	트랜잭션 내에 롤백할 위치인 저장점을 지정

▶ CREATE TABLE

```
CREATE TABLE 테이블명 (
    속성명 데이터타입 [DEFAULT 기본값] [NOT NULL]
    , ...
    [,PRIMARY KEY(기본키_속성명)]
    [,UNIQUE(대체키_속성명)]
    [,FOREIGN KEY(외래키_속성명) REFERENCES 참조테이블명(참조테이블기본키_속성명)]
    [,CHECK(제약조건)]
);
```

- **테이블명** : 임의로 지정한 테이블 이름
- **속성명** : 테이블에 포함될 임의로 지정한 속성 이름
- **데이터타입** : 속성의 데이터 형식
- **DEFAULT 기본값** : 해당 속성의 기본값을 지정
- **NOT NULL** : 해당 속성은 NULL 값을 가질 수 없음
- **PRIMARY KEY(기본키_속성명)** : 해당 속성을 기본키로 지정
- **UNIQUE(대체키_속성명)** : 해당 속성을 대체키로 지

정

- **FOREIGN KEY(외래키_속성명)** : 해당 속성을 외래키로 지정
- **REFERENCES 참조테이블명(참조테이블기본키_속성명)** : 외래키로 지정된 속성은 참조테이블의 기본키를 참조함
- **CHECK(제약조건)** : 데이터베이스에 저장된 데이터의 정확성을 보장하기 위해 정확하지 않은 데이터가 데이터베이스 내에 저장되는 것을 방지하기 위한 조건

▶ ALTER TABLE

```
① ALTER TABLE 테이블명 ADD 속성명 데이터타입;
② ALTER TABLE 테이블명 ALTER 속성명 SET DEFAULT 기본값;
③ ALTER TABLE 테이블명 DROP 속성명;
```

- ① : 테이블에 새로운 속성(열)을 추가한다.
- ② : 테이블의 속성(열)의 기본값을 변경한다.
- ③ : 테이블의 속성(열)을 제거한다.

▶ DROP TABLE

```
DROP TABLE 테이블명 [CASCADE|RESTRICT];
```

- **DROP TABLE 테이블명** : 테이블 삭제
- **CASCADE** : 참조하는 테이블을 연쇄적으로 제거
- **RESTRICT** : 참조하는 테이블이 있을 경우 제거 안 됨

▶ INSERT

```
INSERT INTO 테이블명([속성])
VALUES (데이터1, 데이터2, ...);
```

- **INSERT INTO 테이블명** : 튜플을 삽입할 테이블명
- **속성** : 데이터를 삽입할 속성
- **데이터** : 테이블에 삽입할 데이터값

▶ UPDATE

```
UPDATE 테이블명
SET 속성명 = 데이터
[WHERE 조건];
```

- **UPDATE 테이블명** : 튜플을 수정할 테이블명
- **SET 속성명 = 데이터** : 해당 속성을 데이터값으로 수정
- **WHERE 조건** : 조건에 만족하는 튜플 수정

▶ DELETE

```
DELETE FROM 테이블명
[WHERE 조건];
```

- **DELETE FROM 테이블명** : 튜플을 삭제할 테이블명

- **WHERE 조건** : 조건에 만족하는 튜플 삭제

▶ SELECT

```
SELECT [DISTINCT] 속성 | 집계함수 [AS 별칭]
FROM 테이블명
[WHERE 조건]
[GROUP BY 속성 [HAVING 조건]]
[ORDER BY 속성 [ASC | DESC]];
```

- **SELECT 속성** : 해당 속성을 검색
- **SELECT 집계함수** : GROUP BY절에 지정된 그룹별로 속성의 합계, 평균 등을 검색하기 위한 함수
- **DISTINCT** : 중복되는 튜플을 배제하여 출력
- **AS 별칭** : 해당 속성의 이름을 별칭으로 표시
- **FROM 테이블명** : 튜플을 검색할 테이블명
- **WHERE 조건** : 조건에 만족하는 튜플 검색
- **GROUP BY 속성** : 해당 속성을 그룹화
- **HAVING 조건** : 그룹에 대한 조건
- **ORDER BY 속성** : 해당 속성을 정렬
 - ASC : 오름차순 정렬
 - DESC : 내림차순 정렬

▶ SELECT - 조건

- **비교 연산자**

A = B 또는 A IN B	A와 B는 같다.
A <> B 또는 A NOT IN B	A 와 B는 같지 않다.
A > B	A는 B보다 크다.
A >= B	A는 B보다 크거나 같다.
A < B	A는 B보다 작다.
A <= B	A는 B보다 작거나 같다.

- **논리 연산자**

조건1 AND 조건2	조건1과 조건2가 모두 참일 경우 참을 반환
조건1 OR 조건2	조건1, 조건2 둘 중 하나라도 참일 경우 참을 반환
NOT 조건	조건의 부정을 반환

- **LIKE**

%	모든 문자를 의미 • 예 김% : 김으로 시작하는 모든 문자
_	한 글자를 의미 • 예 김_ : 김으로 시작하고 2 글자인 문자

- **기타 연산자**

BETWEEN A AND B	A와 B 사이의 값
IS NULL	NULL 값
IS NOT NULL	NULL 값이 아닌 값

▶ SELECT - 부속(Subquery, 서브쿼리)

SQL문 내부에 사용하는 SELECT문

- 부속질의 결과를 WHERE절의 조건으로 사용

예 SELECT ~ FROM ~ WHERE ~
(SELECT ~ FROM ~ WHERE ~);

▶ 집계 함수(Aggregate Function)

COUNT(속성)	그룹별 튜플 수
AVG(속성)	그룹별 평균
SUM(속성)	그룹별 합계
MAX(속성)	그룹별 최댓값
MIN(속성)	그룹별 최솟값
STDDEV(속성)	그룹별 표준편차
VARIAN(속성)	그룹별 분산

▶ 집합 연산자

UNION	중복을 제거한 합집합 연산자
UNION ALL	중복을 포함한 합집합 연산자
INTERSECT	교집합 연산자
EXCEPT	차집합 연산자

예 SELECT ~ FROM ~
UNION | UNION ALL | INTERSECT | EXCEPT
SELECT ~ FROM ~

▶ GRANT

```
GRANT 권한리스트
ON 테이블명
TO 사용자명
[WITH GRANT OPTION];
```

- **GRANT 권한리스트** : 부여할 권한 종류
- **ON 테이블명** : 사용자가 권한을 가질 테이블명
- **TO 사용자명** : 권한을 가질 사용자
- **WITH GRANT OPTION** : 부여받은 권한을 다른 사용자에게 부여할 수 있는 권한을 가짐

▶ REVOKE

```
REVOKE 권한리스트
ON 테이블명
FROM 사용자명
[CASCADE];
```

- **REVOKE 권한리스트** : 취소할 권한 종류
- **ON 테이블명** : 사용자로부터 권한을 취소할 테이블명
- **FROM 사용자명** : 권한을 취소할 사용자
- **CASCADE** : 해당 사용자가 다른 사용자에게 권한을 부여했을 경우 연쇄적으로 권한 취소

▶ COMMIT

COMMIT;

▶ ROLLBACK

- ① ROLLBACK;
- ② ROLLBACK TO 저장점명;

- ① : COMMIT 시점 이후의 데이터베이스 상태로 되돌림
- ② : 저장점을 지정한 시점 이후의 데이터베이스 상태로 되돌림

▶ SAVEPOINT

SAVEPOINT 저장점명;

- **SAVEPOINT 저장점명** : 임의로 지정한 저장점명

▶ 관계 데이터 연산

원하는 데이터를 얻기 위해 릴레이션에 필요한 처리 요구를 수행하는 데이터 언어

• 종류

관계 대수	<ul style="list-style-type: none"> 절차적 언어(절차 중심) 원하는 정보를 어떻게 유도하는가를 연산자와 연산 규칙 이용하여 기술
관계 해석	<ul style="list-style-type: none"> 비절차적 언어(결과 중심) 원하는 정보가 무엇이라는 것만 정의

▶ 관계 대수의 분류

• 순수 관계 연산자

SELECT(σ)	릴레이션에서 주어진 조건을 만족하는 튜플들을 검색하는 것
PROJECT(π)	릴레이션에서 주어진 조건을 만족하는 속성들을 검색하는 것
JOIN(\bowtie)	두 개의 릴레이션에서 공통된 속성을 연결하는 것
DIVISION(\div)	두 개의 릴레이션 R(X)와 S(Y)가 있을 때, R의 속성이 S의 속성 값을 모두 가진 튜플에서 S가 가진 속성을 제외한 속성만을 구하는 것

• 일반 집합 연산자

합집합 (UNION, \cup)	두 릴레이션에 존재하는 튜플의 합 집합을 구하는 연산
교집합 (INTERSECTION, \cap)	두 릴레이션에 존재하는 튜플의 교 집합을 구하는 연산
차집합 (DIFFERENCE, $-$)	두 릴레이션에 존재하는 튜플의 차 집합을 구하는 연산

카티션
프로덕트
(Cartesian
Product,
교차곱, \times)

두 릴레이션에 있는 튜플들의 순서 쌍을 구하는 연산

▶ 프로시저(Procedure)

특정 기능을 수행하는 일종의 트랜잭션 언어로, 호출을 통해 미리 저장해 놓은 SQL문을 실행하는 프로그램

▶ 사용자 정의 함수(Function)

SQL문을 사용하여 일련의 작업을 연속적으로 처리하며, 종료 시 예약어 RETURN을 사용하여 처리 결과를 단일 값으로 반환하는 프로그램

▶ 트리거(Trigger)

데이터베이스의 데이터 삽입, 수정, 삭제 등의 이벤트가 발생할 때마다 관련 작업이 자동으로 수행되는 프로그램

▶ 트랜잭션(Transaction)

데이터베이스의 상태를 변화시키는 논리적 연산의 집합

- 데이터베이스 접근 기본 단위
- 병행 제어의 기본 단위

• 특성

원자성 (Atomicity)	트랜잭션이 데이터베이스에 모두 반영되거나 아니면 전혀 반영되지 않아야 된다.
일관성 (Consistency)	트랜잭션이 그 실행을 성공적으로 완료하면 항상 일관성 있는 데이터베이스 상태로 변환해야 한다.
독립성, 격리성 (Isolation)	둘 이상의 트랜잭션이 동시에 병행 실행되고 있을 때, 또 다른 하나의 트랜잭션의 연산이 끼어들 수 없다.
영속성, 지속성 (Durability)	트랜잭션의 결과는 영구적으로 반영되어야 한다.

- **상태** : 활동(Active), 부분적 완료(Partially Committed), 실패(Failed), 완료(Committed), 철회(Aborted)

▶ 병행 제어(Concurrency Control, 동시성 제어)

트랜잭션을 동시에 여러 개 수행할 때, 데이터베이스 일관성 유지를 위해 트랜잭션 간의 상호 작용을 제어하는 것

- **필요성** : 갱신 내용 손실(Lost Update), 모순성(Inconsistency), 연쇄적인 복귀(Cascading Rollback), 현행 파악 오류(Dirty Read)

• 종류

로킹(Locking) 기법	특정 트랜잭션이 데이터 항목에 대하여 잠금(Lock)을 설정하여, 잠금을 설정한 트랜잭션은 잠금을 해제(Unlock)할 때까지 데이터 항목을 독점적으로 사용하는 기법 • 로킹 단위 : 잠금 연산의 대상으로, 전체 데이터베이스부터 최소 단위인 속성까지 다양함
타임 스탬프 순서 (Time Stamp Ordering) 기법	시스템에서 생성하는 고유 번호인 타임 스탬프를 트랜잭션에 부여하는 기법
낙관적 기법	읽기 전용(Read Only) 트랜잭션이 대부분인 경우 트랜잭션 간의 충돌률이 매우 낮아 병행 제어를 하지 않아도 문제가 없는 점을 이용한 기법
다중 버전 기법	타임 스탬프 기법을 이용하며 버전을 부여하여 관리하는 기법

전체 백업 (Full Backup)	데이터 전체를 백업받는 방식
증분 백업 (Incremental Backup)	백업 대상 데이터 영역 중 변경되거나 증가된 데이터만을 백업받는 방식
차등 백업 (Differential Backup)	전체 백업 이후 변경 사항을 모두 백업받는 방식

▶ 장애

시스템이 제대로 동작하지 않는 상태

- **유형** : 트랜잭션 장애, 시스템 장애, 미디어 장애

▶ 회복(Recovery, 복구)

장애가 발생했을 때 데이터베이스를 장애가 발생하기 전의 상태로 복구시키는 것

- **로그를 이용한 회복 기법**

즉시 갱신(Immediate Update) 기법	트랜잭션 수행 도중에 데이터에 변경이 생기면 즉시 데이터베이스에 해당 변경 사항을 반영하는 기법
연기 갱신(Deferred Update) 기법	트랜잭션이 성공적으로 완료되기 전까지 실제 데이터베이스에 적용을 연기하는 기법

- **그 외 회복 기법**

검사점(Check Point) 기법	트랜잭션 실행 중 주기적으로 변경되는 내용이나 시스템 상황에 관한 정보와 함께 검사점을 로그에 보관하는 기법
미디어 회복(Media Recovery) 기법	전체 데이터베이스의 내용을 일정 주기마다 다른 안전한 저장 장치에 복사해두는 덤프(Dump)를 이용하는 기법
그림자 페이징(Shadow Paging) 기법	로그를 사용하지 않고 데이터베이스를 일정 크기의 페이지로 나누어 각 페이지마다 복사하여 그림자 페이지를 보관하는 기법

▶ 백업(Backup)

데이터베이스에 저장된 데이터를 임시로 복제해 두어, 전산 장비의 장애 발생 시 데이터를 보호하고 복구하기 위한 것

- **방식**

4과목

프로그래밍 언어 활용

▶ 프레임워크(Framework)

소프트웨어에서 특정 기능을 수행하기 위해 필요한 클래스나 인터페이스 등을 모아둔 집합체

- **특징** : 모듈화(Modularity), 재사용성(Reusability), 확장성(Extensibility), 제어의 역흐름(Inversion Of Control)

▶ 소프트웨어 개발 프레임워크

소프트웨어 개발에 공통적으로 사용되는 구성 요소와 아키텍처를 일반화하여 손쉽게 구현할 수 있도록 여러 가지 기능들을 제공해 주는 시스템

- **종류** : 스프링 프레임워크(Spring Framework), 전자정부 프레임워크, 닷넷 프레임워크(.NET Framework)

▶ 서버 개발 프레임워크

서버 프로그램 개발을 쉽게 처리할 수 있도록 여러 가지 기능들을 제공해 주는 시스템

- **종류** : 스프링(Spring), Node.js, 장고(Django), Ruby on Rails, 코드이그나이터(Codeigniter)

▶ 배치 프로그램(Batch Program)

일련의 작업을 정기적으로 반복 수행하거나 정해진 규칙에 따라 일괄 처리하는 프로그램

- **필수 요소** : 대용량 데이터, 자동화, 견고함, 안정성, 성능

▶ 데이터 타입(Data Type, 자료형)

변수에 저장될 데이터 형식

• C/C++ 데이터 타입

종류	데이터 타입	크기
정수형	short	2Byte
	int	4Byte
	long	4Byte
	long long	8Byte
부호 없는 정수형	unsigned short	2Byte
	unsigned int	4Byte
	unsigned long	4Byte
실수형	float	4Byte
	double	8Byte
	long double	8Byte
문자형	char	1Byte
부호 없는 문자형	unsigned char	1Byte

• JAVA 데이터 타입

종류	데이터 타입	크기
정수형	byte	1Byte
	short	2Byte
	int	4Byte
	long	8Byte
실수형	float	4Byte
	double	8Byte
문자형	char	1Byte
논리형	boolean	1Byte

• Python 데이터 타입

종류	데이터 타입	크기
정수형	int	제한 없음
실수형	float	8Byte
	complex	16Byte
문자형	str	제한 없음

▶ 변수(Variable)

프로그램에 필요한 값을 저장하기 위한 공간

• 변수명 작성 규칙

- 영문자, 숫자, _(언더스코어)를 사용한다.
- 첫 글자는 영문자 또는 _를 사용하며, 숫자는 올 수 없다.
- 공백이나 *, +, -, / 등의 특수문자를 사용할 수 없다.
- 대·소문자를 구분한다.
- 예약어를 변수명으로 사용할 수 없다.
- 변수 선언 시 문장 끝에 반드시 세미콜론(;)을 붙여야 한다.
- 글자 수에 제한이 없다.

• 변수 명명법(Casing)

- 카멜 케이스(Camel Casing) **예** camelCasing
- 파스칼 케이스(Pascal Casing) **예** PascalCasing
- 스네이크 케이스(Snake Casing) **예** snake_casing
- 헝가리안 표기법(Hungarian Notation)
예 strHungarianCasing
- GNU Naming Convention **예** gnu_naming_convention
- 상수 표기법 **예** MACRO_CASING

▶ 연산자

• 증가/감소 연산자

기호	사용법	의미
++	++A	(전치) A를 1 증가시킨 후 사용
	A++	(후치) A를 사용 후 1 증가
--	--A	(전치) A를 1 감소시킨 후 사용
	A--	(후치) A를 사용 후 1 감소

• 산술 연산자

기호	의미	예
+	덧셈	10+5=15
-	뺄셈	4-2=2
*	곱셈	6*3=18
/	나눗셈	8/2=4
%	나머지	5%2=1

• 논리 연산자

기호	사용법	의미
&&	A && B	A와 B를 AND 연산한다.
!!	A B	A와 B를 OR 연산한다.
!	!A	A를 NOT 연산한다.

• 관계 연산자

기호	사용법	의미
>	A > B	A가 B보다 크다.
>=	A >= B	A가 B보다 크거나 같다.
<	A < B	A가 B보다 작다.
<=	A <= B	A가 B보다 작거나 같다.
==	A == B	A와 B는 같다.
!=	A != B	A와 B는 같지 않다.

- **대입 연산자(할당 연산자)** : 대입 연산자에는 산술/관계/논리/비트 연산자 모두 사용할 수 있다.

기호	사용법	의미
+=	A += B	A = A + B
-=	A -= B	A = A - B
*=	A *= B	A = A * B
/=	A /= B	A = A / B

• 비트 연산자

기호	사용법	의미
&	피연산자1 & 피연산자2	피연산자1과 피연산자2를 각각 2진수로 표현한 후, 각 비트 단위로 AND 시킨다.
	피연산자1 피연산자2	피연산자1과 피연산자2를 각각 2진수로 표현한 후, 각 비트 단위로 OR 시킨다.
^	피연산자1 ^ 피연산자2	피연산자1과 피연산자2를 각각 2진수로 표현한 후, 각 비트 단위로 XOR 시킨다.
~	~피연산자	피연산자를 2진수로 표현한 후, 피연산자의 각 비트를 반전시킨다. 즉 1의 보수를 구한다.
<<	피연산자1 << 피연산자2	피연산자1을 2진수로 표현한 후, 피연산자2만큼 왼쪽으로 이동시킨 값이다.
>>	피연산자1 >> 피연산자2	피연산자1을 2진수로 표현한 후, 피연산자2만큼

		오른쪽으로 이동시킨 값이다.
--	--	-----------------

• 조건 연산자(삼항 연산자)

조건 ? 수식1 : 수식2;

- [조건]이 참(True)이면 [수식1]을 실행하고, 거짓(False)이면 [수식2]를 실행한다.

- **cast 연산자(형 변환 연산자)** : 데이터 타입을 변경한다.
- **sizeof 연산자** : 자료형의 크기를 Byte 단위로 변환한다.

▶ 연산자 우선순위

우선 순위	연산자	설명	결합법칙
1	++ -- ! ~ (자료형) sizeof	단항	←
2	* / %	산술	→
3	+ -		
4	<< >>	시프트	
5	< <= > >=	관계	
6	== !=		
7	& ^	비트	
8	&&	논리	
9	? :	조건(삼항)	
10	= += -= *= /= 등	대입(할당)	←
11	,	침표	→

▶ 출력 함수 printf()

printf(서식 문자열, 변수);

- **서식 문자열** : 변수의 데이터 타입에 맞는 서식 문자열을 입력
- **변수** : 서식 문자열의 순서에 맞게 출력할 변수를 입력

▶ 입력 함수 scanf()

scanf(서식 문자열, 변수의 주소);

- **서식 문자열** : 변수의 데이터 타입에 맞는 서식 문자열
- **변수의 주소** : 데이터를 입력받아 저장할 변수를 입력
 - 단, 변수의 주소로 입력받아야 하므로 변수에 주소 연산자 &를 붙임

▶ 서식 문자열

서식 문자	의미	설명
%o	octal	8진수 정수
%d	decimal	10진수 정수
%x	hexadecimal	16진수 정수
%c	character	문자
%s	string	문자열

▶ 제어 문자

제어 문자	의미	설명
\n	new line	커서를 다음 줄로 바꿈(개행)
\r	carriage return	커서를 그 줄의 맨 앞으로 이동
\f	form feed	한 페이지를 넘김
\b	backspace	커서를 그 줄의 한 문자만큼 앞으로 이동
\t	tab	커서를 그 줄의 tab만큼 이동

▶ 제어문

프로그래밍 언어에서 조건에 따라 실행해야 할 문장을 제어하거나 실행 순서를 변경시키기 위해 사용하는 것

• 종류

선택적 실행문	if, if~else, 다중 if, switch~case
반복적 실행문	while, do~while, for
그 외	break, continue, goto

▶ if문

```
if(조건) {
    문장;
}
```

- 조건이 참인 경우 문장을 실행

▶ if~else문

```
if(조건) {
    문장1;
}
else {
    문장2;
}
```

- 조건이 참인 경우에는 문장1을 실행, 조건이 거짓인 경우에는 문장2를 실행

▶ 다중 if문

```
if(조건1) {
    문장1;
}
else if(조건2) {
    문장2;
}
else if(조건3) {
    문장3;
}
else {
    문장4;
}
```

- 조건에 따라 문장을 실행하고, 모든 조건에 만족하지 않으면 else의 문장4를 실행

▶ switch~case문

```
switch (조건) {
    case 조건값1:
        문장1;
        break;
    case 조건값2:
        문장2;
        break;
    case 조건값3:
        문장3;
        break;
    default:
        문장4;
        break;
}
```

- 조건값에는 한 개의 상수만 지정할 수 있으며 int, char, enum형의 상수값만 가능
- break : switch~case문 종료 문법

▶ while문

```
while(조건) {
    문장;
}
```

- 조건이 참인 경우 문장을 반복 실행

▶ do~while문

```
do {
    문장;
} while(조건)
```

- 조건이 참인 경우 문장을 반복 실행
- 조건을 먼저 판단하는 while문과는 달리, 문장을 우선 한 번 실행한 후 조건을 판단

▶ for문

```
for(초기값; 조건; 증감값) {
    문장;
}
```

- 초기값 : for문 실행 시 가장 먼저 한 번만 실행되는 값
- 조건 : 조건이 참이면 문장을 반복 실행
- 증감값 : 문장 실행 후 증감식을 실행
- for문 실행 순서 : 초기값 → 조건 → 문장 → 증감값

▶ 무한 반복(무한 루프)

```
① while(true)
② while(1)
③ for(;;)
```

- ① : while문의 조건식이 true인 경우 무한 반복
- ② : while문의 조건식이 1인 경우 무한 반복
- ③ : for문의 초기값, 조건, 증감값을 생략하면 무한 반복

▶ break문

for문, while문, do~while문, switch~case문의 제어를 벗어나기 위해 사용하는 명령문

- 가장 가까운 곳에 있는 하나의 루프만 벗어남

▶ continue문

for문, while문, do~while문에서 다음 반복을 실행하기 위해 사용하는 명령문

- continue문 다음 문장을 실행하지 않고, 바로 그 루프의 선두로 되돌아가서 실행

▶ goto문

```
goto 레이블명;
레이블명:
문장;
```

- 지정된 레이블(Label)로 무조건 분기하는 명령문
- 레이블(Label) : 실행하고자 하는 일부 문장들의 집합으로, 레이블명은 임의로 지정

▶ 배열(Array)

한 가지 데이터 타입을 연속적으로 나열한 것

• 선언

```
① 데이터타입 배열명[배열의크기];
   배열명[인덱스] = 값;
② 데이터타입 배열명[배열의크기] = {값1, 값2, ...};
```

- 데이터타입 : 배열에 저장될 자료형
- 배열명 : 배열 이름을 임의로 지정
- 배열의크기 : 배열에 저장될 최대 요소 개수
- 인덱스 : 배열에서 특정 위치

- 값 : 배열에 저장할 값

• 사용

```
배열명[인덱스];
```

예 int a[3] = {19, 21, 18};

a[0]	a[1]	a[2]
19	21	18

▶ 2차원 배열

행과 열로 조합한 배열

• 선언

```
① 데이터타입 배열명[행개수][열개수];
   배열명[행][열] = 값;
② 데이터타입 배열명[행개수][열개수] = {{값1, 값2, ...}, {값3, 값4, ...}};
```

- 데이터타입 : 배열에 저장될 자료형
- 배열명 : 배열 이름을 임의로 지정
- 행개수, 열개수 : 배열의 행 크기와 열 크기
- 행, 열 : 배열에서 특정 위치
- 값 : 배열에 저장할 값

예 char ch[2][2] = {{'a', 'b'}, {'c', 'd'}};

↓ a[0][0]	↓ a[0][1]
a	b
c	d
↑ a[1][0]	↑ a[1][1]

▶ 구조체(Struct)

관련 정보를 하나의 의미로 묶어 데이터를 체계적으로 관리하기 위한 것

• 선언

```
struct 구조체명{
    구조체멤버
};
```

- 구조체명 : 구조체 이름을 임의로 지정
- 구조체멤버 : 구조체에 저장할 변수

• 사용

```
① struct 구조체명 구조체변수;
② struct 구조체명 구조체변수 = {값};
```

- 구조체명 : 구조체 이름
- 구조체변수 : 구조체 멤버에 접근하기 위한 변수
- 값 : 구조체에 저장할 값

```
예 struct PERSON {
    char name[50];
    int age;
    char address[100];
};

struct PERSON p1 = {"기사퍼스트", 10, "대한민국"};
```


	p1
p1.name	"기사퍼스트"
p1.age	10
p1.address	"대한민국"

▶ 포인터(Pointer)

변수의 메모리 주소

- **포인터 변수** : 메모리 주소를 저장하기 위한 변수

데이터타입 *포인터변수명

- **주소 연산자(&)** : 변수의 메모리 주소를 반환하는 연산자
- **간접 참조 연산자(*)** : 변수가 가리키는 메모리 주소의 값을 반환하는 연산자

예) `int a = 10;`

10	100
100번지	200번지

`int *p;`

10	100
100번지	200번지

`p = &a;`

20	100
100번지	200번지

`*p = *p + 10;`

▶ 함수(Function)

특정한 목적의 작업을 수행하기 위한 프로그램 코드의 집합

- **종류** : 표준 함수, 사용자 정의 함수
- **사용자 정의 함수**

반환형 함수명(매개변수)

```
{
    수행할 동작
    return;
}
```

- 반환형(리턴형) : 함수가 반환하는 값의 데이터 타입
- 함수명 : 함수 이름을 임의로 지정
- 매개변수(Parameter, 파라미터) : 함수가 전달받는 값을 저장하는 변수
- return : 반환 값

예) `int add(int x, int y) {`
`int z = x + y;`
`return z;`
`}`

▶ 클래스(Class)

for문, while문, do~while문, switch~case문의 제어를 벗어나기 위해 사용하는 명령문

- 가장 가까운 곳에 있는 하나의 루프만 벗어남

▶ continue문

for문, while문, do~while문에서 다음 반복을 실행하기 위해 사용하는 명령문

- continue문 다음 문장을 실행하지 않고, 바로 그 루

프의 선두로 되돌아가서 실행

▶ JAVA언어 기본 구조

• 출력 함수 System.out.print()

- ① System.out.print(출력값);
- ② System.out.println(출력값);
- ③ System.out.printf(서식 문자열, 변수);

- 출력값 : 숫자, 문자 등의 값이나 변수, 식 입력
- 서식 문자열 : 변수의 데이터 타입에 맞는 서식 문자열 입력
- 변수 : 서식 문자열의 순서에 맞게 출력할 변수 입력
- ①, ③ : 자동 개행 X, ② : 자동 개행 O

• 입력 함수 Scanner 클래스

Scanner 객체변수명 = new Scanner(System.in);

- 객체변수명 : 데이터를 입력받아 저장할 객체 변수 입력

• 클래스(Class)

접근제한자 class 클래스명

- 접근제한자 : 내·외부로부터 클래스 멤버에 대한 접근 범위 설정
- 클래스명 : 클래스 이름을 임의로 지정

• 객체(Object)

클래스명 객체변수명 = new 클래스명();

- 클래스명 : 객체를 생성할 클래스명
- 객체변수명 : 객체 변수 이름을 임의로 지정

• 생성자(Constructor) : 초기화 작업을 위한 함수

```
public 클래스명(매개변수) {
    수행할 동작
}
```

- 클래스명 : 생성자를 포함하는 클래스명
- 매개변수(Parameter, 파라미터) : 생성자가 전달받는 값을 저장하는 변수

• 접근 제한자(접근 제어자)

접근 범위	public	protected	default	private
자신을 포함한 클래스	O	O	O	O
동일 패키지	O	O	O	X
상속받은 클래스	O	O	X	X
모든 영역	O	X	X	X

• 배열(Array)

- ① 데이터타입 배열명[] = new 데이터타입[배열의크기];
배열명[인덱스] = 값;
- ② 데이터타입[] 배열명 = new 데이터타입[배열의크기];
배열명[인덱스] = 값;

③ 데이터타입 배열명[] = {값1, 값2, ...};
 ④ 데이터타입[] 배열명 = {값1, 값2, ...};

- 데이터타입 : 배열에 저장될 자료형
- 배열명 : 배열 이름을 임의로 지정
- 배열크기 : 배열에 저장될 최대 요소 개수
- 인덱스 : 배열에서 특정 위치
- 값 : 배열에 저장할 값

• 문자열

① String 변수명;
 변수명 = "문자열";
 ② String 변수명 = "문자열";

- 변수명 : 문자열을 저장할 변수명을 임의로 지정
- 문자열 : 반드시 큰따옴표(" ")를 사용하여 문자열 표현
- **개선된 for문** : 배열의 요소를 순서대로 변수에 저장하며 문장을 반복 실행

```
for(데이터타입 변수명:배열명) {
    문장;
}
```

- 데이터타입 : 배열의 자료형
- 변수명 : 배열의 요소를 저장할 변수명
- 배열명 : 사용할 배열명

▶ Python언어 기본 구조

• 출력 함수 print()

① print(출력값, sep = 분리문자, end = 종료문자)
 ② print(서식 문자열 % 변수)

- 출력값 : 숫자, 문자 등의 값이나 변수, 식 입력
- sep : 여러 값을 출력할 때 값과 값 사이를 구분하기 위한 문자
- end : 맨 마지막에 표시할 문자
- 서식 문자열 : 변수의 데이터 타입에 맞는 서식 문자열 입력
- 변수 : 서식 문자열의 순서에 맞게 출력할 변수 입력

• 입력 함수 input()

① 변수명 = input(출력문자)
 ② 변수명1, 변수명2, ... = input(출력문자).split(분리문자)

- 변수명 : 데이터를 입력받아 저장할 변수 입력
- 출력문자 : 데이터를 입력받기 전 화면에 출력할 문자 입력
- 분리문자 : 여러 값을 입력할 때 값과 값 사이를 구분하기 위한 문자

• 인덱싱 : 특정 위치의 값을 반환

변수명[인덱스]

- **슬라이싱** : 지정한 시작 위치에서 지정한 끝 위치까지를 반환

변수명[시작인덱스:끝인덱스]

• if문

```
if 조건1:
    문장1
elif 조건2:
    문장2
elif 조건3:
    문장3
else:
    문장4
```

- 조건에 따라 문장을 실행하고, 모든 조건에 만족하지 않으면 else의 문장4를 실행

• for문

```
for 변수명 in range(초기값, 최종값, 증감값):
    문장
```

- 변수명 : 변수 이름을 임의로 지정
- 초기값 : 변수에 처음으로 저장할 초기값
- 최종값 : 변수에 마지막으로 저장할 최종값
- 증감값 : 문장 실행 후 증감 값 실행

• 개선된 for문

```
for 변수 in 리스트명:
    문장
```

- 변수명 : 리스트의 요소를 저장할 변수 이름
- 리스트명 : 사용할 리스트 이름

• 함수(Function)

```
def 함수명(매개변수):
    수행할 동작
    return
```

- 함수명 : 함수 이름을 임의로 지정
- 매개변수(Parameter, 파라미터) : 함수가 전달받는 값을 저장하는 변수
- return : 반환 값

• 클래스(Class)

```
class 클래스명:
```

- 클래스명 : 클래스 이름을 임의로 지정

• 객체(Object)

```
변수명 = 클래스명();
```

- 변수명 : 변수 이름을 임의로 지정
- 클래스명 : 객체를 생성할 클래스명

• 생성자(Constructor)

```
class 클래스명:
    def __init__(self, 매개변수):
        수행할 동작
```

- 클래스명 : 생성자를 포함하는 클래스명
- 매개변수(Parameter, 파라미터) : 생성자가 전달받는 값을 저장하는 변수

▶ 프로그래밍 언어

고급 프로그래밍 언어	사람이 이해하기 쉽게 작성된 프로그래밍 언어 • 예 C, JAVA, Python, Basic 등
저급 프로그래밍 언어	컴퓨터가 이해하기 쉽게 작성된 프로그래밍 언어 • 예 기계어, 어셈블리어

- **컴파일러(Compiler), 인터프리터(Interpreter)** : 고급 프로그래밍 언어를 기계어로 변환하는 기능을 수행
- 컴파일러는 전체를 번역, 인터프리터는 줄 단위로 번역

▶ 절차적 프로그래밍 언어

일련의 처리 절차를 정해진 문법에 따라 순서대로 기술해 나가는 언어

- **종류** : C, ALGOL, COBOL, FORTRAN, BASIC

▶ 객체 지향 프로그래밍 언어

현실 세계의 개체(Entity)를 기계의 부품처럼 하나의 객체(Object)라는 기본 단위로 나뉘, 이 객체들의 상호작용으로 프로그래밍하는 방식으로 기술해 나가는 언어

- **종류** : JAVA, C++, C#, Smalltalk

▶ 스크립트 언어

응용 소프트웨어를 제어하는 컴퓨터 프로그래밍 언어

- **종류** : Python, PHP, ASP, JSP, JCL, JavaScript

▶ 명령형 언어

프로그램이 어떤 방법으로 해야 하는지를 나타내기보다 무엇(WHAT)과 같은지에 초점을 두는 언어

- **종류** : 절차적 언어, 객체 지향 언어

▶ 선언형 언어

순차적인 명령 수행을 기본으로, 어떤 방법으로(HOW) 문제 처리를 할 것인지에 초점을 두는 언어

- **종류** : HTML, LISP, PROLOG, XML, Haskell, SQL

▶ 라이브러리(Library)

효율적인 프로그램 개발을 위해 필요한 함수, 데이터 등 프로그램을 모아 놓은 집합체

• 구분

표준 라이브러리	<ul style="list-style-type: none"> • 프로그래밍 언어가 기본적으로 가지고 있는 라이브러리 • 별도의 파일 설치 없이 일반적으로 많이 사용하는 날짜와 시간 등의 기능을 이용 가능
외부 라이브러리	<ul style="list-style-type: none"> • 별도의 파일을 설치하여야 하는 라이브러리 • 누구나 개발하여 설치할 수 있으며, 인터넷 등을 이용하여 공유 가능

- **C 표준 라이브러리** : stdio.h, string.h, math.h, stdlib.h, time.h
- **JAVA 표준 라이브러리** : java.lang, java.util, java.io, java.sql, java.net, java.awt
- **Python 표준 라이브러리** : string, re, math, random, datetime, logging

▶ 운영체제(OS; Operating System)

사용자가 컴퓨터의 하드웨어를 쉽게 사용할 수 있도록 인터페이스를 제공해 주는 소프트웨어

• 목적

- 처리 능력(Throughput) 향상
- 신뢰도(Reliability) 향상
- 사용 가능도(Availability) 향상
- 응답시간(반환시간, Turn Around Time) 단축

- **종류** : MS-DOS, Windows, MacOS, Windows NT, UNIX, LINUX, Android, iOS 등

▶ 운영체제의 기능적 분류

- **제어 프로그램** : 시스템 전체의 작동 상태 감시, 작업의 순서 지정, 작업에 사용되는 데이터 관리 등의 역할을 수행하는 프로그램
- 종류 : 감시 프로그램, 작업 제어 프로그램, 데이터 관리 프로그램
- **처리 프로그램** : 제어 프로그램의 지시를 받아 사용자가 요구한 문제를 해결하기 위한 프로그램
- 종류 : 서비스 프로그램, 문제 프로그램, 언어 번역 프로그램

▶ Windows

당시 널리 쓰이던 MS-DOS에서 멀티태스킹과 GUI 환경을 제공하기 위한 응용 프로그램으로 출시된 운영체제

• 특징

- GUI(Graphic User Interface, 그래픽 사용자 인터페이스)
- 선점형 멀티태스킹(Preemptive MultiTasking, 양보)
- PnP(Plug and Play, 자동 감지 기능)
- OLE(Object Linking and Embedding, 개체 연결 및 삽입)
- Single-User 시스템

▶ UNIX(유닉스)

1960년대 말에 미국 AT&T 벨(Bell) 연구소에서 개발한 운영체제

- **특징** : 개방형 시스템, 다중 사용자 시스템, 대화식 시분할 시스템, 다중 작업(멀티태스킹), 계층(트리) 구조의 파일 시스템, 다양한 네트워킹 기능

• 구성

사용자
셸
커널
하드웨어

- 커널(Kernel) : UNIX의 가장 핵심적인 부분으로, 주기억장치에 적재된 후 상주하며 실행
- 셸(Shell) : 시스템과 사용자 간의 인터페이스를 담당하는 명령어 해석기

▶ LINUX(리눅스)

1991년 리누스 토발즈(Linus Torvalds)가 만든 운영체제

- 특징 : 오픈 소스 소프트웨어, UNIX와 호환 가능

▶ Mac OS(매킨토시 OS)

UNIX 기반으로 만들어져 애플(Apple)의 제품군에서만 사용이 가능한 그래픽 기반 운영체제

▶ 안드로이드(Android)

리눅스 커널 위에서 동작하며 자바 및 코틀린 언어로 앱을 만들어 작동하는 휴대 전화나 소형 기기에서 사용되는 운영체제

- 구글이 공개한 개방형 모바일 운영체제

▶ iOS

OS X을 기반으로 만들어져 있고, 멀티 터치를 비롯한 스마트 폰에는 없었던 사용자 인터페이스로 구현한 운영체제

- 애플의 모바일 운영체제

▶ 기억장치 관리 전략

- 반입(Fetch) 전략 : 보조기억장치의 프로그램이나 데이터를 언제 주기억장치로 적재할 것인지를 결정

요구 반입	요구할 때 적재하는 방법
예상 반입	미리 예상하여 적재하는 방법

- 배치(Placement) 전략 : 보조기억장치의 프로그램이나 데이터를 주기억장치의 어디에 위치시킬 것인지를 결정

최초 적합 (First Fit)	첫 번째에 배치시키는 방법
최적 적합 (Best Fit)	단편화를 가장 작게 남기는 분할 영역에 배치시키는 방법
최악 적합 (Worst Fit)	단편화를 가장 많이 남기는 분할 영역에 배치시키는 방법

- 교체(Replacement) 전략 : 주기억장치의 이미 사용되고 있는 영역 중에서 어느 영역을 교체하여 사용할 것인지를 결정

- 종류 : FIFO, OPT, LRU, LFU, NUR, SCR 등

▶ 주기억장치 할당 기법

- 연속 할당 기법 : 프로그램을 주기억장치에 연속으로 할당하는 기법
 - 종류 : 단일 분할 할당(단일 프로그래밍), 다중 분할 할당(다중 프로그래밍)

단일 분할 할당 (단일 프로그래밍)	<ul style="list-style-type: none"> • 스와핑(Swapping) : 하나의 프로그램 전체를 주기억장치에 할당하여 사용하다가, 필요에 따라 다른 프로그램과 교체하는 기법 • 오버레이(Overlay) : 프로그램의 모든 부분이 동시에 주기억장치에 상주해 있을 필요가 없으므로 작업을 분할하여 필요한 부분만 교체하는 기법
다중 분할 할당 (다중 프로그래밍)	<ul style="list-style-type: none"> • 고정 분할(정적 분할) : 주기억장치를 미리 여러 개의 고정된 크기로 분할하고, 프로그램을 각 영역에 할당하는 기법 • 가변 분할(동적 분할) : 프로그램을 주기억장치에 적재하면서 필요한 만큼의 크기로 영역을 분할하는 기법

- 분산 할당 기법(가상기억장치 기법) : 프로그램을 특정 단위의 조각으로 나누어 주기억장치 내에 분산하여 할당하는 기법

- 종류

페이징 (Paging)	가상기억장치에 보관되어 있는 프로그램과 주기억장치의 영역을 동일한 크기로 나눈 후, 나뉜 프로그램을 주기억장치의 영역에 동일하게 적재시켜 실행하는 기법 <ul style="list-style-type: none"> • 단위 : 페이지(Page)
세그멘테이션 (Segmentation)	가상기억장치에 보관되어 있는 프로그램을 다양한 크기의 논리적인 단위로 나눈 후, 주기억장치에 적재시켜 실행시키는 기법 <ul style="list-style-type: none"> • 단위 : 세그먼트(Segment)

▶ 페이지 교체(Replacement) 알고리즘

페이지 부재(Page Fault)가 발생할 경우, 가상기억장치의 필요한 페이지를 주기억장치의 어떤 페이지 프레임을 선택하여 교체해야 하는가를 결정하는 기법

- 종류

FIFO (First In First Out)	가장 먼저 들어온 페이지를 먼저 교체시키는 기법
------------------------------	----------------------------

OPT (OPTimal Replacement)	앞으로 가장 오랫동안 사용하지 않을 페이지를 교체하는 기법
LRU (Least Recently Used)	최근에 가장 오랫동안 사용하지 않은 페이지를 교체하는 기법
LFU (Least Frequently Used)	사용 횟수가 가장 적은 페이지를 교체하는 기법
NUR (Not Used Recently)	최근에 사용하지 않은 페이지를 교체하는 기법

▶ 워킹 셋(Working Set)

프로세스가 일정 시간 동안 자주 참조하는 페이지들의 집합

- 자주 참조되는 워킹 셋을 주기억장치에 상주시킴으로써, 페이지 부재 및 페이지 교체 현상을 줄임

▶ 스래싱(Thrashing)

프로세스의 처리 시간보다 페이지 교체 시간이 더 많아져 CPU 이용률이 저하되는 현상

• **방지 방법** : 다중 프로그래밍의 정도를 줄임, CPU 이용률을 높임, 워킹 셋 방법 사용

▶ 구역성(Locality, 국부성),

참조 국부성(Locality Of Reference)

프로세스가 실행되는 동안 일부 페이지만 집중적으로 참조하는 성질

• 구분

시간 구역성	최근에 참조된 기억장소가 가까운 미래에도 계속 참조될 가능성이 높다.
공간 구역성	하나의 기억장소가 참조되면 그 근처의 기억장소가 계속 참조될 가능성이 높다.

▶ 프로세스(Process)

주기억장치에 저장된 현재 실행 중인 프로그램

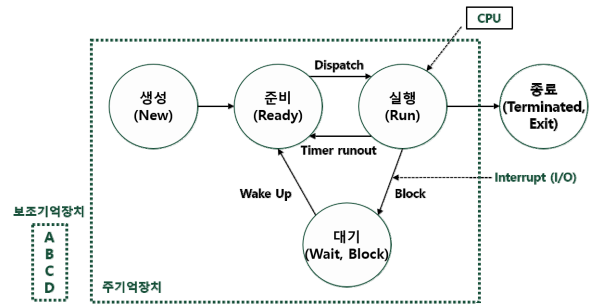
• **스레드(Thread)** : 프로세스를 분할하여 하나의 프로세스 내에 병행성을 증대시키기 위한 기법

- 경량 프로세스

• **프로세스 제어 블록(PCB; Process Control Block)** : 운영체제가 프로세스에 대한 중요한 정보를 저장해 놓은 곳

- 저장 정보 : 프로세스의 현재 상태, 프로세스 우선 순위, 프로세스 식별자(고유번호), 레지스터 저장 장소, 관련 레지스터 정보, 할당된 자원에 대한 포인터

▶ 프로세스 상태 전이도



• 상태

생성 상태 (New)	프로세스가 막 생성된 상태
준비 상태 (Ready)	프로세스가 CPU를 사용하여 실행될 수 있는 상태
실행 상태 (Run)	프로세스가 CPU를 차지하여 실행 중인 상태
대기 상태 (Wait, Block)	어떤 사건이 발생하기를 기다리는 상태
종료 상태 (Terminated, Exit)	프로세스가 CPU를 할당받아 주어진 시간 내에 완전히 수행을 종료한 상태

• 프로세스 상태 관련 용어

디스패치 (Dispatch)	준비상태에 있는 여러 프로세스 중 프로세스를 선정하여 CPU를 할당하는 것
시간 만료 (Timer Runout)	CPU를 할당받아 실행 중인 프로세스가 할당 시간을 초과하면, CPU를 다른 프로세스에게 양도하고 자신은 준비 상태로 전이되는 것
인터럽트 (Interrupt)	예기치 않은 일, 응급 사태 등 어떠한 특수한 상태가 발생하면, 현재 실행하고 있는 프로그램이 일시 중단되고, 그 특수한 상태를 처리하는 프로그램으로 옮겨져 처리한 후 다시 원래의 프로그램을 처리하는 현상
Block	실행 중인 프로세스가 지정된 시간 이전에 다른 작업을 위해 스스로 프로세서를 양도하고 대기 상태로 전이되는 것
Wake Up	실행 중인 프로세스가 완료되어 대기 중인 프로세스를 준비 상태로 전이하는 것
교통량 제어기 (Traffic Controller)	프로세스의 상태에 대한 조사와 통보 담당
스풀링 (Spooling)	다중 프로그래밍 환경에서 디스크를 이용한 저속의 입-출력 장치와 고속의 CPU 간의 속도 차이를 해소하기 위한 방법 • 디스크 일부를 버퍼처럼 사용

▶ 프로세스 스케줄링(CPU 스케줄링)

주 기억장치에 저장되어 있는 프로그램의 CPU 사용 순서를 결정하기 위한 정책

• 종류

비선점 (Non Preemptive) 스케줄링	프로세스에게 이미 할당된 CPU를 강제로 빼앗을 수 없고, 사용이 끝날 때까지 기다려야 하는 방법 • 대표 기법 : FIFO, SJF, HRN
선점 (Preemptive) 스케줄링	우선순위가 높은 다른 프로세스가 할당된 CPU를 강제로 빼앗을 수 있는 방법 • 대표 기법 : RR, SRT

• FIFO(First-In First-Out) = FCFS(First-Come

First-Service) : 준비 상태에서 도착한 순서에 따라 CPU를 할당하는 기법

- 평균 실행시간 : 프로세스의 평균 실행시간
- 평균 대기시간 : 각 프로세스가 실행되기까지의 평균 대기시간
- 평균 반환시간 : 프로세스의 대기시간과 실행시간의 합(평균 반환시간 = 평균 실행시간 + 평균 대기시간)

• SJF(Shortest Job First) : 작업이 끝나기까지의 실행시간 추정치가 가장 작은 작업을 먼저 실행하는 기법

• HRN(Highest Response ratio Next) : 우선순위를 부여하고 그 중 가장 높은 프로세스에게 먼저 CPU를 할당하는 기법

- 우선순위 계산식 : (대기시간 + 서비스시간) / 서비스시간

• RR(Round Robin, 라운드 로빈) : 대화식 시분할 시스템을 위해 고안된 방식으로, Time Slice를 지정하여 할당하는 기법

- FIFO 기법의 선점형 기법

• SRT(Shortest Remaining Time) : 현재 실행 중인 프로세스의 남은 시간과 준비상태 큐에 새로 도착한 프로세스의 실행시간을 비교하여 가장 짧은 실행시간을 요구하는 프로세스에게 CPU를 할당하는 기법

- SJF 기법의 선점형 기법

▶ 교착상태(Dead Lock)

둘 이상의 프로세스들이 자원을 점유한 상태에서 서로 다른 프로세스가 점유하고 있는 자원을 요구하며 무한정 기다리는 현상

• 4가지 필요충분조건

- 상호 배제(Mutual Exclusion)
- 점유와 대기(Hold & Wait)
- 비선점(Nonpreemption)
- 환형 대기(Circular Wait)

• 해결 방안

- 예방 기법(Prevention)
- 회피 기법(Avoidance)

- 발견 기법(Detection)

- 회복 기법(Recovery)

▶ Windows 기본 명령어

dir	파일 목록 표시(directory)
copy	파일 복사
type	파일 내용 확인
ren	파일 이름 변경(rename)
del	파일 삭제(delete)
attrib	파일의 속성 변경(attribute)
find	파일 찾기
move	파일 이동
comp	파일 비교(compare)
md	디렉터리 생성(make directory)
cd	디렉터리 위치 변경(change directory)
chkdsk	디스크 상태 점검(check disk)
format	디스크 초기화
cls	화면 내용 지움(clear screen)
exit	명령 프롬프트 종료

▶ Unix / Linux 주요 명령어

fork	프로세스 생성
exec	프로세스 실행(execute)
wait	부모 프로세스가 자식 프로세스 종료를 기다리며 일시 중지
kill	프로세스 제거
ps	현재 프로세스 상태 확인(process status)
getpid	자신의 프로세스 아이디 확인
getppid	부모 프로세스 아이디 확인
chmod	파일 접근 권한 모드 설정(change mode)
chown	파일 소유자 변경(change owner)
cat	파일 내용을 화면에 표시
grep	파일 내용에서 지정한 문자열 찾기
cp	파일 복사(copy)
find	파일 찾기
rm	파일 삭제(remove)
ls	디렉터리 내용 보기(list)
mount	새로운 파일 시스템을 기존 파일 시스템의 서브 디렉터리에 연결
unmount	새로운 파일 시스템을 기존 파일 시스템의 서브 디렉터리에서 해제
chdir	디렉터리 위치 변경(change directory)
fsck	파일 시스템 검사 및 보수(file system check)
uname	시스템 정보(커널 이름, 커널 버전, 사용자 이름 등) 출력
who	현재 시스템에 접속한 사용자 정보 출력
sleep	지정한 시간동안 대기

▶ 셸 스크립트(Shell Script)

셸(Shell)에서 사용할 수 있는 명령어들의 조합을 모아
서 만든 파일

• 셸 스크립트의 제어문 종류

선택적 실행문	if, case
반복적 실행문	for, while, until, select

예) wow 사용자가 로그인할 때까지 반복문을 수행하
는 셸 스크립트

```
until who | grep wow
do
sleep 5
done
```

▶ IP 주소(Internet Protocol Address)

인터넷에 연결된 모든 컴퓨터의 자원을 구분하기 위한
고유한 주소

- **IPv4(IP version 4)** : 8비트씩 4부분, 총 32비트로 구
성
 - 각 부분을 마침표(.)로 구분하여 표현하고, 각 구분
은 10진수로 표현
 - 총 5개의 클래스로 나뉨

A 클래스	<ul style="list-style-type: none"> • 국가나 대형 통신망에 사용 • 시작 주소 : 0~127 • 연결 가능 호스트 수 : 256X256X256
B 클래스	<ul style="list-style-type: none"> • 중대형 통신망에 사용 • 시작 주소 : 128~191 • 연결 가능 호스트 수 : 256X256
C 클래스	<ul style="list-style-type: none"> • 소규모 통신망에 사용 • 시작 주소 : 192~223 • 연결 가능 호스트 수 : 256
D 클래스	<ul style="list-style-type: none"> • 멀티캐스트용으로 사용 • 시작 주소 : 224~239
E 클래스	<ul style="list-style-type: none"> • 실험적 주소로 공용되지 않음

▶ IPv6(Internet Protocol version 6)

IPv4의 주소 고갈 문제를 해결하기 위해 기존의 IPv4
주소 체계를 128비트 크기로 확장한 인터넷 프로토콜
주소

- 16비트씩 8부분, 총 128비트로 구성
- 각 부분을 콜론(:)으로 구분하여 표현하고, 각 구분은
16진수로 표현
- **주소 체계** : 유니캐스트(Unicast), 멀티캐스트(Mul
ticast), 애니캐스트(Anycast)

▶ 서브넷(Subnet)

IP 주소에서 하나의 네트워크가 분할되어 나뉘진 작은
네트워크

네트워크 주소	호스트 주소
---------	--------



네트워크 주소	서브넷 주소	호스트 주소
---------	--------	--------

- **서브넷 마스크(Subnet Mask, 마스크)** : IP 주소에서
네트워크 주소와 호스트 주소를 구별하는 방식
 - 4Byte(32bit)의 IP 주소 중 네트워크 주소와 호스트
주소를 구분하기 위한 비트
 - IP 주소와 서브넷 마스크를 AND 연산하여 네트워
크 주소를 얻음

구분	서브넷 마스크
A 클래스	255.0.0.0
B 클래스	255.255.0.0
C 클래스	255.255.255.0

▶ OSI 7계층(Open System Interconnection

7 Layer, OSI 참조 모델)

다른 시스템 간의 원활한 통신을 위해 국제표준화기구
(ISO)에서 제안한 7단계의 표준화 프로토콜

• 구조

OSI 7계층		
상위 계층	Layer 7	응용 계층
	Layer 6	표현 계층
	Layer 5	세션 계층
	Layer 4	전송 계층
하위 계층	Layer 3	네트워크 계층
	Layer 2	데이터 링크 계층
	Layer 1	물리 계층

- **물리 계층(Physical Layer)**
 - 물리적인 연결 방식 규정
 - 매체 간의 전기적, 기능적, 절차적 기능 및 인터페
이스 정의
 - PDU : 비트(Bit)
- **데이터 링크 계층(Data Link Layer)**
 - 두 컴퓨터 간 데이터 통신 규정
 - 흐름 제어, 프레임 동기화, 오류 제어, 에러 검출
및 정정, 순서 제어
 - PDU : 프레임(Frame)
- **네트워크 계층(Network Layer)**
 - 네트워크 연결을 설정, 유지, 해제하는 기능 및 데
이터 통신 규정
 - 교환 기술, 경로 설정, 패킷 정보 전송
 - PDU : 패킷(Packet)
- **전송 계층(Transport Layer)**
 - 종단 시스템(End-to-End) 간에 신뢰성 있는 데이터
전송을 하기 위한 규정
 - 주소 설정, 다중화, 오류 제어, 흐름 제어
 - PDU : 세그먼트(Segment)

• 세션 계층(Session Layer)

- 데이터 교환 관리 및 대화 제어를 위한 규정
- 전송하는 정보의 일정한 부분에 동기점을 두어 대화(회화) 동기 조절
- PDU : 메시지(Message, Data)

• 표현 계층(Presentation Layer)

- 데이터 변환 및 데이터 표현 규정
- 코드 변환, 구문 검색, 암호화, 형식 변환, 압축
- PDU : 메시지(Message, Data)

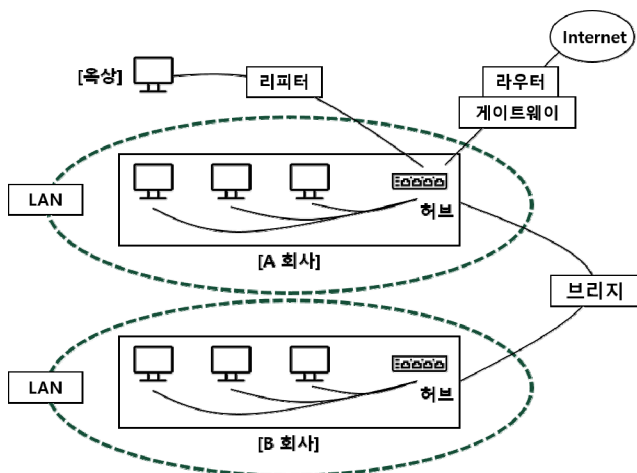
• 응용 계층(Application Layer)

- 사용자가 OSI 7계층 환경에 접근할 수 있도록 서비스 제공
- PDU : 메시지(Message, Data)

▶ OSI 7계층의 계층별 표준 프로토콜

응용 계층	HTTP, TELNET, FTP, DNS, DHCP, SNMP, SMTP, SSH, POP3
표현 계층	ASCII, MIME, JPEG, MPEG, SSL, TLS
세션 계층	SSL, TLS
전송 계층	TCP, UDP, RTP, RTCP, SCTP
네트워크 계층	X.25, IP, ICMP, IGMP, ARP, RARP
데이터 링크 계층	HDLC, LAPB, LLC, ADCCP, BSC, ISDN, PPP, Ethernet, ATM, ARQ, IEEE 802
물리 계층	RS-232C, X.21

▶ 네트워크 장비



허브 (Hub)	여러 대의 컴퓨터를 연결하여 네트워크로 보내거나, 하나의 네트워크로 수신된 정보를 여러 대의 컴퓨터로 송신하기 위한 장비 • OSI 1계층 장비
리피터 (Repeater)	장거리 데이터 전송에서 신호를 증폭시키는 장비 • OSI 1계층 장비
브리지 (Bridge), 스위치 (Switch)	두 개의 LAN을 연결하여 훨씬 더 큰 LAN을 만들기 위한 장비 • OSI 2계층 장비

라우터 (Router)	네트워크 계층에서 연동하여 경로를 설정하고 전달하는 기능을 제공하는 장비 • 게이트웨이 기능 지원 • OSI 3계층 장비
게이트웨이 (Gateway)	프로토콜 구조가 전혀 다른 외부 네트워크와 접속하기 위한 장비 • OSI 4계층 장비

▶ 네트워크 토폴로지(Network Topology, 네트워크 구성)

네트워크를 구성하는 장비들을 공간적으로 배치하는 방법

• 분류

- 성형 = 스타형(Star) : 모든 사이트가 하나의 중앙 사이트에 직접 연결되는 중앙 집중형 방식의 구조
- 버스형(Bus) : 공유 버스에 연결된 구조
- 링형(Ring) = 환형, 루프형 : 인접하는 다른 두 사이트와만 직접 연결된 구조
- 계층형(Hierarchy) = 트리형, 분산형 : 여러 개의 사이트를 계층적으로 연결한 구조
- 망형(Mesh) : 각 사이트가 시스템 내의 다른 모든 사이트와 직접 연결된 구조

▶ TCP/IP 프로토콜

서로 다른 기종의 컴퓨터들이 데이터를 주고받을 수 있도록 하는 인터넷 표준 프로토콜

• 구조

OSI 7계층	TCP/IP 계층
응용 계층 표현 계층 세션 계층	응용 계층
전송 계층	전송 계층
네트워크 계층	인터넷 계층
데이터 링크 계층 물리 계층	네트워크 액세스 계층

• 네트워크 액세스 계층(Network Access Layer)

- 실제 데이터를 송·수신하는 역할
- 관련 표준 프로토콜 : IEEE 802, Ethernet, HDLC, X.25, RS-232C, ARQ 등

• 인터넷 계층(Internet Layer)

- 데이터 전송을 위한 주소 지정, 경로 설정 제공
- 관련 표준 프로토콜 : IP, ICMP, IGMP, ARP, RARP 등

• 전송 계층(Transport Layer)

- 호스트들 간의 신뢰성 있는 통신 제공
- 관련 표준 프로토콜 : TCP, UDP 등

• 응용 계층(Application Layer)

- 응용 프로그램 간의 데이터 송·수신 제공

- 관련 표준 프로토콜 : HTTP, TELNET, FTP, SMTP, SNMP, DNS 등

▶ IP(Internet Protocol, 인터넷 프로토콜)

데이터그램을 기반으로 한 IP 주소에 따라 다른 네트워크 간 패킷의 전송 및 경로 제어를 위한 프로토콜

- 비연결형 서비스

▶ TCP(Transmission Control Protocol,

전송 제어 프로토콜)

네트워크의 정보 전달을 통제하고, 프로세스 간에 신뢰할 수 있는 통신을 제공하는 프로토콜

- 가상 회선 방식, 연결형 서비스

• TCP 헤더

- 송신자 포트 번호(Source Port)
- 수신자 포트 번호(Destination Port)
- 순서 번호(Sequence Number)
- 응답 번호(Acknowledgement Number)
- 헤더 길이(Header Length, HLEN)
- 예약 필드(Reserved)
- 윈도우 크기(Window Size, 전송 패킷의 최대치)
- 체크섬(Checksum)
- 긴급 포인터(Urgent Pointer)

▶ UDP(User Datagram Protocol,

사용자 데이터그램 프로토콜)

프로세스 간의 비연결성, 순서 제어가 없는 통신을 제공하는 데이터그램 방식을 지원하기 위한 프로토콜

- 데이터그램 방식, 비연결형 서비스

• UDP 헤더

- 송신자 포트 번호(Source Port)
- 수신자 포트 번호(Destination Port)
- 길이(UDP Length)
- 체크섬(UDP Checksum)

▶ 패킷 교환 방식

송신측에서 메시지를 일정한 크기의 패킷으로 분해하여 전송하고, 수신측에서 패킷을 재결합(재조립)하는 방식

• 구분

가상 회선 방식	정보 전송 전에 가상 경로를 설정하여 목적지에 미리 연결 후 전달하는 연결형 방식
데이터그램 방식	가상 경로를 설정하지 않고 헤더에 주소, 패킷 번호를 붙여서 전달하는 비연결형 방식

▶ 프로토콜(Protocol)

컴퓨터 상호 간 또는 컴퓨터와 단말기 간에 데이터를 송·수신하기 위한 통신 규약

- 기술적 언어

- **기본 요소** : 구문(Syntax), 타이밍(Timing), 의미(Semantics)
- **기능** : 캡슐화(요약화), 동기 제어, 경로 제어(라우팅), 오류 제어(에러 제어), 흐름 제어, 순서 제어, 주소 지정, 다중화, 단편화, 재결합

▶ 동기 제어

송신측과 수신측의 시점을 맞추는 기능

- **기법** : 동기식 전송, 비동기식 전송

▶ 경로 제어(= 라우팅, Routing)

송·수신측 간의 전송 경로 중에서 최적 패킷 교환 경로를 선택하는 기능

• 라우팅 방식

IGP(Interior Gateway Protocol, 내부 게이트웨이 프로토콜)	자율 시스템 내 라우팅 정보를 주고받는 데 사용되는 프로토콜 • 프로토콜 종류 : RIP, IGRP, OSPF
EGP(Exterior Gateway Protocol, 외부 게이트웨이 프로토콜)	자율 시스템 간 라우팅 정보를 주고받는 데 사용되는 프로토콜 • 프로토콜 종류 : BGP

• 라우팅 알고리즘

거리 벡터 알고리즘(Distance Vector Algorithm)	라우터와 라우터 간의 거리(Distance), 방향(Vector) 정보를 이용하여 최단 경로 찾고, 그 최적 경로를 이용할 수 없을 경우에 다른 경로를 찾는 방식 • 벨만 포드(Bellman-Ford) 알고리즘 사용 • 프로토콜 종류 : RIP, IGRP, BGP
링크 상태 알고리즘(Link State Algorithm)	라우터와 라우터 간의 모든 경로를 파악한 뒤, 대체 경로를 사전에 마련해 두는 방식 • 라우팅 정보에 변화가 생길 경우, 변화된 정보만 네트워크 내의 모든 라우터에게 알리는 방식 • 다익스트라(Dijkstra) 알고리즘 사용 • 프로토콜 종류 : OSPF

▶ RIP(Routing Information Protocol,

경로 선택 정보 프로토콜)

경유하는 라우터의 대수(홉 수)에 따라 최단 경로를 동적으로 결정하는 라우팅 프로토콜

▶ IGRP(Interior Gateway Routing Protocol,

내부 경로 제어 프로토콜)

시스코(Cisco)에서 개발한 전송 속도, 대역폭 등 네트워크 상태를 고려하는 라우팅 프로토콜

▶ OSPF(Open Shortest Path First, 최단 경로 우선 프로토콜)

최단 경로를 선정할 수 있도록 라우팅 정보에 노드 간의 거리 정보, 링크 상태 정보를 실시간으로 조합하고, 발생한 변경 정보에 대해 빠른 업데이트를 제공하는 라우팅 프로토콜

▶ 오류 제어(= 에러 제어)

오류를 검출하고 정정하여, 데이터나 제어 정보의 파손에 대비하는 기능

- **ARQ(Automatic Repeat reQuest, 자동 반복 요청)** : 통신 경로에서 오류 발생 시 수신측은 오류의 발생을 송신측에 통보하고 송신측은 오류가 발생한 프레임을 재전송하는 오류 제어 방식
 - 구분 : 정지-대기(Stop-and-Wait) ARQ, Go-Back-N ARQ, 선택적 재전송 (Selective-Repeat) ARQ, 적응적(Adaptive) ARQ

▶ 트래픽 제어(Traffic Control)

전송되는 패킷의 흐름과 그 양을 조절하는 기능

• 흐름 제어(Flow Control) 기법

정지 및 대기 (Stop-and-Wait)	수신자에게 데이터를 보낸 후 응답이 올 때까지 기다리는 방식 <ul style="list-style-type: none"> • 한 번에 하나의 패킷만 전송
슬라이딩 윈도우 (Sliding Window)	수신측에서 설정한 윈도우 크기만큼 송신측에서 긍정 응답 없이 데이터를 전송할 수 있게 하여 데이터 흐름을 동적으로 조절하는 방식 <ul style="list-style-type: none"> • 한 번에 여러 패킷을 전송

• 혼잡 제어(Congestion Control) 기법

AIMD (Additive Increase/ Multiplicative Decrease)	<ul style="list-style-type: none"> • 패킷이 문제없이 도착하면 혼잡 윈도우 크기를 1씩 증가 • 혼잡 현상이 발생하면 혼잡 윈도우 크기를 반으로 줄임
Slow Start	<ul style="list-style-type: none"> • 패킷이 문제없이 도착하면 혼잡 윈도우 크기를 각 패킷마다 1씩 증가시켜 한 주기가 지나면 혼잡 윈도우 크기가 2배로 됨 • 혼잡 현상 발생 시 혼잡 윈도우 크기를 1로 줄여버리는 방식

- **교착상태 회피** : 교착상태 발생시 교착상태에 있는 한 단말 장치를 선택하여 패킷 버퍼를 폐기

▶ 다중화(Multiplexing)

하나의 통신 회선을 분할하여 여러 대의 단말기가 동시에 사용할 수 있도록 다수의 통신로를 구성하는 기술

• 기법

- 주파수 분할 다중화(FDM; Frequency Division Multiplexing)
- 시분할 다중화(TDM; Time Division Multiplexing)
- 동기식 시분할 다중화(STDM; Synchronous TDM)
- 비동기식 시분할 다중화(ATDM; Asynchronous TDM)
- 파장 분할 다중화(WDM; Wavelength Division Multiplexing)
- 코드 분할 다중화(CDM; Code Division Multiplexing)

5과목

정보시스템 구축 관리

▶ 소프트웨어 개발 방법론

소프트웨어 개발의 전 과정에서 지속적으로 적용할 수 있는 방법과 절차, 기법 등

• 종류

세대	방법론
1970년대	구조적 방법론
1980년대	정보공학 방법론
1990년대	객체 지향 방법론
2000년대	컴포넌트 기반 방법론
2000년대	애자일 방법론
2010년대	제품 계열 방법론

▶ 구조적 방법론

절차(프로세스) 단위로 문제를 해결하는 방법론

- 구조화 프로그래밍, 구조적인 프로그램 작성

- 특징 : 모듈화, 유지보수성 향상
- 주요 관점 : 프로세스 중심
- 개발 방식 : 하향식(Top-down)

▶ 정보공학 방법론

정보시스템 개발을 공학적으로 접근하기 위해 체계화시킨 개발 방법론

- 특징 : 기업업무 중심의 정보시스템 개발 최적화
- 주요 관점 : 데이터 중심
- 개발 방식 : 하향식(Top-down)

▶ 객체 지향 방법론

현실 세계의 실체(Entity, 개체)를 속성과 메소드가 결합된 독립적인 형태의 객체(Object)로 표현하는 개념으로 구현 대상을 하나의 객체로 보고 객체와 객체 간의 관계로 모델링하는 방법

- 특징 : 실세계의 객체 모형을 반영
- 주요 관점 : 객체 중심
- 개발 방식 : 상향식(Bottom-up)

▶ 컴포넌트 기반(CBD; Component Based Design) 방법론

소프트웨어를 구성하는 컴포넌트를 조립해서 하나의 새로운 응용 프로그램을 작성하는 방법론

- 특징 : 재사용성 및 효율성 극대화
- 주요 관점 : 컴포넌트 중심
- 개발 방식 : 상향식(Bottom-up)

▶ 애자일 방법론

고객의 요구사항 변화에 유연하게 대응하기 위해 일정

한 주기를 반복하면서 개발하는 방법론

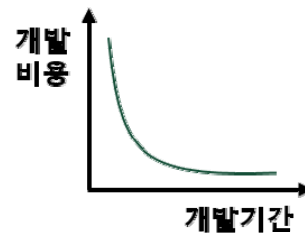
▶ 제품 계열 방법론

특정 제품에 적용하고 싶은 공통된 기능을 정의하여 개발하는 방법론

▶ 비용 산정

소프트웨어 개발에 필요한 기능과 규모를 기반으로 소요되는 인원과 기간, 자원 등을 확인하여 필요한 비용을 예측하는 활동

- 개발 비용과 개발 기간의 상관관계 : 개발 완료 기간을 앞당기면 비용은 더 증가한다.



• 비용 결정 요소

프로젝트 요소	개발 제품의 복잡도, 시스템 크기, 신뢰도
자원 요소	인적 자원, 개발에 필요한 하드웨어/소프트웨어 자원
생산성 요소	개발자의 능력, 개발 기간

• 비용 산정 기법

하향식 산정 기법	프로그램의 규모를 예측하고 과거의 경험을 바탕으로 예측한 규모에 대한 소요 인력과 기간을 추정하는 기법 • 종류 : 전문가 판단 기법, 델파이 기법
상향식 산정 기법	소요 기간을 구하고 여기에 투입되어야 할 인력과 투입 인력의 참여도를 곱하여 최종 인건 비용을 계산하는 기법 • 종류 : LOC 기법, 개발 단계별 노력 기법, 수학적 산정 기법

▶ 전문가 판단 기법

조직 내에 경험이 많은 두 명 이상의 전문가에게 비용 산정을 의뢰하는 기법

▶ 델파이(Delphi) 기법

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

▶ LOC(Line of Code, 원시 코드 라인 수) 기법

예측치를 이용해서 노력(인월)과 개발 기간, 개발 비용, 생산성 등을 산정하는 기법

• LOC 산정 공식

- 노력(인월) = 개발 기간 × 투입 인원,
LOC / 1인당 월평균 생산 코드 라인 수
- 개발 기간 = 노력(인월) / 투입 인원
- 개발 비용 = 노력(인월) × 단위 비용(1인당 월평균 인건비)
- 생산성 = LOC / 노력(인월)

▶ 개발 단계별 노력(Effort Per Task) 기법

LOC 기법을 보완하기 위해 각 기능을 구현시키는 데 필요한 노력을 생명 주기의 각 단계별로 산정하는 기법

▶ 수학적 산정 기법

개발 비용 산정의 자동화를 목표로 하는 기법

- 종류 : COCOMO 모형, 푸트남 모형, 기능 점수 모형

▶ COCOMO(Constructive COst MOdel) 모형

원시 프로그램의 규모인 LOC에 의한 비용 산정 기법

- COCOMO의 소프트웨어 개발 유형 : 소프트웨어의 복잡도나 원시 프로그램의 규모에 따라 분류
- 종류

조직형 (Organic Mode)	5만(= 50KDSI) 라인 이하의 소프트웨어
반분리형 (Semi-Detached Mode)	30만(= 300KDSI) 라인 이하의 소프트웨어
내장형 (Embedded Mode)	30만(= 300KDSI) 라인 이상의 소프트웨어

- COCOMO 모형의 종류 : 비용 산정 단계 및 적용 변수의 구체화 정도에 따라 분류
- 종류 : 기본형(Basic) COCOMO, 중간형(Intermediate) COCOMO, 발전형(Detailed) COCOMO

▶ 푸트남(Putnam) 모형

레이리-노든(Rayleigh-Norden) 곡선의 분포도를 기초로 하여 소프트웨어 생명 주기의 전 과정에 사용될 노력의 분포를 가정해 주는 모형

- 생명 주기 예측 모형
- 자동화 추정 도구 : SLIM

▶ 기능 점수(FP; Function Point) 모형

소프트웨어의 기능을 증대시키는 요인별로 가중치를 부여하고 합산하여 총 기능 점수를 산출한 다음, 총 기능 점수와 영향도를 이용하여 기능 점수(FP)를 구한 후 비용을 산정하는 기법

- 자동화 추정 도구 : ESTIMACS
- 소프트웨어 기능 증대 요인(비용 산정에 이용되는 요인)
 - 자료 입력(입력 양식)
 - 정보 출력(출력 보고서)

- 명령어(사용자 질의 수)
- 데이터 파일
- 필요한 외부 루틴과의 인터페이스

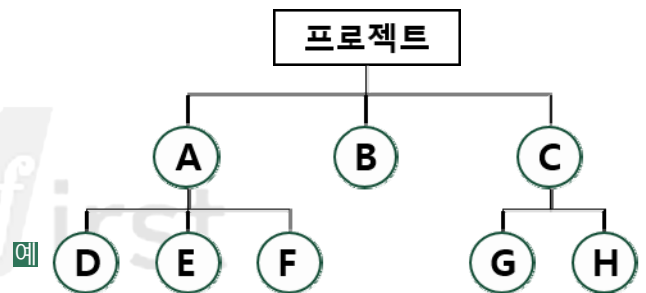
▶ 일정 계획

프로젝트의 프로세스를 이루는 소작업을 파악하고 예측된 노력을 각 소작업에 분배하며, 소작업의 순서와 일정을 정하는 활동

- 브룩스(Brooks) 법칙 : 진행 중인 소프트웨어 개발 프로젝트에 새로운 개발 인력을 추가로 투입할 경우 의사소통 채널의 증가와 작업 적응 기간 등 부작용으로 인해 개발 기간이 더 길어진다.
- 순서 : 프로젝트 규모 추정 → 소단위 작업 분해(WBS) → PERT/CPM 네트워크로 표현 → 간트 차트로 표현

▶ WBS(Work Breakdown Structure, 작업 분해 구조도)

프로젝트를 여러 개의 작은 소단위로 분해하여 계층 구조로 표현한 것

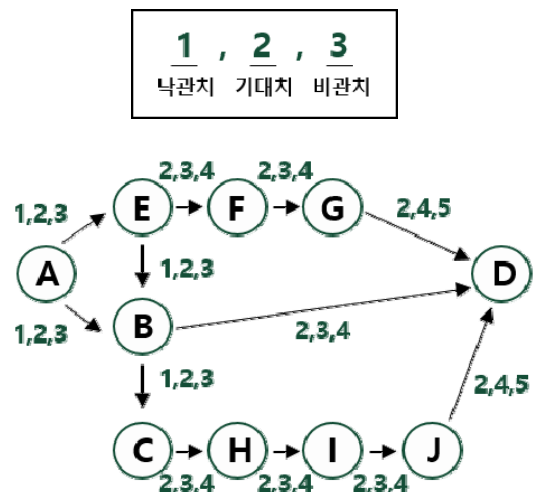


▶ PERT/CPM

프로젝트의 지연을 방지하고 계획대로 진행되도록 일정을 계획하는 것으로, 적은 비용으로 빠른 기간 내에 프로젝트를 완성하기 위한 방법

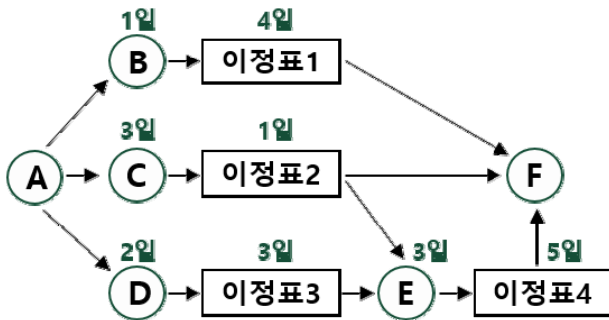
- PERT(Program Evaluation and Review Technique, 프로그램 평가 및 검토 기술) : 소요 기간 예측이 어려운 소프트웨어에서 사용하는 기법

예



- **CPM(Critical Path Method, 임계 경로 기법)** : 작업 시간이 정확하게 주어졌을 때 사용하는 기법

예



- 임계 경로(Critical Path) : 여러 단계의 과정을 거치는 작업을 완수하기 위한 여러 경로 중 시간이 가장 많이 걸리는 경로

예

기능 경로	소요 기간(일)
A-B-이정표1-F	5
A-C-이정표2-F	4
A-C-이정표2-E-이정표4-F	12
A-D-이정표3-E-이정표4-F	13

↑ 임계 경로

▶ 간트 차트(Gantt Chart)

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

- 시간선(Time-Line) 차트

- **포함되는 내용** : 이정표, 작업 일정, 작업 기간, 산출물

예

작업 단계	작업 일정		이정표					산출물
	1	2	3	4	5			
A								AA
B								BB
C								CC

▶ 소프트웨어 개발 보안

보안 취약점을 소프트웨어 개발 단계에서 미리 제거하고, 소프트웨어 개발 생명 주기에서 단계적으로 보안 업무를 수행하는 개발 방법

- **보안의 3요소**

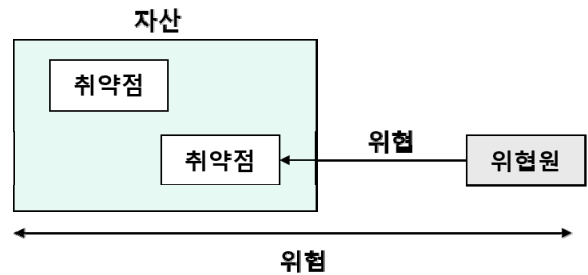
기밀성 (Confidentiality)	인가된 사용자만 정보 자산에 접근할 수 있는 것
무결성 (Integrity)	적절한 권한을 가진 사용자에 의해 인가된 방법으로만 정보를 변경할 수 있도록 하는 것
가용성 (Availability)	정보 자산에 대해 적절한 시간 또는 시점에 접근 가능한 것

- **그 외 보안 요소**

인증 (Authentication)	시스템이 각 사용자를 정확히 식별하고자 할 때 사용하는 방법
----------------------------	-----------------------------------

부인 방지 (Non-repudiation)	메시지 송·수신이나 교환 후 또는 통신이나 처리가 실행된 후에 그 사실을 증명함으로써 사실 부인을 방지하는 보안 기술
--------------------------------	---

- **소프트웨어 개발 보안 관련 용어**



자산 (Asset)	조직의 데이터 또는 조직의 소유자가 가치를 부여한 대상
위협원 (Threat Agents)	조직 자산의 파괴와 손해가 발생하는 행동을 할 수 있는 내·외부의 주체
위협 (Threat)	조직의 자산에 대한 위협이 되는 위협원의 공격 행동
취약점 (Vulnerability)	위협이 발생하기 위한 사전 조건 및 상황
위험 (Risk)	위협원이 취약점을 이용해 위협적인 행동으로 자산에 나쁜 영향의 결과를 가져올 확률과 영향도

▶ Secure SDLC(Software Development Life Cycle, 소프트웨어 개발 보안 생명주기)

소프트웨어 개발의 각 단계별로 요구되는 보안 활동을 수행함으로써 안전한 소프트웨어를 만들 수 있도록 하는 방법론

- **대표적인 방법론**

MS-SDL (Microsoft-Secure Development Lifecycle)	마이크로소프트에서 보안 수준이 높은 안전한 소프트웨어를 개발하기 위해 자체 수립한 개발 방법론
Seven Touchpoints	실무적으로 검증된 개발 보안 방법론 중 하나로, 소프트웨어 보안의 모범 사례를 SDLC에 통합한 방법론
CLASP(Comprehensive, Lightweight Application Security Process)	소프트웨어 개발 생명 주기 초기 단계에 보안 강화를 목적으로 하는 정형화된 프로세스로, 활동 중심 및 역할 기반의 프로세스로 구성된 방법론

▶ Secure OS(Secure Operating System, 보안 운영체제)

운영체제상에 내재된 보안상의 결함으로 발생할 수 있는 각종 해킹으로부터 시스템을 보호하기 위해 기존 운영체제의 커널에 보안 기능을 추가한 보안 운영체제

- **기능**

기능	예시
사용자 인증	사용자 신분 인증 및 검증
계정 관리	비밀번호 관리
통합 관리	다수 서버 보안 관리
접근 통제	권한, 특성 등에 따른 접근 통제 및 보안 관련 작업 시 안전한 경로 제공
감사 기록 축소	보안 관련 사건 기록 보호, 막대한 양의 감사 기록 분석/축소
변경 감사	보안 커널 변경 금지
해킹 감시 (침입 탐지)	해킹 즉각 탐지 및 차단, 실시간 모니터링
객체 재사용 방지	메모리에 이전 사용자가 사용하였던 정보가 남아있지 않도록 기억공간 정리

▶ 취약점(Vulnerability)

컴퓨터나 네트워크에 침입하여 자원에 대한 허가되지 않은 접근을 시도하려는 공격자에게 열린 문을 제공할 수 있는 소프트웨어, 하드웨어, 절차 혹은 인력상의 약점

• 소프트웨어 취약점

버퍼 오버플로 (Buffer Overflow)	메모리 버퍼의 경계값을 넘어서 메모리값을 읽거나 저장하여 예기치 않은 결과를 발생시킬 수 있는 보안 약점
허상 포인터 (Dangling Pointer)	유효한 객체를 가리키고 있지 않은 포인터로, 현재 전혀 다른 데이터를 갖고 있어 예측할 수 없는 행동을 발생시킬 수 있는 보안 약점
포맷 스트링 버그 (Format String Bug)	printf() 등 외부 입력값에 포맷 스트링을 제어할 수 있는 함수를 사용하여 발생할 수 있는 보안 약점
SQL 인젝션 (SQL Injection, SQL 삽입)	검증되지 않은 외부 입력값이 SQL 문 생성에 사용되어 악의적인 쿼리가 실행될 수 있는 보안 약점
코드 인젝션 (Code Injection)	유효하지 않은 데이터를 실행함으로써 악의적인 결과를 초래하는 보안 약점
이메일 인젝션 (Email Injection)	이메일을 보낼 때 받는 사람 목록을 추가하거나 본문에 완전히 다른 메시지를 부정하게 추가할 수 있는 보안 약점
HTTP 헤더 인젝션 (Header Injection)	공격자가 HTTP 헤더에 개행 문자(CR/LF) 등을 삽입해 공격하는 보안 약점
HTTP 응답 분할 (Response Splitting)	HTTP 요청에 들어있는 파라미터가 HTTP 응답 헤더에 포함되어 사용자에게 다시 전달될 때, 입력값에 CR이나 LF와 같은 개행 문자가 존

	재하면 HTTP 응답이 2개 이상 분리될 수 있는데, 두 번째 응답에 악의적인 코드를 주입해 공격하는 보안 약점
디렉터리 접근 공격 (Directory Traversal)	비공개 디렉터리 파일에 대해 부정하게 디렉터리 패스를 가로질러 접근하여 공격하는 보안 약점
사이트 간 스크립팅 (XSS; Cross Site Scripting)	검증되지 않은 외부 입력값에 의해 사용자 브라우저에서 악의적인 스크립트가 실행될 수 있는 보안 약점
검사 시점과 사용 시점	멀티 프로세스상에서 자원을 검사하는 시점과 사용하는 시점이 달라서 발생하는 보안 약점
심볼릭 링크 경쟁 (Symbolic Link Race)	심볼릭 링크 파일을 수정하여 원본 파일을 수정할 수 있는 보안 약점
사이트 간 요청 위조 (CSRF; Cross Site Request Forgery)	사용자가 자신의 의지와는 무관하게 공격자가 의도한 행위를 특정 웹 사이트에 요청하게 하는 공격
클릭재킹 (Clickjacking)	투명한 버튼이나 링크를 함정으로 사용할 웹 페이지에 심어두어 의도치 않은 콘텐츠에 접근하게 하는 공격
FTP 바운스 공격 (FTP Bounce Attack)	FTP 프로토콜 구조의 허점을 이용한 공격으로, 전송 목적지 주소를 임의로 지정해 임의의 목적지로 메시지가 전송될 수 있는 공격

▶ 시큐어 코딩(Secure Coding)

소프트웨어 개발 보안 가이드를 참고하여 소프트웨어의 보안 취약점을 사전에 제거 및 보완하면서 프로그래밍하는 것

- **소프트웨어 보안 약점 항목** : 입력 데이터 검증 및 표현, 보안 기능, 시간 및 상태, 에러 처리, 코드 오류, 캡슐화, API 오용, 세션 통제

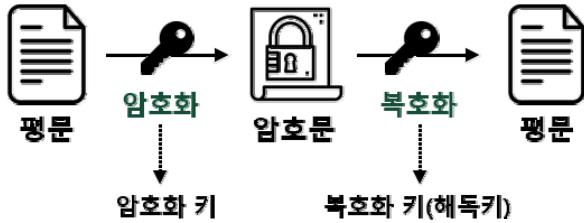
▶ 암호화(Encryption)

평문을 암호문으로 변환하는 과정

- **평문(Plaintext, Cleartext)** : 누구나 읽을 수 있는 암호화되기 전의 원본 메시지
- **암호문(Ciphertext, Cyphertext)** : 의미를 알 수 없는 형식으로, 평문으로 된 정보를 암호 처리하여 특정인만 이용할 수 있도록 암호화한 문서
- **복호화(Decryption)** : 암호화의 역과정으로, 암호문을 평문으로 변환하는 과정

▶ 양방향 암호화

평문을 암호문으로 암호화하는 것과 암호문을 평문으로 복호화하는 것이 모두 가능한 암호 방식

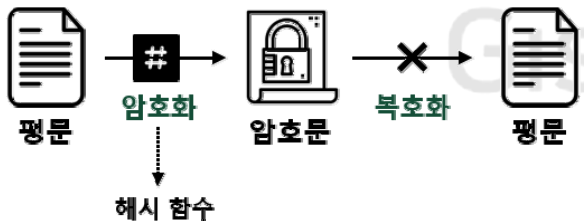


- **개인키(Private) 암호화 기법** : 암호화 키와 복호화 키가 같은 대칭키 암호화 기법
 - 비공개키 암호화 기법
 - 종류

Stream 방식	1bit씩 암호화 • 대표적인 알고리즘 : LFSR, RC4
Block 방식	2bit 이상씩 묶음 암호화 • 대표적인 알고리즘 : DES, 3DES, AES, SEED, ARIA

- **공개키(Public) 암호화 기법** : 암호화 키와 복호화 키가 다른 비대칭키 암호화 기법
 - 대표적인 알고리즘 : RSA, ECC

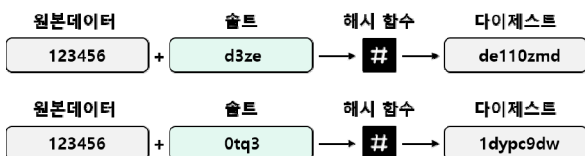
▶ 단방향 암호화



평문을 암호화하는 것은 가능하지만, 암호문을 평문으로 복호화하는 것은 불가능한 암호 방식

- 대표적인 알고리즘 : SHA-256 등 SHA 시리즈, MD5, SNEFRU, N-NASH, HAVAL
- **단방향 해시 함수** : 원본 데이터를 암호화된 데이터로 변환해 주는 알고리즘
 - 해시(Hash) : 임의 길이의 입력 데이터를 받아 고정된 길이의 출력 값(해시 값)으로 변환하는 것
 - 솔트(Salt) : 일종의 랜덤 문자열로, 원본 데이터에 임의의 문자열을 덧붙이는 것

예



▶ DES(Data Encryption Standard, 데이터 암호화 표준)

1975년 미국 국립 표준국(NBS)에서 발표한 개인키 암호

화 알고리즘으로, 56bit의 암호화키를 이용하여 64bit의 평문(블록)을 암호화 및 복호화하는 방식

- **3DES(Triple DES)** : 각 데이터 블록에 DES 알고리즘을 세 번 적용하여 보안을 강화한 암호화 알고리즘

▶ AES(Advanced Encryption Standard, 고급 암호 표준)

DES를 보완하기 위해 2001년 미국 국립 표준 기술 연구소(NIST)에서 발표한 개인키 암호화 알고리즘으로, 128, 192, 256bit의 암호화키를 이용하여 128bit의 평문(블록)을 암호화 및 복호화하는 방식

▶ SEED

1999년 한국인터넷진흥원(KISA)에서 개발한 블록 암호화 알고리즘

▶ ARIA(아리아)

학계(Academy), 연구소(Research Institute), 정부 기관(Agency)의 머리글자를 딴 것으로, 2004년 국가정보원과 산학연협회가 개발한 국가 표준 블록 암호화 알고리즘

▶ RSA

1978년에 MIT 공과 대학의 Rivest, Shamir, Adelman 등 3인이 공동 개발 한 RSA 법(RSA scheme)이라는 암호화 알고리즘을 사용하는 공개키 암호화 알고리즘으로, 큰 수의 소인수 분해에는 많은 시간이 소요되지만, 소인수 분해의 결과를 알면 원래의 수는 곱셈으로 간단히 구해지는 사실에 바탕을 둔 방식

▶ SHA-256 암호 알고리즘

SHA의 한 종류로, 비밀번호 등 임의의 길이 메시지를 256bit의 축약된 메시지로 만들어 내는 해시 알고리즘

▶ MD5(Message Digest Algorithm 5)

미국 MIT의 로널드 라이베스트(Ronald Rivest)가 MD4를 대체하기 위해 개발한 정보보호를 위해 임의 길이의 입력 데이터를 128bit 고정 길이로 메시지를 압축하는 단방향 해시 알고리즘

▶ 시스템 보안

시스템 구성 요소 및 자원들의 기밀성, 무결성, 가용성을 보장하기 위해 취해지는 활동

- **시스템 보안 설계 대상**

계정 관리	<ul style="list-style-type: none"> • 계정(Account) : 사용자에게 소유되는 모든 파일과 자원, 그리고 정보 • 계정 관리 : 적절한 권한을 가진 사용자를 식별하기 위한 가장 기본적인 인증 수단
--------------	---

패스워드	<ul style="list-style-type: none"> 패스워드 : 사용자가 컴퓨터 시스템 또는 통신망에 접속할 때 사용자 ID와 함께 입력하여 정당한 사용자라는 것을 식별할 수 있도록 컴퓨터에 전달해야 하는 고유의 문자열
세션 관리	<ul style="list-style-type: none"> 세션(Session) : 네트워크 환경에서 사용자 또는 컴퓨터 간의 대화를 위한 논리적 연결 세션 관리 : 사용자와 시스템 또는 두 시스템 간의 활성화된 접속에 대한 관리
접근 제어	<ul style="list-style-type: none"> 접근(Access) : 일반적으로 데이터를 얻는 것으로, 데이터가 있는 곳을 알아내어 그곳에 가서 데이터를 가져오는 것 접근 제어(AC; Access Control) : 시스템이 연결된 다른 시스템으로부터 적절히 보호되고 꼭 필요한 사람에 의해서만 필요한 서비스를 제공할 수 있도록 전체 시스템 차원에서 접근을 통제하는 것
권한 관리	<ul style="list-style-type: none"> 권한(Right) : 책임을 수행하기 위해 필요한 권한으로, 직책 수행에 필요한 결정권, 기타 재능의 수단을 합한 것 권한 관리 : 시스템의 각 사용자가 적절한 권한으로 적절한 정보 자산에 접근할 수 있도록 통제하는 것
로그 관리	<ul style="list-style-type: none"> 로그(Log) : 시스템 사용에 대한 모든 내역을 기록해 놓은 것 로그 관리 : 컴퓨터 생성된 대량의 로그 메시지를 처리하는 것
취약점 관리	<ul style="list-style-type: none"> 취약점 : 공격자가 시스템의 정보 보안을 낮추는 데 사용되는 약점이다. 취약점 관리 : 취약점을 확인하고 분류, 치료, 완화시키는 주기적인 과정

• **시스템 보안 구현 환경**

관리적 보안	<ul style="list-style-type: none"> 인적 자산에 대한 보안 각종 관리 절차 및 규정
물리적 보안	<ul style="list-style-type: none"> 설비/시설 자산에 대한 보안 물리적 위협으로부터 보호하는 것
기술적 보안	<ul style="list-style-type: none"> 정보 자산에 대한 보안 실제 정보시스템에 적용된 기술에 특화하여 기술적으로 마련할 수 있는 정보 보호 대책

• **시스템 보안 구현 도구(보안 취약점 도구)**

MBSA(Microsoft Baseline Security Analyzer)	일반적으로 윈도우 시스템에서 트러거 쉬운 보안 관련 설정을 간단히 확인하는 기능을 갖춘 도구
---	---

Nmap (Network mapper)	고든 라이온(Gordon Lyon)이 작성한 보안 스캐너로, 서버 관리자가 시스템 자체 스캔을 통해 자신이 운영하는 서버에 자신도 모르는 다른 포트가 열려 있는지 등을 확인하는 도구
NBTScan	네트워크를 점검(Scan)하는 프로그램으로, 점검하고자 하는 대상 IP에 대해서 질의를 보내면 해당 시스템은 IP 주소와 NetBIOS 컴퓨터 이름, 사용자 이름, MAC 주소 등의 정보를 반송하는 도구

▶ **보안 솔루션**

접근 통제, 침입 차단 및 탐지, DDos 탐지 등을 수행하여 외부로부터 불법적인 침입을 막는 기술이나 시스템 또는 장비

- **종류** : 방화벽, IDS, DMZ, IPS, NAC, TMS, UTM, ESM, DLP 등

방화벽(Firewall)	해킹 등에 의해 외부로 정보 유출을 막기 위해 사용하는 보안 시스템
IDS(Intrusion Detection System, 침입 탐지 시스템)	정보 시스템의 보안을 위협하는 침입 행위가 발생할 경우 이를 탐지 및 적극 대응하기 위한 보안 시스템
DMZ(Demilitarized Zone, 비무장지대)	내·외부 공격으로부터 중요 데이터를 보호하거나 서버의 서비스 중단을 방지하기 위한 침입 차단 기능을 하는 보안 시스템
IPS(Intrusion Prevention System, 침입 방지 시스템)	방화벽, 침입 탐지 시스템과 같은 네트워크 기반의 차단 솔루션을 논리적으로 결합한 시스템
WIPS(Wireless Intrusion Prevention System, 무선 침입 방지 시스템)	무선 공유기 사용 현황을 실시간으로 관찰하여 허용하지 않은 접속을 막고, 보안 취약점을 야기할 수 있는 부적절한 접속을 방지하는 무선 랜(Wi-Fi) 침입 방지 시스템
웹 방화벽(Web Firewall)	일반 방화벽이 탐지하지 못하는 SQL 삽입 공격, XSS 등의 웹 기반 공격을 방어해 주는 보안 솔루션
NAC(Network Access Control, 네트워크 접근 제어)	사전에 인가받지 않은 사용자나 보안 체계를 갖추지 않은 정보 기기의 네트워크 접속을 차단하는 솔루션
TMS(Threat Management System, 위협 관리 시스템)	사이버 공격에 대한 침입 탐지와 트래픽 분석 등의 기술을 통해 로컬 네트워크의 위협 분석과 취약성 정보 등을 사전에 관리자에게 통보 및 실시간으로 관제하고 대응할 수 있는 시스템

UTM(Unified Threat Management, 통합 위협 관리)	침입 차단 시스템, 가상 사설망 등 다양한 보안 솔루션 기능을 하나로 통합한 보안 솔루션
ESM(Enterprise Security Management, 기업 보안 관리)	방화벽, 침입 탐지 시스템, 가상 사설망 등의 보안 솔루션을 하나로 모은 통합 보안 관리 시스템
DLP(Data Loss Prevention, 데이터 유출 방지)	기업 내부자의 고의나 실수로 인한 외부로의 정보 유출을 방지하는 솔루션
스팸 차단 솔루션(Anti-Spam Solution)	스팸 메일 차단 기능뿐 아니라 메일에 대한 바이러스 검사, 내부 유출방지 등의 확장 기능을 가지고 있는 보안 솔루션
보안 USB(Security USB)	정보 유출 방지 등 보안 기능을 갖춘 USB 메모리
DRM(Digital Rights Management, 디지털 저작권 관리)	웹을 통해 유통되는 각종 디지털 콘텐츠의 안전 분배와 불법 복제 방지를 위한 저작권 보호 방식
VPN(Virtual Private Network, 가상 사설망)	인터넷망과 같은 공중망을 사설망처럼 이용해 회선 비용을 크게 절감할 수 있는 기업 통신 서비스

▶ 네트워크 구축 관련 신기술 및 트렌드 정보

클라우드 컴퓨팅 (Cloud Computing)	IT 자원을 구매하거나 소유할 필요 없이 필요한 만큼 사용료를 주고 쓰는 서비스
인터클라우드 컴퓨팅(Inter-Cloud Computing)	둘 이상의 클라우드 서비스 제공자 간의 상호 연동을 가능케 하는 기술
멀티 클라우드 (Multi Cloud)	서로 다른 업체에서 2개 이상의 퍼블릭 클라우드를 이용해 하나의 서비스를 운영하는 것
모바일 컴퓨팅(Mobile Computing, 이동형 컴퓨팅)	무선 이동 통신과 PDA, 인터넷을 통해 컴퓨터와 통신 기술을 효과적으로 연계시켜 언제, 어디서나 이동하면서 정보 교환이나 수집, 검색, 정리, 저장하는 기술
모바일 클라우드 컴퓨팅 (Mobile Cloud Computing)	클라우드 컴퓨팅의 경제성과 모바일의 이동성이 결합된 것
IoT(Internet of Things, 사물 인터넷)	가전제품, 전자 기기뿐만 아니라 헬스케어, 원격 감침, 스마트홈, 스마트카 등 다양한 분야에서 사물을 네트워크로 연결해 정보를 공유하는 기술
신 클라이언트 (Thin Client)	각종 프로그램 및 데이터를 네트워크로 연결된 서버로부터 받아서 사용하는 PC 대체 컴퓨터
VLAN(Virtual	물리적 배치와 상관없이 논리적으로

Local Area Network, 가상 랜)	LAN을 구성할 수 있는 기술
BLE(Bluetooth Low Energy, 저전력 블루투스)	약 10m 도달 반경을 가진 2.4GHz 주파수 대역에서 저전력, 저용량 데이터 송·수신이 가능한 저전력 블루투스 기술
비컨(Beacon)	주변의 일정 반경 범위 내(최대 50m)에서 블루투스 4.0을 기반으로 사물의 정보를 주기적으로 전송하는 근거리 무선 통신 기술
RFID(Radio Frequency Identification, 전자 태그)	극소형 칩에 상품 정보를 저장하고 안테나를 달아 무선으로 데이터를 송신하는 장치로써 일반적으로 유통 분야에서 물품 관리를 위해 사용된 바코드를 대체할 차세대 인식 기술
NFC(Near Field Communication, 근접 무선 통신)	10cm 이내의 가까운 거리에서 다양한 무선 데이터를 주고받는 통신 기술
피코넷(Piconet)	여러 개의 독립된 통신 장치가 블루투스 기술이나 UWB 통신 기술을 사용하여 통신망을 형성하는 무선 네트워크 기술
애드혹 네트워크 (Ad-Hoc Network)	노드들에 의해 자율적으로 구성되는 기반 구조가 없는 네트워크
SON(Self-Organizing Network, 자동 구성 네트워크)	주변 상황에 자동적으로 적응하여 스스로 망을 구성하는 네트워크
NGN(Next Generation Network, 차세대 통신망)	ITU-T에서 개발하고 있는 유선망 기반의 차세대 통신망
NDN(Named Data Networking, 엔디엔)	인터넷에서 콘텐츠 자체의 정보와 라우터 기능만을 이용하여 목적지로 데이터를 전송하는 기술
지능형 초연결망	네트워크 전체에 소프트웨어 정의 기술(SDx)을 적용하는 차세대 국가망
스마트 그리드 (Smart Grid)	전기 및 정보통신 기술을 활용하여 전력망을 지능화, 고도화함으로써 고품질의 전력 서비스를 제공하고 에너지 이용 효율을 극대화하는 차세대 지능형 전력망
와이선(Wi-SUN)	스마트 그리드 서비스를 제공하기 위한 와이파이 기반의 저전력 장거리 통신 기술
네트워크 슬라이싱 (Network Slicing)	하나의 물리적 코어 네트워크를 독립된 다수 가상 네트워크로 분리한 뒤 고객 맞춤형 서비스를 제공하는 네트워크 기술
메시 네트워크 (Mesh Network)	기존 무선 랜의 한계 극복을 위해 등장한 다 대 다 디바이스 간 통신을 지원하는 네트워크 기술
포스퀘어 (Foursquare)	포스퀘어(Foursquare)사의 위치 기반 소셜 네트워킹 서비스(SNS)

NAT(Network Address Translation, 네트워크 주소 변환)	사내의 개별 IP 주소인 사설 IP와 정식 IP 주소인 공식 IP를 상호 변환하는 기능
지그비(Zigbee)	IEEE 802.15 표준을 기반으로 만들어진 것으로 저속, 저비용, 저전력 무선망을 위한 기술
MQTT(Message Queuing Telemetry Transport)	사물 통신, 사물 인터넷(IoT)과 같이 대역폭이 제한된 통신 환경에 최적화하여 개발된 푸시 기술 기반의 경량 메시지 전송 프로토콜

▶ SW 구축 관련 신기술 및 트렌드 정보

VR(Virtual Reality, 가상 현실)	어떤 특정한 환경이나 상황을 컴퓨터로 만들어 그것을 사용하는 사람이 마치 실제 주변 상황 및 환경과 상호작용을 하고 있는 것처럼 만들어 주는 인간과 컴퓨터 사이의 인터페이스
AR(Augmented Reality, 증강 현실)	컴퓨터 세상의 환경을 복제하는 것을 목적으로 하는 가상 현실의 유형
MR(Mixed Reality, 혼합 현실)	현실을 기반으로 가상 정보를 추가하는 증강 현실(AR)과 가상 환경에 현실 정보를 추가하는 증강 가상(AV)의 의미를 포함한 것
메타버스 (Metaverse)	3차원 가상 세계
AI(Artificial Intelligence, 인공지능)	컴퓨터에 의한 인간 지능 프로세스의 시뮬레이션으로 컴퓨터가 인간의 지능 활동을 모방할 수 있도록 하는 것
텐서플로 (TensorFlow)	구글(Google)사에서 개발한 머신 러닝을 위한 오픈 소스 소프트웨어 라이브러리
뉴럴링크 (Neuralink)	테슬라 CEO 일론 머스크(Elon Musk)가 설립한 뇌 연구 스타트업으로 신경 레이스(Neural Lace)라고 부르는 기술을 개발하는 기업
온톨로지 (Ontology)	존재하는 사물과 사물 간의 관계 등 여러 개념을 컴퓨터가 처리할 수 있는 형태로 표현하는 것
시맨틱 웹 (Semantic Web, 의미론적 웹)	컴퓨터가 정보 자원의 뜻을 이해하고, 논리적 추론까지 할 수 있는 차세대 지능형 웹
디지털 트윈 (Digital Twin)	물리적인 사물과 컴퓨터에 동일하게 표현되는 가상 모델
SDN(Software Defined Network, 소프트웨어 정의망)	소프트웨어 프로그래밍을 통해 네트워크 경로 설정과 제어 및 복잡한 운용 관리를 편리하게 처리할 수 있는 차세대 네트워킹 기술
SDS(Software Defined Storage, 소프트웨어 정의 스토리지)	소프트웨어를 이용하여 전체 스토리지 자원을 관리하는 데이터 저장 장치 체계

SDDC(Software-Defined Data Center, 소프트웨어 정의 데이터 센터)	데이터 센터를 효율적으로 운영하고 편리하게 관리하기 위해 등장한 모든 컴퓨팅 인프라를 가상화하여 서비스하는 데이터 센터
SDP(Software Defined Perimeter, 소프트웨어 정의 경계)	네트워크 장치, 단말 상태, 사용자 ID를 체크하여 권한이 있는 사용자 및 디바이스에 대해서만 접근 권한을 부여하며 인증 받지 못한 단말기에 대해서는 그 어떠한 서비스 연결 정보도 얻지 못하게 접근을 제어하는 프레임워크
SSO(Single Sign-On, 싱글 사인 온)	하나의 시스템에서 인증에 성공하면 다른 시스템에 대한 접근 권한도 얻는 시스템
DSA(Digital Signature Algorithm, 전자 서명 알고리즘)	미국 국립 표준 기술 연구소(NIST)에서 전자 서명 표준안으로 개발된 엘가말 암호 방식 기반의 전자 서명 알고리즘
CC(Common Criteria, 공통 평가 기준)	ISO/IEC 15408이라고도 불리는 정보 보호 제품의 평가 기준을 규정한 국제 표준
DLT(Distributed Ledger Technology, 분산 원장 기술)	중앙 관리자나 중앙 데이터 저장소가 존재하지 않고 P2P 망 내의 참여자들에게 모든 거래 목적이 분산 저장되어 거래가 발생할 때마다 지속적으로 갱신되는 디지털 원장
블록체인(BlockChain)	P2P 네트워크 분산 환경에서 온라인 금융 거래 정보를 블록으로 연결하여 중앙 관리 서버가 아닌 참여자(피어, Peer)들의 개인 디지털 장비에 분산 저장시켜 공동으로 관리하는 방식
매시업(Mashup)	웹에서 제공하는 정보 및 서비스를 이용하여 새로운 소프트웨어나 서비스, 데이터베이스 등을 만드는 기술
RIA(Rich Internet Application)	데스크톱 환경처럼 응답 속도가 빠르고 사용하기 쉬운 기능과 특징을 제공하는 웹 제작 기술
PET(Privacy Enhancing Technology, 개인정보 강화 기술)	개인정보 침해 위험을 관리하기 위한 핵심 기술로, 암호화·익명화 등 개인정보를 보호하는 기술에서 사용자가 직접 개인정보를 통제하기 위한 기술까지 다양한 사용자 프라이버시 보호 기술을 통칭한 것

▶ HW 구축 관련 신기술 및 트렌드 정보

3D 프린팅(3D Printing, 3차원 인쇄)	디지털화된 디자인 데이터를 활용해 인쇄를 하듯 물체를 만들어 내는 방식
4D 프린팅(4D Printing, 4차원 인쇄)	미리 설계된 시간이나 임의 환경 조건이 충족되면 스스로 모양을 변경하거나 제조하여 새로운 형태로 바뀌는 제품을 3D 프린팅 하는 기술
엔 스크린 (N screen)	하나의 콘텐츠를 PC·TV·휴대폰 등 여러 단말기에 공유하여 끊김 없이 이용하는 체계

컴패니언 스크린 (Companion Screen)	TV 방송 시청에 동반되어 이용되는 보조 기기
패블릿(Phablet)	폰(Phone)과 태블릿(Tablet)의 합성어로, 5인치 이상의 대화면 스마트폰
멤리스터(Memristor)	메모리(Memory)와 레지스터(Resistor)의 합성어로, 전류의 방향과 크기 등 기존의 상태를 모두 기억하는 소자
RAID(Redundant Array of Inexpensive Disk, 복수 배열 독립 디스크)	디스크의 고장에 대비해 데이터의 안정성을 높인 컴퓨터의 저장 장치로서, 하나의 대형 저장 장치 대신 다수의 일반 하드디스크를 배열로 구성하고, 데이터를 분할해서 분산 저장하거나 다중화한 저장 장치
M-DISC(Millennial DISC)	한 번의 기록만으로 자료를 영구 보관할 수 있는 광 저장장치
C형 유에스비(USB Type-C, USB-C, Universal Serial Bus Type-C)	기기 간 데이터 전송을 위한 USB 케이블 단자의 위아래가 동일한 24핀의 USB
MEMS(멤스, Micro-Electro-Mechanical Systems, 초소형 정밀기계 기술)	실리콘이나 수정, 유리 등을 가공하여 초고밀도 집적 회로, 머리카락 절반 두께의 초소형 기어, 손톱 크기의 하드 디스크 등 초미세 기계 구조물을 만드는 기술
트러스트존(TrustZone)	암(ARM)사에서 개발한 것으로, 프로세서(CPU) 안에 독립적인 보안 구역을 따로 두어 중요한 정보를 보호하는 하드웨어 기반의 보안 기술

▶ DB 구축 관련 신기술 및 트렌드 정보

빅 데이터(Big Data)	기존의 관리 방법이나 분석 체계로는 처리하기 어려운 막대한 양의 정형 또는 비정형 데이터 집합
브로드 데이터(Broad Data)	거대한 자료라는 의미를 가진 빅 데이터와 달리 기업 마케팅에 효율적인 다양한 정보
하둡(Hadoop)	일반 컴퓨터들을 연결하여 하나의 시스템처럼 작동하도록 묶어 다양한 대용량 데이터들을 분산 처리하는 자유자바 소프트웨어 프레임워크
맵리듀스(MapReduce)	대용량 데이터를 분산 처리하기 위한 목적으로 개발된 프로그래밍 모델
스콧(Sqoop)	관계형 데이터베이스와 하둡 사이에서 데이터 이관을 지원하는 툴
타조(Tajo)	하둡 기반의 대용량 데이터 웨어하우스 시스템
CEP(Complex Event Processing, 복잡 이벤트 처리)	실시간으로 발생하는 많은 사건들 중 의미가 있는 것만을 추출할 수 있도록 사건 발생 조건을 정의하는 데이터 처리 방법
데이터 다이어트	데이터를 삭제하는 것이 아니라 압축

(Data Diet)	하고, 겹친 정보는 중복을 배제하고, 새로운 기준에 따라 나누어 저장하는 작업
디지털 아카이빙(Digital Archiving)	디지털 정보 자원을 장기적으로 보존하기 위한 작업
LOD(Linked Open Data, 개방형 링크드 데이터)	연계 데이터(Linked Data)와 오픈 데이터(Open Data)가 결합된 단어로, 사용자가 정확하게 원하는 정보를 찾을 수 있도록 웹상의 모든 데이터와 데이터베이스를 공개하고 연결하는 것
RTO(Recovery Time Objective, 목표 복구 시간)	비상사태 또는 업무 중단 시점부터 업무가 복구되어 다시 정상 가동 될 때까지의 시간
RPO(Recovery Point Objective, 목표 복구 시점)	조직에서 발생한 여러 가지 재난 상황으로 인해 IT 시스템이 마비되었을 때, 각 업무에 필요한 데이터를 여러 백업 수단을 활용하여 복구할 수 있는 기준점

▶ 보안 구축 관련 신기술 및 트렌드 정보

DoS(Denial Of Service, 서비스 거부 공격)	정당한 사용자가 적절한 대기 시간 내에 정보 시스템의 데이터나 자원을 사용하는 것을 방해하는 공격 방법
죽음의 핑(Ping of Death)	인터넷 프로토콜 허용 범위인 65,536 바이트 이상의 큰 패킷을 고의로 전송하여 발생하는 서비스 거부 공격 방법
SYN 플러딩(SYN Flooding)	대량의 SYN 패킷을 이용해서 타겟 서버의 서비스를 더 이상 사용할 수 없도록 만드는 공격 방법
스머핑(Smurfing, Ping Flood, Ping 홍수)	IP, ICMP의 특성을 악용하여 고성능 컴퓨터를 이용해 초당 엄청난 양의 접속 신호를 한 사이트에 집중적으로 보냄으로써 상대 컴퓨터의 서버를 접속 불능 상태로 만들어 버리는 공격 방법
LAND 공격(Local Area Network Denial Attack)	'나쁜 상태에 빠지게 하다.'의 의미로, 공격자가 패킷의 출발지 주소(Address)나 포트(Port)를 임의로 변경하여 출발지와 목적지 주소 또는 포트를 동일하게 하여 무한 응답을 발생시키는 공격 방법
TearDrop 공격	패킷 제어 로직을 악용하여 시스템의 자원을 고갈시키는 공격으로, 데이터의 송수신 과정에서 패킷의 크기가 커서 여러 개로 분할되어 전송될 때 분할 순서를 알 수 있도록 Fragment Offset 값을 함께 전송하는데, 이 값을 변경시켜 수신 측에서 패킷 재조립 시 과부하가 발생하는 공격 방법

DDoS(Distribute d Denial of Service Attack, 분산 서비스 거부 공격)	감염된 대량의 숙주 컴퓨터를 이용해 특정 시스템을 마비시키는 사이버 공격 방법
피싱(Phishing)	개인정보(Pprivate Data)와 낚시(Fishing)의 합성어로, 낚시하듯이 개인정보를 몰래 빼내는 것
스피어 피싱(Spear Phishing)	조직 내에 신뢰할 만한 발신인으로 위장해 ID 및 패스워드 정보를 요구하는 것
사회 공학적 해킹(Social Engineering Hacking)	시스템이 아닌 사람의 취약점을 공략하여 원하는 정보를 얻는 공격 기법
파밍(Pharming)	공식적으로 운영하고 있는 도메인 자체를 탈취하여 사용자는 방문한 사이트를 진짜 사이트로 착각하게 하여 아이디와 패스워드 등의 개인정보를 노출하게 하는 수법
큐싱(Qshing)	QR 코드와 피싱(Phishing)의 합성어로, QR 코드를 통해 악성 링크로 접속을 유도하거나 직접 악성코드를 심는 방법
스미싱(SMishing)	SMS와 피싱(Phishing)의 합성어로, 문자 메시지를 이용하여 피싱하는 방법
트랩도어(Trap Door) = 백도어(Back Door)	시스템 보안이 제거된 비밀 통로로, 서비스 기술자나 유지보수 프로그램 작성자의 접근 편의를 위해 시스템 설계자가 고의로 만들어 놓은 시스템의 보안 구멍
루트킷(Rootkit)	시스템 침입 후 침입 사실을 숨긴 채 차후의 침입을 위한 백도어, 트로이 목마 설치, 그리고 원격 접근, 내부 사용 흔적 삭제, 관리자 권한 획득 등 주로 불법적인 해킹에 사용되는 기능들을 제공하는 프로그램들의 모음
트립와이어(Tripwire)	크래커가 침입하여 백도어를 만들어 놓거나 설정 파일을 변경했을 때 분석하는 도구
랜섬웨어(Ransomware)	인터넷 사용자의 컴퓨터에 잠입해 내부 문서나 스프레드시트, 그림 파일 등을 암호화해 열지 못하도록 만든 후 돈을 보내주면 해독용 열쇠 프로그램을 전송해 준다면 금품을 요구하는 악성 프로그램
웜 바이러스(Worm Virus)	스스로를 복제하는 악성 소프트웨어 컴퓨터 프로그램
스턱스넷(Stuxnet)	독일 지멘스사의 원격 감시 제어 시스템(SCADA)의 제어 소프트웨어에 침투하여 시스템을 마비시키는 바이러스

님다(Nimda)	윈도(Windows) 계열의 서버를 사용하는 PC를 공격 대상으로 하고, 파일을 통해 서버를 감염시키는 공격 방법
제로 데이 공격(Zero Day Attack)	시스템의 보안 취약점이 발견된 상태에서 이를 보완할 수 있는 보안 패치가 발표되기 전에 해당 취약점을 이용해 이뤄지는 해킹이나 악성코드 공격 방법
세션 하이재킹(Session Hijacking, 세션 가로채기)	다른 사람의 세션 상태를 훔치거나 도용하여 액세스하는 해킹 기법
APT(Advanced Persistent Threat, 지능형 지속 공격)	다양한 IT 기술과 방식들을 이용해 특정 기업이나 조직 네트워크에 침투해 활동 거점을 마련한 뒤, 때를 기다리면서 보안을 무력화시키고 정보를 수집한 다음 외부로 빼돌리는 형태의 공격 방법
무작위 대입 공격(Brute Force Attack, 브루트 포스 공격)	조합 가능한 모든 경우의 수를 전부 대입해보는 공격 방법
키로거 공격(Key Logger Attack)	컴퓨터 사용자의 키보드 움직임을 탐지해 ID나 패스워드, 계좌 번호, 카드 번호 등과 같은 개인의 중요한 정보를 몰래 빼가는 해킹 공격 방법
이블 트윈 공격(Evil Twin Attack)	소셜 네트워크에서 악의적인 사용자가 지인 또는 특정 유명인으로 가장하여 활동하는 공격 방법
논리 폭탄(Logic Bomb)	특정 조건이 만족되면 트리거에 의해 특정 형태의 공격을 하는 코드로, 일반 프로그램에 오류를 발생시키는 프로그램 루틴을 무단으로 삽입하여 부정한 행위를 하는 것
사이버 협박(Cyber-bullying, 사이버 불링)	인터넷에서 특정인을 집단적으로 따돌리거나 욕설/모욕/위협/소문/사진 등으로 집요하게 괴롭히는 행위
스니핑(Sniffing)	네트워크의 중간에서 남의 패킷 정보를 도청하는 해킹 유형의 하나
스누핑(Snooping)	네트워크상에서 남의 정보를 엿탐하여 불법으로 가로채는 행위
스푸핑(Spoofing)	승인받은 사용자인 것처럼 시스템에 접근하거나 네트워크상에서 허가된 주소로 가장하여 접근 제어를 우회하는 공격 행위
그레이웨어(Grayware)	정상 소프트웨어와 바이러스 소프트웨어의 중간에 해당하는 일종의 악성 소프트웨어
스택가드(StackGuard)	메모리상에서 프로그램의 복귀 주소와 변수 사이에 특정 값을 저장해 두었다가 그 값이 변경되었을 경우 오버플로 상태로 가정하여 프로그램 실행을 중단하는 기술
모드 체크(Mode Check)	입력할 수 있는 문자가 제한된 경우 입력 문자를 확인하여 이상 유무를 검색하는 것

리커버리 통제(Recovery Control, 복구 통제)	데이터 백업과 같이 부적절한 사건· 상황으로 인해 발생할 피해를 막거 나, 장애를 해결하고 정상적인 운영 상태로 회복하는 것
PEM(Privacy Enhanced Mail, 프라이버시 향상 전자 우편)	전자 우편의 내용을 암호화하여 전송 하고 특정한 키가 있어야만 내용을 볼 수 있도록 하는 인터넷 전자 우편 (E-Mail)의 표준
시스로그(Syslog)	다양한 프로그램들이 생성하는 메시 지들을 저장하고, 이 메시지들을 이 용해서 다양한 분석 등이 가능하도록 하는 로그 메시지들의 집합
SRM(Security Reference Monitor, 보안 참조 모니터)	사용자가 특정 객체에 접근할 권한이 있는지, 해당 객체에 특정 행위를 할 수 있는지를 검사하는 기능 또는 장 치
SOS(Security Operating Service, 보안 운영 서비스)	고객의 IT 자원 및 보안 시스템에 대 한 종합적 보안 관리를 원격으로 제 공하는 서비스
OWASP(The Open Web Application Security Project)	주로 웹을 통한 정보 유출, 악성파일 및 스크립트, 보안 취약점을 연구하 는 오픈소스 웹 애플리케이션 보안 프로젝트

Gisa1first