

1과목 정보시스템 기반 기술

001 운영체제의 개요

- 컴퓨터 시스템의 자원들을 효율적으로 관리한다.
- 사용자가 컴퓨터를 편리하고 효과적으로 사용할 수 있
도록 환경을 제공한다.

002 운영체제의 목적

- 처리 능력(Throughput) 향상
- 반환 시간(Turn Around Time) 단축
- 사용 가능도(Availability) 향상
- 신뢰도(Reliability) 향상

003 운영체제의 기능

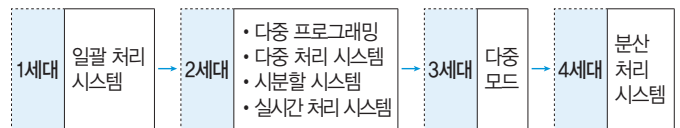
- 프로세서, 기억장치, 입·출력장치, 파일 및 정보 등의 자원을 관리한다.
- 자원의 스케줄링 기능을 제공한다.
- 사용자와 시스템 간의 인터페이스를 제공한다.
- 데이터를 관리하고, 데이터 및 자원의 공유 기능을 제
공한다.

004 운영체제 운용 기법

- 실시간 처리 시스템 : 처리할 데이터가 생겨날 때마다 바
로 처리하는 방식
- 분산 처리 시스템 : 지역적으로 분산된 여러 대의 컴퓨터
를 연결하여 작업을 분담하여 처리하는 방식
- 다중 프로그래밍 시스템 : 한 개의 CPU로 여러 개의 프로
그램을 동시에 처리하는 방식

- 다중 처리 시스템 : 하나의 컴퓨터에 여러 개의 CPU를
설치하여 프로그램을 처리하는 방식
- 임베디드 시스템 : 마이크로프로세서에 특정 기능을 수
행하는 응용 프로그램을 탑재하여 컴퓨터의 기능을 수
행하는 방식

005 운영체제 운용 기법의 발달 과정



006 프로세스의 정의

- PCB를 가진 프로그램
- 주기억장치에 저장된 프로그램
- 프로세서가 할당되는 실체
- 프로시저가 활동중인 것
- 비동기적 행위를 일으키는 주체
- 지정된 결과를 얻기 위한 일련의 계통적 동작
- 목적 또는 결과에 따라 발생하는 사건들의 과정

007 프로세스 상태 전이

- 제출(Submit) : 사용자가 작업을 시스템에 제출한 상태
- 접수(Hold) : 제출된 작업이 디스크의 할당 위치에 저장
된 상태
- 준비(Ready) : 프로세서를 할당받기 위해 기다리고 있는
상태
- 실행(Run) : 프로세서를 할당받아 실행되는 상태
- 대기(Wait, 보류, 블록(Block)) : 입·출력 처리가 완료될 때
까지 대기하고 있는 상태
- 종료(Terminated, Exit) : 실행이 끝나고 프로세스 할당이
해제된 상태

008

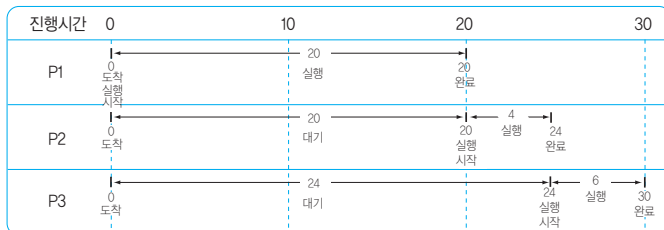


스케줄링 - FCFS(FIFO)

준비상태 큐에 도착한 순서에 따라 차례로 CPU를 할당하는 기법이다.

예제 다음과 같은 프로세스들이 차례로 준비상태 큐에 들어왔다고 가정할 때, FCFS 기법을 이용하여 평균 실행 시간, 평균 대기 시간, 평균 반환 시간을 구하시오(제출시간은 없으며 시간의 단위는 초임).

프로세스 번호	P1	P2	P3
실행 시간	20	4	6



- 평균 실행 시간 : $(20+4+6)/3 = 10$
- 평균 대기 시간 : $(0+20+24)/3 = 14.6$
- 평균 반환 시간 : $(20+24+30)/3 = 24.6$

009

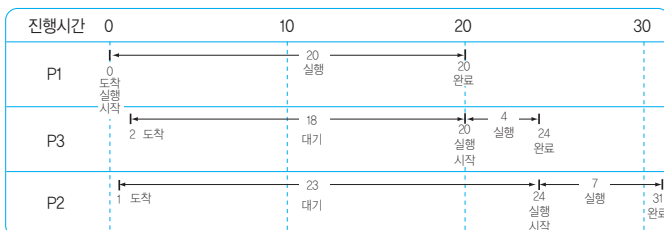


스케줄링 - SJF

실행 시간이 가장 짧은 프로세스에게 먼저 CPU를 할당하는 기법이다.

예제 다음과 같은 프로세스들이 차례로 준비상태 큐에 들어왔다고 가정할 때, SJF 기법을 이용하여 평균 실행 시간, 평균 대기 시간, 평균 반환 시간을 구하시오(제출 시간이 있을 경우).

프로세스 번호	P1	P2	P3
실행 시간	20	7	4
도착 시간	0	1	2



- 평균 실행 시간 : $(20+4+7)/3 = 10.3$
- 평균 대기 시간 : $(0+18+23)/3 = 13.6$
- 평균 반환 시간 : $(20+22+30)/3 = 24$

010



스케줄링 - RR(Round Robin)

- 시분할 시스템을 위해 고안된 방식이다.
- FCFS 알고리즘을 선점 형태로 변형한 기법이다.
- 할당되는 시간이 클 경우 FCFS 기법과 같아진다.

011



교착상태 발생의 필요 충분 조건

- 상호 배제(Mutual Exclusion) : 한 번에 한 개의 프로세스만이 공유 자원을 사용할 수 있어야 함
- 점유와 대기(Hold and Wait) : 하나의 자원을 점유하고 있으면서 다른 프로세스에 할당되어 사용되고 있는 자원을 추가로 점유하기 위해 대기하는 프로세스가 있어야 함
- 비선점(Non-preemption) : 다른 프로세스에 할당된 자원은 사용이 끝날 때까지 강제로 빼앗을 수 없어야 함
- 환형 대기(Circular Wait) : 공유 자원과 공유 자원을 사용하기 위해 대기하는 프로세스들이 원형으로 구성되어 있어야 함

012



기억장치의 배치 전략

- 최초 적합(First Fit) : 첫 번째 분할 영역에 배치
- 최적 적합(Best Fit) : 단편화를 가장 작게 남기는 분할 영역에 배치
- 최악 적합(Worst Fit) : 단편화를 가장 많이 남기는 분할 영역에 배치

013



가상기억장치 - 페이징 기법

- 프로그램과 주기억장치의 영역을 동일한 크기로 나눈 후 적재시켜 실행하는 기법이다.
- 프로그램을 일정한 크기로 나눈 단위를 페이지라고 한다.
- 내부 단편화가 발생할 수 있다.

014



가상기억장치 - 세그먼테이션 기법

- 다양한 크기의 논리적인 단위로 나눈 후 적재시켜 실행시키는 기법이다.
- 프로그램을 논리적인 크기로 나눈 단위를 세그먼트라고 한다.
- 외부 단편화가 발생할 수 있다.

015



주요 페이지 교체 알고리즘

- FIFO : 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 기법
- LRU : 최근에 가장 오랫동안 사용하지 않은 페이지를 교체하는 기법
- NUR : 최근에 사용하지 않은 페이지를 교체하는 기법, 참조 비트와 변형 비트가 사용됨

016



주요 디스크 스케줄링

예 초기 헤드 위치가 53번 트랙이고, 디스크 대기 큐에 다음과 같은 순서의 액세스 요청이 대기 중일 때 디스크 스케줄링별 헤드의 이동 순서와 총 이동 거리를 구하시오.

디스크 대기 큐 : 98, 183, 37, 122, 14, 124, 65, 67

FCFS

- 가장 먼저 들어온 트랙에 대한 요청을 먼저 서비스하는 기법이다.
- 이동 순서 : 53 → 98 → 183 → 37 → 122 → 14 → 124 → 65 → 67
- 총 이동 거리 : $45+85+146+85+108+110+59+2 = 640$

SSTF

- 탐색 거리가 가장 짧은 트랙에 대한 요청을 먼저 서비스하는 기법이다.
- 이동 순서 : 53 → 65 → 67 → 37 → 14 → 98 → 122 → 124 → 183
- 총 이동 거리 : $12+2+30+23+84+24+2+59 = 236$

017



직접 파일(Direct File)

- 레코드를 임의의 물리적 저장공간에 기록하는 것이다.
- 해싱 함수를 이용하여 물리적 상대주소를 계산한다.

018



2단계 디렉터리

- 중앙에 마스터 파일 디렉터리가 있고, 그 아래에 사용자별로 서로 다른 파일 디렉터리가 있는 2계층 구조이다.
- 마스터 파일 디렉터리는 사용자 파일 디렉터리를 관리한다.
- 사용자 파일 디렉터리는 사용자별 파일을 관리한다.

019



권한(자격) 리스트

- 영역을 중심으로 권한 리스트를 구성한 것이다.
- 각 영역에 대한 권한 리스트는 객체와 그 객체에 허용된 조작 리스트로 구성된다.

020



UNIX의 특징

- 대부분 C 언어로 작성되어 있어 이식성이 높다.
- 장치와 프로세스 간의 호환성이 높다.
- 다중 사용자(Multi-User), 다중 작업(Multi-Tasking)을 지원한다.
- 트리 구조의 파일 시스템을 갖는다.

021 초치기 UNIX - 커널(Kernel)의 기능

- 프로세스(CPU 스케줄링) 관리
- 기억장치 관리
- 파일 시스템 관리
- 입 · 출력 관리

022 초치기 UNIX - 셸(Shell)

- 사용자의 명령어를 인식하여 프로그램을 호출하고 명령을 수행하는 명령어 해석기이다.
- 시스템과 사용자 간의 인터페이스를 담당한다.

023 초치기 UNIX의 주요 명령어

- cat : 파일 내용을 화면에 표시함
- chmod : 파일의 보호 모드를 설정하여 파일의 사용 허가를 지정함
- chown : 소유자를 변경함
- exec : 새로운 프로세스를 수행함
- fork : 새로운 프로세스를 생성함
- ls : 현재 디렉터리 내의 파일 목록을 확인함

024 초치기 비동기식 전송

- 시작 비트와 정지 비트를 붙여서 전송하는 방식이다.
- 문자와 문자 사이의 휴지 시간이 불규칙하다.

025 초치기 해밍 코드

- 수신 측에서 오류가 발생한 비트를 검출한 후 직접 수정하는 방식이다.
- 1비트의 오류만 수정이 가능하다.

026 초치기 통신 프로토콜

서로 다른 기기들 간의 데이터 교환을 정확하고 원활하게 수행할 수 있도록 표준화된 통신 규약이다.

027 초치기 OSI 7 계층(1계층 → 7계층)

물리 계층 → 데이터 링크 계층 → 네트워크 계층 → 전송 계층 → 세션 계층 → 표현 계층 → 응용 계층

028 초치기 OSI 7계층의 주요 계층

- 데이터 링크 계층 : 신뢰성 있고 효율적인 정보 전송을 할 수 있도록 시스템 간 연결 설정과 유지 및 종료를 담당함
- 네트워크 계층 : 개방 시스템들 간의 네트워크 연결을 관리하는 기능과 데이터의 교환 및 중계 기능을 함
- 전송 계층 : 종단 시스템 간에 투명한 데이터 전송을 가능하게 함

029 초치기 X.25의 계층 구조

- 물리 계층
- 프레임 계층
- 패킷 계층

030 초치기 TCP/IP 계층별 주요 프로토콜

- 응용 계층 : TELNET, FTP, SMTP, SNMP, E-Mail 등
- 전송 계층 : TCP, UDP 등
- 인터넷 계층 : IP, ICMP, IGMP, ARP, RARP 등
- 네트워크 액세스 계층 : Ethernet, IEEE 802, HDLC, X.25, RS-232C 등

031 초치기 경로 제어(라우팅) 프로토콜

- 종류 : RIP, OSPF, EGP, BGP, EIGRP 등
- 거리 벡터 방식 : RIP, EIGRP, BGP 등
- 링크 상태 방식 : OSPF

032 초치기 패킷 교환 방식

- 메시지를 일정한 길이의 패킷으로 잘라서 전송하는 방식이다.
- 음성 전송보다 데이터 전송에 더 적합하다.
- 장애가 발생하여도 다른 정상적인 경로를 선택하여 우회할 수 있다.
- 대량의 데이터 전송 시 전송 지연이 많아진다.

033 초치기 고속 이더넷

- 100 BASE T라고도 불리는 이더넷의 고속 버전이다.
- CSMA/CD를 사용하며, UTP 케이블을 이용해 100Mbps의 속도로 전송한다.

034 초치기 IPv6 주소

- 16비트씩 8부분, 총 128비트로 구성되어 있다.
- 각 부분을 16진수로 표현하고, 콜론(:)으로 구분한다.
- 인증성, 기밀성, 데이터 무결성의 지원으로 보안 문제를 해결할 수 있다.
- 주소의 확장성, 융통성, 연동성이 뛰어나다.

035 초치기 소프트웨어 공학의 기본 원칙

- 현대적인 프로그래밍 기술을 계속적으로 적용해야 한다.
- 개발된 소프트웨어의 품질이 유지되도록 지속적으로 검증해야 한다.
- 소프트웨어 개발 관련 사항 및 결과에 대한 명확한 기록을 유지해야 한다.

036 초치기 폭포수 모형

- 이전 단계로 돌아갈 수 없다는 전제하에 각 단계를 확실히 매듭짓고 다음 단계를 진행하는 개발 방법론이다.
- 보웬이 제시한 고전적 생명 주기 모형이다.
- 요구사항을 반영하기 어렵다.

037 초치기 나선형 모형

- 나선을 따라 돌듯이 점진적으로 완벽한 최종 소프트웨어를 개발하는 것이다.
- ‘계획 수립 → 위험 분석 → 개발 및 검증 → 고객 평가’ 과정이 반복적으로 수행된다.

정보처리산업기사 핵심 요약

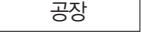
038 애자일 개발 4가지 핵심 가치

- 프로세스와 도구보다는 개인과 상호작용에 더 가치를 둔다.
- 방대한 문서보다는 실행되는 SW에 더 가치를 둔다.
- 계약 협상보다는 고객과 협업에 더 가치를 둔다.
- 계획을 따르기 보다는 변화에 반응하는 것에 더 가치를 둔다.

039 XP의 핵심 가치

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

040 자료 흐름도의 구성 요소

기호	표기법
프로세스(Process)	
자료 흐름(Data Flow)	
자료 저장소(Data Store)	
단말(Terminator)	

041 자료 사전의 표기 기호

- = : 정의
- + : 연결
- () : 생략

- [] : 선택
- { } : 반복
- * * : 설명

042 UML

- 시스템 개발자와 고객 또는 개발자 상호 간의 의사소통이 원활하게 이루어지도록 표준화한 대표적인 객체지향 모델링 언어이다.
- 구성 요소 : 사물(Things), 관계(Relationships), 다이어그램(Diagram)

043 UML의 주요 관계

- 일반화(Generalization) 관계 : 하나의 사물이 다른 사물에 비해 더 일반적인지 구체적인지를 표현
- 의존(Dependency) 관계 : 필요에 의해 서로에게 영향을 주는 짧은 시간 동안만 연관을 유지하는 관계를 표현
- 실체화(Realization) 관계 : 사물이 할 수 있거나 해야 하는 기능으로 서로를 그룹화 할 수 있는 관계를 표현

044 소프트웨어 아키텍처 뷰의 종류

- 유스케이스(Use Case) 뷰
- 논리적(Logical) 뷰
- 구현(Implementation) 뷰
- 프로세스(Process) 뷰
- 배포(Deployment) 뷰

045 초치기 파이프 - 필터 패턴

- 각 단계를 필터 컴포넌트로 캡슐화하여 파이프를 통해 데이터를 전송하는 패턴이다.
- 서버 시스템이 입력 데이터를 받아 처리하고 결과를 다음 서버 시스템으로 넘겨주는 과정을 반복한다.

046 초치기 메소드

객체가 메시지를 받아 실행해야 할 때 객체의 구체적인 연산을 정의한 것이다.

047 초치기 클래스

- 공통된 속성과 연산을 갖는 객체의 집합이다.
- 클래스에 속한 각각의 객체를 인스턴스라 한다.

048 초치기 추상 클래스

- 구체 클래스에서 구현하려는 기능들의 공통점만을 모아 추상화한 클래스이다.
- 인스턴스 생성이 불가능하다.

049 초치기 캡슐화(Encapsulation)

- 데이터와 데이터를 처리하는 함수를 하나로 묶는 것을 의미한다.
- 외부 모듈의 변경으로 인한 파급 효과가 적다.
- 재사용이 용이하다.

050 초치기 상속(Inheritance)

이미 정의된 상위 클래스의 모든 속성과 연산을 하위 클래스가 물려받는 것이다.

051 초치기 림바우의 분석 기법

- 객체(Object) 모델링 : 정보 모델링이라고도 하며, 속성과 연산 식별 및 객체들 간의 관계를 규정하여 객체 다이어그램으로 표시하는 것
- 동적(Dynamic) 모델링 : 상태 다이어그램을 이용하여 객체들 간의 동적인 행위를 표현하는 모델링
- 기능(Functional) 모델링 : 자료 흐름도를 이용하여 프로세스들 간의 처리 과정을 표현한 모델링

052 초치기 생성 패턴 - 빌더(Builder)

- 인스턴스를 건축 하듯이 조합하여 객체를 생성한다.
- 객체의 생성 과정과 표현 방법을 분리하고 있다.

053 초치기 생성 패턴 - 프로토타입(Prototype)

- 원본 객체를 복제하는 방법으로 객체를 생성하는 패턴이다.
- 비용이 큰 경우 주로 이용한다.

054 초치기 생성 패턴 - 싱글톤(Singleton)

- 생성된 객체를 어디서든 참조할 수 있지만, 여러 프로세스가 동시에 참조할 수는 없다.
- 불필요한 메모리 낭비를 최소화할 수 있다.

055 구조 패턴 - 데코레이터(Decorator)

- 객체 간의 결합을 통해 능동적으로 기능들을 확장할 수 있는 패턴이다.
- 객체에 부가적인 기능을 추가하기 위해 다른 객체들을 덧붙이는 방식으로 구현한다.

056 화이트박스 테스트

- 원시 코드의 논리적인 모든 경로를 테스트하여 테스트 케이스를 설계하는 방법이다.
- 원시 코드의 모든 문장을 한 번 이상 실행함으로써 수행된다.

057 화이트박스 테스트 - 루프 검사

- 프로그램의 반복(Loop) 구조에 초점을 맞춰 실시하는 테스트 케이스 설계 기법이다.
- 반복 구조 : 단순 루프, 중첩 루프, 연결 루프, 비구조적 루프

058 화이트박스 테스트의 종류

- 기초 경로 검사
- 제어 구조 검사
 - 조건 검사
 - 루프 검사
 - 데이터 흐름 검사

059 블랙박스 테스트 종류

- 동치 분할 검사
- 경계값 분석
- 원인-효과 그래프 검사
- 오류 예측 검사
- 비교 검사

060 소프트웨어 테스트 순서

단위 테스트 → 통합 테스트 → 시스템 테스트 → 인수 테스트

061 단위 테스트

- 모듈이나 컴포넌트에 초점을 맞춰 테스트하는 것이다.
- 사용자의 요구사항을 기반으로 한 기능성 테스트를 최우선으로 수행한다.

062 통합 테스트

- 단위 테스트가 끝난 모듈을 통합하는 과정에서 발생하는 오류 및 결함을 찾는 테스트 기법이다.
- 종류 : 하향식 통합 테스트, 상향식 통합 테스트, 혼합식 통합 테스트

063 사용자 인터페이스의 특징

- 사용자의 편리성과 가독성을 높여준다.
- 작업 시간을 단축시킨다.
- 업무에 대한 이해도를 높여준다.
- 사용자 중심으로 설계되어 있다.

064 사용자 인터페이스의 구분

- CLI(Command Line Interface) : 명령과 출력이 텍스트 형 태로 이뤄지는 인터페이스
- GUI(Graphical User Interface) : 아이콘이나 메뉴를 마우스 로 선택하여 작업을 수행하는 그래픽 환경의 인터페이스
- NUI(Natural User Interface) : 사용자의 말이나 행동으로 기기를 조작하는 인터페이스

065 형상 관리

- 소프트웨어의 변경 사항을 관리하기 위해 개발된 일련의 활동이다.
- 관리 항목 : 소스 코드, 프로젝트 계획, 분석서, 설계서, 지침서, 프로그램, 테스트 케이스 등

066 빌드 자동화 도구

- 빌드를 포함하여 테스트 및 배포를 자동화하는 도구이다.
- 종류 : Ant, Make, Maven, Gradle, Jenkins 등



2과목 프로그래밍 언어 활용

067 C/C++의 데이터 타입과 크기

종류	데이터 타입	크기
문자	char	1Byte
부호없는 문자형	unsigned char	1Byte
정수	short	2Byte
	int	4Byte
부호없는 정수형	unsigned short	2Byte
	unsigned int	4Byte
실수	float	4Byte
	double	8Byte

068 변수명 작성 규칙

- 첫 글자로 숫자는 올 수 없다.
- 공백이나 *, +, -, / 등의 특수문자를 사용할 수 없다.
- 예약어를 변수명으로 사용할 수 없다.

069 상수의 생성

C언어에서 상수를 만들 때는 const 또는 #define 예약어를 사용한다.

070 외부 변수

- 변수나 함수를 참조(reference)하기 위한 변수이다.
- 함수가 종료된 뒤에도 값이 소멸되지 않는다.

071



산술 연산자

연산자	의미	비고
%	나머지	정수만 연산할 수 있으며, 실수를 사용하면 오류가 발생함
++	증가	• 전치 : 변수 앞에 증감 연산자가 오는 형태로 먼저 변수의 값을 증감시킨 후 변수를 연산에 사용함(++a, --a)
--	감소	• 후치 : 변수 뒤에 증감 연산자가 오는 형태로 먼저 변수를 연산에 사용한 후 변수의 값을 증감시킴(a++, a--)

072



관계 연산자

- == : 같다
- != : 같지 않다
- > : (왼쪽이) 크다
- >= : (왼쪽이) 크거나 같다
- < : (왼쪽이) 작다
- <= : (왼쪽이) 작거나 같다

073



비트 연산자

- & (and) : 모든 비트가 1일 때만 1
- ^ (xor) : 모든 비트가 같으면 0, 하나라도 다르면 1
- | (or) : 모든 비트 중 한 비트라도 1이면 1
- ~ (not) : 각 비트의 부정, 0이면 1, 1이면 0
- << (왼쪽 시프트) : 비트를 왼쪽으로 이동
- >> (오른쪽 시프트) : 비트를 오른쪽으로 이동

074



논리 연산자

- ! (not) : 부정
- && (and) : 모두 참이면 참
- || (or) : 하나라도 참이면 참

075



조건 연산자

조건에 따라 서로 다른 수식을 수행한다.

예) $mx = a < b ? b : a;$

a가 b보다 작으면 mx에 b를 저장하고 그렇지 않으면 mx에 a를 저장한다.

076



연산자 우선순위

대분류	중분류	연산자	결합 규칙	우선 순위
단항 연산자	단항 연산자	! ~ ++ -- sizeof	←	높음
이항 연산자	산술 연산자	* / % + -	→	↑
	시프트 연산자	<< >>		
	관계 연산자	< <= >= > == !=		
	비트 연산자	& ^ 		
	논리 연산자	&& 		
삼항 연산자	조건 연산자	? :	→	↓
대입 연산자	대입 연산자	= += -= *= /= %= <<= >>= 등	←	
순서 연산자	순서 연산자	,	→	
				낮음

077



scanf() 함수

키보드로 입력받아 변수에 저장하는 함수이다.

예) `scanf("%d %f", &i, &j);`

정수를 입력받아 i에 저장하고, 실수를 입력받아 j에 저장한다.

078 주요 서식 문자열

- %d : 정수형 10진수를 입 · 출력하기 위해 지정함
- %o : 정수형 8진수를 입 · 출력하기 위해 지정함
- %x : 정수형 16진수를 입 · 출력하기 위해 지정함
- %c : 문자를 입 · 출력하기 위해 지정함
- %s : 문자열을 입 · 출력하기 위해 지정함
- %f : 소수점을 포함하는 실수를 입 · 출력하기 위해 지정함

079 printf() 함수

인수로 주어진 값을 화면에 출력하는 함수이다.

예) `printf("%d, %c", a, b);`

a의 값을 정수로 출력하고 쉼표(.)와 공백 한 칸을 띄운 후, b의 값을 문자로 출력한다.

080 주요 제어문자

- \n : 다음 줄 앞으로 이동함
- \b : 왼쪽으로 한 칸 이동함
- \t : 일정 간격 띄움
- \r : 현재 줄의 처음으로 이동함
- \0 : 널 문자를 출력함
- \a : 스피커로 벨 소리를 출력함
- \\ : 역 슬래시를 출력함
- \f : 한 페이지를 넘김

081 Java에서의 표준 출력

- `printf()`
 - 예) `System.out.printf("%d", r);`
r의 값을 10진수 정수로 출력한다
- `print()`
 - 예) `System.out.print(r + s);`
r과 s를 더한 값을 출력한다.
- `println()`
 - 예) `System.out.println(r + "은(는) 소수");`
r의 값과 은(는) 소수를 출력한 후, 커서를 다음 줄의 처음으로 옮긴다.

082 기타 표준 입·출력 함수

- `getchar()` : 키보드로 한 문자를 입력받아 변수에 저장하는 함수
- `gets()` : 키보드로 문자열을 입력받아 변수에 저장하는 함수로, [Enter]를 누르기 전까지를 하나의 문자열로 인식하여 저장함
- `putchar()` : 인수로 주어진 한 문자를 화면에 출력하는 함수
- `puts()` : 인수로 주어진 문자열을 화면에 출력한 후 커서를 자동으로 다음 줄 앞으로 이동하는 함수

083 파일 입력 함수

- `fscanf()` : 파일 포인터 변수가 가리키는 위치에서 데이터를 가져와 지정한 자료형으로 변수에 저장하는 함수
- `fgetc()` : 파일로부터 한 문자를 입력받아 변수에 저장하는 함수
- `fgets()` : 파일로부터 문자열을 입력받아 변수에 저장하는 함수

084 초치기 단순 if문

- 조건이 한 개일 때 사용하는 제어문이다.
- 조건이 참일 때만 실행하는 경우

예

```
if (a > b)
    printf("Gilbut");
```

a가 b보다 크면 Gilbut을 출력하고, 아니면 if문을 벗어난다.

- 조건이 참일 때와 거짓일 때 실행할 문장이 다른 경우

예

```
if (a > b)
    printf("참");
else
    printf("거짓");
```

a가 b보다 크면 참을 출력하고, 아니면 거짓을 출력한다.

085 초치기 switch문

- 조건에 따라 분기할 곳이 여러 곳인 경우 간단하게 처리할 수 있는 제어문이다.
- break문이 생략되면 수식과 레이블이 일치할 때 실행할 문장부터 break문 또는 switch문이 종료될 때까지 모든 문장이 실행된다.

예

```
switch(a) {
    case 1:
        printf("바나나");
        break;
    case 2:
        printf("딸기");
        break;
    default:
        printf("없음");
}
```

- a가 1이면 바나나를 출력하고 switch문을 탈출한다.
- a가 2면 딸기를 출력하고 switch문을 탈출한다.
- a가 1이나 2가 아니면 없음을 출력하고 switch문을 탈출한다.

086 초치기 for문

초기값, 최종값, 증가값을 지정하는 수식을 이용해 정해진 횟수를 반복하는 제어문이다.

예

```
for (i = 1; i <= 10; i++)
    sum = sum + i;
```

반복 변수 i가 1부터 1씩 증가하면서 10보다 작거나 같은 동안 sum에 i의 값을 누적시킨다.

087 초치기 while문

조건이 참인 동안 실행할 문장을 반복 수행하는 제어문이다.

예

```
while (i <= 10)
    i = i + 1;
```

i가 10보다 작거나 같은 동안 i의 값을 1씩 누적시킨다.

088 초치기 do~while문

- 조건이 참인 동안 정해진 문장을 반복 수행하다가 조건이 거짓이면 반복문을 벗어난다.
- 실행할 문장을 무조건 한 번 실행한 다음 조건을 판단하여 탈출 여부를 결정한다.

예

```
do
    i = i + 1;
while (i <= 10);
```

i가 10보다 작거나 같은 동안 i의 값을 1씩 누적시킨다.

089 초치기 1차원 배열

변수들을 일직선상의 개념으로 조합한 배열이다.

예 char a[3] = {'A', 'B', 'C'};

3개의 요소를 갖는 문자형 배열 a를 선언한다.

	a[0]	a[1]	a[2]
배열 a	A	B	C

090 초치기 2차원 배열

변수들을 평면, 즉 행과 열로 조합한 배열이다.

예 int b[2][3] = {{11, 22, 33}, {44, 55, 66}};

2개의 행과 3개의 열을 갖는 정수형 배열 b를 선언한다.

	a[0][0]	a[0][1]	a[0][2]
배열 b	11	22	33
	44	55	66
	a[1][0]	a[1][1]	a[1][2]

091 초치기 배열 형태의 문자열 변수

• C언어에서는 큰따옴표(“ ”)로 묶인 글자는 글자 수에 관계없이 문자열로 처리된다.

• 배열에 문자열을 저장하면 문자열의 끝을 알리기 위한 널 문자('\0')가 문자열 끝에 자동으로 삽입된다.

예 char a[5] = "love";

5개의 요소를 갖는 문자형 배열 a를 선언하고, "love"로 초기화한다.

배열 a	l	o	v	e	\0
	a[0]	a[1]	a[2]	a[3]	a[4]

092 초치기 포인터와 포인터 변수

- 포인터 변수를 선언할 때는 자료의 형을 먼저 쓰고 변수명 앞에 간접 연산자 *를 붙인다(예 int *a;).
- 포인터 변수에 주소를 저장하기 위해 변수의 주소를 알아낼 때는 변수 앞에 번지 연산자 &를 붙인다(예 a = &b;).
- 실행문에서 포인터 변수에 간접 연산자 *를 붙이면 해당 포인터 변수가 가리키는 곳의 값을 말한다(예 c = *a;).

예제 다음 C언어로 구현된 프로그램의 출력 결과를 확인하시오.

```
main( )
{
    int a = 50; ①
    int *b; ②
    b = &a; ③
    *b = *b+20; ④
    printf("%d, %d", a, *b); ⑤
}
```

- ① 정수형 변수 a를 선언하고 50으로 초기화한다.
- ② 정수형 변수가 저장된 곳의 주소를 기억할 포인터 변수 b를 선언한다.
- ③ 정수형 변수 a의 주소를 포인터 변수 b에 기억시킨다. b에는 a의 주소가 저장된다.
- ④ b가 가리키는 곳의 값에 20을 더한다. b가 가리키는 곳이 a이므로 결국 a의 값도 바뀌는 것이다.
- ⑤ 결과 70, 70



093 포인터와 배열

- 배열을 포인터 변수에 저장한 후 포인터를 이용해 배열의 요소에 접근할 수 있다.
- 배열 위치를 나타내는 첨자를 생략하고 배열의 대표명만 지정하면 배열의 첫 번째 요소의 주소를 지정하는 것과 같다.

예

```
1 int a[5], *b
2 b = a;
3 b = &a[0];
```

- 5개의 요소를 갖는 정수형 배열 a와 정수형 포인터 변수 b를 선언한다.
- 배열의 대표명을 적었으므로 a 배열의 시작 주소인 a[0]의 주소를 b에 저장한다.
- a 배열의 첫 번째 요소인 a[0]의 주소(&)를 b에 저장한다.

	a[0]	a[1]	a[2]	a[3]	a[4]	← 배열 표기 방법
배열 a	첫 번째	두 번째	세 번째	네 번째	다섯 번째	
	*(a+0)	*(a+1)	*(a+2)	*(a+3)	*(a+4)	← 포인터 표기 방법

094 사용자 정의 함수

사용자가 필요한 기능을 취향대로 만들어 사용할 수 있는 함수이다.

```
#include <stdio.h>
2 int func(int x, int y) {
3     int sum = x + y;
4     return sum;
5 }
main() {
15 int r = func(3, 5);
6     printf("%d", r);
}
```

- 정수형 변수 r을 선언하고, 3과 5를 인수로 func() 함수를 호출한 후 돌려받은 값으로 초기화한다.
- 정수를 반환하는 func() 함수의 시작점이다. 1번에서 전달 받은 3과 5를 x와 y가 받는다. (x = 3, y = 5)

- 정수형 변수 sum을 선언하고 x와 y를 더한 값 8로 초기화한다. (sum = 8)
- sum의 값 8을 함수를 호출했던 5번으로 반환한다.
- 4번에서 돌려받은 값 8을 r에 저장한다. (r = 8)
- r의 값 8을 출력한다.

결과 8

095 Python의 기본 문법

- if나 for와 같이 코드 블록을 포함하는 명령문을 작성할 때 코드 블록은 콜론(:)과 여백으로 구분한다.
- 문자열에 따옴표가 포함되는 경우 다른 따옴표를 이용하여 문자열을 묶어줘야 한다.

예 'She said "I like it" '

096 Python의 input() 함수

- 키보드로 입력받아 변수에 저장하는 함수이다.
- 입력되는 값은 문자열로 취급되어 저장된다.

예 a = input('입력하세요.')

- 입력하세요.가 출력되고 커서가 깜빡거리며 입력을 기다린다.
- 키보드로 값을 입력하면 변수 a에 저장된다.

097 Python의 print() 함수

인수로 주어진 값을 화면에 출력하는 함수이다.

예 print(82, 24, sep = '-', end = ',')

82와 24 사이에 분리문자 '-'가 출력되고, 마지막에 종료문자 ','가 출력된다.

결과 82_24,

098



Range

연속된 숫자를 생성하는 것으로, 리스트, 반복문 등에서 많이 사용된다.

예1 다음 C언어 `a = list(range(5))`

0에서 4까지 연속된 숫자를 리스트 a로 저장한다.

리스트 a

0	1	2	3	4
---	---	---	---	---

예2 `b = list(range(4, 9))`

4에서 8까지 연속된 숫자를 리스트 b로 저장한다.

리스트 b

4	5	6	7	8
---	---	---	---	---

예3 `c = list(range(1, 15, 3))`

1에서 14까지 3씩 증가하는 숫자들을 리스트 c로 저장한다.

리스트 c

1	4	7	10	13
---	---	---	----	----

099



슬라이스

문자열이나 리스트와 같은 순차형 객체에서 일부를 잘라(slicing) 반환하는 기능이다.

예 `a = ['a', 'b', 'c', 'd', 'e']` 일때

`a[1:3] → ['b', 'c']`

`a[0:5:2] → ['a', 'c', 'e']`

`a[3:] → ['d', 'e']`

`a[:3] → ['a', 'b', 'c']`

`a[::-3] → ['a', 'd']`

100



Python의 for문

• range를 이용하는 방식

예

```
for i in range(1, 11):
    sum = sum + i
```

i에 1부터 10까지 순서대로 저장하며 sum에 i의 값을 누적시키는 실행문을 반복 수행한다.

• 리스트(List)를 이용하는 방식

예

```
a = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ]
for i in a:
    sum = sum + i
```

a 리스트에 저장된 10개의 요소를 i에 순서대로 저장하며 sum에 i의 값을 누적시키는 실행문을 반복 수행한다.

101



Python의 클래스

클래스를 사용하려면 클래스 이름을 정하고 객체 생성을 위한 속성과 메소드(함수)를 정의한 후, 객체를 선언하면 된다.

예제 다음 Python으로 구현된 프로그램의 출력 결과를 확인하시오.

```
class Cls:
    x = 10
    ④ def add(self, a):
    ⑤     return a + self.x
    ① a = Cls( )
    ② a.x = 5
    ③⑥ print(a.add(5))
```

- ① Cls 클래스의 객체 a를 생성한다.
- ② 객체 a의 변수 x에 5를 저장한다. (a.x = 5)
- ③ 5를 인수로 a 객체의 add() 메소드를 호출한 후 돌려받은 값을 출력한다.
- ④ add() 메소드의 시작점이다. ③번에서 전달받은 5를 a가 받는다.
- ⑤ a와 객체의 변수 x를 더한 값 10을 메소드를 호출했던 ⑥번으로 반환한다.
- ⑥ ⑤번에서 돌려받은 값 10을 출력한다.

결과 10

102 초치기 HTML - 프레임의 주요 태그

〈frameset〉 태그

- 화면을 분할한다.
- 〈frameset〉...〈/frameset〉 태그 사이에는 분할할 프레임의 개수만큼 〈frame〉 태그를 사용한다.

〈frame〉 태그

- 분할된 각각의 프레임에 표시할 HTML 문서를 지정한다.
- 〈frameset〉으로 분할된 영역에 〈frame〉 태그가 적용되는 순서는 다음과 같다.
 - 화면이 가로로 분할된 경우 : 위쪽 → 아래쪽
 - 화면이 세로로 분할된 경우 : 왼쪽 → 오른쪽
- 분할된 프레임의 개수는 〈frameset〉의 rows 또는 cols 속성으로 알 수 있다.

예 rows="20%,*" → 2개, cols="200, *, 500" → 3개

103 초치기 TML - 테이블의 주요 태그

- 〈table〉 : 테이블에 관한 세부사항을 설정함
- 〈thead〉 : 테이블의 머리글 부분을 정의함
- 〈tbody〉 : 테이블의 본문 부분을 정의함
- 〈tfoot〉 : 테이블의 바닥글 부분을 정의함
- 〈tr〉 : 행을 만들
- 〈td〉 : 셀을 만들
- 〈th〉 : 셀을 만들면서 제목 스타일을 적용함

104 초치기 HTML - 〈style〉 태그

- 서식을 지정하는 태그로, 〈head〉 부분에 지정하면 테이블 전체에 공통으로 적용된다.
- 서식 지정 형식

요소이름:선택자 { 속성1:속성값1; 속성2:속성값2; ...
속성n:속성값n }

- 요소이름 : 태그 이름에서 '〈'와 '〉'를 제외하고 입력함
- 선택자 : 요소 중 일부에만 서식을 지정할 때 사용하는 옵션으로, 생략이 가능함
 - ▶ first-child : 첫 번째 요소에 적용
 - ▶ last-child : 마지막 요소에 적용
 - ▶ nth-child(N) : N번째 요소마다 적용
- 속성:속성값 : 요소에 적용할 속성과 속성값을 입력함. 2개 이상의 속성을 지정할 때는 세미콜론(;)을 이용하여 구분함

105 초치기 HTML - 〈form〉 태그

- 사용자로부터 정보를 입력받는 틀을 정의한다.
- 관련 속성
 - method : 데이터 전송 방식을 지정함
 - ▶ get : 입력받은 데이터를 URL에 첨부하여 전송함
 - ▶ post : 입력받은 데이터를 메시지 형식으로 전송함
 - action : 데이터를 전송할 URL을 지정함

106 초치기 JavaScript - document.write()

가장 일반적인 출력 메소드로, 인수로 출력할 데이터를 입력하면 화면에 데이터가 출력된다.

예 document.write("Sinagong"); → 화면에 Sinagong이 출력됨

107



JavaScript - 대화상자

- 알림 대화상자

alert(내용);

대화상자 본문에 '내용'이 표시되고, 아래쪽에 <확인> 단추가 표시된다.

- 확인 대화상자

confirm(내용);

대화상자 본문에 '내용'이 표시되고, 아래쪽에 <확인> 과 <취소> 단추가 표시된다.

- 입력 대화상자

prompt(내용, 기본값);

- 대화상자 본문에 '내용'이 표시되고, '내용' 아래에 '기본값'이 입력된 텍스트 상자가 표시된다.
- 대화상자 아래쪽에 <확인>과 <취소> 단추가 표시된다.

108



JavaScript - 배열의 주요 메소드

- pop() : 배열의 마지막 요소를 제거함
- push() : 배열의 마지막에 요소를 추가함
- join() : 배열의 모든 요소를 하나의 문자열로 변환함
- shift() : 배열의 첫 번째 요소를 제거함
- splice() : 배열에서 지정한 범위의 요소를 제거한 후 제거된 위치에 지정한 값을 저장함

109



객체지향 프로그래밍 언어의 종류

- JAVA
 - 분산 네트워크 환경에 적용할 수 있다.
 - 캡슐화가 가능하고 재사용성 높다.
- C++ : C 언어에 객체지향 개념을 적용한 언어
- Smalltalk : 1세대 객체지향 프로그래밍 언어 중 하나로 순수한 객체지향 프로그래밍 언어

110



객체지향 프로그래밍 언어의 구성 요소

- 객체(Object) : 데이터(속성)와 이를 처리하기 위한 연산(메소드)을 결합시킨 실체
- 클래스(Class) : 두 개 이상의 유사한 객체들을 묶어서 하나의 공통된 특성을 표현하는 요소

111



객체지향 프로그래밍 언어의 주요 특징

- 캡슐화(Encapsulation) : 데이터(속성)와 함수를 하나로 묶는 것
- 상속성(Inheritance) : 이미 정의된 상위 클래스의 모든 속성과 연산을 하위 클래스가 물려받는 것

112



스크립트 언어의 종류

- 자바스크립트 : 웹 페이지의 동작을 제어하는 데 사용되는 클라이언트용 스크립트 언어
- PHP : Linux, Unix, Windows 운영체제에서 사용 가능한 서버용 스크립트 언어
- 파이썬(Python) : 귀도 반 로섬이 발표한 대화형 인터프리터 언어
- 셸 스크립트 : 셸에서 사용되는 명령어들의 조합으로 구성된 스크립트 언어
- Basic : 절차지향 기능을 지원하는 대화형 인터프리터 언어

113



웹 스크립트에서 사용되는 제어문

- 선택형 : if, case
- 반복형 : for, while, until

114 JavaScript 프레임워크의 종류

- jQuery : 웹 브라우저 간의 호환성 문제를 해결하고 명령어를 단순화한 프레임워크
- React : 메타(META)에서 개발한 프레임워크
- Angular : 구글(Google)에서 개발한 프레임워크
- Node.js : 인터넷 브라우저 외에도 JavaScript가 동작하도록 함
- Ember : 웹 애플리케이션을 위한 다양한 기능 제공

115 C언어의 대표적인 표준 라이브러리

- math.h
 - 수학 함수들을 제공한다.
 - 주요 함수 : sqrt, pow, abs 등
- stdlib.h
 - 자료형 변환, 난수 발생, 메모리 할당에 사용되는 기능들을 제공한다.
 - 주요 함수 : atoi, atof, srand, rand, malloc, free 등

116 try ~ catch 문

- C++, C#, Java, JavaScript 등의 언어에서 예외 처리 기능을 수행하는 명령문이다.
- 일반적으로 예외가 발생한 경우에는 'try문 → 해당 예외 catch문 → finally문' 순으로 진행한다.
- finally 블록은 예외의 발생과 관계없이 무조건 수행되는데, C++에서는 사용할 수 없다.

117 프레임워크의 특성

- 모듈화(Modularity) : 캡슐화를 통해 모듈화를 강화하고 설계 및 구현의 변경에 따른 영향을 최소화함으로써 소프트웨어의 품질을 향상시키고 유지 보수를 용이하게 함

- 재사용성(Reusability) : 재사용 가능한 모듈들을 제공함으로써 예산 절감, 생산성 향상, 품질 보증이 가능함
- 확장성(Extensibility) : 다형성(Polymorphism)을 통한 인터페이스 확장이 가능하여 다양한 형태와 기능을 가진 애플리케이션 개발이 가능함
- 제어의 역흐름(Inversion of Control) : 개발자가 관리하고 통제해야 하는 객체들의 제어를 프레임워크에 넘김으로써 생산성을 향상시킴

118 결합도

- 모듈 간에 상호 의존하는 정도를 나타낸다.
- 독립적인 모듈이 되기 위해서는 각 모듈 간의 결합도가 약해야 하며 의존하는 모듈이 적어야 한다.
- 결합도의 종류(약함(강함)) : 자료 결합도 < 스템프 결합도 < 제어 결합도 < 외부 결합도 < 공통 결합도 < 내용 결합도
 - 자료 결합도 : 서로 다른 모듈 간에 매개변수 또는 인수를 통해 꼭 필요한 자료만을 교환하는 경우의 결합도
 - 스템프 결합도 : 서로 다른 모듈이 동일한 자료 구조를 참조하는 경우의 결합도
 - 내용 결합도 : 한 모듈이 다른 모듈의 내부 자료를 직접적으로 참조하는 경우의 결합도

119 응집도

- 모듈 안의 요소들이 서로 관련되어 있는 정도를 나타낸다.
- 응집도의 종류(약함(강함)) : 우연적 응집도 < 논리적 응집도 < 시간적 응집도 < 절차적 응집도 < 교환적 응집도 < 순차적 응집도 < 기능적 응집도
 - 논리적 응집도 : 논리적으로 서로 관련 있는 요소들을 모아 하나의 모듈로 작성한 경우의 응집도
 - 절차적 응집도 : 일정한 순서에 의해 처리되어야 할 요소들을 하나의 모듈로 구성한 경우의 응집도로, 전달 데이터와 반환 데이터 사이에 관련이 없음
 - 기능적 응집도 : 모듈 내부의 모든 기능 요소가 한 가지의 작업만을 수행하는 경우의 응집도

120 초치기 재사용

- 이미 개발된 기능들을 파악하고 재구성하여 새로운 시스템 또는 기능 개발에 사용하기 적합하도록 최적화시키는 작업이다.
- 재사용 규모에 따른 분류 : 함수와 객체, 컴포넌트, 애플리케이션

121 초치기 효과적인 모듈 설계 방안

- 결합도는 줄이고 응집도는 높여서 모듈의 독립성과 재사용성을 높인다.
- 복잡도와 중복성을 줄이고 일관성을 유지시킨다.

122 초치기 보안 취약점 (Security Vulnerability)

시스템 기능이나 설계, 구현 단계에서의 문제점 등으로 인해 시스템이 가지게 되는 약점을 의미한다.

123 초치기 보안 3대 요소

- 기밀성 : 시스템 내의 정보와 자원은 인가된 사용자에 제한 접근이 허용되며, 정보가 전송 중에 노출되더라도 데이터를 읽을 수 없음
- 무결성 : 시스템 내의 정보는 오직 인가된 사용자만 수정할 수 있음
- 가용성 : 인가받은 사용자는 언제라도 사용할 수 있음

124 초치기 API(Application Programming Interface)

- 운영체제나 프로그래밍 언어 등에 있는 라이브러리를 이용할 수 있도록 규칙 등을 정의해 놓은 인터페이스를 의미한다.
- Open API : 누구나 무료로 사용할 수 있게 공개된 API

3과목 데이터베이스 활용

125 초치기 자료 구조의 분류

- 선형 구조 : 배열, 선형 리스트, 스택, 큐, 데크
- 비선형 구조 : 트리, 그래프

126 초치기 스택(Stack)

- 리스트의 한쪽 끝으로만 자료의 삽입, 삭제 작업이 이루어지는 자료 구조이다.
- 가장 나중에 삽입된 자료가 가장 먼저 삭제되는 후입선출(LIFO) 방식으로 자료를 처리한다.

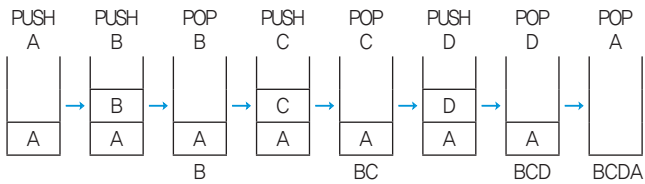
127 초치기 스택의 응용 분야

- 함수 호출의 순서 제어
- 인터럽트의 처리
- 수식 계산 및 수식 표기법
- 컴파일러를 이용한 언어 번역
- 부 프로그램 호출 시 복귀주소 저장
- 서브루틴 호출 및 복귀 주소 저장

128 초치기 스택의 삽입(Push)과 삭제(Pop)

- PUSH : 스택에 자료를 입력하는 명령
- POP : 스택에서 자료를 출력하는 명령

예제 순서가 A, B, C, D로 정해진 입력 자료를 스택에 입력하였다가 B, C, D, A 순서로 출력하는 과정을 나열하시오.



129 큐(Queue)

- 리스트의 한쪽에서는 삽입 작업이 이루어지고 다른 한쪽에서는 삭제 작업이 이루어지는 자료 구조이다.
- 가장 먼저 삽입된 자료가 가장 먼저 삭제되는 선입선출(FIFO) 방식으로 처리한다.

130 데크(Deque)

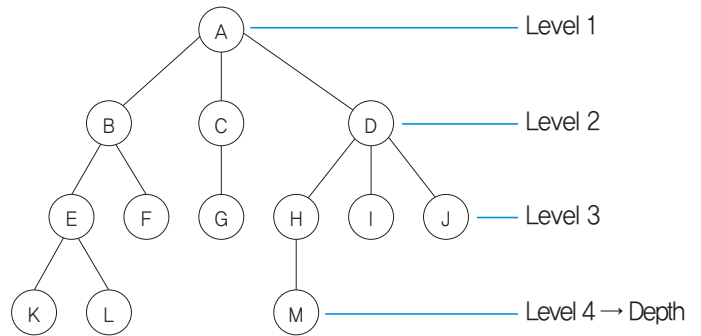
- 삽입과 삭제가 리스트의 양쪽 끝에서 모두 발생할 수 있는 자료 구조이다.
- 입력 제한: 입력은 한쪽에서만 일어나고 출력은 양쪽에서 일어남
- 출력 제한: 입력은 양쪽에서 일어나고 출력은 한쪽에서만 일어남

131 방향/무방향 그래프의 최대 간선 수

- 무방향 그래프의 최대 간선 수: $n(n-1)/2$
- 방향 그래프의 최대 간선 수: $n(n-1)$

132 트리(Tree)

정점(Node, 노드)과 선분(Branch, 가지)을 이용하여 사이클을 이루지 않도록 구성한 그래프(Graph)의 특수한 형태이다.



- 노드(Node): 트리의 기본 요소로서 자료 항목과 다른 항목에 대한 가지(Branch)를 합친 것

예 A, B, C, D, E, F, G, H, I, J, K, L, M

- 디그리(Degree, 차수): 각 노드에서 뻗어 나온 가지의 수

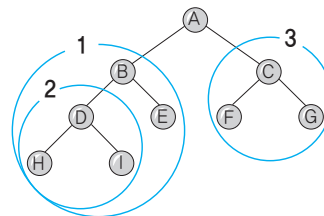
예 A = 3, B = 2, C = 1, D = 3

- 단말 노드(Terminal Node) = 잎 노드(Leaf Node): 자식이 하나도 없는 노드, 즉 디그리가 0인 노드

예 K, L, F, G, M, I, J

133 이진 트리의 운행법

예 다음 트리를 Inorder, Preorder, Postorder 방법으로 운행했을 때 각 노드를 방문한 순서는?



Preorder 운행법의 방문 순서

- ① Preorder는 Root → Left → Right이므로 A13이 된다.
 - ② 1은 B2E이므로 AB2E3이 된다.
 - ③ 2는 DHI이므로 ABDHIE3이 된다.
 - ④ 3은 CFG이므로 ABDHIECFG가 된다.
- ∴ 방문 순서: ABDHIECFG

Inorder 운행법의 방문 순서

- ① Inorder는 Left → Root → Right이므로 1A3이 된다.
- ② 1은 2BE이므로 2BEA3이 된다.
- ③ 2는 HDI이므로 HDIBEA3이 된다.
- ④ 3은 FCG이므로 HDIBEAFCG가 된다.

∴ 방문 순서 : HDIBEAFCG

Postorder의 방문 순서

- ① Postorder는 Left → Right → Root이므로 13A가 된다.
- ② 1은 2EB이므로 2EB3A가 된다.
- ③ 2는 HID이므로 HIDEB3A가 된다.
- ④ 3은 FGC이므로 HIDEBFGCA가 된다.

∴ 방문 순서 : HIDEBFGCA

134 수식의 표기법(Infix → Postfix)

Infix로 표기된 수식에서 연산자를 해당 피연산자 두 개의 뒤(오른쪽)에 오도록 이동하면 Postfix가 된다.

$$X = A / B * (C + D) + E \rightarrow X A B / C D + * E + =$$

- ① 연산 우선순위에 따라 괄호로 묶는다.
 $(X = ((A / B) * (C + D)) + E)$
- ② 연산자를 해당 괄호의 뒤로 옮긴다.

$$X = ((A / B) * (C + D)) + E$$

$$(X ((A B) / (C D) +) * E) + =$$

- ③ 괄호를 제거한다.
 $X A B / C D + * E + =$

135 수식의 표기법(Infix → Prefix)

Infix로 표기된 수식에서 연산자를 해당 피연산자 두 개의 앞(왼쪽)에 오도록 이동하면 Prefix가 된다.

$$X = A / B * (C + D) + E \rightarrow X + * / A B + C D E$$

- ① 연산 우선순위에 따라 괄호로 묶는다.
 $(X = ((A / B) * (C + D)) + E)$
- ② 연산자를 해당 괄호의 앞으로 옮긴다.

$$(X = ((A / B) * (C + D)) + E)$$

$$= (X + (* (/ (A B) + (C D)) E))$$

- ③ 괄호를 제거한다.
 $= X + * / A B + C D E$

136 수식의 표기법(Postfix → Infix)

Postfix는 Infix 표기법에서 연산자를 해당 피연산자 2개의 뒤(오른쪽)로 이동한 것이므로 연산자를 다시 해당 피연산자 2개의 가운데로 옮기면 된다.

$$A B C - / D E F + * + \rightarrow A / (B - C) + D * (E + F)$$

- ① 인접한 피연산자 2개와 오른쪽의 연산자를 괄호로 묶는다.
 $((A (B C -) /) (D (E F +) *) +)$
- ② 연산자를 해당 피연산자의 가운데로 이동시킨다.

$$((A (B C -) /) (D (E F +) *) +)$$

$$((A / (B - C)) + (D * (E + F)))$$

- ③ 필요 없는 괄호를 제거한다.
 $A / (B - C) + D * (E + F)$

137 삽입 정렬(Insertion Sort)

예제 8, 5, 6, 2, 4를 삽입 정렬로 정렬하시오.

• 초기 상태 : 8 5 6 2 4

• 1회전 : 8 5 6 2 4 → 5 8 6 2 4

두 번째 값을 첫 번째 값과 비교하여 5를 첫 번째 자리에 삽입하고 8을 한 칸 뒤로 이동시킨다.

• 2회전 : 5 8 6 2 4 → 5 6 8 2 4

세 번째 값을 첫 번째, 두 번째 값과 비교하여 6을 8자리에 삽입하고 8을 한 칸 뒤로 이동시킨다.

5 6 8 2 4 → 2 5 6 8 4

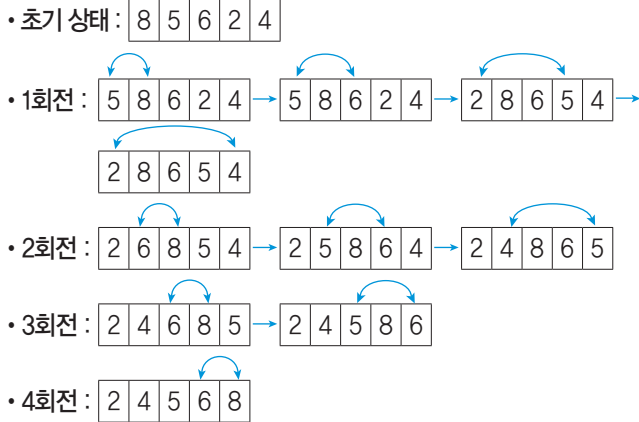
네 번째 값 2를 처음부터 비교하여 맨 처음에 삽입하고 나머지를 한 칸씩 뒤로 이동시킨다.

• 4회전 : 2 5 6 8 4 → 2 4 5 6 8

다섯 번째 값 4를 처음부터 비교하여 5자리에 삽입하고 나머지를 한 칸씩 뒤로 이동시킨다.

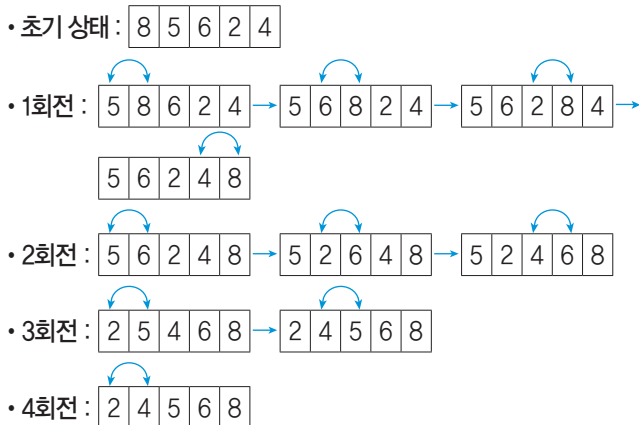
138 선택 정렬(Selection Sort)

예제 8, 5, 6, 2, 4를 선택 정렬로 정렬하시오.



139 버블 정렬(Bubble Sort)

예제 8, 5, 6, 2, 4를 버블 정렬로 정렬하시오.



140 해시 테이블 관련 용어

- Collision(충돌 현상) : 서로 다른 두 개 이상의 레코드가 같은 주소를 갖는 현상
- Synonym : 충돌로 인해 같은 Home Address를 갖는 레코드들의 집합

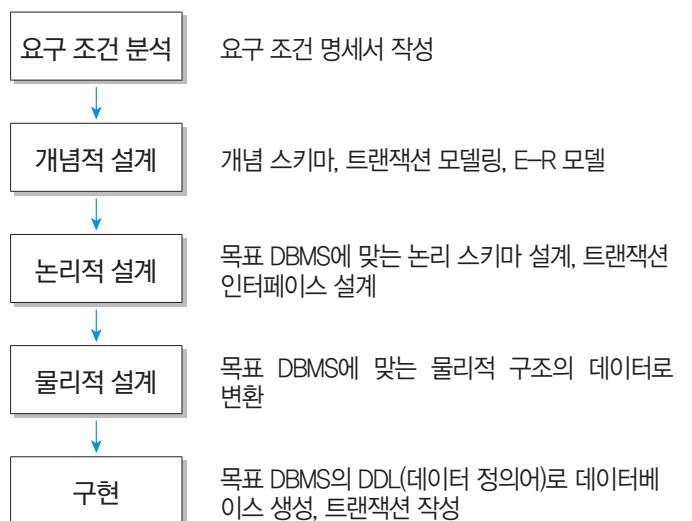
141 해싱 함수(Hashing Function)

- 제산법(Division)
- 제곱법(Mid-Square)
- 폴딩법(Folding)
- 기수 변환법(Radix)
- 대수적 코딩법(Algebraic Coding)
- 숫자 분석법(Digit Analysis, 계수 분석법)
- 무작위법(Random)

142 DBMS의 필수 기능

- 정의 기능 : 모든 응용 프로그램들이 요구하는 데이터 구조를 지원하기 위해 데이터베이스에 저장될 데이터의 형(Type)과 구조에 대한 정의, 이용 방식, 제약 조건 등을 명시하는 기능
- 조작 기능 : 데이터 검색, 갱신, 삽입, 삭제 등을 체계적으로 처리하기 위해 사용자와 데이터베이스 사이의 인터페이스 수단을 제공하는 기능
- 제어 기능 : 데이터베이스를 접근하는 갱신, 삽입, 삭제 작업이 정확하게 수행되어 데이터의 무결성이 유지되도록 제어해야 함

143 데이터베이스 설계 순서



144



물리적 설계 옵션 선택 시 고려 사항

- 반응 시간(Response Time)
- 공간 활용도(Space Utilization)
- 트랜잭션 처리량(Transaction Throughput)

145



데이터 모델에 표시할 요소

- 구조(Structure) : 논리적으로 표현된 개체 타입들 간의 관계로서 데이터 구조 및 정적 성질을 표현함
- 연산(Operation) : 데이터베이스에 저장된 실제 데이터를 처리하는 작업에 대한 명세로서 데이터베이스를 조작하는 기본 도구
- 제약 조건(Constraint) : 데이터베이스에 저장될 수 있는 실제 데이터의 논리적인 제약 조건

146



E-R 다이어그램

기호	기호 이름	의미
	사각형	개체(Entity) 타입
	마름모	관계(Relationship) 타입
	타원	속성(Attribute)
	선, 링크	개체 타입과 속성을 연결

147



튜플(Tuple)

- 릴레이션을 구성하는 각각의 행을 말한다.
- 튜플의 수 = 카디널리티(Cardinality)

148



속성(Attribute)

- 데이터베이스를 구성하는 가장 작은 논리적 단위이다.
- 속성의 수 = 디그리(Degree) = 차수

149



도메인(Domain)

하나의 애트리뷰트가 취할 수 있는 같은 타입의 원자(Atomic)값들의 집합이다.

150



릴레이션의 특징

- 한 릴레이션에는 똑같은 튜플이 포함될 수 없으므로 릴레이션에 포함된 튜플들은 모두 상이하다.
- 한 릴레이션에 포함된 튜플 사이에는 순서가 없다.
- 속성의 유일한 식별을 위해 속성의 명칭은 유일해야 한다.
- 속성의 값은 논리적으로 더 이상 쪼갤 수 없는 원자값만을 저장한다.

151



후보키(Candidate Key)

- 릴레이션을 구성하는 속성들 중에서 튜플을 유일하게 식별하기 위해 사용하는 속성들의 부분집합, 즉 기본키로 사용할 수 있는 속성들을 말한다.
- 릴레이션에 있는 모든 튜플에 대해서 유일성과 최소성을 만족시켜야 한다.

152



널 값(NULL Value)

데이터베이스에서 아직 알려지지 않거나 모르는 값으로서 '해당 없음' 등의 이유로 정보 부재를 나타내기 위해 사용하는, 이론적으로 아무것도 없는 특수한 데이터를 말한다.

153 초치기 기본키(Primary Key)

- 후보키 중에서 특별히 선정된 주키(Main Key)로 중복된 값을 가질 수 없다.
- NULL 값을 가질 수 없다.

154 초치기 대체키(Alternate Key)

- 후보키가 둘 이상일 때 기본키를 제외한 나머지 후보키를 의미한다.
- 보조키라고도 한다.

155 초치기 슈퍼키(Super Key)

- 한 릴레이션 내에 있는 속성들의 집합으로 구성된 키이다.
- 릴레이션을 구성하는 모든 튜플에 대해 유일성은 만족시키지만, 최소성은 만족시키지 못한다.

156 초치기 외래키(Foreign Key)

- 다른 릴레이션의 기본키를 참조하는 속성 또는 속성들의 집합을 의미한다.
- 한 릴레이션에 속한 속성 A와 참조 릴레이션의 기본키인 B가 동일한 도메인 상에서 정의되었을 때의 속성 A를 외래키라고 한다.

157 초치기 무결성

- 개체 무결성 : 기본 테이블의 기본키를 구성하는 어떤 속성도 Null 값이나 중복값을 가질 수 없다는 규정
- 참조 무결성 : 외래키 값은 Null이거나 참조 릴레이션의 기본키 값과 동일해야 함. 즉 릴레이션은 참조할 수 없는 외래키 값을 가질 수 없다는 규정

158 초치기 관계대수

- 관계형 데이터베이스에서 원하는 정보와 그 정보를 검색하기 위해서 어떻게 유도하는가를 기술하는 절차적인 언어이다.
- 질의에 대한 해를 구하기 위해 수행해야 할 연산의 순서를 명시한다.

159 초치기 순수 관계 연산자 - Select

- 릴레이션에 존재하는 튜플 중에서 선택 조건을 만족하는 튜플의 부분집합을 구하여 새로운 릴레이션을 만드는 연산이다.
- 기호 : 시그마(σ)

160 초치기 순수 관계 연산자 - Project

- 주어진 릴레이션에서 속성 리스트에 제시된 속성 값만을 추출하여 새로운 릴레이션을 만드는 연산이다.
- 기호 : 파이(π)

161 초치기 순수 관계 연산자 - Join

- 공통 속성을 중심으로 두 개의 릴레이션을 하나로 합쳐서 새로운 릴레이션을 만드는 연산이다.
- 기호 : \bowtie

162 초치기 순수 관계 연산자 - Division

- $X \supset Y$ 인 두 개의 릴레이션 $R(X)$ 와 $S(Y)$ 가 있을 때, R 의 속성이 S 의 속성값을 모두 가진 튜플에서 S 가 가진 속성을 제외한 속성만을 구하는 연산이다.
- 기호 : \div

163 일반 집합 연산자 - 기호

연산자	기호
합집합(UNION)	U
교집합(INTERSECTION)	\cap
차집합(DIFFERENCE)	-
교차곱(CARTESIAN PRODUCT)	\times

164 일반 집합 연산자 - 교차곱(CARTESIAN PRODUCT)

- 두 릴레이션에 있는 튜플들의 순서쌍을 구하는 연산이다.
- 디그리는 두 릴레이션의 디그리를 더한 것과 같고, 카디널리티는 두 릴레이션의 카디널리티를 곱한 것과 같다.

165 관계해석

- 관계 데이터 모델의 제안자인 코드(Codd)가 수학의 Predicate Calculus(술어 해석)에 기반을 두고 관계 데이터베이스를 위해 제안했다.
- 원하는 정보가 무엇이라는 것만 정의하는 비절차적 특성을 지닌다.

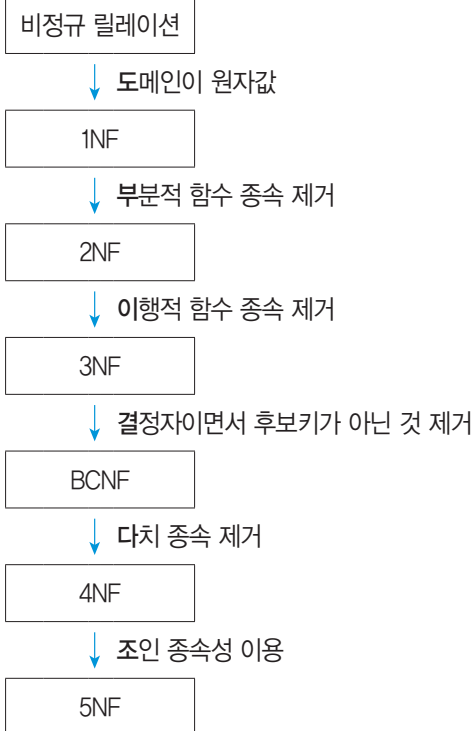
166 정규화(Normalization)

- 함수적 종속성 등의 종속성 이론을 이용하여 잘못 설계된 관계형 스키마를 더 작은 속성의 세트로 쪼개어 바람직한 스키마로 만들어 가는 과정이다.
- 데이터 삽입 시 릴레이션을 재구성할 필요성을 줄인다.

167 이상(Anomaly)

- 정규화를 거치지 않으면 데이터베이스 내에 데이터들이 불필요하게 중복되어 릴레이션 조작 시 예기치 못한 곤란한 현상이 발생하는 것을 의미한다.
- 종류 : 삽입 이상, 삭제 이상, 갱신 이상

168 정규화 과정



169 이행적 종속 관계

$A \rightarrow B$ 이고 $B \rightarrow C$ 일 때 $A \rightarrow C$ 를 만족하는 관계를 의미한다.

170



함수적 종속

데이터들이 어떤 기준값에 의해 종속되는 것을 의미한다.

예 <수강> 릴레이션이 (학번, 이름, 과목명)으로 되어 있을 때, '학번'이 결정되면 '과목명'에 상관없이 '학번'에는 항상 같은 '이름'이 대응된다. '학번'에 따라 '이름'이 결정될 때 '이름'을 '학번'에 함수 종속적이라고 하며 '학번 → 이름'과 같이 쓴다.

171



뷰(View)

- 사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위해 하나 이상의 기본 테이블로부터 유도된, 이름을 가지는 가상 테이블이다.
- 기본 테이블의 기본키를 포함한 속성(열) 집합으로 뷰를 구성해야만 삽입, 삭제, 갱신 연산이 가능하다.
- 뷰를 정의할 때는 CREATE문, 제거할 때는 DROP문을 사용한다.

172



뷰의 장·단점

장점

- 논리적 데이터 독립성이 제공된다.
- 접근 제어를 통한 자동 보안이 제공된다.

단점

- 독립적인 인덱스를 가질 수 없다.
- 뷰의 정의를 변경할 수 없다.
- 삽입, 삭제, 갱신 연산에 제약이 따른다.

173



시스템 카탈로그
(System Catalog)

- 시스템 그 자체에 관련이 있는 다양한 객체에 관한 정보를 포함하는 시스템 데이터베이스이다.
- 사용자가 시스템 카탈로그 내용을 검색할 수는 있지만 갱신할 수는 없다.

174



트랜잭션의 특성

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

175



DDL(데이터 정의어)

- 스키마, 도메인, 테이블, 뷰, 인덱스를 정의하거나 변경 또는 삭제할 때 사용하는 언어이다.
- CREATE : 스키마, 도메인, 테이블, 뷰, 인덱스를 정의함
- ALTER : TABLE에 대한 정의를 변경하는 데 사용함
- DROP : 스키마, 도메인, 테이블, 뷰, 인덱스를 삭제함

정보처리산업기사 핵심 요약

176 DML(데이터 조작어)

- 데이터베이스 사용자가 응용 프로그램이나 질의어를 통하여 저장된 데이터를 실질적으로 처리하는 데 사용되는 언어이다.
- SELECT : 테이블에서 조건에 맞는 튜플을 검색함
- INSERT : 테이블에 새로운 튜플을 삽입함
- DELETE : 테이블에서 조건에 맞는 튜플을 삭제함
- UPDATE : 테이블에서 조건에 맞는 튜플의 내용을 변경함

177 DCL(데이터 제어어)

- 데이터의 보안, 무결성, 회복, 병행 수행 제어 등을 정의하는 데 사용되는 언어이다.
- COMMIT : 명령에 의해 수행된 결과를 실제 물리적 디스크로 저장하고, 데이터베이스 조작 작업이 정상적으로 완료되었음을 관리자에게 알려줌
- ROLLBACK : 데이터베이스 조작 작업이 비정상적으로 종료되었을 때 원래의 상태로 복구함
- GRANT : 데이터베이스 사용자에게 사용 권한을 부여함
- REVOKE : 데이터베이스 사용자의 사용 권한을 취소함

178 CREATE TABLE

- 테이블을 정의하는 명령문이다.
- 표기 형식

```
CREATE TABLE 테이블명
(속성명 데이터_타입 [DEFAULT 기본값] [NOT NULL], ...
[, PRIMARY KEY(기본키_속성명, ...)]
[, UNIQUE(대체키_속성명, ...)]
[, FOREIGN KEY(외래키_속성명, ...)
  [REFERENCES 참조테이블(기본키_속성명, ...)
  [ON DELETE 옵션]
  [ON UPDATE 옵션]
[, CONSTRAINT 제약조건명] [CHECK (조건식)];
```

179 ALTER TABLE

- 테이블에 대한 정의를 변경하는 명령문이다.
- 표기 형식

```
ALTER TABLE 테이블명 ADD 속성명 데이터_타입 [DEFAULT '기본값'];
ALTER TABLE 테이블명 ALTER 속성명 [SET DEFAULT '기본값'];
ALTER TABLE 테이블명 DROP COLUMN 속성명 [CASCADE];
```

180 DROP TABLE

- 기본 테이블을 제거하는 명령문이다.
- 표기 형식

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

- CASCADE : 제거할 요소를 참조하는 다른 모든 개체를 함께 제거함
- RESTRICT : 다른 개체가 제거할 요소를 참조중일 때는 제거를 취소함

181 삽입문(INSERT INTO~)

- 기본 테이블에 새로운 튜플을 삽입할 때 사용한다.
- 표기 형식

```
INSERT INTO 테이블명(속성명1, 속성명2, ...)
VALUES (데이터1, 데이터2, ...);
```

182



삭제문(DELETE FROM~)

- 기본 테이블에 있는 튜플들 중에서 특정 튜플(행)을 삭제할 때 사용한다.
- 표기 형식

```
DELETE
FROM 테이블명
[WHERE 조건];
```

183



갱신문(UPDATE~ SET~)

- 기본 테이블에 있는 튜플들 중에서 특정 튜플의 내용을 변경할 때 사용한다.
- 표기 형식

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

184



데이터 조작문의 네 가지 유형

- SELECT(검색) : SELECT~ FROM~ WHERE~
- INSERT(삽입) : INSERT INTO~ VALUES~
- DELETE(삭제) : DELETE~ FROM~ WHERE~
- UPDATE(변경) : UPDATE~ SET~ WHERE~

185



Select문

```
SELECT [PREDICATE] [테이블명].속성명1, [테이블명].속성명2,...
FROM 테이블명1, 테이블명2,...
[WHERE 조건]
[GROUP BY 속성명1, 속성명2,...]
[HAVING 조건]
[ORDER BY 속성명 [ASC | DESC]];
```

• SELECT절

– Predicate : 불러올 튜플 수를 제한할 명령어

▶ DISTINCT : 중복된 튜플이 있으면 그 중 첫 번째 한 개만 검색

– 속성명 : 검색하여 불러올 속성(열) 및 수식들

• FROM절 : 질의에 의해 검색될 데이터들을 포함하는 테이블명

• WHERE절 : 검색할 조건

• GROUP BY절 : 특정 속성을 기준으로 그룹화하여 검색할 때 그룹화 할 속성

• HAVING절 : 그룹에 대한 조건

• ORDER BY절

– 속성명 : 정렬의 기준이 되는 속성명

– [ASC | DESC] : 정렬 방식(ASC는 오름차순, DESC 또는 생략하면 내림차순)

186



조건 연산자 - LIKE

- 대표 문자를 이용해 지정된 속성의 값이 문자 패턴과 일치하는 튜플을 검색하기 위해 사용된다.
- 대표 문자
 - % : 모든 문자를 대표함
 - _ : 문자 하나를 대표함
 - # : 숫자 하나를 대표함

187



그룹 함수

- GROUP BY절에 지정된 그룹별로 속성의 값을 집계할 때 사용된다.
- COUNT/SUM/AVG/MAX/MIN(속성명) : 그룹별 튜플 수/합계/평균/최대값/최소값을 구하는 함수

188



집합 연산자

2개 이상의 테이블의 데이터를 하나로 통합하는 연산자이다.

189



CROSS JOIN(교차 조인)

- 조인하는 두 테이블에 있는 튜플들의 순서쌍을 결과로 반환된다.
- 교차 조인의 결과로 반환되는 테이블의 행의 수는 두 테이블의 행 수를 곱한 것과 같다.

190



테스트와 디버깅의 목적

테스트(Test)를 통해 오류를 발견한 후 디버깅(Debugging)을 통해 오류가 발생한 소스 코드를 추적하며 수정하는 것이다.

191



프로시저(Procedure)

- 절차형 SQL을 활용하여 특정 기능을 수행하는 일종의 트랜잭션 언어이다.
- 호출을 통해 실행되어 미리 저장해 놓은 SQL 작업을 수행한다.

192



옵티마이저(Optimizer)

- 작성된 SQL이 가장 효율적으로 수행되도록 최적의 경로를 찾아 주는 모듈이다.
- 종류 : RBO(규칙 기반 옵티마이저), CBO(비용 기반 옵티마이저)

193



APM

애플리케이션의 성능 관리를 위해 접속자, 자원 현황, 트랜잭션 수행 내역, 장애 진단 등 다양한 모니터링 기능을 제공하는 도구이다.

