**01 테이블의 특정 행을 입력할 수 있는 DML 구문은?**

​

​

​

**02 트랜잭션 제어 명령어 중 ‘트랜잭션 진행 중 특정 지점으로 되돌리기 위해 설정하는 명령어’는 무엇인가?**

​

​

​

**03 다음 보기 중 DDL 구문을 선택하시오**

|  |
| --- |
| **INSERT CREATE SELECT ALTER DROP DELETE UPDATE** |

​

​

​

**[4~11]**

|  |  |  |  |
| --- | --- | --- | --- |
| **PRIMARY KEY** | **FOREIGN KEY** | **UNIQUE** | **CHECK** |
| **TABLE** | **VIEW** | **INDEX** | **SEQUENCE** |
| **SYNONYM** | **DESCRIBE** | **CASE** | **DECODE** |
| **SYSDAYE** | **SELECT** | **FROM** | **WHERE** |
| **USER\_TABLES** | **DISTINCT** | **GROUP BY** | **HAVING** |
| **NVL** | **NOT NULL** | **SUBSTR** | **ORDER BY** |

​

**04 중복된 행을 자동으로 제거해주는 명령어(=키워드)는 무엇인가?**

​

​

​

**05 테이블 또는 뷰의 구조를 조회할 때 사용하는 명령어는 무엇인가?**

​

​

​

**06 데이터 검색 시 결과를 정렬할 때 사용하는 명령어는 무엇인가?**

​

​

​

**07 물리적인 테이블을 기반으로 생성되며, 데이터 딕셔너리에 SELECT 구문 형태로 저장되는 객체는 무엇인가?**

​

​

​

**08 그룹에 제하는 조건문을 작성할 수 있는 절은 무엇인가?**

​

​

​

**09 테이블에 NULL 값이 들어가지 못하게 막아주는 제약조건은 무엇인가?**

​

​

​

**10 NOT NULL과 UNIQUE의 성격을 모두 가지고 있는 제약조건은 무엇인가?**

​

​

​

**11 자신의 테이블이나 다른 테이블의 특정 컬럼을 참조하는 제약조건은 무엇인가?**

​

​

**12 사원 테이블로부터 부서번호가 50인 사원들의 사원번호, 이름, 급여를 조회하는 '사원뷰50' 뷰(VIEW)를 생성하는 DDL 구문을 작성하시오.**

**​**

**​**

**​**

**​**

**​**

**​**

**13 사원 테이블을 사용하여 사원의 이름과 커미션율 컬럼을 출력하되 커미션을 받지 않는 사원을 제외한 정보만 출력하는 쿼리 구문을 작성하시오. (커미션을 받지 않는 사원의 커미션 컬럼의 값은 NULL이다)**

​

​

​

​

​

​

**14 사원2 테이블을 삭제하는 쿼리구문을 작성하시오.**

​

​

​

​

**15 TITLE 테이블에 아래와 같이 데이터를 삽입하는 SQL(DML) 문장을 작성하시오. (테이블에는 아래 4개의 컬럼만 있다.)**

​

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **테이블명** | **TITLE** | | | |
| **컬럼명** | **TITLE\_ID** | **TITLE** | **RATING** | **RELEASE DATE** |
| **데이터값** | **1** | **ORACLE** | **R** | **05-JUN-17** |

​

​

​

​

​

**16 사원 테이블로부터 이름이 ‘박’으로 시작되는 사원들의 회원번호, 이름, 부서번호를 출력하는 구문을 작성하시오.**

​

​

​

​

​

**17 다음 빈칸에 알맞은 키워드를 넣어 쿼리 구문을 완성하시오.**

|  |
| --- |
| **SELECT 부서번호, AVG(급여) as 평균급여**  **FROM 사원**  **GROUP BY 부서번호**  **( ) AVG(급여) > 6000;** |

​

​

**18 조건에 맞추어 SQL문을 작성하시오. 2020.2차 기출**

|  |
| --- |
| **① 학생 테이블에서 3, 4학년인 학번, 이름을 검색**  **② IN 함수를 사용하여 작성할 것**  **③ 조건에 맞지 않을 경우, 오답으로 처리됨** |

**테이블명: 학생**

|  |  |  |
| --- | --- | --- |
| **학번** | **이름** | **학년** |
| **20171234** | **개구리** | **4** |
| **20183456** | **맹꽁이** | **3** |
| **20195678** | **두꺼비** | **2** |

**​**

​

**19. student 테이블의 name 속성에 idx\_name 이름으로 인덱스로 생성하는 명령어를 작성하시오. 2020.2차 기출**

​

​

​

​

​

**20. 사원 테이블에서 부서번호가 30, 60인 사원의 부서번호와 급여를 부서번호 기준으로 내림차순, 급여 기준으로 오름차순 정렬하여 조회하시오.**

**[출처]** [**(SQL응용 대비) 단답-서술형 20제 (수제비- IT 커뮤니티 (정보처리기사,빅데이터분석기사 등))**](https://cafe.naver.com/soojebi/27264) **| 작성자** [**페코페코**](https://cafe.naver.com/soojebi.cafe?iframe_url=%2FCafeMemberNetworkView.nhn%3Fm%3Dview%26memberid%3Dpucoroc)

1. 리팩토링의 목적은 무엇인가?

리팩토링의 목적은 소프트웨어를 보다 이해하기 쉽고 수정하기 쉽도록 만드는 것이다. 결과의 변경 없이 코드의 구조를 재조정하여 가독성을 높이고 유지보수를 쉽게 하기 위한 목적이다

​

추가답안: 리팩토링은 유지보수 향상, 유연한 시스템, 생산성향상, 품질향상을 위해 사용한다.

​

2. 출력값

#include <stdio.h>

voidmain(){

int i, c=0;

while(i<10){

i++

c \*= i

}

printf("%d", c);

}

답:0

​

3. 대표적인 내부 라우팅 프로토콜로 다익스트라 알고리즘을 이용 대규모 네트워크에 적합한 링크 상태 라우팅 프로토콜로도 불리는 라우팅 프로토콜은 무엇인가?

답: OSPF

​

\*RIP: 기업의 근거리 통신망, 소규모 네트워크

​

4. 심리학자 톰 마릴은 컴퓨터가 메시지를 전달하고, 메시지가 제대로 도착했는지 확인하며, 도착하지 않았을 경우 메시지를 재전송하는 일련의 방법을 '기술적 은어'를 뜻하는 ( ① )라는 용어로 정의하였다.

답 : 프로토콜

​

​

5. 형상통제?

답: 형상 항목의 버전관리를 위해서 변경여부나, 변경활동을 통제하는 활동

추가답안: 솔직히 기억 안날 수 있기 때문에...(양심찔림) 개발업체와 외주업체에 대한 형상 통제 및 지원을 위해 사용한다.

​

​

6. 네트워크 계층에서 TCP/IP에서 IP 패킷을 처리할 때 발생되는 문제를 알려주는 프로토콜.?

답: ICMP

​

7. 분기커버리지

1), 2)

​

편하게 다 YES~ 다.NO로 외우시는 게 깔끔 :))

​

1)

YES-YES

1234561

NO-NO

124567

​

2)

YES-NO

1234561

NO-YES

124561

​

8. sql min max avg group by having 사용문제

select 과목이름, min(점수) as '최소점수', max(점수) as '최대점수' from 성적 group by 과목이름 having avg(점수) >= 90;

​

9. SQL DELETE 튜플 삭제 이름 테이블에서 민수‘를 삭제하시오.

DELETE FROM 학생 WHERE 이름=‘민수’;

​

10. 릴레이션에 A, B가 있을 경우에 릴레이션 B조건에 맞는 것들만 릴레이션 A에서 튜플을 꺼내 프로젝션하는 관계대수는 무엇인가?

÷

​

[추가정보]

합집합(union, ∪)

합병 가능한 두 릴레이션 합집합

​

교집합(intersection, ∩)

합병 가능한 두 릴레이션 교집합.

차집합(difference, -)

합병 가능한 두 릴레이션 차집합.

​

셀렉트(select, σ)

릴레이션에서 주어진 조건을 만족하는 튜플들을 선택하는 연산이다.

​

프로젝트(project, ∏ )

프로젝트는 릴레이션의 애트리뷰트를 연산 대상.

​

조인(join, ⋈)

두 개의 릴레이션으로부터 연관된 투플들을 결합하는 연산자

​

디비전(dibision, ÷)

출제답안

​

개명 연산(rename, ρ)

애트리뷰트 이름을 변경할 때 사용한다.

​

11. 헝가리안 표기법이란?

프로그래밍 언어에서 변수 및 함수의 인자 이름 앞에 데이터 타입을 명시하는 코딩 규칙이다.

추가답안: 위의 것이 기억이 안나신다면! MS도 공식 가이드라인에서 사용하지 말 것을 권고하고 있는 사장된 코딩규칙이다.

​

12. 동치 분할 테스트, 경계값 테스트 등의 종류가 있는 것은?

블랙박스테스트

​

13. DB의 스키마 설명?

스키마는 데이터베이스 구조와 제약조건에 관한 전반적인 명세를 기술한 메타데이터 집합이다.

추가답안: 위에게 기억 안난다면... 스키마는 외부, 내부, 개념 스키마로 구성되어있다. 이것도 기억 안난다면 DB구성요소의 속성 및 관계 구조를 정의한 정보다. 이것도 기억 안난다면 그냥 메타데이터의 집합이다.

​

14. 출력값은?

#include <stdio.h>

intr1(){

return 4; 4

}

intr10(){

return (30+r1( )); 34

}

intr100(){

return (200+r10( )); 234

}

intmain(){

printf("%d\n", r100( )); 출력은 234가 됨 이해?

return 0; return 돌아가야 하잖아요! 4부터 돌아가서!~~

}

답: 234

​

15. 출력값?

i=0

while(i<10)

{

i++

if(i%2==1)

continue;

sum += i; 자바코드

}

​

짝수의 합계를 구하는 문제죠? %에 2 나왔다.그 다음 ==에 0아니면 1 나올 확률이 높죠

​

2 4 6 8 10 <--------------다 더하면 30이 나옵니다.

2 6 12 20 30 (누적)

​

답: 30

​

16. 자바 클래스 상속문제

자바 클래스 상속문제

​

abstract class Vehicle{

String name;

abstract public String getName(String val);

public String getName(){

return "Vehicle name:" + name;

}

}

class Car extends Vehicle{

public Car(String val){

name = super.name = val;

}

public String getName(String val){

return "Car name :" + val;

}

public String getName(byte val[]){

return "Car name :" + val;

}

}

public class Test{

public static void main(String[] args){

Vehicle obj = new Car("Spark");

System.out.printf(obj.getName() );

}

}

​

답: Vehicle name: Spark

​

17. EAI 유형

Point to Point, Hub&Spoke

​

[추가정보]

EAI(정보, 전달, 연계, 통합 등 상호 연동이 가능하게 해주는 솔루션)

포인트포인투: 점대점 연결, 변경 및 재사용 어려움

허브앤스포크: 허브 장애 발생시 시스템 전체에 영향

Message Bus: 애플리케이션 미들웨어를 두어 처리하는 방식.

Hyberid:허브 앤 스포크와 메시지 버스를 혼합방식

​

ESB(애플리케이션 연계, 데이터 변환 지원 등 인터페이스를 제공하는 솔루션, 서비스 중심의 통합)

​

포허메하!!

​

18. UI의 직관성이랑? = 외우는 법 (쉽게 이사)하자

누구나 쉽게 이해하고 쉽게 사용할 수 있어야한다.

​

[추가정보]

직관성(Intuitiveness) : 누구나 쉽게 이해하고 쉽게 사용할 수 있어야한다

일관성(Consistency) : UI의 사용감을 높일 수 있다

효율성(Effectiveness) : 익숙해진 다음에 더 효율적으로 사용할 수 있다면 가장 좋다

유연성(Flexibility) : 예상하지 못한 다양한 인터랙션을 포용할 수 있는 유연성이 중요하다

​

19. C++에서 생성자란?

객체초기화 함수 객체생성시 자동호출되는 메소드

또는 초기화 용도로 주로 사용하는 메소드(이게 더 외우기 쉬움)

​

20. DB에 칼럼 추가

ALTER, ADD

​

예) ALTER TABLE 수제비 ADD (주소(80));

​

[추가정보]

ADD, DROP COLUMN, MODIFY COLUMN, RENAME COLUMN, ADDCONSTRAINT

DROP CONTSTRAINT

[출처] [정보처리기사 2020년 3회 실기 기출 정리해봤습니다! (수제비- IT 커뮤니티 (정보처리기사,빅데이터분석기사 등))](https://cafe.naver.com/soojebi/33462) | 작성자 [취린이폭풍](https://cafe.naver.com/soojebi.cafe?iframe_url=%2FCafeMemberNetworkView.nhn%3Fm%3Dview%26memberid%3Dunike1004)

관계 대수 연산

SELECT 'σ'

PROJECT 'π'

JOIN '⋈'

DIVISION 'R'

​

​

자료사전 표기법

​

= 정의 (is composed of)

+ 구성 (and, along with)

{ } 반복 (iteration of)

[ ] 택일 (choose only one of)

\*\* 주석(comment)

​

​

UML 관계표현 방법

​

n : n개의 객체가 연관됨

0..1 : 연관된 객체가 없거나 1개만 존재

0..\* / \* : 연관된 객체가 없거나 다수 일 수 있음

n..\* : 연관된 객체가 적어도 n개 이상

n..m : 연관된 객체가 n - m개

​

​

UML 구성요소 관계 표현

​

**연관관계** -----

UML구성요소 관계 표현의 하나로써 2개이상의 사물이 서로 관련되어 있음을 표현하며 실선과 화살표로 연결하여 표현하시만 양방향관계의 경우 화살표 없이 실선으로 연결하여 표현하는 관계

​

**포함관계** ◆ㅡㅡ

UML구성요소 관계 표현의 하나로써 집합 관계의 특수한 형태로 포함하는 사물의 변화가 포함되는 사물에게 영향을

미치는 관계 부분(포함되는 쪽)에서 전체(포함하는 쪽)로 속이 채워진 마름모를 연결하여 표현되는 관계

​

**집합 관계** ◇ㅡㅡ

하나의 사물이 다른 사물에 포함되어 있는 관계 부분(포함되는 쪽)에서 전체(포함하는 쪽)로 속이 빈 마름모를 연결하여 표현

​

**의존관계** <-----

사물 사이에 연관은 있으나 필요에 의해서 서로에게 영향을 주는 짧은 시간 동안만 연관을 유지하는 관계 영향을 주는 사물이 영향을 받는 사물 쪽으로 점선 화살표 연결

​

**실체화관계** ◁-----

사물이 할 수 있거나 해야 하는 기능으로 서로를 그룹화할 수 있는 관계 사물에서 기능 쪽으로 속이 빈 점선 화살표 연결

​

**일반화 관계** ◁ㅡㅡ

하나의 사물이 다른 사물에 비해 일반적인지 구체적인지 표현 구체적인 사물에서 일반적인 사물 쪽으로 속이 빈 실선 화살표를 연결

​

​

UML 구조(정적) 다이어그램의 종류

​

**클래스 다이어그램(Class Diagram)** : 클래스가 가지는 속성, 클래스사이 관계표현

​

**객체 다이어그램(Object Diagram)** : 인스턴스를 특정 시점의 객체와 객체사이 관계표현

​

**컴포넌트 다이어그램(Component Diagram)** : 구현 단계에서 사용되며 컴포넌트 간의 관계나 인터페이스를 표현

​

**배치 다이어그램(Deployment Diagram)** : 구현단계에서 사용되며 결과물, 프로세스, 컴포넌트 등 물리적 요소들의 위치 표현

​

**복합체 구조 다이어그램(Composite Structure Diagram)** : 복잡한 구조를 가지는 클래스 혹은 컴포넌트의 내부 구조 표현

​

**패키지 다이어그램(Package Diagram)** : 유스케이스나 클래스 등의 모델 요소들을 그룹화한 패키지들의 관계 표현

​

​

UML 행위(동적) 다이어그램의 종류

​

**유스케이스(Use Case Diagram)** : 사용자의 요구를 분석하여 기능 모델링 작업에 사용됨

​

**시퀀스(Sequence Diagram)** : 상호 작용하는 시스템이나 객체들이 주고받는 메시지 표현

​

**커뮤니케이션(Communication Diagram)** : 객체들이 주고받는 메시지를 표현 할 뿐 아니라 객체들 간의 연관까지 표현

​

**상태(State Machine Diagram)** : 하나의 객체가 자신이 속한 클래스의 상태 변화 혹은 다른 객체와의 상호 작용에 따라 어떻게 변화하는지 표현

​

**활동** : 객체의 처리 로직이나 조건에 따른 처리의 흐름을 순서에 따라 표현

​

**상호작용 개요 (Interaction Overview Diagram)​** : 상호작용 다이어그램간의 제어흐름표현

​

**타이밍 다이어그램(Timing Diagram)** : 객체 상태 변화와 시간 제약을 명시적으로 표현

​

​

논리 모델링 속성

​

**개체(Entity)** : 관리할 대상이 되는 실체

**속성(Attiributes)** : 관리할 정보의 구체적 항목

**관계(Relationship)** : 개체 간의 대응 관계

​

​

개체 - 관계 속성

​

개체 : □ (사각형)

관계 : ◇ (마름모)

속성 : ○ (타원)

관계-속성연결 : - (선)

​

​

이상 현상의 종류

​

**삽입** : 정보 저장 시 해당 정보의 불필요한 세부 정보를 입력해야하는 경우

**삭제** :정보 삭제시 원치 않는 다른 정보가 같이 삭제되는 경우

**갱신** : 중복 데이터 중에서 특정 부분만 수정되어 중복된 값이 모순을 일으키는 경우

​

​

정규화 단계

​

**1정규형 (속성값은 원자값)** 속성값이 1개값이 되도록 구성

​

**2정규형 (부분함수종속제거=완전함수적종속)** A->B,C 일때 B->D 조건일때 부분함수종속에 해당

**3정규형 (이행적함수종속제거)** A->B B->C A->C 일때 이행함수종속에 해당

​

**BCNF정규형(후보키가 여러개인 경우)**

​

**4정규형 (다치종속제거)**

​

**5정규형 (조인종속제거)**

​

​

DDL (데이터 정의어) 의 종류

​

**CREATE** : 데이터베이스, 테이블등을 생성하는 역할

**ALTER** : 테이블을 수정하는 역할

**DROP** : 데이터베이스, 테이블을 삭제하는 역할

**TRUNCATE** : 테이블을 초기화 시키는 역할

​

​

키의 종류

​

**기본키( Primary Key)**

후보키 중 유일성과 최소성을 가지며 튜플을 식별하기 위해 반드시 필요한 키로 중복된 값을 가질 수 없는키

​

**후보키(Candidate Key)**

릴레이션을 구성하는 속성들 중에서 튜플을 유일하게 식별하기 위해 사용되는 속성들의 부분집합의 키

​

**외래키(Foreign Key)**

다른 릴레이션의 기본키를 참조하는 속성 또는 속성들의 집합을 의미하고 릴레이션 간의 관계를 표현

​

**복합키(Composite Key)**

2개 이상의 필드를 조합하여 만든 키

​

**유일키(Unique Key)**

기본키와 마찬가지로 중복성이 허용되지 않지만 NULL 에 대한 허용이 가능한 키

​

​

개발 언어 선정 기준

​

**적정성**

대상 업무의 성격, 즉 개발하고자 하는 시스템이나 응용 프로그램의 목적에 적합해야 한다.

​

**효율성**

프로그래밍의 효율성이 고려되어야 한다.

​

**이식성**

일반적인 PC 및 OS에 개발환경이 설치 가능해야 한다.

​

**친밀성**

프로그래머가 그 언어를 이해하고 사용할 수 있어야 한다.

​

**범용성**

다양한 과거 개발 실적이나 사례가 존재하고, 광범위한 분야에 사용되고 있어야 한다.

​

​

MVC패턴

​

**모델(Model)**

애플리케이션이 무엇을 할 것인지 정의하는 역할

​

**뷰(View)**

화면에 무엇인가 보여주기 위한 역할

​

**컨트롤러(Controller)**

모델이 어떻게 처리할지를 알려주는 역할

​

​

공통 모듈 구현 절차

​

**DTO/VO**

화면에서 전달받은 회원정보로 데이터베이스에 저장하는 객체를 구현

​

**SQL문**

테이블 정의 및 생성 및 검색, 입력 구현

​

**DAO**

SQL을 구현한 XML id를 호출하여 조작을 수행

​

**Service**

회원등록을 위해 joinDAO를 호출 JoinService를 선언 및 구현

​

**Controller**

핵심이 되는 컨트롤러에 회원가입 단위 모듈의 메인 로직을 구현

​

입/출력 검증 로직 구현-회원가입이 성공/실패 시 다른 메시지를 호출하는 검증로직 구현

​

​

UI 설계 원칙

​

**직관성**

누구나 쉽게 이해하고 사용할 수 있어야 한다.

**유효성**

사용자의 목적을 정확하게 달성하여야 한다.

**학습성**

누구나 쉽게 배우고 익힐 수 있어야 한다.

**유연성**

사용자의 요구사항을 최대한 수용하며, 오류를 최소화하여야 한다.

​

​

테스트 오라클의 종류

​

**참(True) 오라클**

모든 테스트 케이스의 입력 값에 발생된 모든 오류를 검출할 수 있음

​

**휴리스틱(Heuristic) 오라클**

특정 입력 값에 대해 올바른 결과를 제공하고, 나머지 값들에 대해서는 추정으로 처리하는 오라클이다.

​

**샘플링(Sampling) 오라클​​**

특정한 몇 개의 입력값에 대해서만 기대하는 결과를 제공해 주는 오라클이다.

​

**일관성 검사(Consistent)**

오라클애플리케이션 변경이 있을 때, 수행 전과 후의 결과 값이 동일한지 확인하는 오라클이다.

​

​

클린코드의 특징

​

**가독성** : 누구든지 읽기 쉽게 코드를 작성한다.

**단순성** : 소스 코드는 복잡하지 않고 간단하게 작성한다.

**의존성** : 다른 모듈에 미치는 영향을 최소화하도록 작성한다.

**중복성** : 중복을 최소화할 수 있는 코드를 작성한다.

**추상화** : 상위 클래스/메소드/함수를 통해 애플리케이션의 특성을 간략하게 나타내고,

상세 내용은 하위 클래스/메소드/함수에서 구현한다.

​

​

집계 함수의 분류

​

COUNT(입력값) 복수 행의 줄(ROW) 수

SUM(입력값) 복수 행의 해당 컬럼(COLUMN) 간의 합계

AVG(입력값) 복수 행의 해당 컬럼 간의 평균

MAX(입력값) 복수 행의 해당 컬럼 중 최댓값

MIN(입력값) 복수 행의 해당 컬럼 중 최솟값

STDDEV(입력값) 복수 행의 해당 컬럼 간의 표준편차

VARIAN(입력값) 복수 행의 해당 컬럼 간의 분산

​

​

DRM의 구성요소

​

**콘텐츠 제공자 (Contents Provider)**

콘텐츠를 제공하는 저작권자

​

**콘텐츠 분배자 (Contents Distributor)**

쇼핑몰 등으로써 암호화된 콘텐츠 제공

​

**패키저(Packager)**

콘텐츠를 메타 데이터와 함께 배포 가능한 단위로 보안 컨테이너 원본을 안전하게 유통 하기위한 전자적 보안 장치

​

**DRM 컨트롤러**

배포된 콘텐츠의 이용 권한을 통제

​

**클리어링 하우스(Clearing House)**

키 관리 및 라이선스 발급 관리

​

​

애자일 방법론의 유형

​

**스크럼(SCRUM)**

매일 정해진 시간 장소에서 짦은 시간의 개발을 하는 팀을 위한 프로젝트 관리 중심 방법론

​

**린(LEAN)**

도요타의 린 시스템 품질 기법을 소프트웨어 개발 프로세스에 적용해서 낭비 요소를 제거하여 품질을 향상시킨 방법론

​

**백로그(BACKLOG)**

제품과 프로젝트에 대한 요구사항

​

**스프린트(SPRINT)**

2~4주의 짦은 개발 기간으로 반복적 수행으로 개발 품질 향상 개념

​

**스크럼 미팅**

매일 15분 정도 미팅으로 TO-DO LIST 계획 수립 개념

​

​

요구사항의 관리 도구

​

**헬릭스 RM(Helix RM)**

요구사항 추적성 매트릭스를 제공하고 요구사항 재사용을 통해 검증 시간과 반복 업무를 줄이는 관리 도구

​

**지라(Jira)**

애플리케이션 생명주기 관리(ALM)와 요구사항 관리에 사용하는 도구

​

**레드마인(redmine)**

웹 기반의 프로젝트 관리와 버그 추적 기능을 제공하는 도구

​

​

감성공학

인간의 감성을 정성적 .정량적으로 측정 및 평가하고 과학적으로 분석하여 이를 구체적인 제품 설계로 실현해 내는 공학

​

감성공학의 종류

​

**1류 접근 방법**

인간의 감성을 표현하는 어휘를 이용하여 제품 디자인 요소와 연계시키는 접근 방법

​

**2류 접근 방법**

개인의 연령 성병 등의 개별적 특성과 생활 방식으로부터 개인이 갖고 있는 이미지를 구체화하는 방법

​

**3류 접근 방법**

기존의 감성적 어휘 대신 공학적인 방법으로 접근하여 인간의 감각을 측정 이를 바탕으로 수학적 모델을 구축하는 방법

​

​

객체 지향 설계 원칙의 종류

​

**단일 책임의 법칙**

하나의 클래스는 하나의 목적을 위해서 생성 / 클래스가 제공하는 모든 서비스는 하나의 책임을 수행하는데 집중되어 있어야하는 법칙

​

**개방 폐쇄 원칙**

소프트웨어의 구성요소는 확장에는 열려 있고 변경에는 닫혀있어야 한다는 법칙

​

**리스코프 치환의 원칙**

자식 클래스는 언제나 자신의 부모 클래스를 대체한다는 원칙

​

**의존성 역전의 원칙**

실제 사용 관계는 바뀌지 않으며 추상을 매개로 메시지를 주고받음으로써 관계를 최대한 느슨하게 만드는 원칙

​

​

미들웨어 솔루션

​

**RPC**

응용 프로그램의 프로시저를 사용하여 원격 프로시저를 로컬 프로시저처럼 호출하는 방식의 미들웨어

​

**MOM**

메시지 기반의 비동기형 메시지 전달 방식 미들웨어

​

**TP-monitor**

온라인 업무에서 트랜잭션을 처리 감시하는 미들웨어

분산 환경 핵심 기술인 분산 트랜잭션을 처리하기 위한 미들웨어

​

**Legacyware**

기존의 애플리케이션이나 DB 기반에 새로운 업데이트된 기능을 덧붙이고자 할 때 사용되는 미들웨어

​

**ORB**

코바(CORBA) 표준 스펙을 구현한 객체 지향 미들웨어

​

​

선형 구조의 종류

​

**스택**

LIFO / 리스트의 한쪽 끝으로만 자료의 삽입 / 삭제가 진행되는 선형 자료 구조

​

**큐**

한쪽에서는 삽입 작업 / 다른 한쪽에서는 삭제 작업이 이루어지도록 구성된 선형 자료 구조

​

**데크**

삽입과 삭제가 리스트의 양쪽 끝에서 모두 발생하도록 구성된 선형 자료 구조

​

**이진트리의 계산법 ( 2N승 - 1 )** (깊이가 N일때 노드의 수) EX) 깊이가 4 이면 -> 16-1 =15

**방향그래프 계산법 N(N-1)**

**무방향 그래프 계산법 N(N-1) / 2**

​

​

알고리즘 기법

​

**분할과 정복**

문제를 나눌 수 없을 때까지 나누고 각각을 풀면서 다시 병합하여 문제의 답을 얻는 알고리즘

​

**동적 계획법**

어떤 문제를 풀기 위해 그 문제를 더 작은 문제의 연장선으로 생각하고 과거에 구한 해를 활용하는 방식의 알고리즘

​

**탐욕법**

결정을 해야할 때마다 그 순간에 가장 좋다고 생각 되는 것을 해답으로 선택함으로써 최종적인 해답에 도달하는 방식의 알고리즘

​

**백트래킹**

어떤 노드의 유망성 점검 후 유망하지 않으면 그 노드의 부모 노드로 되돌아간 후 다른 자손 노드를 검색하는 알고리즘

​

​

개발도구의 유형

​

**빌드 도구**

작성한 코드의 빌드 및 배포를 수행하는 도구

​

**구현 도구**

코드의 작성과 디버깅 수정 등과 같은 작업 시 사용되는 도구

​

**테스트 도구**

코드의 기능 검증과 전체의 품질을 높이기 위해 사용하는 도구

​

**형상 관리 도구**

개발자들이 작성한 코드와 리소스 등 산출물에 대한 버전 관리를 위한 도구

​

​

프레임워크의 특징

​

**모듈화**

프레임워크는 구현을 인터페이스 뒤에 감춰 모듈화를 강화

​

**재사용성**

인터페이스를 통해 여러 애플리케이션에서 반복적으로 사용하는 일반적 컴포넌트를 정의하여 재사용성을 높힌다

​

**확장성**

다형성을 통해 애플리케이션이 프레임워크의 인터페이스를 확장

​

**제어의 역행**

프레임워크 코드가 전체 애플리케이션의 처리 흐름을 제어

​

​

단위 모듈 구현 원리

​

**정보 은닉**

어렵거나 변경 가능성이 있는 모듈을 타 모듈로부터 은폐

​

**분할과 정복**

복합한 문제를 분해 / 모듈 단위로 문제 해결

​

**데이터 추상화**

각 모듈 자료 구조를 액서스하고 수정하는 함수 내에 자료 구조의 표현내역을 은폐

​

**모듈 독립성**

낮은 결합도와 높은 응집도를 가진 특성

​

​

리눅스/유닉스 운영체제의 환경 변수

​

**$PATH**

콜론으로 구분된 디렉터리 목록을 포함하여 쉘은 이름에 슬래시 / 가 포함되지 않는 명령어를 검색

​

**$HOME**

사용자의 홈 디렉터리의 위치를 포함

​

**$PWD**

현재 디렉터리의 절대 경로명

​

**$SHELL**

로그엔 쉘의 절대 경로명

​

​

전송기술 방식 종류

​

**멀티캐스트**

하나 이상의 송신자들이 특정한 하나 이상의 수신자들에게 데이터를 전송하는 방식

​

**유니캐스트**

고유 주소로 식별된 하나의 네트워크 목적지에 1:1로 트래픽 또는 메시지를 전송하는 전송 기술

​

**브로드캐스트**

하나의 송신자가 같은 서브네트워크상의 모든 수신자에게 데이터를 전송하는 전송 기술

​

**애니 캐스트**

단일 송신자로부터의 데이터그램들을 토폴로지상의 잠재적인 수신자 그룹 안에서 가장 가까운 노드로 연결시키는 전송 기술

​

​

소프트웨어 개발방법론의 종류

​

**구조적 방법론**

기능에 따라 나누어 개발, 분할과 정복 방식의 접근 프로세스 중심의 하향식 방법론

​

**정보공학 방법론**

관리 절차와 작업 기법을 체계화. 개발 주기로 대형 프로젝트를 수행하는 방법론

​

**객체지향 방법론**

데이터 관련 동작을 모두 포함하는 방법론

​

**컴포넌트 기반 방법론**

컴포넌트를 조립해서 하나의 새로운 응용 프로그램 작성하는 방법론 / 컴포넌트는 원하는 DB와 SW의 개발된 모듈 단위

​

**애자일 방법론**

사람이 중심이 되고 변화에 유연하고 신속하게 적응하는 신속 적응적 경량 개발 방법론

​

**제품 계열 방법론**

특정 제품에 적용하고 싶은 공통된 기능을 정의하여 개발하는 방법론 / 임베디드의 적용

​

​

네트워크 설치 구조

​

**버스형**

하나의 네트워크 회선에 여러 대의 노드가 멀티 포인트로 연결되어 있는 구조 형태

​

**트리형**

각 노드가 계층적으로 연결 , 나뭇가지가 사방으로 뻗은 모양과 유사하고 허브만 있으면 쉽게 연결 가능한 구조 형태

​

**링형**

모든 노드가 하나의 링에 순차적으로 연결되어 있는 구조 형태

​

**성형**

허브에 점대점 연결 , 소규모 네트워크에 간편하고 중앙 허브 고장 시 전체 네트워크에 영향이 있는 구조 형태

​

​

네트워크 구축 모델

​

**코어 계층**

각 네트워크 사이에 높은 속도의 스위칭을 목적으로 하여 각 분배 계층을 연결하는 계층

​

**분배 계층**

액서스 계층의 트래픽을 처리하기 위한 정책을 결정하는 계층 / 코어 계층 - 액서스 계층 사이의 통신을 담당

​

**액서스 계층**

말단 계층의 장비 쪽에 위치하여 트래픽이 발생하는 계층 / 네트워크 자원에 대한 사용자 접근을 제어하는 기능

​

​

현실 종류

​

**가상 현실 (VR)**

인공적인 기술로 만들어낸 실제와 유사하지만 실제가 아닌 어떤 특정한 환경이나 상황 혹은 그 기술

​

**증강 현실(AR)**

가상 현실의 한 분야로 실제로 존재하는 환경에 가상의 사물이나 정보를 합성하여 마치 원래의 환경에 존재하는 사물처럼 보이도록 하는 컴퓨터 그래픽 기술

​

**혼합 현실(MR)**

실세계의 물리적 환경과 가상환경을 혼합한 경험을 제공하는 하이브리드 현실

실시간으로 현실과 가상에 존재하는 것 사이에서 실시간으로 상호작용 할 수 잇는 것을 말할 때 개념을 사용

​

**확장 현실(XR)**

현실과 가상 간의 인터렉션이 더욱 강화되고 현실 공간에 배치된 가상 물체를 물리적으로 만드는 기술

​

​

블록체인 2.0 알고리즘

​

**PoW(Proof of Work)**

어려운 문제를 빨리 해결한 사람에게 블록을 만들수 있도록 허가하는 합의 알고리즘

​

**PoS(Proof of stake)**

화폐량을 더 많이 소유하고 있는 승인자가 우선적으로 블록을 생성할 수 있는 알고리즘

​

**PBFT**

참가자 중 1명이 리더가 되어 모든 참가자에게 요청을 보내고 요청 결과를 집계한 뒤 다수의 값을 사용해 블록을 확정하는 합의 알고리즘

​

​

저장장치

​

**DAS**

하드 디스크 같은 데이터 저장 장치를 호스트 버스 어뎁터에 직접 연결하는 방식

저장 장치와 호스트 기기 사이에는 네트워크 디바이스가 있지 말아야 하고 직접 연결하는 방식으로 구성

​

**NAS**

서버와 저장 장치를 네트워크로 연결하는 방식

저장 장치와 서버를 직접 연결하는 것이 아니라 네트워크를 통해 스토리지에 접속하고 파일 단위로 관리

​

**SAN**

서버와 스토리지를 저장 장치 전용 네트워크로 상호 구성하여 고가용성,고성능, 융통성,확장성을 보장

데이터를 블록 단위로 관리하는 기술

DAS의 빠른 처리와 NAS의 스토리지 공유 방식을 합친 방식

​

​

데이터베이스 회복

데이터베이스 운영 도중 예기치 못한 장애가 발생할 경우 데이터베이스를 장애 발생 이전의 일관된 상태로 복원 시키는 작업

​

데이터베이스 회복의 종류

​

**체크포인트 회복기법**

체크포인트 이전에 처리된 트랜잭션은 회복작업에서 제외하고 체크포인트 이후에 처리된 내용에 대해서만 회복작업을

수행

​

**그림자페이지 회복기법**

로그를 이용하지 않고 DB에 대한 페이지 테이블 전체를 교체하는 방법

​

​

데이터베이스 회복을 위한 조치

​

**REDO**

데이터베이스 내용 손상 시 가장 최근 복제 본을 적재 후 변경된 로그를 이용하여 재실행해 데이터베이스를 복원하는 방법

​

**UNDO**

데이터베이스 내용 자체는 손상되지 않았지만 변경 중이거나 변경된 내용에 대한 신뢰성을 잃어버린 경우

모든 변경 내용을 취소하여 복원하는 방법

​

​

서버 접근 통제의 과정

​

**임의적 접근통제(DAC)**

시스템에 대한 접근을 사용자/그룹의 신분 기반으로 제한하는 방법

​

**강제적 접근통제(MAC)**

시스템 정보의 허용등급을 기준으로 사용자가 갖는 접근 허가 권한에 근거하여 시스템에 대한 접근을 제한하는 방법

​

**역할기반 접근통제(RBAC)**

중앙 관리자가 사용자와 시스템의 상호관계를 통제하며 조직 내 맡은 역할에 기초하여 자원에 대한 접근을 제한하는 방법

1. **데이터웨어하우스**- 업무시스템에서 모아진 정보를 일관된 스키마로 저장한 저장소

2. **사용성 테스트**- 사용자가 직접 제품을 사용하면서, 사전에 정의된 시나리오에 맞춰 과제를 수행한 후 질문에 답하도록 하는 테스트 기법

3. **빅데이터**- 기존에 처리 가능한 데이터 범위를 넘어서는 수십 페타바이트 크기의 비정형 데이터

4. **UI**- 사용자와 시스템 사이에 의사소통할 수 있도록 고안된 물리적 가상의 매개체

5. **정규화**- 관계형 데이터 베이스의 설계에서, 중복을 최소화하여 데이터를 구조화하여 이상현상이 생기지 않도록 하는 데이터 모델링 기법

6. **인스펙션** - 원시코드 등을 저작자 이외의 전문가나 팀이 검사하여 오류를 찾아내는 공식적인 검토 방법

7. **워크스루** - 검토 자료를 회의 전에 배포한 후, 사전 검토 후 짧은 회의를 통해 오류를 발견하는 비공식적인 검토 방법

8. **동료검토** - 2~3명이 진행하며, 요구사항 명세서 작성자가 명세서를 설명하고 이해관계자들이 이를 들으면서 오류를 발견하는 기법

9. **WSDL** - 웹서비스명, 프로토콜 정보 등 웹서비스에 관한 상세정보를 XML 형식으로 기술한 언어 및 파일

10. **프로토타입** - 정적인 와이어프레임과 스토리보드에 동적 효과를 적용해 실제 구현한 것처럼 시뮬레이션 할 수 있는 모형

11. **와이어프레임** - 화면 단위의 레이아웃을 설계하는 작업

12. **스토리보드** - 와이어프레임과 같은 서비스 구축을 위한 모든 정보를 담은 문서

13. **BCNF** - 3차 정규화를 만족하면서, 모든 결정자가 후보키가 되도록 하여 결정자 함수 종속을 제거하는 것

14. **인덱스** - 검색 연산의 최적화를 위해, 데이터베이스 내 열에 대한 정보를 구성한 데이터 구조

15. **반정규화** - 시스템의 성능 향상과 운영의 단순화를 위해, 테이블의 중복, 통합, 분리 등을 수행하는 데이터 모델링 기법으로 역정규화 및 비정규화라고도 부름

16. **DBMS\_OUTPUT 패키지** - 메시지를 버퍼에 저장하고, 버퍼로부터 메시지를 읽어오기 위한 인터페이스를 제공하는 패키지

17. **옵티마이저** - SQL을 빠르고 효율적으로 수행할 최적의 경로를 생성하는 DBMS 내부의 핵심 엔진

18. **JDBC -** 자바에서 데이터베이스를 사용할 수 있도록 연결해주는 응용프로그램 인터페이스

19. **ODBC** - 모든 언어에서 사용할 수 있는 DB 접속 기술

20. **DBMS** - 데이터베이스 내의 데이터를 접근할 수 있도록 해주는 시스템

21. **E-R 모델** - 실제 세계에 존재하는 데이터와 그들간의 관계를 사람이 이해할 수 있는 형태로 표현한 모델

22. **프로시저** - SQL쿼리의 집합으로, 절차형 SQL을 통해 특정 기능을 수행하는 트랜잭션 언어

23. **트리거** - 특정 테이블에 데이터 변경 이벤트가 발생했을 때, DBMS에서 자동적으로 실행되도록 구현된 프로그램

24. **XML** - HTML, SGML 의 단점을 보완한 특수 목적용 마크업 언어

25. **데이터마이닝** - 대규모로 저장한 데이터 안에서 체계적으로 통계적 규칙이나 데이터 간의 관계 및 추세를 발견하는 기술

26. **군집화** - 유사한 특성을 갖는 데이터들을 집단화하는 방법

27. **SAN** - 광케이블을 활용해 근거리 네트워크 환경을 구축하여, 빠른 속도로 데이터를 전송할 수 있는 특수 목적용 고속 네트워크

28. **IPsec** - 3계층(네트워크 계층)에서 AH와 ESP를 통해 기밀성, 무결성, 인증의 기능을 수행하는 보안 프로토콜

29. **ESB** - 기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션들을 하나로 연계해 서비스 중심의 통합을 지향하는 기술

30. **EAI** - 기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션들 간의 정보 전달 및 연계를 가능하게 해주는 솔루션

31. **형상관리** - 소프트웨어 개발의 전체 과정에서 발생하는 모든 변경사항들을 관리하는 활동

32. **Mybatis** - 자바의 관계형 DB 프로그래밍을 조금 더 쉽게할 수 있도록 도와주는 프레임워크

33. **Tear Drop** - IP패킷의 재조합 과정에서, 잘못된 Fragment offset 정보로 인해 타겟시스템에서 문제를 발생하도록 만드는 공격 기법

34. **메시지 버스(Message Bus)** - 애플리케이션 사이에 미들웨어(버스)를 두어 연계하는 미들웨어 통합 방식

35. **AJAX** - 자바스크립트를 사용한 비동기 통신 기술로, 서버와 클라이언트 간 XML 데이터를 주고받는 기술

36. **스니핑** - 공격 대상에게 직접적으로 공격을 가하지 않고, 네트워크 중간에서 남의 패킷을 도청하는 수동적 공격 기법

37. **UX** - UI를 포함한 개념으로, 사용자가 직/간접적으로 경험하고 느끼는 총체적 경험

38. **살충제 패러독스** - 동일한 테스트 케이스로 반복적으로 테스트를 수행하면, 더이상 결함을 발견할 수 없다는 테스트 원리

39. **오류-부재의 궤변** - 요구사항을 만족하지 못한다면, 결함이 없다고 해도 품질이 높다고 볼 수 없다는 테스트의 원리

40. **트랜잭션** - 하나의 논리적인 기능을 수행하기 위한 작업의 기본 단위로, 한꺼번에 수행되는 일련의 연산

41. **테일러링** - 소프트웨어의 기법, 프로세스 등을 기술적 요구에 맞춰 최적화 하는 기법

42. **CSS(XSS) 공격** - 공격자가 웹 사이트에 악의적인 스크립트를 포함해서 사용자가 열어보도록 유도한 후, 사용자 정보를 탈취하는 공격 기법

43. **CSRF 공격** - 공격자가 웹 서버의 취약점을 이용해 악성 스크립트를 삽입하고, 정상적인 사용자로 하여금 자신의 의지와 무관하게 공격자가 의도한 행위를 특정 웹사이트에 요청하게 하는 공격 기법

44. **회귀 테스트** - 오류를 수정하거나 제거한 시스템에서 오류 수정 및 제거 이후 새로 유입된 오류가 없는지 확인하는 반복테스트 기법

45. **병행 테스트** - 변경된 시스템과 기존의 시스템에 동일한 데이터를 입력한 후 비교하는 테스트 기법

46. **클린 코드** - 잘 작성되어 가독성이 높고, 의존성을 줄이고 중복성을 최소화한 코드

47. **외계인 코드 -** 베드코드의 한 유형으로, 아주 오래되거나 참고 문서가 없어 유지보수가 어려운 코드

48. **테스트 케이스** - 특정 요구사항에 준수하는 지 확인하기 위해 만들어진 입력 값, 실행 조건, 예상된 결과의 집합

49. **테스트 시나리오** - 테스트 케이스들의 집합으로, 테스트 절차를 명세한 문서

50. **목 오브젝트** - 사용자의 행위를 사전에 입력해두면, 그 상황에 예정된 행위를 수행하는 객체

51. **결함 에이징 분석** - 등록된 결함에 대해, 특정한 결함 상태의 지속 시간을 측정하여 분석하는 것

52. **파티셔닝** - 대용량 데이터베이스를 파티션이라는 보다 작은 단위로 분할한 것

53. **가상화** - 물리적인 리소스들을 사용자에게 하나로 보이게 하거나, 하나의 물리적 리소스를 여러개로 보이게 하는 기술

54. **테스트 오라클** - 테스트 결과의 참/거짓을 판단하기 위해, 사전에 정의된 참 값을 대입해 비교하는 기법

55. **동등분할테스트** - 유사한 도메인별로 무효/유효값을 그룹핑하여, 대표 값을 테스트 케이스로 도출하여 테스트하는 기법

56. **경계 값 테스트** - 경계 값을 포함하여 테스트 케이스를 설계

57. **솔트(Salt)** - 일방향 해시 함수에서, 다이제스트를 생성할 때 추가되는 임의의 문자열

58. **SQL 삽입(Injection)** - 공격자가 SQL에 악의적인 구문을 삽입해서, DB로 부터 정보를 열람 및 조작할 수 있는 공격기법

58. **구조체** - 사용자가 기본 타입을 가지고, 새롭게 정의할 수 있는 자료형

59. **열거체** - 서로 연관된 정수형 상수들의 집합

60. **공용체** - 모든 멤버 변수가 하나의 메모리 공간을 공유하는 자료형

60. **NoSQL** - 전통적인 RDBMS와 다르게 고정적인 테이블 스키마가 필요하지 않으며, 수평적으로 확장이 가능한 DBMS

61. **시큐어코딩** - 설계 및 구현 단계에서 해킹 등의 보안 취약점을 사전에 제거하고, 외부로부터 안전한 소프트웨어를 개발하는 것

62. **VPN** - 공중망에서 터널링 기법을 사용해 사설망과 같은 서비스를 제공하는 가상 네트워크

63. **REST** - HTTP URI 를 통해 자원을 명시하고, HTTP 메소드를 통해 해당 자원에 대한 명령을 적용할 수 있는 기술

64. **MVC 패턴** - 모델, 뷰, 컨트롤러로 구성되어 비즈니스 로직을 서로 영향없이 쉽게 고칠 수 있는 패턴

65. **델파이 기법** - 전문가의 경험적 지식을 통한 미래 예측 및 문제 해결 기법

66. **SSO** - 한번의 시스템 인증을 통해, 여러 정보 시스템에 재인증절차 없이 접근이 가능한 통합 로그인 기술

67. **스머핑/스머프 공격** - 출발지 주소를 공격대상의 IP로 설정해서, 네트워크 전체에 ICMP Echo를 직접 브로드캐스팅하여 타겟 시스템을 마비시키는 공격 기법

68. **알파테스트** - 개발자의 환경에서 통제된 상태로, 선택된 사용자가 개발자와 함께 수행하는 인수테스트

69. **베타테스트** - 실제 환경에서, 일정 수의 사용자에게 직접 해당 소프트웨어를 사용하게 하여 피드백을 받는 인수테스트

70. **CRUD Matrix** - 관계의존성을 Create, Read, Update, Delete로 표현한 매트릭스

71. 트랜잭션의 **일관성(consistency)** - 트랜잭션이 성공적으로 완료되면, 언제나 일관성 있는 DB상태로 변환

72. 트랜잭션의 **원자성(Atomicity)** - 트랜잭션의 연산은 모두 성공하거나, 모두 실패해야한다

73. 트랜잭션의 **격리성(Isolation)** - 트랜잭션이 실행 중일 때, 중간 결과를 다른 트랜잭션이 접근할 수 없다

74. 트랜잭션의 **영속성(Durability)** - 트랜잭션이 성공적으로 완료되면, 시스템에 오류가 발생하더라도 변경된 내용은 계속 보존

75. **이상현상의 발생 원인과 종류** - 원인: 테이블에서 일부 속성의 종속으로 인해 테이블 조작 시 중복이 발생하기 때문에 발생함

종류: 삽입, 삭제, 갱신이상

76. **파밍** - 사용자들이 진짜 사이트로 오인하도록 유도하여 개인정보를 훔치는 신종 인터넷 사기 수법

77. **병행제어** - 하나의 트랜잭션이 연산 중, 다른 트랜잭션의 영향을 받지 않도록 제어하는 것

78. **로킹기법** - 하나의 트랜잭션이 데이터 항목에 액세스중일 때, 다른 트랜잭션이 해당 데이터 항목에 접근하지 못하도록 제어하는 기법

79. **소프트웨어 아키텍처** - 소프트웨어의 구성요소와 그들간의 관계를 표현한 시스템 구조

80. **WAS** - 웹에서 애플리케이션이 구동될 수 있는 환경을 제공하는 서버

81. **멀티캐스트 프로토콜** - 인터넷에서 같은 내용의 데이터를 여러명의 특정 그룹의 수신자에게 동시에 전송할 수 있는 프로토콜

82. **UML** - 객체지향 소프트웨어 개발 과정에서, 산출물을 명세화/문서화하기 위한 모델링 언어

83. **링크- 상태 알고리즘** - 다익스트라 알고리즘을 이용해, 상태 정보를 모든 라우터에 전달하여 최단 경로 트리를 구성하는 알고리즘

84. **거리-벡터 알고리즘** - 벨만-포드 알고리즘을 이용해, 인접 라우터와 정보를 공유해 목적지까지의 거리와 방향을 결정하는 알고리즘

85. **은행가 알고리즘** - 교착상태의 회피기법으로, 사전에 필요한 자원 수를 명시하고 안정적인 상태에만 자원을 할당하는 기법

86. **스레싱** - 계속적으로 페이지 부재가 발생하여, 프로세스의 실제 처리 시간보다 페이지 교체 시간이 더 많아지는 현상

87. **맵리듀스** - 구글에서 대용량 데이터 처리를 분산 병렬 컴퓨팅에서 처리하기 위한 목적으로 만든 프레임워크

88. **API** - 응용프로그램에서 사용할 수 있도록, 기능을 제어할 수 있게 만든 인터페이스

89. **비트락커** - 윈도즈 OS에 포함되어, 볼륨 전체의 암호화를 통한 완전한 디스크 암호화 기능을 제공

90. **UI의 설계 원칙 -직관성** (쉽게 이사) - 사용자가 쉽게 이용하고 사용할 수 있다

**- 유효성(정사목달)** - 정확하게 사용자의 목표 달성

**- 학습성** - 누구나 쉽게 배우고 사용

**- 유연성(사인최포실방)** - 사용자의 인터랙션을 최대한 포용하고, 실수를 방지

91. **미들웨어** - 컴퓨터와 컴퓨터간 연결을 쉽고 안전하게 도와주고, 이에 대한 관리를 도와주는 소프트웨어

92. **DRM** - 디지털 콘텐츠의 저작권을 보호하고, 권한을 허용하는 기술

93. 테스트 **드라이버** - 상향식 통합 테스트에서, 하위 모듈을 호출하는 상위의 더미 모듈

94. **tPmc** - 하드웨어 성능 지표로 사용되며, 1분당 최대 처리건수를 나타낸 수치

95. **시멘틱웹** - 온톨로지를 사용하여, 컴퓨터가 이해할 수 있는 형태로 제작된 지능형 웹

95. **온톨로지** - 실세계에 존재하는 모든 개념들과 그들간의 관계 정보를 컴퓨터가 이해할 수 있도록 서술한 것

96. **랜섬웨어** - 사용자 컴퓨터에 몰래 침입해 문서를 암호화하고, 사용자가 열지 못하도록 하여 암호 해독용 프로그램의 전달을 조건으로 사용자에게 돈을 요구하는 공격

97. **제어구조 테스트** (제그내) - 제어구조를 그래프 형태로 나타내어, 내부 로직을 테스트하는 기법

98. **JSON** - AJAX를 위해, '속성-값', '키-값' 쌍으로 이루어진 데이터를 전달하기 위한 개방형 표준 포맷

99. **프로토콜** - 데이터 교환을 원활하게 하기 위한 표준화된 통신 규약

100. **LOC 기법** - 각 기능의 원시코드 라인 수의 낙관치, 비관치, 중간치를 측정하여 예측치를 구하고, 이를 이용해 비용을 산정하는 비용산정 모델

[출처] [정보처리기사 약술형/서술형 대비 100제 (수제비- IT 커뮤니티 (정보처리기사,빅데이터분석기사 등))](https://cafe.naver.com/soojebi/35556) | 작성자 [수제비와 취뽀](https://cafe.naver.com/soojebi.cafe?iframe_url=%2FCafeMemberNetworkView.nhn%3Fm%3Dview%26memberid%3Dksp4064)