C 언어

포인터, 구조체,

```
switch (x) {
    case1:
        printf ("1")
    case2 :
        printf ("2")
        break
    default:
        printf ("default")
}
```

이거 없으면 → 여기부터 실행됨

포인터

(%c, *p) : 문자출력

(%s, p) : 문자열출력 (시작주소)

계산 우선순위 ++>*

p++	주소뒤, 값	++*p	주소, 값+1
(*p)++	주소, 값뒤	++(*p)	주소, 값+1
*p++	다음주소, 현재값	*++p	주소뒤, 값

구조체

```
struct 이름 {
    char name[10]
    struct insa *s
}
```

1. 기본 구조체 p1.name
2. 구조체가 포인터 s->name 아니면 (*s).name
3. 구조체 내부의값이포인터 *(s.name)

Java

```
class parent {
    name() }

class child extends parent {
    name() }

public class test {
    public static void main (String[] args) {
        parent a = new child()
    }
}
```

1. child로 a가 parent클래스에서 재정의 x 부모의 name함수는 super.name() 으로가능
3. static은 main 실행전에 선언됨

Python

range

range(최종값) 0 ~ 최종-1

range(초기값, 최종값) 초기 ~ 최종-1

range(초기값, 최종값, 증감값) 초기 초기증가 최종-1

slice

객체[초기위치:최종위치] 초기 ~ 최종-1

객체[초기위치:최종위치:증감값] 초기 초기증가 최종-1

객체[:] 아 객체[::] 모든값

객체[초기위치:] 초기 ~ 마지막

객체[:최종위치] 0 ~ 최종위치-1

객체[::증가값] 0 ~ 마지막까지 증감값으로 증가

SQL

1. 도메인 생성

create Domain 이름
default 값
Constraint 제약조건이름 check(범위)

2. 테이블 생성

create table 테이블명
(속성명 변수형(최대수) NULL 여부,
primary key (속성명)
foreign key (속성명) References 다른테이블명(속성명)
)

3. 인덱스 생성

create[unique] Index 인덱스명
on 테이블명(속성 정렬)
CLUSTER

where 서브쿼리 조건 연산자

IN	- 여러값중하나일때
NOT IN	- 여러값모두 일때
EXIST	- 서브쿼리가 결과반환 O
NOT EXIST	- 〃 결과반환 X
ALL	- 모든값 조건을 만족
ANY/SOME	- 하나라도 만족하면

Commit
Save point 이름
RollBack to 이름
Roll Back

4. alter

ALTER Table 테이블명 ADD 속성명 타입[기본값]
ALTER Table 테이블명 ALTER 속성명 변화할사항
ALTER Table 테이블명 DROP column 속성명[cascade]

참조되는 개체도 삭제

5. DROP

DROP 원하는종류 이름[cascade|restrict]

Grant, Rebake

등급
GRANT 등급 To 부여된사람
REVoke 등급 From 제한된사람

등급
- DBA: 데이관리자
- RESource: 테이블,데이생성가능
- Connect: 단순 사용자

테이블및속성 권한
GRANT 권한 ON 개체 To 사용자[with GrANT option]
REVoke 권한 ON 개체 From 사용자[cascade]

권한
- All
- Select
- Insert
- Delete
- Update

부여받은 권한을 부여할지

해당 사용자에게 받은권한 다른 사용자의 권한을 취소

Insert

```
Insert Into 테이블명(속성)
Values (값);
```

Delete - 레코드 삭제

```
Delete
From 테이블명
[where 조건]
```

Update

```
UPDATE
SET 속성명 = 값, 속성2 = 값
[where 조건]
```

합집합

- Union - 중복 X
- Union ALL - 중복 O
- (결과 동합)
- Intersect - 공통된 항
- Except - 1번에서 2번을 제외 (select) (select)

Join

inner join

조건없을때 = cross join과 같음

equi

1. where table1.속성1 = table2.속성
2. From table1 join table2 using 속성명

non-equi

= 이거의 조건

Outer join

- Left Outer Join - 우측에 릴레이션과 안맞으면 좌측릴레이션에 NULL로 추가 (좌측은 모든데이터)
- Right Outer Join - 좌측에 릴레이션과 안맞으면 우측릴레이션에 NULL로 추가 (우측은 모든데이터)
- Full Outer Join - 모든데이터 추출

정규화

비정규 릴레이션 → 1NF (도메인의 원자값) → 2NF (부분함수적 종속 제거) → 3NF (이행함수적 종속제거)

* 함수적 종속 A의값을 정하면 B의 값도 유일하게 정해짐 ex) 학번 → 이름
* 부분함수적 종속 복합키의 일부만으로 값이 정해짐 ex) (학번, 과목) → 이름
* 이행 함수적 종속 A → B → C 처럼 중간키를 거쳐서 종속 ex) 학과 → 학번 코드 → 학과이름

학번	과목명	교수이름	학과코드	학과명
1001	DB	김교수	CS	컴퓨터공학
1001	C언어	김교수	CS	컴퓨터공학
1002	DB	김교수	CS	컴퓨터공학

1. 셀마다 하나의 값이다 1NF 만족
2. 학번, 과목명 → 학과코드 결정

즉, 하나의 키로 종속되게 연결

학번	학과코드	학과명
1001	CS	컴퓨터공학
1002	CS	컴퓨터공학

학번	과목명	교수이름
1001	DB	김교수
1001	C언어	김교수
1002	DB	김교수

제 2 정규화 만족

학과코드	학과명
CS	컴퓨터공학

학번 → 학과코드 → 학과명

학번	학과코드
1001	CS
1002	CS

다음과 같이 학번으로 두개의 컬럼값이 종속

제 3 정규화 만족

결합도, 객체지향 설계원칙

객체지향 분석 방법론 : 럼바우

분석활동 : 객체 모델링 → 동적 모델링 → 기능 모델링

- 객체 모델링 : 객체의 속성, 연산의 정의, 관계를 규정
- 동적 모델링 : 사건에 초점을 맞추어 객체의 제어, 상호작용, 작동순서
- 기능 모델링 : 자료흐름(DFD)를 활용한 프로세스 간의 자료흐름 정의

객체지향 설계원칙

1. Single Responsibility Principle (단일 책임 원칙)
   객체는 단 하나의 책임만 가져야 한다.
2. Open Close Principle (개방-폐쇄 원칙)
   기존 코드를 변경하지 않고 기능을 추가 할 수 있어야한다.
3. Liskov Substitution Principle (리스코프 치환 원칙)
   그식클래스는 목모 클래스의 기능을 수행할 수 있어야한다
4. Interface Seperate Principle (인터페이스 분리 원칙)
   그선이 사용하지 않는 인터페이스와 의존 관계를 맺거나 영향을 받지 않아야 한다는 원칙
5. Dependency Inversion Principle (의존 성 역전 원칙)
   의존 관계 성립시 추상성이 높은 클래스와 의존관계를 맺어야 한다는 원칙

결합도

독립성↑

- 자료 결합도 : 모듈인터페이스가 자료 오소로만 구성
- 스탬프 결합도 : 모듈간 배열, 레코드등이 전달
- 제어 결합도 : 논리적 흐름을 제어하기 위해 선호나 제어요소를 전달
- 외부 결합도 : 어떤 모듈에 선언된 데이터를 외부에서참조
- 공유 결합도 : 공통데이터 영역에서 여러모듈이 사용
- 내부 결합도 : 한모듈이 다른 모듈의 기능이나 고로를 직접참조, 수정

독립성↓

응집도

독립성↑

- 기능적응집도 : 오 듈 내부의 모든 기능 오소들이 단일 문제와 연관되어 수행
- 순차적응집도 : 하나의 활동으로 부터 나온 출력 데이터를 그 다음 활동의 입력데이터로 사용
- 교환적응집도 : 동일 입출력으로 다른 기능 수행
- 절차적응집도 : 모듈이 관련 기능을 가질때 모듈내부에서 순 차적으로 수행될 경우
- 시간적응집도 : 특정시간에 처리되는 몇개의 기능을 모아 하나의 모듈로 구성
- 논리적응집도 : 유사한 성격을 갖거나 특정 카테고 묶음되는 처리 오소들로 하나의 모듈 형성
- 우연적응집도 : 모듈 내부의 각각의 오소들이 서로 관련 없는 오소로만 구성된 경우

독립성↓

디자인 패턴

생성 패턴 : 객체생성 및 참조

1. Abstract Factory : 인터페이스를 통해 서로 연관,의존 하는 객체들의 그룹으로 추상적표현
2. Builder : 인터페이스를 점층적또 조합하여 객체생성
3. Factory Method : 상위 클래스에서 인터페이스만 정의, 실제 생성은 서브클래스에서 담당
4. Prototype : 원복객체를 복사
5. Singleton : 하나의 객체를 어디서든 참조, 여러 프로세스 에서 동시 참조X

구조 패턴 : 객체 조합으로 더 큰 구조로 만드는 패턴

Adapter : 호환성 없는 클래스의 인터페이스를 다른 클래스가 이용할 수 있게 변환

Bridge : 구현부에서 추상층을 분리 독립적으로 확장이 가능한 패턴

Proxy : 접근이 어려운 객체와 연결하고자하는 객체 사이의 인터 페이스 (대리자)

행위패턴 : 객체 상호작용, 책임분배

1. observer : 이벤트 생성·발행(Publish), 이를 수신(Subscribe)하여 협력
2. Mediator : 수많은 상호 작용(Interface)을 캡슐화 하여 객체로 정의
3. Template Method : 상위 클래스에서 골격을 정의 하위에서 세부 처리를 구체화

test

화이트 박스 테스트

기준

1. 문장 검증 기준 : 코드의 모든 구문이 한 번 이상 수행
   (Statement Coverage)
2. 분기 검증 기준 : 모든 조건문에 조건식 결과가 True, False인 경우를 한 번씩 수행
   (Branch Coverage)
3. 조건 검증 기준 : 조건문에 포함된 개별 조건식의 결과가 True, False인 경우를 한 번씩 수행
   (Condition Coverage)

ex)

블랙 박스 테스트

종류

1. 동치 분할 검사 : 입력 조건에 타당한 것과 그렇지 않은 것의 기초 개수를 공평하게 하고 입력 조건에 맞는 출력 결과와 함께
   (Equivalence Partitioning Test)
2. 경계값 분석 : 입력 조건의 경계값을 테스트
   (Boundary Value Analysis)
3. 원인-효과 그래프 : 입력 데이터 간의 관계와 출력에 영향을 미치는 상황을 체계적으로 분석한 다음 최소의 비용으로 효과적인 테스트 케이스 선정
   (Cause-Effect Graphing Testing)

ex)

Secure

Network