자료구조

한국폴리텍대학 대구캠퍼스
AI엔지니어링학과 강현우

자료구조
– 1강
Introduction

한국폴리텍대학
대구캠퍼스

Basic Concept

- **Engineer?**
  - 엔지니어 → 기술자? 공학자?

Basic Concept

- SW도 제품이다.

- Requirements
  - Gathering / Analysis
    - ✓ Elimination of ambiguities
    - ✓ Partition of requirements

- Specification
  - Define what the System is to do
  - Functional / Performance specification

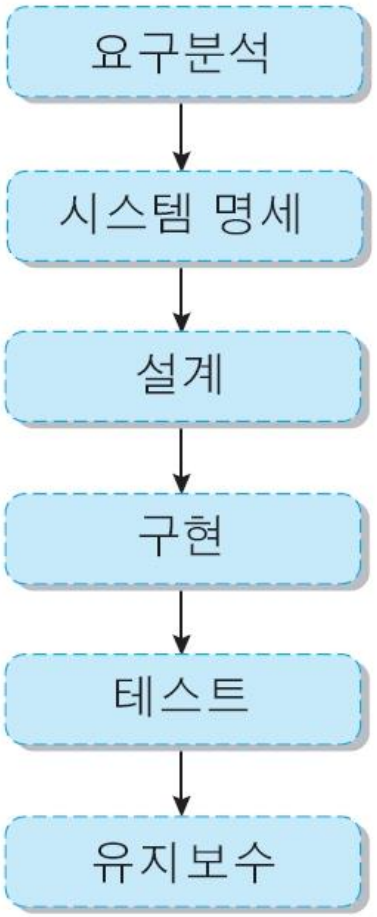
Basic Concept

- Design → 자료구조, 알고리즘
  - Description of how to achieve the Spec.
  - Data objects / Operations
- Coding
- Verification
  - Test → Trouble Shooting
- Operation and maintenance
  - DevOps

Software Life Cycle

◆소프트웨어 생명주기 (Software Life Cycle)

- ➢소프트웨어를 체계적으로 개발하고 관리하기 위해서 개발 과정을 단계별로 나누어 구분한 것
- ➢일반적으로 6단계로 구분



Software Life Cycle

◆요구분석 단계

- ➢문제 분석 단계
- ➢개발할 소프트웨어의 기능과 제약조건, 목표 등을 소프트웨어 사용자와 함께 명확히 정의하는 단계
- ➢개발할 소프트웨어의 성격을 정확히 이해하고 개발 방법과 필요한 개발 자원 및 예산 예측
- ➢요구명세서 작성

◆시스템 명세

- ➢시스템이 무엇을 수행해야 하는가를 정의하는 단계
- ➢입력 자료, 처리 내용, 생성되는 출력 정의
- ➢시스템 기능 명세서 작성

Software Life Cycle

◆설계 단계

- **시스템 명세 단계에서 정의한 기능을 실제로 수행하기 위한 방법을 논리적으로 결정하는 단계**

- **시스템 구조 설계**
  - ✓시스템을 구성하는 내부 프로그램이나 모듈 간의 관계와 구조 설계

- **프로그램 설계**
  - ✓프로그램 내의 각 모듈에서의 처리 절차 알고리즘을 설계

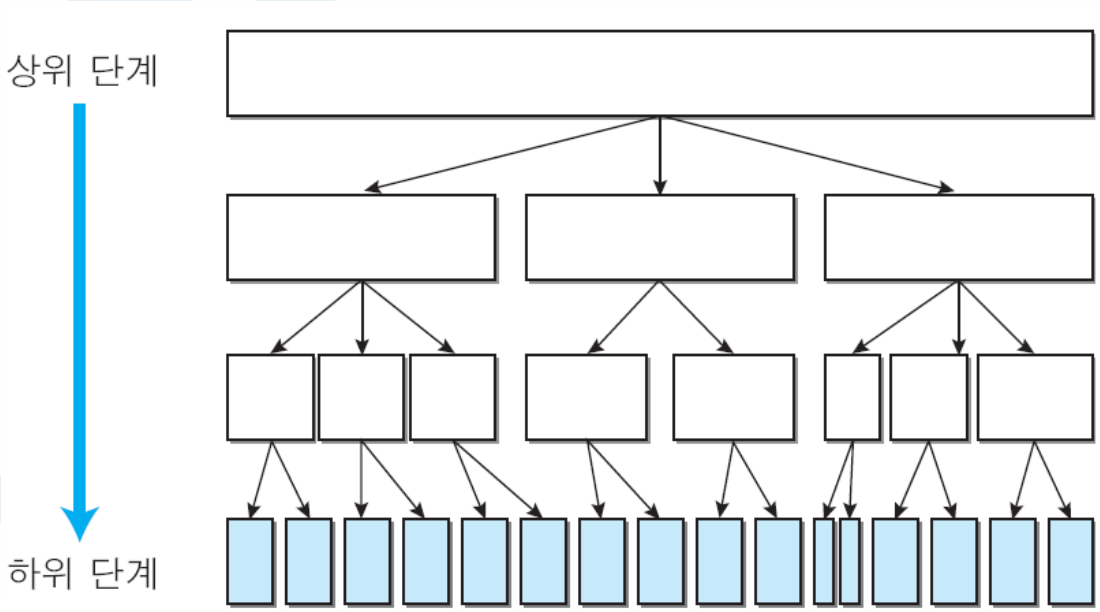
- **사용자 인터페이스 설계**
  - ✓사용자가 시스템을 사용하기 위해 보여지는 부분 설계

Software Life Cycle

➢설계 방법

✓하향식 설계 방법 (Top-Down)

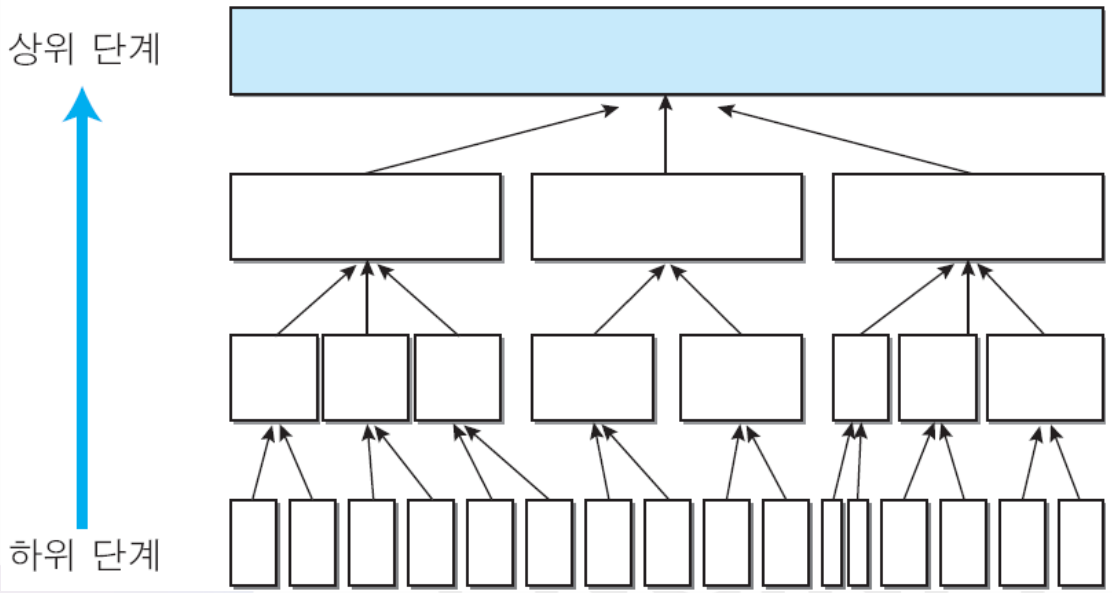
- 상위단계에서 하위단계로 설계해가면서 점차 구체적으로 설계하는 방법



Software Life Cycle

✓ 상향식 설계 방법 (Bottom-Up)

- 하위단계의 작은 단위의 문제를 먼저 해결하고 이를 이용하여 상위단계의 큰 단위의 문제를 해결하는 방법



✓ 객체지향 설계 방법

- 하위단위의 문제해결 도구를 객체로 만들어 재사용하는 방법으로 전체 문제를 해결하는 방법

Software Life Cycle

알고리즘

◆구현 단계

➢**설계 단계에서 논리적으로 결정한 문제 해결 방법을**
➢**실제 프로그램으로 작성하는 단계**

➢**프로그래밍 기법**

✓구조화 프로그래밍

- 지정문과 조건문, 반복문 만을 사용하여 프로그램을 작성
- 순차구조, 선택구조, 반복구조의 세 가지 제어구조로 표현
- 구조가 명확하여 정확성 검증과 테스트 및 유지보수 용이

✓모듈러 프로그래밍

- 프로그램을 여러 개의 작은 모듈로 나누어 계층 관계로 구성하는 프로그래밍 기법
- 모듈별로 개발과 테스트 및 유지보수 가능
- 모듈의 재사용 가능

Software Life Cycle

◆테스트 단계

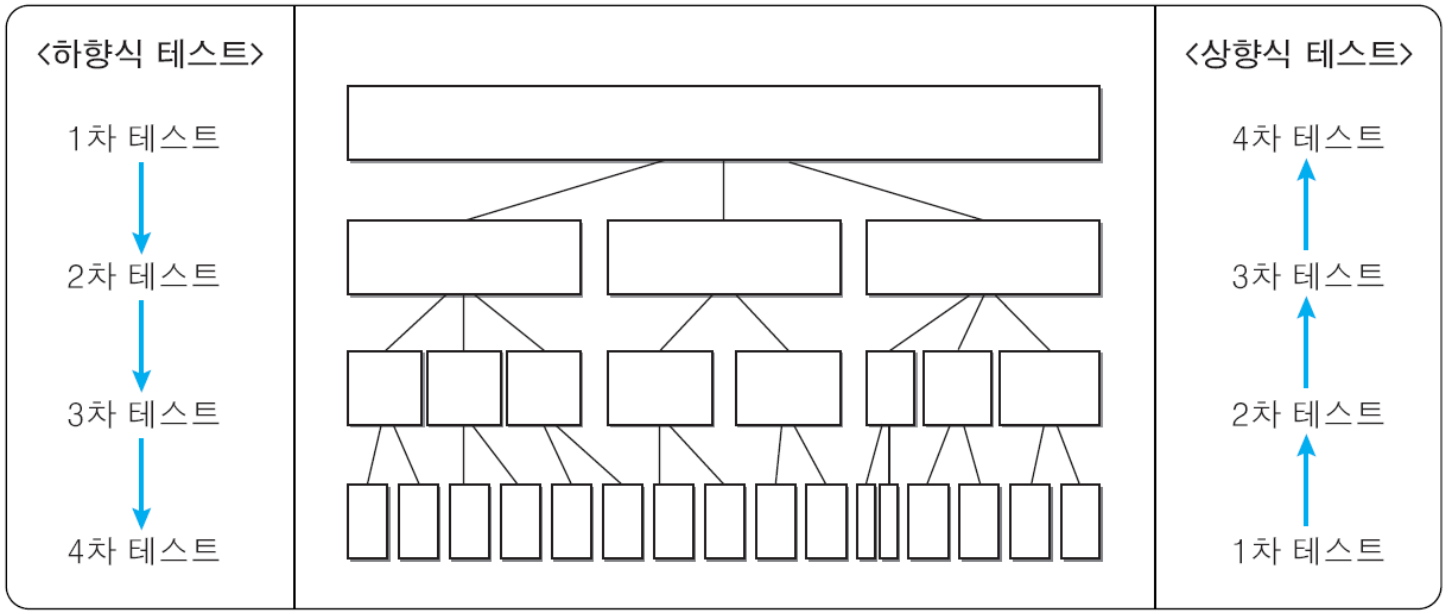
- **개발한 시스템이 요구사항을 만족하는지**
- **실행결과가 예상한 결과와 정확하게 맞는지를**
- **검사하고 평가하는 일련의 과정**
- **숨어있는 오류를 최대한 찾아낸다.**
  - ✓ 시스템의 완성도를 높임

- **1단계 : 단위 테스트(Unit Test)**
  - ✓ 시스템의 최소 구성요소가 되는 모듈에 대해서

- **2단계 : 통합테스트(Integration test)**
  - ✓ 단위 테스트를 통과한 모듈을 연결, 전체 시스템으로 완성
  - ✓ 통합적으로 시행하는 테스트
  - ✓ 구성요소 연결을 점진적으로 확장하면서 테스트 시행
    - 하향식 테스트
    - 상향식 테스트

Software Life Cycle

✓하향식/상향식 점진적 테스트



➢**3단계 : 인수 테스트**

✓완성된 시스템을 인수하기 위해서 실제 자료를 사용한 최종 테스트

Software Life Cycle

◆유지보수 단계

➢시스템이 인수되고 설치된 후 일어나는 모든 활동

✓소프트웨어 생명주기에서 **가장 긴 기간**

➢유지보수의 유형

✓수정형 유지보수
- 사용 중에 발견한 프로그램의 오류 수정 작업

✓적응형 유지보수
- 시스템과 관련한 환경적 변화에 적응하기 위한 재조정 작업

✓완전형 유지보수
- 시스템의 성능을 향상시키기 위한 개선 작업

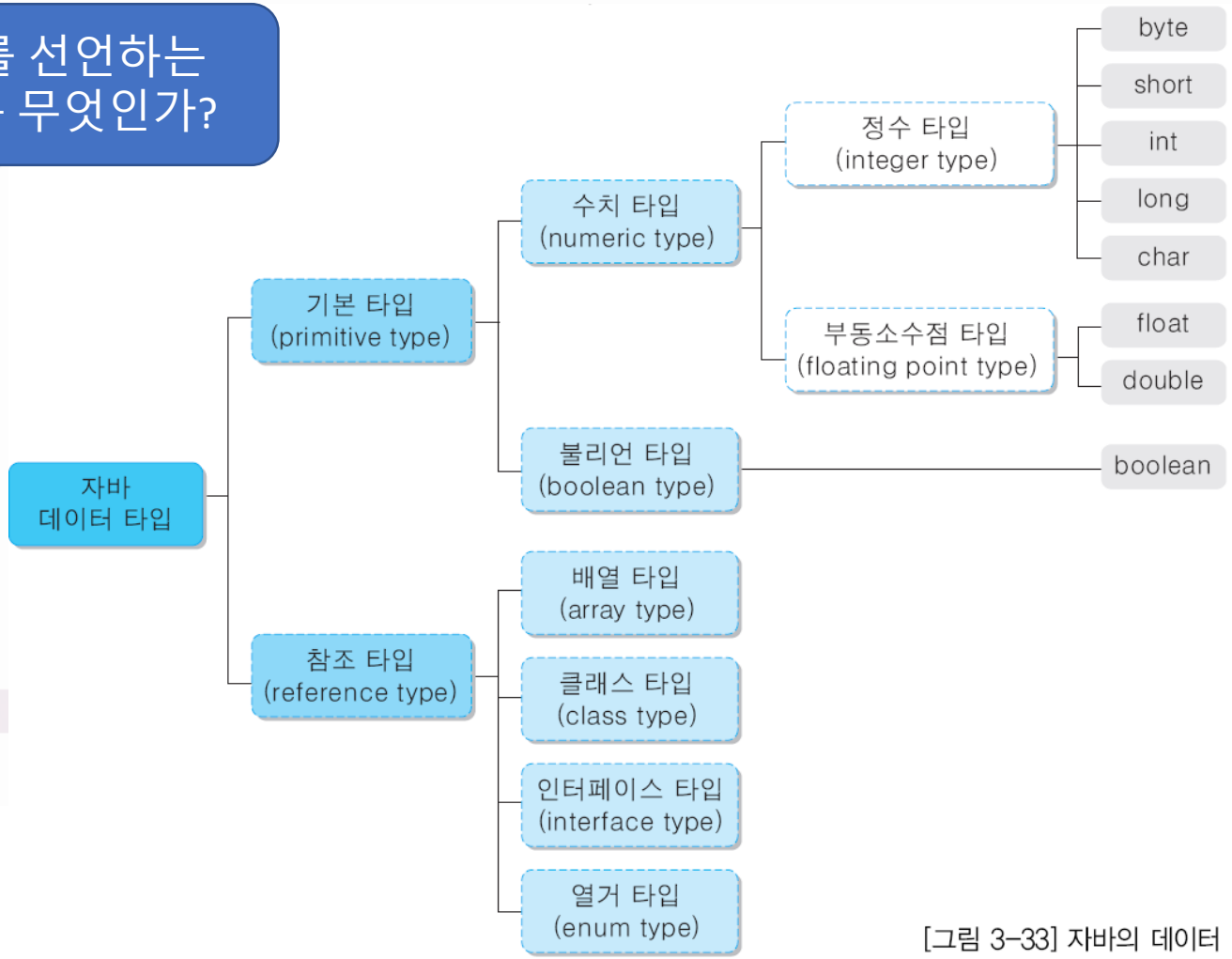
✓예방형 유지보수
- 앞으로 발생할지 모를 변경 사항을 수용하기 위한 대비 작업

자료구조
– 1강
Data Type

한국폴리텍대학
대구캠퍼스

지난 시간 복습

변수를 선언하는
의미는 무엇인가?



[그림 3-33] 자바의 데이터 타입

지난 시간 복습 – 조건문

숙제는 하였는가?
- BMI에 따라 등급...

Else는
단독으로
쓰일 수 없다.

If 구문의
scope

```
if (Boolean 값으로 치환될수 있는 코드, 즉 조건문)
{
    System.out.println("조건이 true 일경우에만 실행됩니다.");
}

else
{
    System.out.println("위의 if 가 true 가 아닌 경우에 실행됩니다.");
}
```

Primitive Type

◆키워드로 정의된 8개의 기본 데이터 타입

➢정의된 형태의 값을 변수에 저장하기 위한 데이터 타입

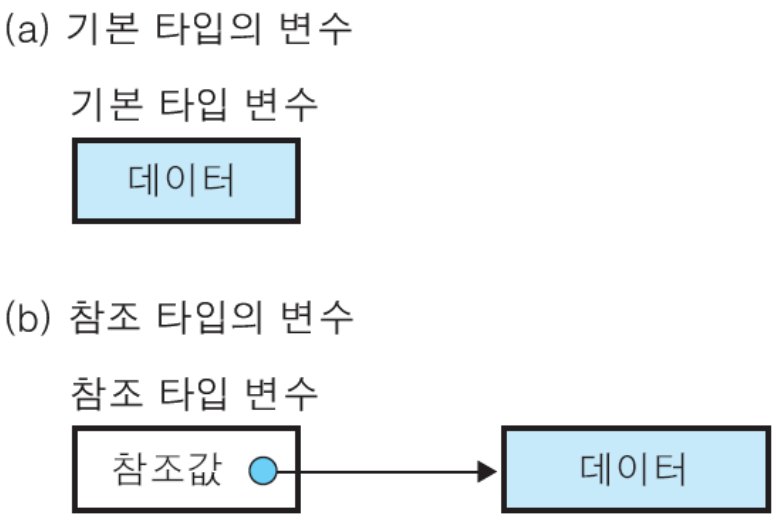
- ✓문자 데이터를 저장할 경우 유니코드 값을 사용
- ✓데이터 타입을 사용하여 변수를 선언하면, 해당 타입에 따라 정해진 메모리 크기가 변수에 할당

구분	기본 타입	메모리 크기
정수 타입	byte	1바이트
	short	2바이트
	int	4바이트
	long	8바이트
	char	2바이트
부동 소수점 타입	float	4바이트
	double	8바이트
불리언 타입	boolean	정해져있지 않음

Reference Type

◆참조값(reference value)을 다루기 위한 데이터 타입

- ➢참조값 : 메모리 주소를 계산할 수 있는 값
- ➢기본 타입을 제외한 나머지 타입
  - ✓배열, 클래스, 인터페이스, 열거 타입
- ➢기본 타입과 참조 타입의 비교



배열(Array) Type

◆배열 타입

- ➢**같은 데이터 타입의 변수들을 메모리에 연속으로 저장한 데이터들의 그룹**
- ➢**인덱스**
  - ✓배열의 요소를 구별하기 위해 사용하는 번호
  - ✓**0부터 시작**
  - ✓1차원 배열의 선언과 생성 및 사용
    - 기본형 변수와 마찬가지로 배열 변수를 선언한 후에 사용

① 데이터 타입 배열이름 [ ] ; ❶ ❷ ❸
② 데이터 타입 [ ] 배열이름 ; ❶ ❸ ❷

❶ 데이터 타입 : 배열을 구성하는 각 요소에 대한 공통된 데이터 타입
❷ 배열 이름 : 배열 변수의 이름으로 사용자가 정한 식별자
❸ 대괄호 : 보통의 변수가 아니라 배열 변수임을 나타내는 기호

배열(Array) Type

배열 선언	char c[ ]; char[ ] c;	int i[ ]; int[ ] i;	long l[ ]; long[ ] l;
의미	char형 배열 요소들에 대한 배열 c 선언	int형 배열 요소들에 대한 배열 i 선언	long형 배열 요소들에 대한 배열 l 선언

➢배열 생성

✓기본 타입의 변수를 선언하면 그에 대한 메모리 공간이 자동 생성되지만, 참조 타입의 변수는 메모리 참조값을 저장할 수 있는 공간만 확보

✓**new 연산자**를 사용하여 데이터가 저장될 전체 메모리 생성 작업 필요

- 생성되는 메모리 크기 = 데이터 타입에 지정된 메모리 크기 x 배열 크기

```
new 데이터타입 [배열크기] ;
❶   ❷      ❸
```

❶ new : 메모리 생성(할당) 연산자

❷ 데이터 타입 : 생성할 배열의 데이터 타입

❸ 배열크기 : 생성할 배열의 내부 요소의 개수

배열(Array) Type

배열 생성	c = new char[10];	i = new int[10];	l = new long[10];
의미	char형 배열 요소 10개에 대한 메모리가 생성된 배열 c	int형 배열 요소 10개에 대한 메모리가 생성된 배열 i	long형 배열 요소 10개에 대한 메모리가 생성된 배열 l
배열 요소	[0]~c[9]	[0]~i[9]	[0]~l[9]
메모리 할당크기	2byte x 10	4byte x 10	8byte x 10

실습

```
public static void main(String[] args) {
    int [] array_test;

    array_test = new int[5];

    array_test[0] = 1;
    array_test[1] = 1;
    array_test[2] = 2;
    array_test[3] = array_test[1] + array_test[2];
    array_test[4] = array_test[2] + array_test[3];

    System.out.println("array_test[0] = " + array_test[0]);
    System.out.println("array_test[1] = " + array_test[1]);
    System.out.println("array_test[2] = " + array_test[2]);
    System.out.println("array_test[3] = " + array_test[3]);
    System.out.println("array_test[4] = " + array_test[4]);
}
```

배열의 선언

메모리 공간에 할당

New 를 안하고 쓰려고 하면? → 오류
java: variable array_test might not have been initialized

인덱스로 접근
일반 변수처럼 사용하면 됨

인덱스를 넘어가면?
Out of Range (bounds)

실습

```
public static void main(String[] args) {
    int [] gugudan_array = new int[9];
    int dan = 2;

    gugudan_array[0] = dan * 1;
    gugudan_array[1] = dan * 2;
    gugudan_array[2] = dan * 3;
    gugudan_array[3] = dan * 4;
    gugudan_array[4] = dan * 5;
    gugudan_array[5] = dan * 6;
    gugudan_array[6] = dan * 7;
    gugudan_array[7] = dan * 8;
    gugudan_array[8] = dan * 9;

    System.out.println(dan + " * 1 = " + gugudan_array[0]);
    System.out.println(dan + " * 2 = " + gugudan_array[1]);
    System.out.println(dan + " * 3 = " + gugudan_array[2]);
    System.out.println(dan + " * 4 = " + gugudan_array[3]);
    System.out.println(dan + " * 5 = " + gugudan_array[4]);
    System.out.println(dan + " * 6 = " + gugudan_array[5]);
    System.out.println(dan + " * 7 = " + gugudan_array[6]);
    System.out.println(dan + " * 8 = " + gugudan_array[7]);
    System.out.println(dan + " * 9 = " + gugudan_array[8]);
}
```

선언과 동시에 메모리 할당

배열에 계산 결과를 assign

Print문 안에서 + 는 숫자 더하기가 아니다.
문자열로 처리 된다.

For 반복문

```
for (초기화식; 조건식; 증감식)
{
    // 실행문
}
```

증감식에 의해서 조건식의 결과가 달라질 수 있어야 한다.

그게 아니라면? 무한 루프

① 초기화식이 실행
② 조건식을 판별
③ 조건이 참 → 4번으로, 거짓 → 5번으로
④ 증감식 실행 → 2번으로
⑤ For 루프 종료

실습

변하는 값을 변수 i 로 사용
일반적으로 array의 index는 i로 쓴다.

```
gugudan_array[0] = dan * 1;
gugudan_array[1] = dan * 2;
gugudan_array[2] = dan * 3;
gugudan_array[3] = dan * 4;
gugudan_array[4] = dan * 5;
gugudan_array[5] = dan * 6;
gugudan_array[6] = dan * 7;
gugudan_array[7] = dan * 8;
gugudan_array[8] = dan * 9;
```

```
for (int i = 0; i < 9; ++i)
{
    gugudan_array[i] = dan * (i + 1);
}
```

배열은 0부터 시작하므로
I + 1 을 곱해주어야 한다.

실습

```
for (int i = 0; i < 9; ++i)
{
    gugudan_array[i] = dan * (i + 1);
}

for (int i = 0; i < 9; ++i)
{
    System.out.println(dan + " * " + (i + 1) + " = " + gugudan_array[i]);
    System.out.printf("%d * %d = %d\n", dan, i+1, gugudan_array[i]);
}
```

위 2개의 print문의 출력은 같다.
편한 것을 사용하면 된다.