

**교육용프로그래밍언어기초(스크래치)**

**4주차 - 1교시**

# 프로시저의 개념과 구현





- 🔗 프로시저의 개념과 구현
- 🔗 프로시저 호출과 실행 방식
- 🔗 프로시저의 인자와 매개변수
- 🔗 프로시저의 인자 전달 방식
- 🔗 프로시저의 지역 변수
- 🔗 프로시저의 반환값
- 🔗 스크래치의 프로시저 구현

# 학습목표

- ① 프로시저의 개념을 이해하고 설명할 수 있다.
- ② 프로시저 호출에 의한 실행 방식을 사용할 수 있다.
- ③ 프로시저의 인자와 매개변수를 설명할 수 있다.
- ④ 프로시저의 다양한 인자 전달 방식을 사용할 수 있다.
- ⑤ 프로시저의 지역 변수에 대해 설명할 수 있다.
- ⑥ 프로시저의 반환값에 대해 설명할 수 있다.
- ⑦ 스크래치의 프로시저를 구현할 수 있다.



## 명제와 조건

- 명제는 진리값(참 또는 거짓)을 분명히 판단할 수 있는 문장이나 식이며, 조건은 포함된 미지수 값에 따라 진리 값이 결정되는 문장이나 식



## 관계식

- $\rangle$ ,  $=$ ,  $\langle$  등의 관계 연산자를 포함한 식으로 진리 값을 결과로 맞춤

# 지난주차 정리



## 논리 연산자

- 명제나 조건을 논리 연산자로 연결하면 복잡한 관계를 논리식으로 표현할 수 있음
- 대표적인 논리연산자로는 논리합( $\vee$ ), 논리곱( $\wedge$ ), 부정( $\sim$ ) 등이 있음



## 조건

- 프로그램 내에서 조건은 두 문장 중 어떤 문장을 선택 실행할 것인지, 어떤 문장을 반복 실행할 것인지를 결정하는 기준, 즉 프로그램의 문장을 제어하는데 핵심적인 역할 수행



# 지난주차 정리



## 선택적 문장 제어 방식

- ‘만약’ 선택형과 ‘만약~아니면’ 선택형이 있음
- 스크래치는 ‘만약’ 선택형과 ‘만약-아니면’ 선택형을 구현할 수 있는 선택적 문장 제어 블록 제공



## 반복적 문장 제어 방식

- ‘무한’ 반복형, ‘횟수’ 반복형, ‘while’ 반복형, ‘until’ 반복형이 있음
- 스크래치는 ‘무한’ 반복형, ‘횟수’ 반복형, ‘while’ 반복형, ‘until’ 반복형을 구현할 수 있는 반복적 문장 제어 블록 제공



**생각해 봅시다**

**“프로시저(함수)란 무엇일까요?”**



# 1

## 프로시저의 개념과 구현





## 1

# 프로시저의 개념과 구현

## 1 프로시저란?

### 프로시저

프로그램의 일부 기능을 담당하는  
연속적인 문장들의 묶음

**절차적 프로그래밍 방식**의 핵심 개념

## 1

# 프로시저의 개념과 구현

## 1 프로시저란?

- 프로시저를 활용하면 공통으로 사용하는 기능을 따로 독립시켜, **필요할 때마다 호출**하여 사용할 수 있음
  - 이렇게 만들어 둔 프로시저는 프로그램 기능을 중복으로 구현해야 하는 번거로움과 작업량을 줄여 줌

## 1

# 프로시저의 개념과 구현

## 2 절차적 프로그래밍 방식이란?

- 프로그램을 **프로시저들 간 호출관계**로 보는  
프로그래밍 관점
  - 프로그래머는 프로그램의 기능을 여러 개의 프로시저들로 나누어 구현
  - 프로시저들 간에 호출관계를 적절히 만들어 프로그램의 기능을 구현

## 1

# 프로시저의 개념과 구현

## 2 절차적 프로그래밍 방식이란?

- 〈계산기의 기능을 절차적 프로그래밍 방식으로 구현〉
  - 사용자와 상호작용 기능을 수행하는 프로시저
    - 사용자로부터 숫자, 연산 종류를 입력 받음
    - 연산의 종류에 따라 해당 프로시저를 호출
    - 계산된 결과를 사용자에게 보여줌

# 1

## 프로시저의 개념과 구현

### 2 절차적 프로그래밍 방식이란?

#### • 〈계산기의 기능을 절차적 프로그래밍 방식으로 구현〉

##### ▪ 사용자와 상호작용 기능을 수행하는 프로시저

- 사용자로부터 숫자, 연산 종류를 입력 받음
- 연산의 종류에 따라 해당 프로시저를 호출
- 계산된 결과를 사용자에게 보여줌

사용자가  
입력한 숫자  
전달

더하기 기능을 수행하는 프로시저  
전달 받은 숫자로 더하기를 수행함

빼기 기능을 수행하는 프로시저  
전달 받은 숫자로 빼기를 수행함

곱하기 기능을 수행하는 프로시저  
전달 받은 숫자로 곱하기를 수행함

나누기 기능을 수행하는 프로시저  
전달받은 숫자로 나누기를 수행함

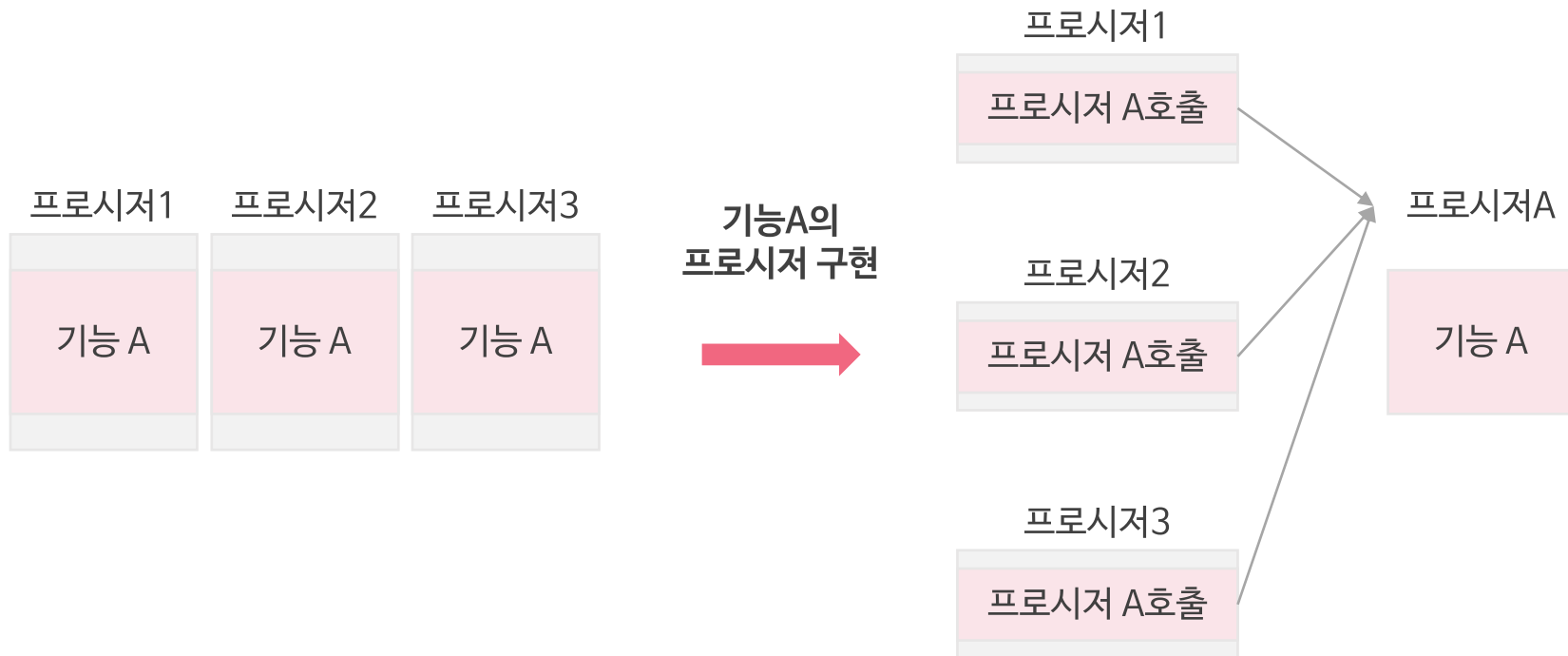
## 1

# 프로시저의 개념과 구현

## ③ 절차적 프로그래밍 방식의 특징

- 1 프로시저들이 공통으로 사용하는 기능을 별도의 프로시저로 구현
- 2 프로그램 기능의 중복적 구현을 방지 (프로그래밍 작업량 감소)
- 3 기능의 수정 작업이 해당 프로시저에서만 이루어짐

## ③ 절차적 프로그래밍 방식의 특징



## 1

# 프로시저의 개념과 구현

## ③ 절차적 프로그래밍 방식의 특징

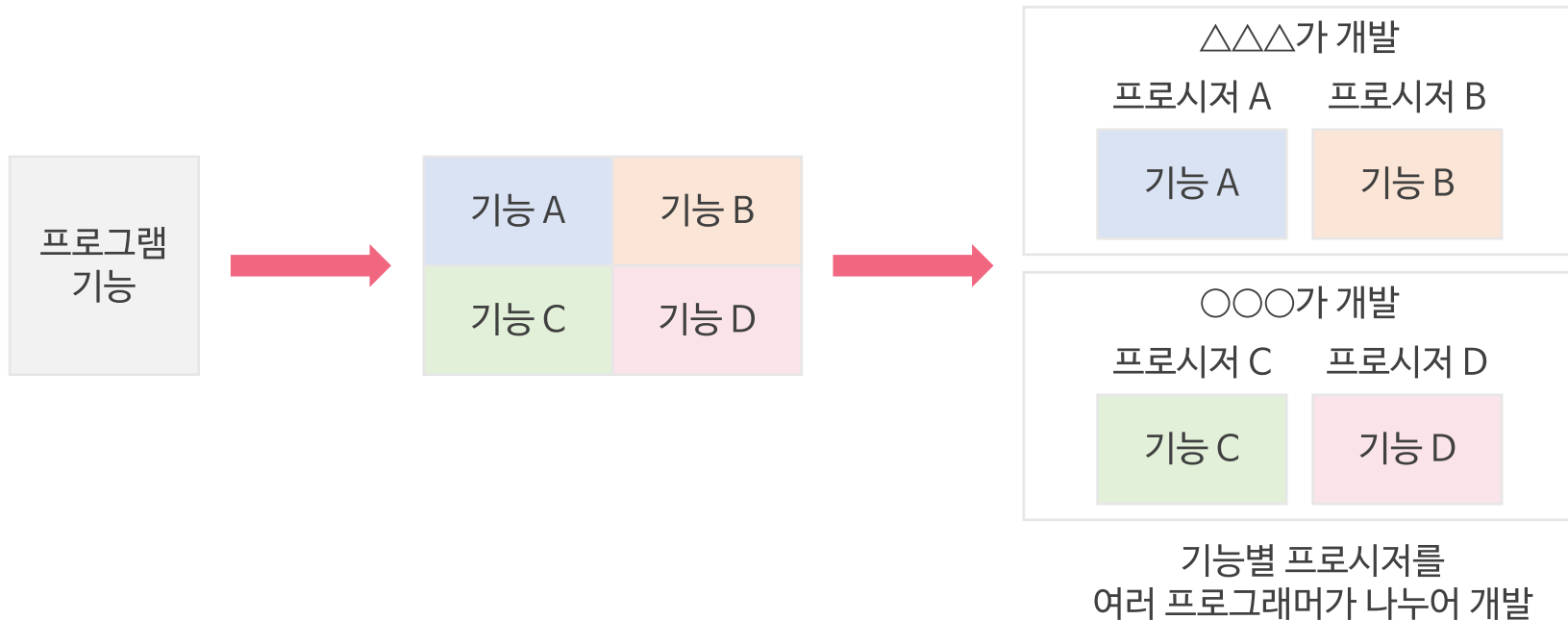
- 4 복잡한 기능을 보다 간단한 작은 단위의 기능들로 분해하는 작업을 도와 줌
- 5 분해된 기능들은 프로시저들로 설계되고 여러 프로그램들에게 할당되어 병렬적으로 구현



# 1

## 프로시저의 개념과 구현

### ③ 절차적 프로그래밍 방식의 특징





## 2

# 프로시저 호출과 실행 방식





## 2

# 프로시저 호출과 실행 방식

## 1 프로시저 이름과 호출

- 프로시저는 자신의 기능을 대표하는 이름을 가짐
- 프로시저는 자신의 이름이 호출되면 내부에 포함하는 문장들을 수행함

## 2

## 프로시저 호출과 실행 방식

## 2 프로시저 호출의 유효범위

- 프로시저는 변수와 마찬가지로 자신이 생성된 장소를 기반으로 호출될 수 있는 유효범위를 가짐

## Point

유효범위 밖에서는 프로시저 호출이 안됨

## 2

## 프로시저 호출과 실행 방식

### ③ 프로시저 실행 방식

- 프로시저가 다른 프로시저를 호출하면, 호출한 프로시저의 실행이 완료될 때까지 기다림
- 호출한 프로시저의 실행이 완료되면 호출했던 지점 이후의 문장들을 이어서 실행

## 2

# 프로시저 호출과 실행 방식

## 3 프로시저 실행 방식



## 2

# 프로시저 호출과 실행 방식

## 3 프로시저 실행 방식

- 아래에 같은 프로시저 호출에서 문자들의 실행 순서 알아 보기

프로시저 1

문장(1)
프로시저 2 호출 문장
문장(2)

프로시저 2

문장(3)
프로시저 3 호출 문장
문장(4)
프로시저 4 호출 문장
문장(5)

프로시저 3

문장(6)
-------

프로시저 4

문장(7)
-------

문장(1) → 프로시저2 호출 문장 → 문장(3) → 프로시저3 호출 문장 → 문장(6) → 문장(4)  
 → 프로시저4 호출 문장 → 문장(7) → 문장(5) → 문장(2)



# 3

## 프로시저의 인자와 매개변수





## 3

## 프로시저의 인자와 매개변수

## 1 프로시저 인자와 매개변수의 역할

- 프로시저를 호출할 때 그 프로시저의 기능 수행에 필요한 자료들을 전달함

## 프로시저 인자

- 프로시저를 호출할 때 전달할 자료들  
(프로시저 이름 바로 옆에 표기)

## 3

## 프로시저의 인자와 매개변수

## 1 프로시저 인자와 매개변수의 역할

- 프로시저를 호출할 때 그 프로시저의 기능 수행에 필요한 자료들을 전달함

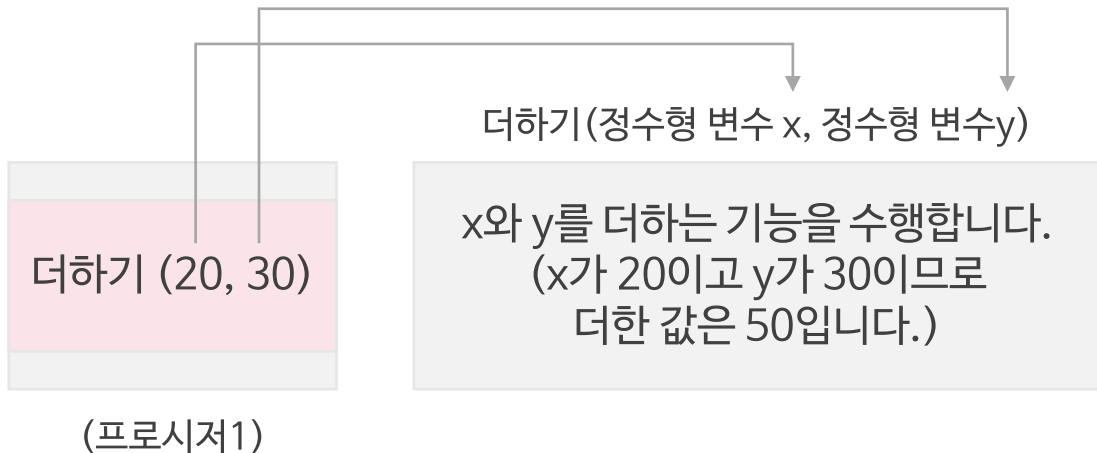
## 프로시저 매개변수

- 자신이 호출될 때 함께 전달되는 자료들을 할당받기 위한 변수들
- 프로시저 인자와 매개변수는 개수가 동일해야 하고 적힌 순서대로 인자가 매개변수에 할당됨
- 인자의 자료형과 그 인자가 할당될 매개변수의 변수형은 동일해야 함
- 매개변수는 프로시저 내부에서 사용 가능

## 3

## 프로시저의 인자와 매개변수

## 1 프로시저 인자와 매개변수의 역할





# 4

## 프로시저의 인자 전달 방식

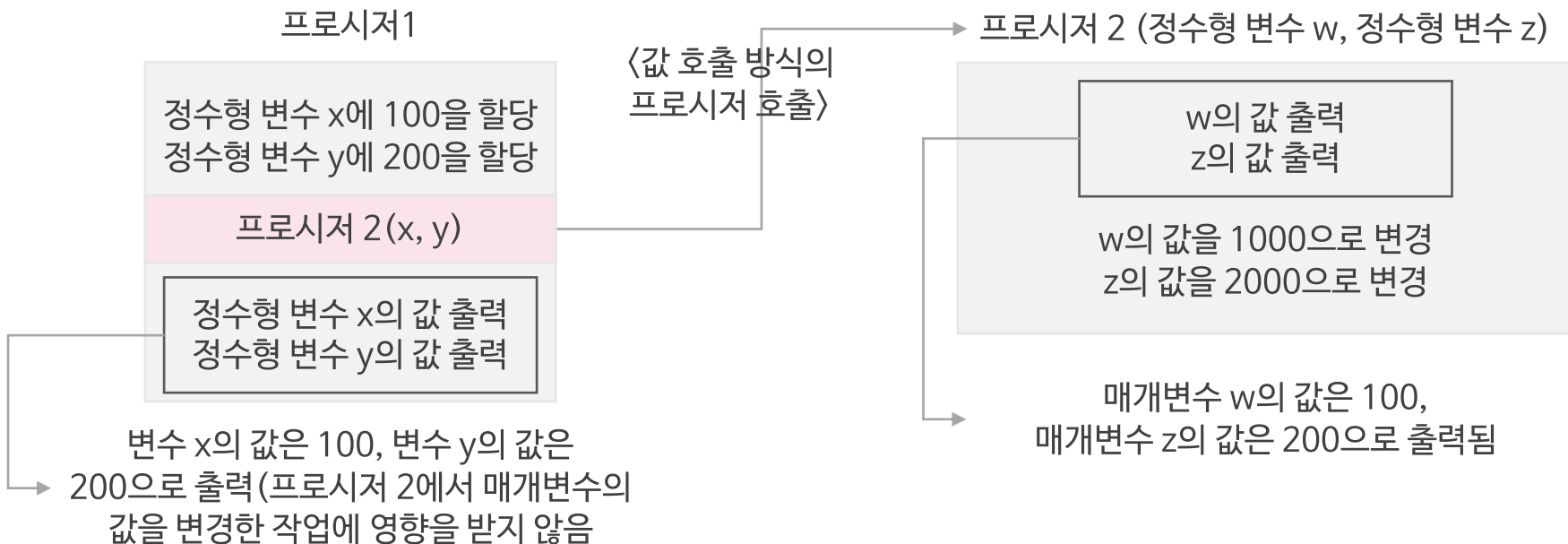


# 4

## 프로시저의 인자 전달 방식

### 1 값 호출 방식

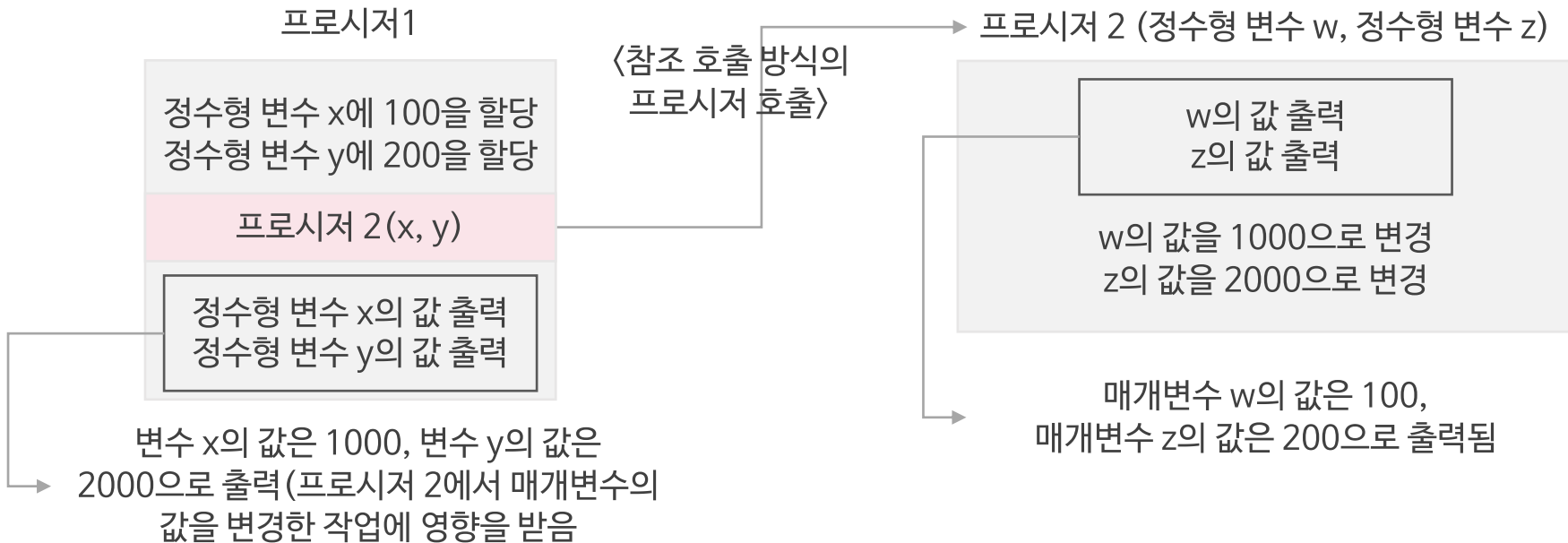
- 프로시저 내에서 매개변수가 변경되어도 **원래 대응하는 인자 값은 변경 없음**



## 4 프로시저의 인자 전달 방식

### 2 참조 호출 방식

- 프로시저 내에서 매개변수가 변경되면 **원래 대응하는 인자 값도 변경됨**





5

# 프로시저의 지역 변수



## 5

## 프로시저의 지역 변수

- 1 대부분의 프로그래밍 언어는 프로시저 내부에 변수 생성 기능 제공
  - 이런 변수는 그 프로시저의 지역변수로서 다른 프로시저에서는 활용하지 못함





# 5

## 프로시저의 지역 변수

### 2 프로시저의 지역 변수들

프로시저 내부에서 생성된 변수들

프로시저의 매개변수들

### ③ 지역 변수 사용의 장점

- 프로시저가 자신의 기능을 수행하는 데 지역 변수만 사용하면 다른 프로시저들 간의 의존성이 낮아지므로 프로시저를 이해하거나 수정하는 데 용이함

프로시저 1 (정수형 변수 x, 정수형 변수 y)

정수형 변수 sum 생성  
x와 y를 더하여 sum에 할당합니다.

프로시저 2

프로시저 1의 지역 변수들을 활용하지 못합니다.

프로시저 1의 매개변수 x, y와 내부에서 생성된 sum은 프로시저 1의 지역변수입니다.  
따라서 다른 프로시저들은 이 변수들을 사용하지 못합니다.



# 6

## 프로시저의 반환값





## 6

# 프로시저의 반환값

- 1 프로시저가 종료되는 순간에 반환하는 자료 값
  - 프로그래밍 언어에 따라 프로시저의 반환값 기능을 지원하기도 하고 안 하기도 함
  - 반환되는 자료 값의 자료형을 프로시저 이름 앞에 붙여 줌



## 6

# 프로시저의 반환값

## 2 반환값은 그 프로시저를 호출한 프로시저에서 변수로 할당받아 사용

- 프로시저의 반환값의 자료형과 반환값을 할당받는 변수의 변수형은 동일해야 함

## 2 반환값은 그 프로시저를 호출한 프로시저에서 변수로 할당받아 사용

### • 〈계산하기〉

정수형 변수 sum을 생성하고 0을 할당  
정수형 변수 x에 100을 할당  
정수형 변수 y에 200을 할당

sum=더하기(x, y)  
(더하기 (x, y)를 호출하고  
반환값을 변수 sum에 할당

변수 sum의 값 출력

변수 sum에 프로시저 더하기에서 반환한 값  
300이 할당되었으므로 300이 출력

반환값의 자료형을 프로시저 이름 앞에 붙임

정수형 더하기(정수형 변수 w, 정수형 변수 z)

정수형 변수 r을 생성  
매개변수 w와 z의 값을 더하여 r에 할당

변수 r의 값을 반환함

변수 r의 값 300이 반환되어  
변수 sum에 할당

# 6

## 프로시저의 반환값

### ③ 프로시저의 반환값

- 아래와 같이 정수를 반환하는 프로시저를 호출 할 때 변수 x의 값 알아보기

정수형 변수 multiply를 생성  
정수형 변수 x에 2를 할당  
정수형 변수 y에 6을 할당

multiply=곱하기(x, y)

변수 multiply의 값 출력

정수형 곱하기(정수형 변수 w, 정수형 변수 z)

정수형 변수 result를 생성  
매개변수 w와 z의 값을 곱하여 result에 할당

변수 result의 값을 반환함

변수 multiply = 6



7

# 스크래치의 프로시저 구현





## 7

## 스크래치의 프로시저 구현

프로시저의 이름과 매개변수들을 담은 절차 블록으로 구현

절차 블록은 [코드] 탭의 [내 블록] 메뉴에서 생성

프로그래머가 실제 생성하는 것은 절차 블록을 호출할 수 있는 문장 블록

절차 블록은 문장 블록의 정보 (프로시저 이름, 프로시저 매개변수 정보)를 활용하여 스크래치가 자동적으로 생성

스크래치의 프로시저는 반환값 기능이 없음

## 7

## 스크래치의 프로시저 구현

## 1 [움직이기] 프로시저 구현하기

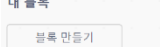
## 1 프로시저 호출 문장 블록 생성

- 프로시저 이름: 움직이기
- 프로시저 매개변수: x좌표, y좌표

## 7

## 스크래치의 프로시저 구현

## 1 [움직이기] 프로시저 구현하기

2 프로시저 호출 문장 생성이 완료되면,  
절차 블록이 자동적으로 생성됨① 코드 탭에서  내 블록 클릭②  클릭③  프로시저의 이름 입력④  입력값 추가하기

## 7

## 스크래치의 프로시저 구현

## 1 [움직이기] 프로시저 구현하기

2 프로시저 호출 문장 생성이 완료되면,  
절차 블록이 자동적으로 생성됨

⑤



매개변수의 이름 입력

⑥

확인 클릭

⑦



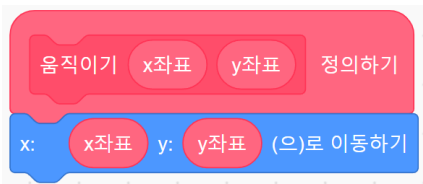
절차 블록 생성

## 7

## 스크래치의 프로시저 구현

## ② 생성된 절차 블록 아래 프로시저의 기능을 프로그래밍 함

## ① 프로시저의 기능을 프로그래밍할 때 매개변수를 활용할 수 있음



## 7

## 스크래치의 프로시저 구현

## 2 생성된 절차 블록 아래 프로시저의 기능을 프로그래밍 함

### 2 프로시저 호출 문장 블록으로 프로시저 호출

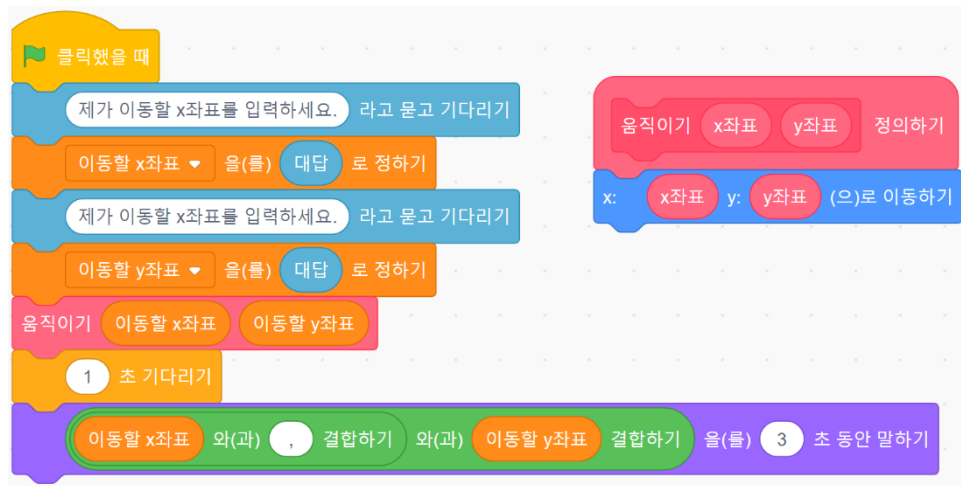
- 프로시저 호출 시, 문장 블록 내 포함된 인자들이 프로시저의 매개변수들에 할당됨
- 프로시저를 호출하면 호출된 프로시저의 작업이 완료될 때까지 기다렸다가 그 후에 호출 이후의 기능을 수행

# 7

## 스크래치의 프로시저 구현

2 생성된 절차 블록 아래 프로시저의 기능을 프로그래밍 함

3 프로시저 [움직이기]를 호출하는 고양이 스프라이트의 행동의 예



### ③ [사칙연산하기] 작업 구현하기

#### 1 고양이 스프라이트의 행동 개요

- ① 프로그램을 실행하면 고양이 스프라이트가 사용자에게 사칙연산 계산에 사용할 첫 번째 수를 요구합니다.
- ② 사용자가 수를 입력하면 고양이 스프라이트는 두 번째 수를 요구합니다.
- ③ 사용자가 두 번째 수를 입력하면 고양이 스프라이트는 더하기, 빼기, 곱하기, 나누기 중 어떤 연산을 원하는지 요구합니다. (사칙연산의 종류는 “더하기”, “빼기”, “곱하기”, “나누기”와 같이 문자열로 입력)
- ④ 사용자가 사칙연산 중 하나를 입력하면 두 수에 그 사칙연산을 계산한 결과값을 말하고 위의 작업을 반복합니다.



## 7

## 스크래치의 프로시저 구현

## ③ [사칙연산하기] 작업 구현하기

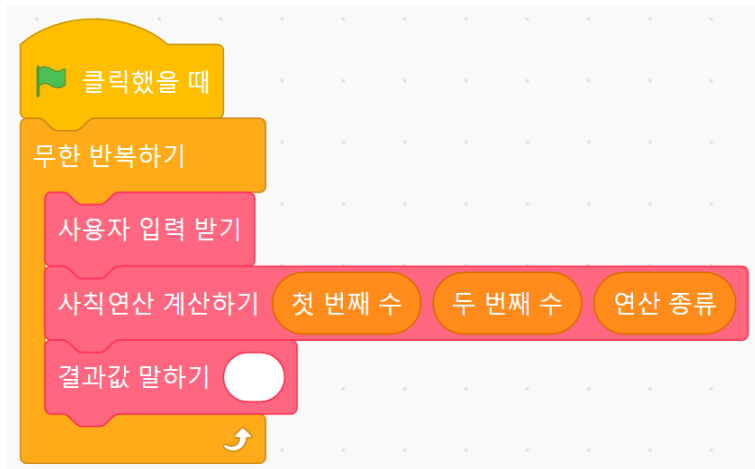
2 이 프로그램 기능의 구현은 아래와 같은 3개의 프로시저를 이용

프로시저	절차블록
사용자 입력 받기	
사칙연산 계산하기	
결과값 말하기	

## 7

## 스크래치의 프로시저 구현

## 4 [사칙연산하기] 작업의 스크립트



## 7

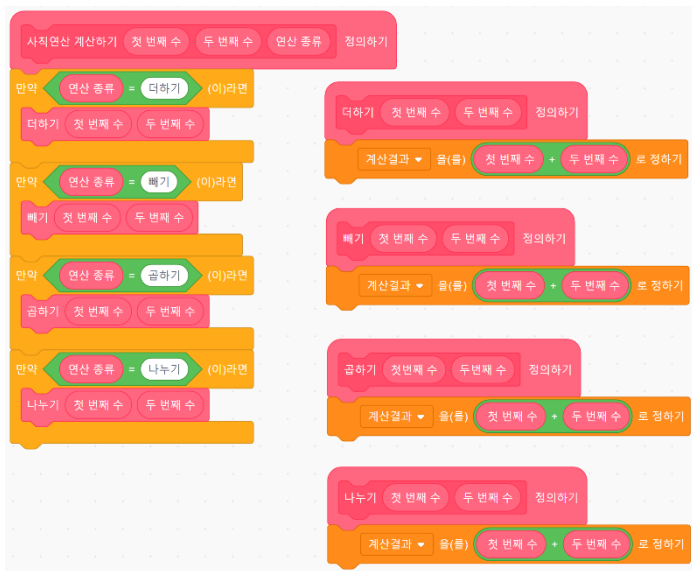
## 스크래치의 프로시저 구현

## 5 [사용자 입력 받기] 프로시저의 스크립트



## 6 [사칙연산 계산하기] 프로시저의 스크립트

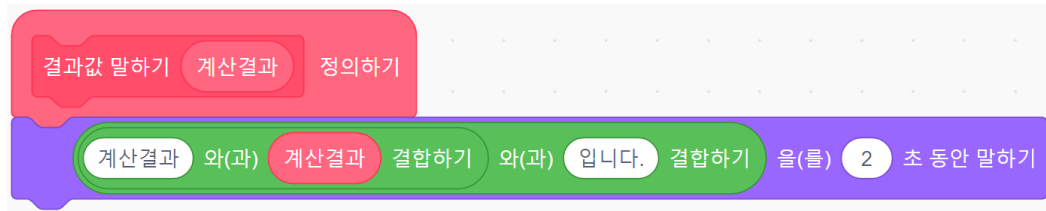
- [더하기], [빼기], [곱하기], [나누기] 프로시저를 생성하여 이용



## 7

## 스크래치의 프로시저 구현

## 7 [결과값 말하기] 프로시저의 스크립트



교육용프로그래밍언어기초(스크래치)

Next

# 병렬 처리의 개념과 구현

