

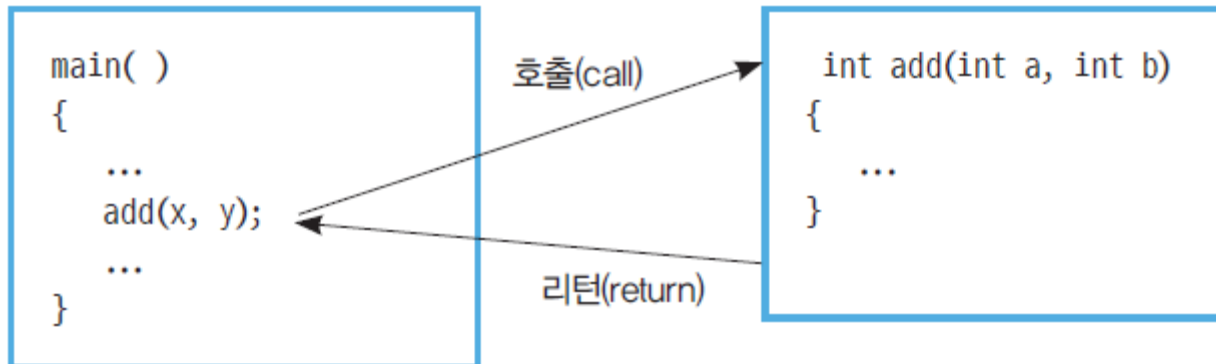
6장. 블록정의와 확장

6.1 재사용 절차

- C 언어의 경우에 자주 사용하는 기능에 해당하는 프로그램 부분은 함수로 정의하여 필요할 때마다 다시 프로그래밍하지 않고 미리 정의된 함수를 호출(call)하여 사용
- 전체 프로그램의 길어도 짧아지고, 함수에 대한 재사용률을 높이기 때문에 프로그램의 생산성이 높아진다.
- 재사용 절차 예
- A B G C D E F T E C D E F A B C D E F ...
- A B G S T E S A B S ... S : C D E F
 - 프로그램 작성이 그만큼 줄어들고, 프로그램 표현이 간결해 짐으로써 프로그램을 이해하는데 더 효과적
- C 언어의 경우엔 이와 같이 활용도가 높은 많은 기능이 함수(function)로 정의되어 라이브러리(Library) 함수로 제공

6.2 함수(1/2)

- C 언어에서 독립적인 기능을 하는 모듈
- 함수를 호출할 때 관련 데이터를 넘기기 위해서 인수(argument)를 사용
- C나 Java 언어에서는 call by value라는 방법을 이용
- call by value : 값을 복사하는 방법으로 인수를 넘긴다는 의미
- 호출하는 쪽에서의 변수나 상수 값이 호출되는 함수의 지역 변수에 값을 복사하는 방식



main() 함수의 지역 변수 x, y 값을
add() 함수 지역 변수인 a, b로 복사

그림 10.1 함수 호출과 리턴(반환)

6.2 함수(2/2)

C 프로그램에서 인수 전달 예

```
int add(int a, int b) // 정수형 형식 매개변수
{
    return(a + b);    // a와 b의 합을 반환한다.(리턴한다.)
}

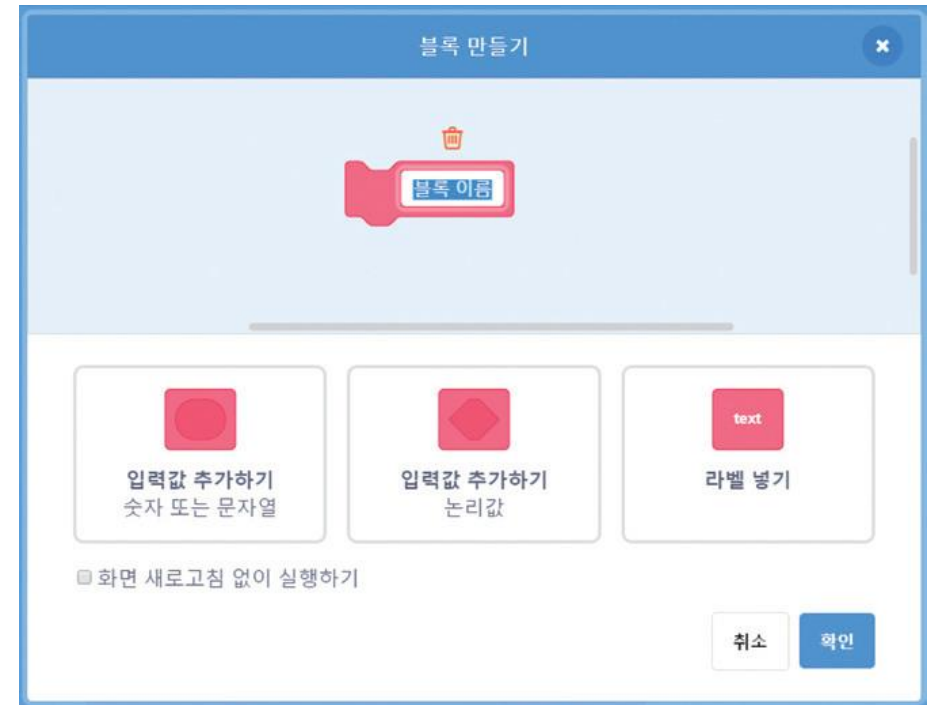
int main(void)
{
    int x = 3;
    int y = 4;
    int sum;

    sum = add(x, y);    // x, y는 정수형 실 매개변수, call by value
    printf("sum = %d\n", sum);
    return 0;
}
```

- add() 호출 시에 매개변수 x, y 값인 3, 4가 각각 형식 매개변수인 a, b로 복사

6.3 블록 만들기(1/20)

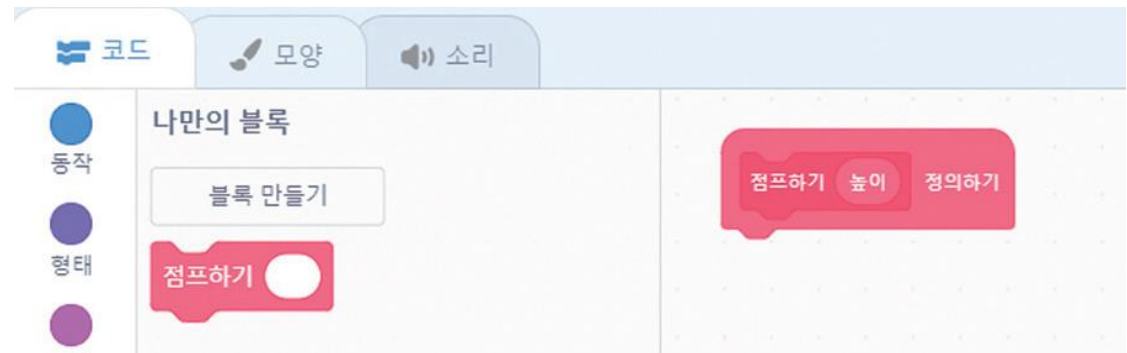
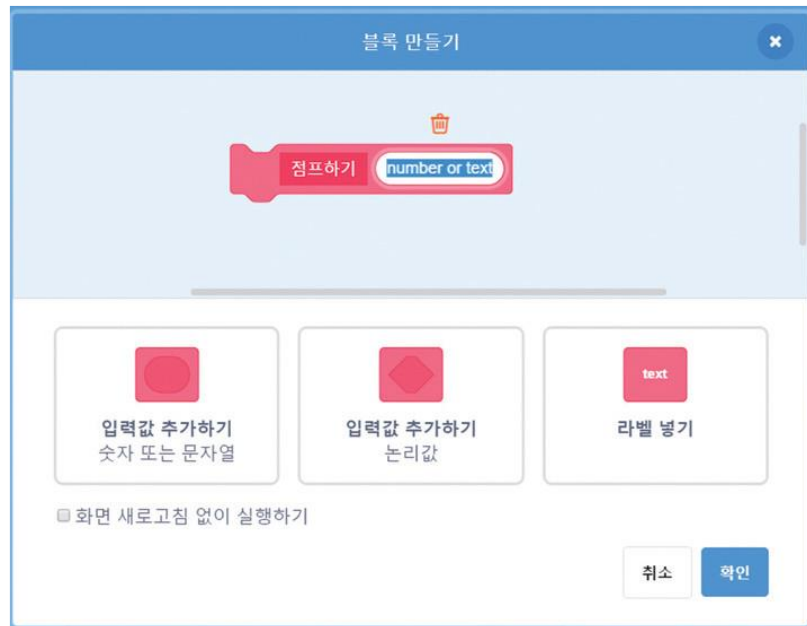
- 스크래치의 나만의 블록 : C 언어의 함수와 유사한 개념의 기능
 - 새로운 블록을 정의하여 필요할 때 사용할 수 있다는 의미
 - 코드 탭에 열거된 마지막 기본 영역이 나만의 블록 영역에는 블록 만들기 버튼만을 포함
-
- 블록 만들기에서는 자신이 만들고자 하는 블록의 이름을 정해주고 그 아래에 매개변수를 정의
 - 매개변수는 함수에 값을 넘겨주기 위해서 사용하는 것으로 call by value 방법에 의하여 값이 넘어간다.



6.3 블록 만들기(2/20)

블록 만들기 예

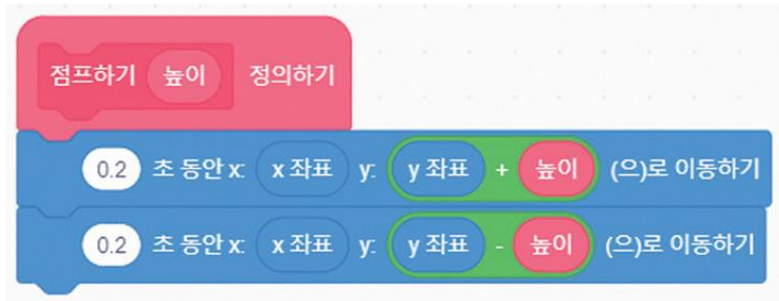
- '점프하기' 이름으로 나만의 블록을 정의하는데 점프하는 높이를 인수로 넘긴다고 가정
- 블록 만들기 대화창에서 블록 이름에 "점프하기"를 입력하고, 입력값 추가하기(숫자 또는 문자열)를 선택
- 블록만들기에서 number or text에 "높이"라고 입력하면 '높이'라는 블록 내의 지역 변수가 정의



6.3 블록 만들기(3/20)

점프하기 블록 정의 스크립트

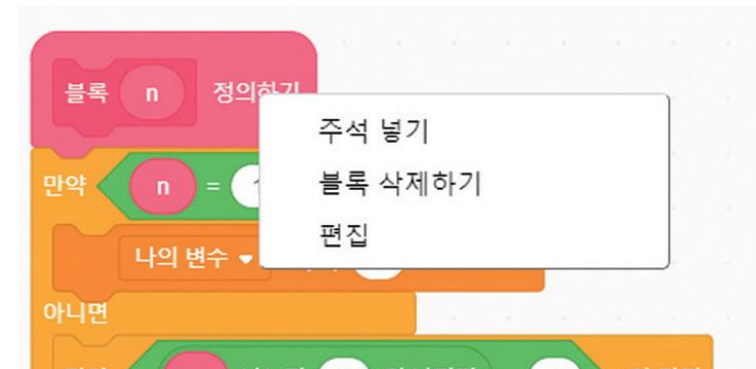
- 우선 점프할 높이는 인수로 들어오고 그 인수는 높이이므로 스프라이트의 y 좌표가 높이만큼 증가되었다 원 위치로 돌아오기위해서 증가한 높이만큼 감소되도록 프로그래밍



- 점프하기 블록 사용 예 : 5번 반복해서 점프하기



- 이미 정의한 블록을 필요에 따라 수정하고자 할 때에는 블록 편집 메뉴를 이용할 수 있다. 이미 정의된 블록 정의하기 위치에 마우스 오른쪽 버튼을 클릭



6.3 블록 만들기(4/20)

블록 활용

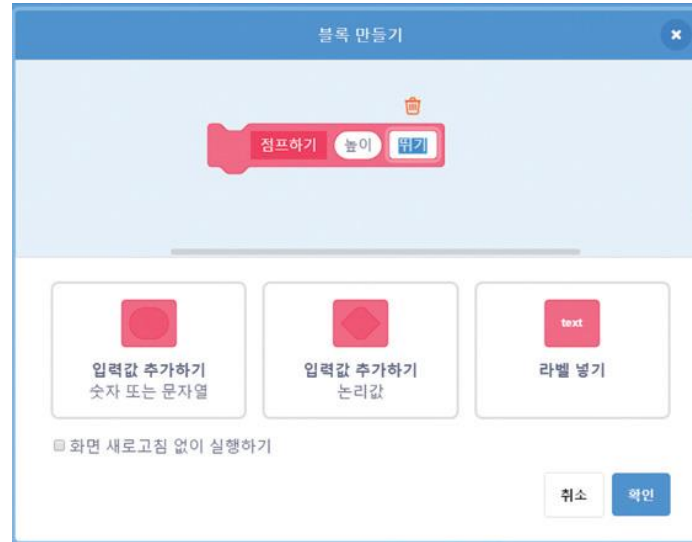
- 스크립트가 복잡할 경우에 각각의 부분을 구분하기 위해서 각 부분을 블록으로 정의하여 전체 구조를 이해하기 쉽게 나타낼 수 있다.
- 전체 스크립트를 두 부분으로 나누어 블록으로 정의



6.3 블록 만들기(5/20)

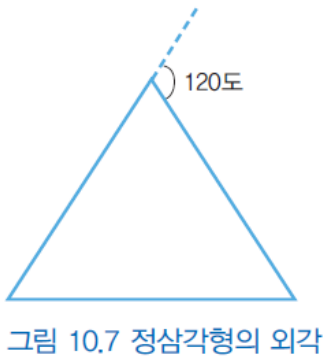
블록 정의에서 라벨 넣기

- 라벨 : 매개변수 뒤에 설명
- 라벨넣기 선택 후, 설명 입력



6.3 블록 만들기(6/20)

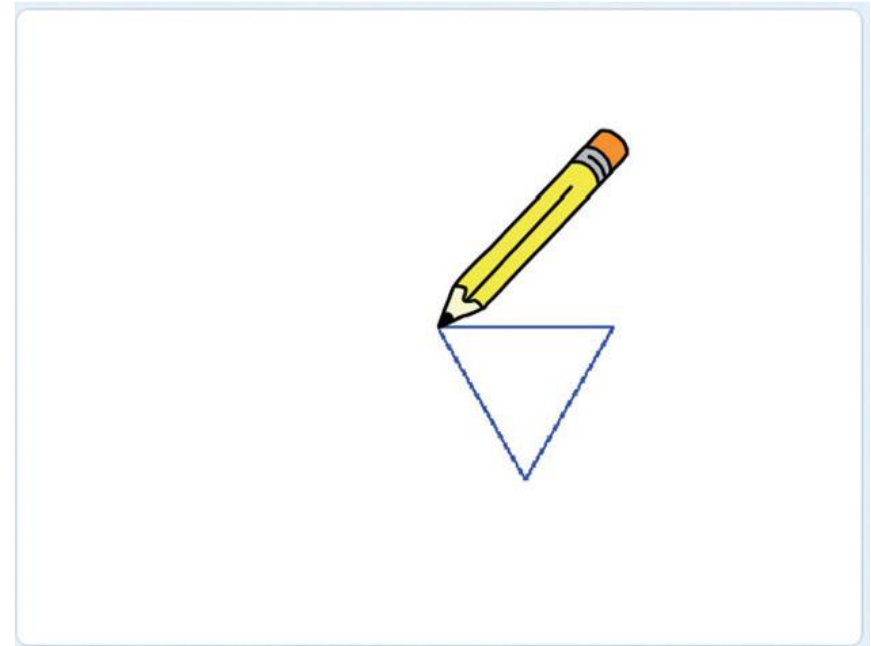
[예제 6.1] 나만의 블록을 이용하여 정삼각형을 그리는 블록을 정의하라. 인수는 정삼각형의 한 변의 길이를 입력 값으로 갖도록 한다.



- 단계 1 : 블록 만들기를 이용하여 정삼각형 블록을 정의한다.
이때 인수는 변의 길이로 입력 값을 추가한다.
- 단계 2 : 선을 그리기 위해서 펜 내리기를 한다.
- 단계 3 : 다음 단계 4, 5를 반복한다.
 - 단계 4 : 변의 길이만큼 이동한다.
 - 단계 5 : 시계방향으로 120도 회전한다.
- 단계 6 : 선 그리기를 마치기 위해서 펜 올리기를 한다.
- 준비 단계
 - 연필 모양을 위해서 pencil 스포라이트를 추가한다.
 - 연필 축 끝이 모양의 중심이 되도록 모양 편집기에서 모양을 이동

6.3 블록 만들기(7/20)

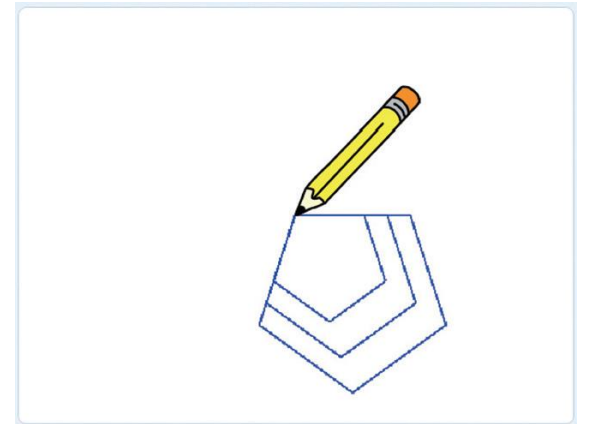
[예제 6.1] 스크립트와 실행화면



6.3 블록 만들기(8/20)

[예제 6.2] 예제 6.1 에서와 같은 방식으로 정오각형을 그리는 나만의 블록을 작성하고 이를 이용하여 다음과 같은 도형을 그리는 프로그램을 작성하라.

- 준비 단계
 - Pencil 스프라이트를 이용하며 모양의 중심이 연필 축 끝이 되도록 이동한다.
- 정오각형을 3번 반복적으로 그린다.
- 3개의 정오각형의 시작점은 모두 화면의 중심인 좌표 (0, 0)이다.
- 정오각형을 그리는 방법은 현재 방향으로 한 선분(변)을 그리고 시계 방향으로 72도 회전을 한 후에 다시 선분을 그리고 72도 회전하기를 반복하여 총 5회를 그리면 된다.

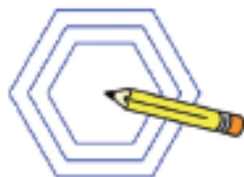


6.3 블록 만들기(9/20)

[예제6.2] 연필 스크립트

연습

다음과 같은 도형을 그리도록 스크립트를 작성하시오.



Pencil 스프라이트의 스크립트

설명



정오각형(변의 길이) 정의하기
그리기를 위한 펜 내리기를 한다.
다음 절차를 5회 반복한다.
 변의 길이만큼 움직여서 한 변을 그린다.
 정오각형을 위하여 72도 회전한다.
그리기를 마치기 위하여 펜을 올린다.



시작하기 버튼을 클릭했을 때
원점 (0, 0)으로 이동한다.
이전에 그린 선은 모두 지운다.
변의 길이를 위해서 나의 변수를 60으로 정한다.
다음 절차를 3번 반복한다.
 정오각형(나의 변수) 블록을 실행한다.
 나의 변수를 20 증가시킨다.

6.3 블록 만들기(10/20)

정n각형 그리기 블록 정의

- 정n각형을 그리기 위한 회전 각도는 $360/n$ 이다.
- 변을 그리고 회전 각도만큼 방향 바꾸기를 n번 반복 한다.



연습

정n각형을 그리는 블록을 이용하여 정사각형을 그려보자.

연습

다음과 같은 별 모양을 그리는 나만의 블록을 정의해보자.



6.3 블록 만들기(11/20)

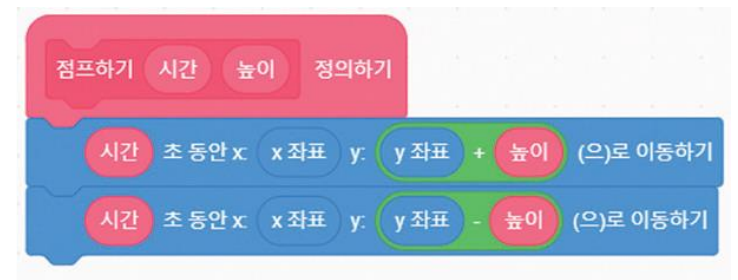
[예제 6.3] 꿈을 피해서 점프하는 고양이

■ 준비 단계

- ① 고양이 스프라이트 외에 곰(Bear-walking) 스프라이트를 추가한다.
- ② 배경은 Forest로 변경한다.



- 시간과 높이를 인수로 갖는 점프하기 블록을 정의
 - 인수로 넘어가는 시간은 점프에서 최고점까지 오르는데 걸리는 시간을 의미한다. 이는 다시 원점으로 돌아올 때 걸리는 시간과 동일하다.
 - 높이는 최고점까지의 거리를 의미한다.



6.3 블록 만들기(12/20)

[예제6.3] 곰 스크립트

- 곰은 화면에서 좌우로 계속 움직인다.
- 벽에 닿으면 튕긴다.
- 만약 고양이에게 닿으면 닿았다 신호를 보낸다.







곰 스프라이트 스크립트	설명
	시작하기 버튼을 클릭했을 때 초기 위치를 (-200, -50)으로 정한다. 크기를 50%로 정한다. 오른쪽 방향 보기를 한다. 다음 절차를 무한 반복한다. 다음 모양으로 바꾼다.(애니메이션 효과) 10만큼 움직인다. 벽에 닿으면 튕긴다. 만약 고양이에게 닿으면 닿았다 신호를 보낸다.

6.3 블록 만들기(13/20)

[예제6.3] 고양이 스크립트

- 스페이스 키를 누르면 점프한다.
- 곶에 닿으면 1초가 사라졌다 나타난다.

고양이 스크립트	설명
	시간과 높이를 인수로 갖는 점프하기 블록 정의하기 시간 초 동안 높이만큼 이동했다가 다시 시간 초 동안 원래 위치로 이동한다.
	시작하기 버튼을 클릭했을 때 회전 방식을 왼쪽-오른쪽으로 정한다. 크기를 60%로 줄인다. 위치를 (200, -50)으로 정한다. 왼쪽을 보게 한다. 화면에 나타낸다.
	스페이스 키를 눌렀을 때 점프하기 블록을 부른다. 0.5초 시간 동안 100을 이동하는 속도로 점프한다.
	달았다 신호를 받았을 때 화면에서 숨겼다(곰과 계속 닿는 것을 피하기 위해서) 1초 지나면 화면에 보이기를 한다.

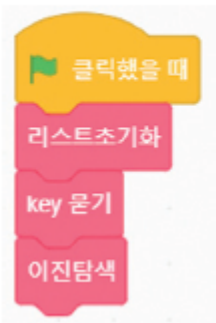


6.3 블록 만들기(14/20)

[예제 6.4] 이진 탐색

- 데이터가 크기에 따라 정렬되어 있을 때는 탐색할 때 순차 탐색 대신에 이진 탐색(Binary Search)을 이용할 수 있다.
- 이진 탐색 과정에서는 first, last, mid, key 등의 변수와 정렬된 데이터를 가지고 있는 리스트를 준비 단계에서 우선 이들을 생성해야 한다.
- 찾고자 하는 데이터를 key 변수가 가지고 있고 값이 5이며
- 정수 리스트의 항목은 2, 4, 5, 8, 10, 11, 14, 18이라고 가정한다.
- 이진 탐색의 반복 회차에 따른 변수의 변화는 다음과 같다.
 - 1회차 first=1, last=8, mid = 버림((first+last)/2) = 버림((1+8)/2) = 4
 - 4번째 항목의 값 8과 key 값 비교, $5 < 8$ 이므로 last = 4 - 1 = 3
 - 2회차 first=1, last=3, mid = 버림((1+3)/2) = 2
 - 2번째 항목의 값 4와 key 값 비교, $5 > 4$ 이므로 first = 2 + 1 = 3
 - 3회차 first=3, last=3 mid = 버림((3+3)/2) = 3
 - 3번째 항목의 값 5와 key 값 비교하여 같으므로 3번째에서 찾는다.

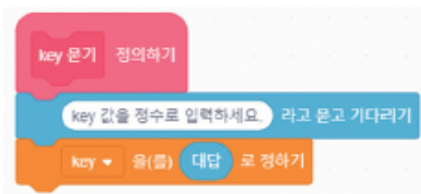
6.3 블록 만들기(15/20)

[예제6.4] 스크립트(1/2)

이진탐색 스크립트	설 명
 A Scratch script starting with a yellow 'when clicked' block, followed by three pink blocks: 'initialize list', 'ask for key', and 'binary search'.	시작하기 버튼을 클릭했을 때 리스트를 초기화하고 key(찾을 값)를 읽어 들이고 이진탐색을 실행한다.
 A Scratch script titled 'initialize list definition'. It starts with an orange block 'erase all contents of 정수리스트', followed by a loop of 18 orange blocks, each adding a value to '정수리스트'.	리스트초기화 블록 정의하기 이전 정수리스트 항목을 모두 삭제한다. 새로운 데이터들을 정수리스트에 추가한다. 데이터들은 크기 순서대로 정렬되어 있어야 한다.
 A Scratch script titled 'ask for key definition'. It starts with a blue block 'ask for key and wait' and an orange block 'set key to answer'.	key 묻기 블록 정의하기 묻고 기다리기를 이용하여 정수를 읽어 들인다. 읽어 들 인 정수를 key 변수에 저장한다.

6.3 블록 만들기(16/20)

[예제6.4] 스크립트(2/2)



key 묻기 블록 정의하기

묻고 기다리기를 이용하여 정수를 읽어 들인다. 읽어 들인 정수를 key 변수에 저장한다.

이진탐색 블록 정의하기

first 변수를 1로 정한다.

last 변수를 정수리스트 크기로 정한다.

first > last가 될 때까지 다음 절차를 반복한다.

mid = 버림((first + last) / 2)으로 정한다.

만약 정수리스트의 mid 번째 항목이 key와 같으면

“mid에서 찾았습니다.”를 2초 동안 말하고 모두 멈추기를 실행한다.

아니면

만약 정수리스트의 mid 번째 항목이 key보다 작으면

first = mid + 1로 정한다.

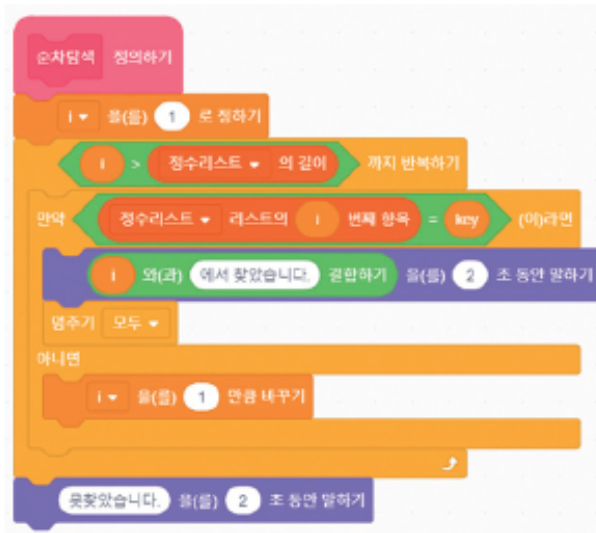
아니면

last = mid - 1로 정한다.

“못찾았습니다.”를 2초 동안 말한다.

6.3 블록 만들기(17/20)

순차탐색 스크립트



순차탐색 블록 정의하기

i 변수를 1로 정한다.

i가 정수리스트의 길이보다 커질 때까지 다음 절차를 반복한다.

만약 정수리스트의 i 번째 항목이 key와 같으면

“i에서 찾았습니다.”를 2초 동안 말한 후

모두 멈추기를 실행하고

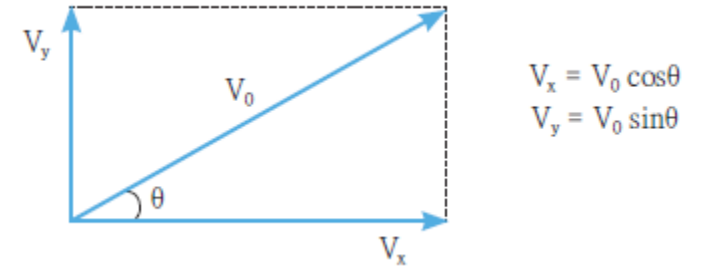
아니면

i 변수를 1 증가시킨다.

“못찾았습니다.”를 2초 동안 말한다.

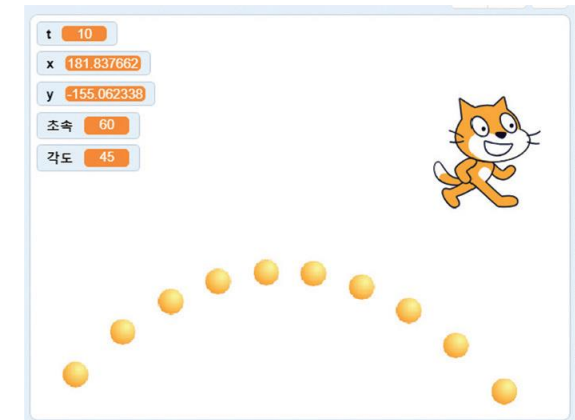
- 이진 탐색은 반복할 때마다 탐색해야 할 범위(first에서 last까지)가 이전 범위의 절반으로 줄어든다.
- 100개의 데이터가 있을 때 한번 반복하면 약 50개로 다시 반복하면 약 25개로 줄어든다. 반복할 때마다 절반씩 줄어서 7번이면 탐색의 결과를 얻을 수 있는 매우 효율적인 알고리즘이다.
- 만약 데이터들이 정렬되어 있지 않으면 순차 탐색으로 찾을 수밖에 없다.

6.3 블록 만들기(18/20)



[예제 6.5] 포물선 궤적으로 공 던지기


- 공을 던지면 지구의 중력이 작용하기 때문에 공의 궤적은 직선이 아니고 포물선의 궤적을 따른다.
- 공을 던질 때 공이 전진하는 각도와 초기 속도에 따라서 포물선의 형태가 정해지게 된다.
- 이외에 중력 가속도 g 가 작용하기 때문에 y 축 방향으로 아래쪽으로 g 만큼의 중력가속도가 계속 가해져서 시간이 감에 따라 아래쪽으로 gt 만큼의 속도 성분이 존재하게 된다. 이로 인해 y 축 방향으로의 속도 변화는 시간이 가면 $V_y - gt$ 가 된다. 이에 따른 이동 거리는 앞의 식을 적분하여 $V_y t - 0.5gt^2$ 이 된다. 여기서 $g = 9.8$ 이다.
- 수평 방향의 이동 거리는 단순히 $V_x t$ 이다.



6.3 블록 만들기(19/20)

[예제 6.5] 스크립트(1/2)


- 준비 단계
 - 사용할 스프라이트
 - ① 고양이 : 초속과 각도를 사용자로부터 묻고 기다리기 블록을 이용하여 들어올려서 공을 던지도록 메시지를 준다.
 - ② 공 : Ball 스프라이트를 이용
 - 초속, 각도 변수를 전역 변수로 생성한다

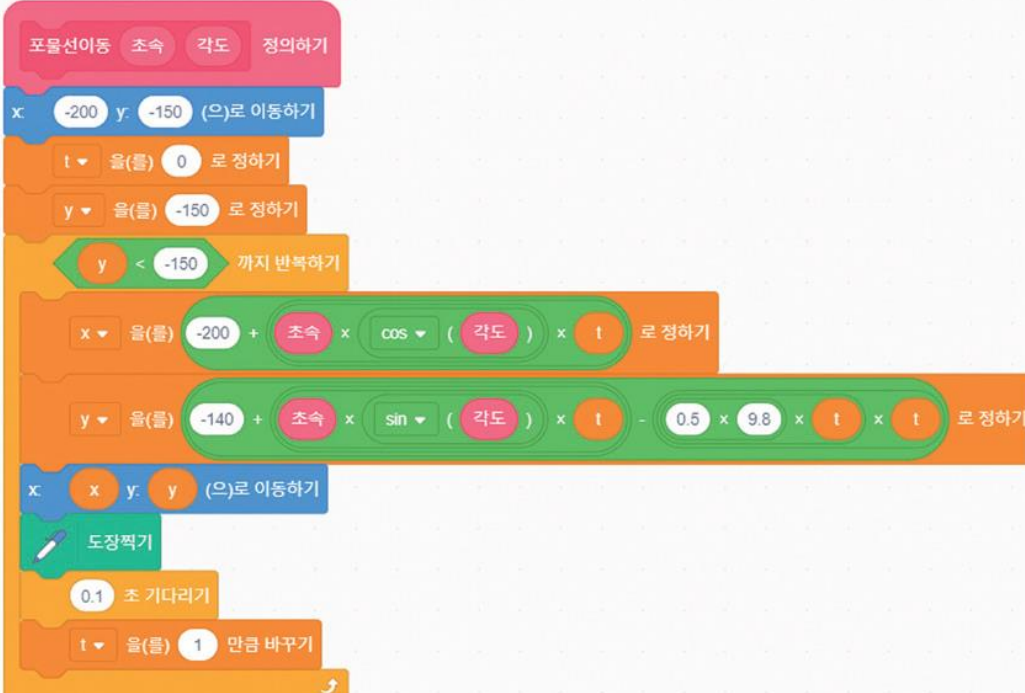
고양이 스크립트	설명
	<p>시작하기 버튼을 클릭했을 때 크기는 원본 크기로 한다. 초속을 묻고 기다린다. 들어오는 대답을 초속 변수에 저장한다. 각도를 묻고 기다린다. 들어오는 대답을 각도 변수에 저장한다. 던지기 신호를 보낸다.(공 스프라이트에게 보내기 위함이다.)</p> <p>던지기 완료 신호를 받았을 때 x 변수를 2초 동안 말한다.(공이 전진한 거리를 알리기 위함이다.)</p>

6.3 블록 만들기(20/20)

[예제 6.5] 스크립트(2/2)

- 공은 초기에 좌표 (-200, -150)에 위치, 시간을 나타내는 t 변수는 처음에 0으로 초기화
- y 좌표는 처음에 -150에서 시작해서 포물선 궤적으로 움직이기 때문에 다시 y 좌표가 -150 위치 아래로 내려오면 종료
- x, y 변수는 시간 t 가 증가함에 따라 움직이는 거리를 계산하여 저장한다. x, y 변수를 공이 이동할 좌표로 정하여 이동한 후 도장 찍기로 화면에서의 위치를 나타냄

공 스크립트	설명
	던지기 신호를 받았을 때 모두 지우기를 실행한다.(이전에 도장 찍기로 그려진 것을 지우기 위함이다.) 포물선이동(초속, 각도) 블록을 실행한다. 던지기 완료 신호를 보낸다.



```
포물선이동 초속 각도 정의하기
x: -200 y: -150 (으)로 이동하기
t: 을(를) 0 로 정하기
y: 을(를) -150 로 정하기
y < -150 까지 반복하기
  x: 을(를) -200 + 초속 x cos ( 각도 ) x t 로 정하기
  y: 을(를) -140 + 초속 x sin ( 각도 ) x t - 0.5 x 9.8 x t x t 로 정하기
x: x y: y (으)로 이동하기
도장찍기
0.1 초 기다리기
t: 을(를) 1 만큼 바꾸기
```


6.4 재귀적 호출(1/7)

- 재귀적 호출(recursive call) : 함수가 자신을 호출하는 것을 의미
- 재귀 호출은 자기 자신을 호출하기 때문에 무한 반복되는 구조가 될 수 있음. 이를 피하기 위해 적절한 조건이 만족 되면 자기 호출을 피하는 논리 구조가 필요



- 말하기 블록은 인수로 n 값을 읽어 들어서 n 값이 0보다 크면 n 값을 1초 동안 말한다. 다음 인수로 n-1을 넘기며 다시 자기 자신을 호출한다. 이때 현재 인수를 1 감소시켜서 자신을 호출하면 반복적으로 자신을 호출하면서 인수의 값은 1씩 감소하게 되고 언젠가는 n이 0 이하가 되게 된다. 그러면 반복은 멈춘다.
- 스크립트는 3부터 역순으로 1까지 말하는 프로그램이 된다.



- 반복 구조를 이용하여 스크립트 수정

6.4 재귀적 호출(2/7)

[예제 6.6] $n!$ (factorial)을 계산하는 재귀적인 블록을 정의하고 이를 이용하는 예제 스크립트로 작성하라.

- $n! = 1 \times 2 \times 3 \times \dots \times (n-1) \times n = (n-1)! \times n$ 이다.
- 이 식으로부터 $n!$ 을 구하기 위해서 $(n-1)!$ 을 이용한다는 점에 착안하여 재귀적인 블록을 만든다.
- 예를 들어서 $4! = 4 \times 3! = 4 \times (3 \times 2!) = 4 \times (3 \times 2 \times 1!)$ 이 된다. $1! = 1$ 이므로 재귀호출이 $1!$ 이 되면 결과는 1이고 반복적인 재귀 호출을 피한다.



factorial 블록 정의하기
만약 $n = 1$ 이면
결과는 1로 정하고
아니면(만약 n 이 1보다 크면)
 $n-1$ 을 인수로 재귀호출하고
결과 = 결과 \times n 으로 정한다.

6.4 재귀적 호출(3/7)

[예제 6.6] 스크립트



C 함수를 이용한 $n!$ 계산은 함수가 값을 반환할 수 있기 때문에 결과를 저장할 전역 변수는 필요 없다. 그러나 스크래치에서는 블록이 값을 되돌리지 않기 때문에 전역 변수를 이용하여 재귀 호출에 의한 계산 결과를 따로 저장해야 한다.

시작하기 버튼을 클릭했을 때
계산할 n 을 읽어 들어서
만약 대답이 0 이하가 되면
잘못된 n 이라 알리고 반복적으로 다시 읽어 들인
다.
대답을 인수로 하여 정의된 블록을 실행하고
결과 변수를 2초 동안 말한다.

```
int fact(int n)
{
    if (n <= 1)
        return 1;
    else
        return (n * fact(n-1));
}
```


6.4 재귀적 호출(4/7)

[예제 6.7] 두 수 x, y 를 읽어 들어서 두 수의 최대공약수(GCD : the Greatest Common Denominator)를 구하여라.

- $\text{GCD}(x, y)$ 는 만약 $x < y$ 이면 $\text{GCD}(y, x)$ 이다.
만약 x 를 y 로 나눈 나머지가 0이 아니면 $\text{GCD}(y, x \text{를 } y \text{로 나눈 나머지})$ 이다.
만약 x 를 y 로 나눈 나머지가 0이면 y 이다.
- $\text{GCD}(16, 24)$ 는 $16 < 24$ 이므로 $\text{GCD}(24, 16)$ 이다.
- $\text{GCD}(24, 16)$ 은 24 나누기 16의 나머지가 8이므로 $\text{GCD}(16, 8)$ 이다.
- $\text{GCD}(16, 8)$ 은 16 나누기 8의 나머지가 0이므로 8이 된다.
- 준비 단계
 - 첫 번째 자연수, 두 번째 자연수, 결과를 위한 변수를 생성한다.

6.4 재귀적 호출(5/7)

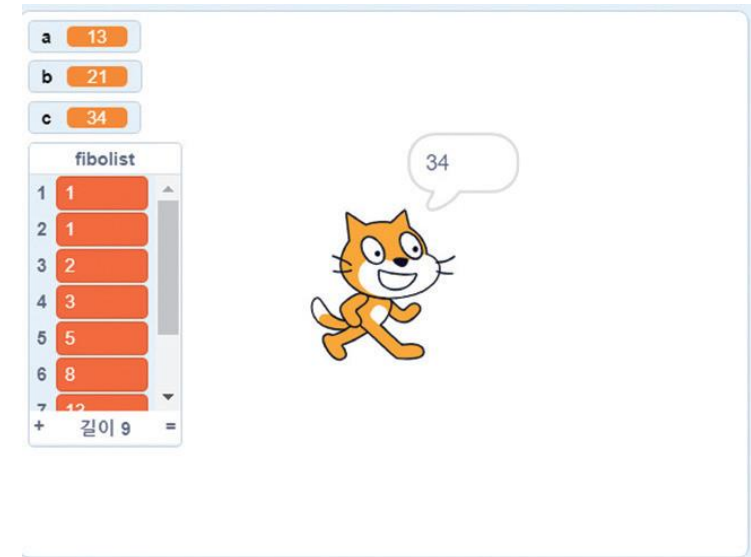
[예제 6.7] 스크립트

최대공약수 스크립트	설 명
	GCD 블록 정의하기 인수는 자연수 x, y 이며 만약 $x < y$ 이면 GCD(y, x) 블록을 실행한다. 아니면 x 를 y 로 나눈 나머지를 $나$ 의 변수에 저장한다. 만약 $나$ 의 변수가 0이면 결과는 y 가 되고 아니면 GCD($y, 나$ 의 변수) 블록을 실행한다.
	시작하기 버튼을 클릭했을 때 묻고 기다리기 블록으로 첫 번째 자연수를 읽어 들어서 첫 번째 자연수 변수에 저장한다. 묻고 기다리기 블록으로 두 번째 자연수를 읽어 들어서 두 번째 자연수 변수에 저장한다. GCD(첫 번째 자연수, 두 번째 자연수) 블록을 실행한다. 결과 변수를 2초 동안 말한다.

6.4 재귀적 호출(6/7)

[예제 6.8] 피보나치 수열

- 앞선 두 수의 합으로 다음 수가 결정되는 규칙을 갖는 수열을 피보나치(Fibonacci) 수열이라 한다. 맨 처음 두 수는 1이다. 이를 나열해보면 다음과 같다. 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...
- 즉, $a_0 = a_1 = 1$, $a_n = a_{n-1} + a_{n-2}$ ($n \geq 2$ 인 정수)이다.
- 수열을 계산하여 스프라이트가 말하기로 보여주면 한 번에 수열을 나타내기 어렵기 때문에 수열을 계산하여 리스트에 넣도록 프로그래밍



6.4 재귀적 호출(7/7)

[예제 6.8] 스크립트



먼저 수열을 저장하기 위해서 fibolist 리스트를 생성한다.

단계 1 이전에 계산된 fibolist를 비우기 위해서 삭제하기를 실행한다.

단계 2 $a = 1$, $b = 1$ 로 정한 후, fibolist에 a , b 를 순서대로 추가한다.

단계 3 fibo 블록을 실행한다. 이때 두 인수 a , b 를 넘긴다.

fibo 블록(인수 a , b) 스크립트에서의 처리 단계는 다음과 같다.

단계 1 $b < 100$ 이라면

$c = a + b$ 를 계산한 다음, 이를 fibolist에 추가한다.

단계 2 그리고 fibo(b , c) 블록을 호출한다.