2002 서울대학교 과학 영재 센터

정보분과
순환교육 교재 1.

차례

5/11 수업 학습목표

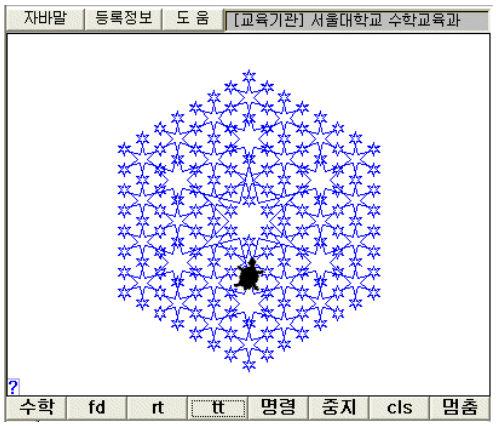
로고 기본 명령, 반복 명령, 함수와 변수 명령을 배운다. 나아가서 주어진 그림과 대응되는 명령의 관계를 생각하면서, 명령어로 생성되는 LOGO GROUP 의 성질을 통해 사고 방식 및 대수학적 개념을 탐구한다.

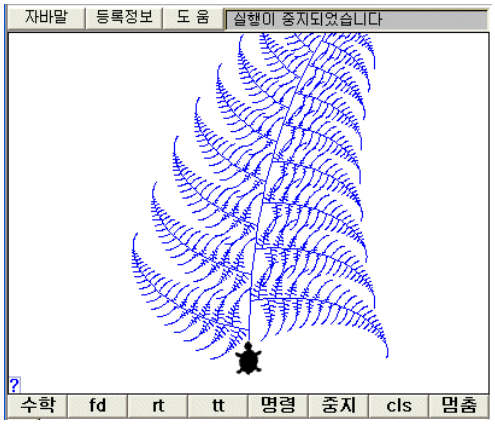
※ 기본 명령Bruner의 EIS 이론이라 불리우는 것은 지식의 기본 원리나 구조 자체를 세 가지 표현 양식, 즉 활동적(Enactive) 표현, 영상적(Iconic) 표현, 상징적(Symbolic) 표현으로 제시하여 발달의 어느 수준에서도 학습이 가능하도록 하는 것이다. 수학적 내용을 자바말로 표현할 때 이 다양한 표현 양식의 적용이 가능하다. 예컨대, 정사각형이라는 수학적 개념을 학교 수학의 전통적인 개념인 '네 변과 네 각이 같은 사각형'이라는 정의가 아니라 아래 표와 같이 자바말의 LOGO 환경을 통한 학습 과정에서 세 가지 표현 양식을 적용하여 구현할 수 있다. 도형은 변과 각으로 이루어져 있다. 거북기하는 도형의 변을 가자로 각을 돌자로 표현하여 이해하는 것이다. Bruner의 이론을 근거로 할 때, 자바말 마이크로월드를 이용한 기존의 기하 교육과정에 나오는 개념과 정리를 움직이는 기하환경이 첨가된 거북기하 환경을 통해 제시할 때 학습자의 조작-탐구를 통한 학습이 크게 증진될 수 있다.

Bruner	활동적(Enactive) 표현	영상적(Iconic) 표현	상징적(Symbolic) 표현
LOGO	앞으로 걸어간 후 왼쪽으로 90도 도는 것을 네 번 반복한다.		반복 4 {가자 10 ; 돌자 90}

Ⅰ. LOGO 기본 명령 (가자, 돌자, 반복, 함수, 변수)

LOGO 는 기본적인 명령어 몇 개만을 이용해 다양한 그래픽을 가능하게 함으로써, 탐구할 수 있는 다양한 소재를 제공해 준다. 다음의 그림의 모두 LOGO에서 기본적인 명령만을 이용해 구현한 것이다.





실제로 이 부분은 프랙탈과 관련된 것으로 다음주에 학습하게 된다. 이러한 것을 구현하기 위해서 먼저 기본적인 명령의 사용법을 알아야 한다. 따라서 LOGO 탐구에 앞서서 기본적인 명령어에 대하여 간단히 알아보자.

1. 기본 버튼 설명

-가자 (발자국 수) : 거북이가 지정한 **발자국의 수** 만큼 **앞으로** 나아간다.
-돌자 (각도) : 거북이의 몸의 방향을 **지정한 각도** 만큼 **돌린다**.
-찍자 (좌표) : 거북이가 지정한 좌표에 점을 찍는다.
-거북 (색깔) : 거북이의 색깔이 청색, 흑색, 적색, 황색, 녹색으로 바뀐다.
-거북 (좌표) : 거북이가 지정한 좌표로 옮겨간다.
-cls : 화면을 청소한다.

위에서 설명된 기본 버튼을 명령어로 직접 명령 입력창에 입력함으로써 거북이에게 지시를 내릴 수 있다.

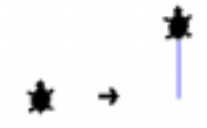
★ 거북이는 항상 현재의 자신의 **상태**를 기준으로 명령을 받아들인다. 거북이의 **상태** 라고 하는 것은 **자신의 위치**와 **머리의 방향**으로 결정된다.
위에서 배운 **가자** 명령을 통해서 **거북이의 위치**를 바꾸어 줄 수가 있고 **돌자** 명령을 통해서 거북이의 **머리의 방향**을 바꿀 수 있다. 따라서 가자와 돌자는 거북이의 상태를 결정짓는 중요한 두 가지 요소이다.
거북이 상태의 초기값은 (0,0) 과 북쪽 방향이다.
즉 초기상태에서 거북이는 원점에 위치해 있으며 거북이의 머리는 북쪽을 향하고 있다는 것이다.

★ 가자 명령 다음에 숫자를 입력할 때, 양수는 앞으로 이동을, 음수는 뒤로 이동을 나타낸다.

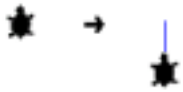
★ 돌자 명령 다음에 숫자를 입력할 때, 양수는 거북이의 왼쪽(반시계 방향)으로 회전을, 음수는 거북이의 오른쪽(시계 방향)으로 회전을 나타낸다.
(방향의 기준을 생각하기 어려우면 자신이 거북이의 등에 타고 있다고 생각하고 그 때를 기준으로 방향을 생각하면 된다.)

★ 거북 명령을 입력할 때에는 명령어와 숫자 사이에 한 칸을 반드시 띄워준다.
그리고 여러 가지 명령을 내릴 경우, 명령과 명령의 사이는 **;** 를 표시해 줌으로써 명령을 구분해준다.

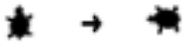
【예】

가자 30; 

- 앞으로 30 만큼 이동한다.

가자 -20; 

- 뒤로 20 만큼 이동한다.

돌자 90; 

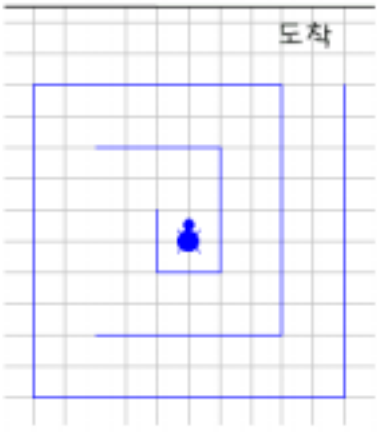
- 왼쪽(반시계 방향)으로 90도 만큼 이동한다.

돌자 -120; 

- 오른쪽(시계방향)으로 120도 만큼 이동한다.

【문제】

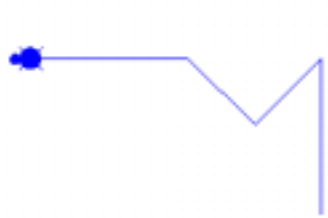
거북이가 다음과 같은 미로를 빠져 나갈 수 있도록 명령을 내려 보자.



【문제】

알파벳 대문자 M을 그리기 위하여 다음과 같이 거북이에게 명령하였더니 결과가 그림과 같다. M자가 제대로 그려지도록 잘못된 부분을 찾아서 고쳐 보자.

```
가자 50; 돌자 135;
가자 30; 돌자 -90;
가자 30; 돌자 45;
가자 50;
```



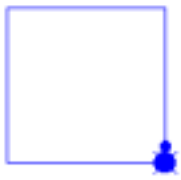
2 . 기본 도형 그리기

지금까지 기본적인 가자, 돌자 명령을 기본적으로 익혀 보았다. 이것을 이용해 기본적인 도형을 그리는 명령을 해 볼 수 있다.

정사각형을 그리는 과정을 생각해 보면 다음의 명령을 내릴수 있다.

① 한 변의 길이 만큼의 선분을 그린다. (첫번째 변)
② 두 번째 변을 그리기 위해 방향을 바꾼다.
③ 한 변의 길이 만큼의 선분을 그린다. (두번째 변)
④ 세 번째 변을 그리기 위해 방향을 바꾼다.
⑤ 한 변의 길이 만큼의 선분을 그린다. (세번째 변)
⑥ 네 번째 변을 그리기 위해 방향을 바꾼다.
⑦ 한 변의 길이 만큼의 선분을 그린다. (네번째 변)
⑧ 거북이를 원래 상태로 돌리기 위해 방향을 바꾼다.

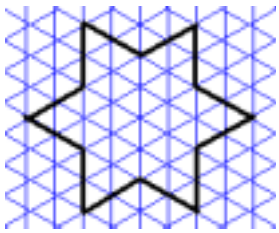
```
가자 50; 돌자 90;
가자 50; 돌자 90;가
자 50; 돌자 90; 가
자 50; 돌자 90;
```



【문제】 반복 명령을 이용하여 원을 그려보자.
(힌트 : 정36각형의 경우 화면에서 보게 되면 원과 같은 모양으로 나타나게 된다.)

【문제】

다음은 정삼각형 모양으로 이루어진 선이다. 그 위에 그려져 있는 별 모양을 그리는 명령을 만들어 보자.(눈금 하나의 크기는 10 이다.)



【문제】

다음의 모양을 그리기 위해 아래의 명령을 실행했다. 그런데 잘못된 부분이 있어 다음과 같은 모양이 만들어졌다. 원래의 모양대로 그림이 그려지게 명령을 고쳐 보자.(거북이의 처음 방향은 북쪽이다.)

입력한 명령

돌자 30; 가자 60; 돌자 120;
가자 60; 돌자 120; 가자 60; 돌자 120;
가자 30; 돌자 120; 가자 30; 돌자 120;
가자 30; 돌자 120;

3 . 반복 명령

위의 정사각형을 그리는 과정을 보면 **가자 50; 돌자 90;** 가 계속 사용되는 것을 알 수 있다. 이렇게 똑같은 패턴을 계속 사용하는 경우에는 **반복** 명령을 쓰면 편리하다.

반복 (반복 횟수){ (반복되는 명령) }

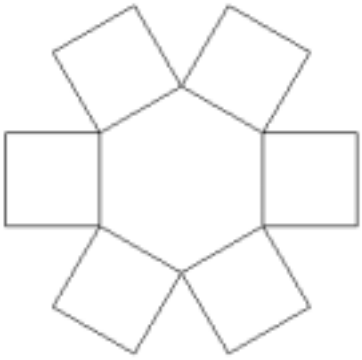
의 형태로 쓰면 된다. 여기서도 반복 명령과 반복 횟수 사이에는 한 칸을 띄워 주어야 하며 반복되는 명령 간에는 **;** 로써 구분을 해 주어야 한다.

위의 정사각형을 그리는 명령을 반복 명령으로 바꾸면 다음과 같이 표현할 수 있다.

반복 4{가자 50; 돌자 90;}

【문제】 반복 명령을 이용하여 다음 그림을 그려 보자.

(이 그림은 정사각형들로 구성된 도형이며 내부의 도형은 정육각형이다.)



4 . 함수

이제는 사각형을 여러번 그려야 할 경우를 생각해 보자.

여러번 그릴 때 마다 반복 4{가자 50; 돌자 90;}를 명령하는 대신 정사각형을 그리는 방법을 거북이에게 기억해 두게 하면 필요할 때마다 그것을 불러서 쓸수 있다.

즉, 함수를 이용한 명령을 할 수 있다.

함수 (함수이름){
(함수로 지정할 명령)
}

이라고 쓰면 거북이는 {} 안의 명령을 ()라는 이름을 가진 명령으로 기억하게 된다. 이것을 '함수를 정의한다.' 고 말한다. 이 경우 함수 이름은 마음대로 정할 수 있다. 그러나 {} 안에 들어갈 명령은 지금까지 배운, 거북이가 알아들을 수 있는 명령이어야 한다. 마찬가지로 명령 사이에는 **;** 로써 구분을 해 주어야 한다. 함수를 정의하는 또다른 방법은,

함수 (함수이름)
(함수로 지정할 명령)
함수끝

에서와 같이 함수를 정의할 때 {}를 사용하는 대신 함수, 함수끝 이라는 말로 대신 하는 것이다.

예를 들어 한변이 50인 정사각형을 그리는 명령을 '정사각형' 이라는 이름으로 기억하도록 하려면

함수 정사각형{
반복 4{가자 50; 돌자 90;}
}

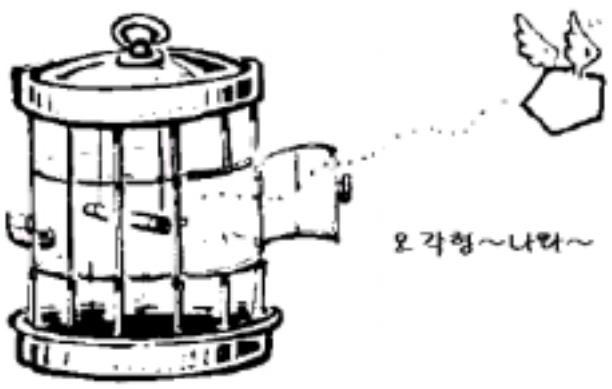
라고 써주면 된다.

지금까지 한 것은 **함수를 정의**하는 일이었다. 함수의 정의는 거북이가 그 함수의 이름에 대응되는 함수를 기억하도록 시키는 것이다. 따라서 그 함수를 실행하라는 명령을 따로 해주

지 않으면 거북이는 단지 그것을 기억만 하고 있을뿐 우리에게 보여줄 수가 없다. 정의된 함수를 부르기 위해서는 함수의 이름을 써주면 된다. 즉,

함수 (함수이름){
(함수로 지정할 명령)
}
으로 함수를 정의한 후

(함수이름);
으로 함수를 부르게 되면 그 함수로 지정된 명령이 실행된다.



정사각형을 그리는 함수를 정의하고 실행하려면

함수 정사각형{
반복 4{가자 50; 돌자 90;}
}
정사각형;

이라고 써주면 된다.

그렇다면 함수를 이용해 크기가 다른 사각형들을 그리는 방법은 없을까?
그러한 작업을 하기 위해서 '변수' 가 필요하다.

함수 (함수이름)(변수){
(변수를 포함한 명령)
}

여기서 변수는 이미 정해져 있는 특정한 값이 아니라 마음대로 바꿀 수 있는 값이다.

함수 정사각형(x){
반복 4{가자 x; 돌자 90;}
}

와 같이 정의하게 되면 거북이는 길이가 x 인 정사각형을 기억하게 된다. 그러나 x 는 정의된 값이 아니므로 함수를 부를 때에는 x 값을 지정해 주어야 한다. 즉,

(함수이름)(x 대신 넣고자 하는 숫자);

로 함수를 불러준다.

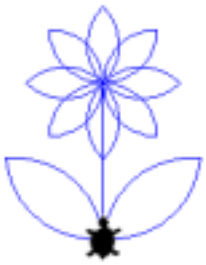
예를 들어 **정사각형(10);** 으로 함수를 부르면 길이가 10인 정사각형이 그려진다. 여기서 x 는 원하는 수로 바꿀 수 있다.

【문제】 다음 그림을 그리기 위해 함수를 사용해서 명령문을 만들어 보자.

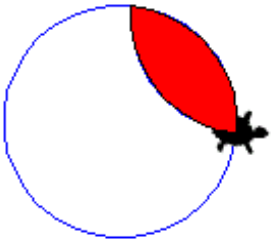
(힌트: 앞에서 그렸던 원의 명령을 이용해서 반원을 그리는 명령을 생각해 보자.
반복 18 {가자 5; 돌자 10;})

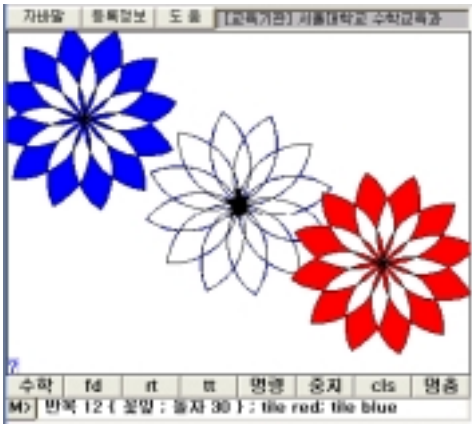
【문제】 변수를 이용한 함수를 사용하여 다음 그림을 그려보자. 여기서 꽃과 잎은 크기가 다르다.

(힌트 : 사분의 일원을 그리는 방법을 생각해 보자.
반복 9{ 가자 5; 돌자 10})



왼쪽의 꽃을 구성하는 작은 단위를 이용해 아래의 꽃그림을 그릴 수 있다.





지금까지 배운 기본 명령을 정리하면 다음과 같다.

가자 x	거북의 머리가 있는 방향으로 x만큼 이동. 가자 명령을 하면 거북의 발자취가 화면에 보인다. 뒷걸음으로 갈 때는 “가자 -20” 과 같이 음의 거리를 대입하면 된다.
돌자 x	수평선을 기준으로 시계반대 방향으로 x도 만큼 회전. 돌자 명령을 하면 거북의 머리 방향이 그 쪽 방향으로 옮겨간다. 시계방향으로 회전하고 싶으면, “돌자 -30”과 같이 음의 각을 대입하면 된다.
반복 x	같은 명령을 반복할 때 사용한다. 예를 들면, 정삼각형을 그릴때 “반복 3 {가자 30 ; 돌자 120}”을 쓸 수 있다.

.

함수	함수를 정의할 때 하는 명령이다. “함수”로 시작해서 “함수끝”이라는 명령을 끝으로 한다. 또는 함수{ }으로 명령한다.
거북 x,y	x,y의 위치로 거북을 이동하라는 명령이다. 이 명령을 하면 거북이 발자취를 남기지 않고 이동한다.
가자	“가자” 명령을 입력한 후, “실행”버튼을 누르고, “가자” 버튼을 다시 누르면 거북이 발자취를 남기지 않고 이동한다. 다시 거북의 자취를 남기고 싶을 때는 다시 한번 “가자” 명령을 명령 입력창에 입력한 후, “실행”버튼을 누르고, 다시 “가자” 버튼을 누르면 된다.

Ⅱ. LOGO GROUP

LOGO 는 수학적으로 생각했을 때 특정한 구조를 가진다.
그 구조를 파악하기 위해서 먼저 구조를 이루는 '대상'이 되는 것과, 그러한 대상간의 '연산'을 생각해보자.

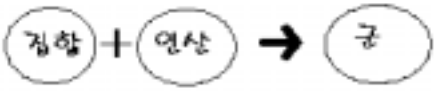
1. 대상과 연산을 갖는 LOGO GROUP

LOGO 는 단순히 그림만을 그리는 도구가 아니다. LOGO를 살펴보면 그 안에서 특정한 구조를 가지고 있다는 것을 알 수 있다.

LOGO 의 구조를 살펴 보기 전에 유리수의 집합(Q)과 덧셈(+)을 예로 들어 생각해 보자.

① 어떤 두 유리수를 더해도 그 값은 다시 유리수이다.
즉, $a, b \in Q$ 인 a, b 에 대해 $a+b \in Q$ 이다.
② 결합 법칙이 성립한다.
즉, $a, b, c \in Q$ 인 a, b, c 에 대해 $(a+b)+c=a+(b+c)$ 이다.
③ 어떤 수에 더해서 원래의 수를 그대로 나오게 하는 수(항등원) 이 존재한다. 실제로 0이 이 조건을 만족한다.
즉, $\forall a \in Q$ 인 a 에 대해 $a+e=e+a=a$ 인 $e \in Q$ 가 존재한다.
④ 어떤 수에 더해서 항등원이 나오게 하는 수(역원) 이 존재한다. 실제로 a 에 대한 역원은 $-a$ 이다.
즉, $\forall a \in Q$ 인 a 에 대해 $a+x=x+a=e$ 인 $x \in Q$ 가 존재한다.

위에서 보는 것처럼 이 네 가지를 만족하는 것을 군(GROUP)이라고 한다. 즉 유리수의 집합은 덧셈에 관하여 군을 이룬다. 이처럼 '군'을 정의하기 위해서는 대상이 되는 **집합** (위의 예에서는 유리수이다.)과 **연산**(위의 예에서는 덧셈이다.) 이 있어야 한다.

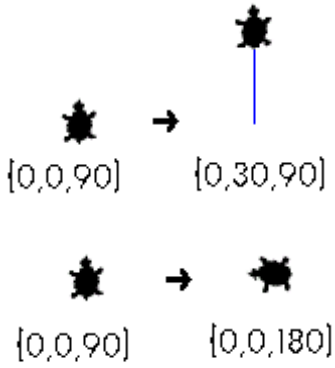


동일하게 LOGO 도 GROUP이 정의 된다.
LOGO에서 대상이 되는 것은 거북이의 상태를 변화시키는 **명령어({가자 A},{돌자 B})의 집합**이다. 그리고 연산에 해당하는 것은 대상인 명령어들을 **합성하는 것**이다. 그러면 다음과 같은 사실을 알 수 있다.

대상 : 유리수의 집합, 연산: 덧셈 ➜ 군
대상 : {가자 A},{돌자 B}의 명령어 집합 연산 : 명령어의 합성 ➜ LOGO GROUP
이 구조를 좀 더 쉽게 파악하기 위해 거북이의 상태를 다음과 같이 생각 할 수 있다.

(x,y,h) : 여기서 x 는 거북이 위치의 x 좌표, y 는 거북이 위치의 y 좌표, h 는 거북이의 머리가 향하는 방향의 각도(screen 의 가운데에 해당하는 부분이 원점이고, 북쪽을 향하고 있을때가 90도이다.) 초기상태를 이렇게 표현하면 (0,0,90) 이 된다. 그러면 가자와 돌자는 다음과 같이 나타낼 수 있다.

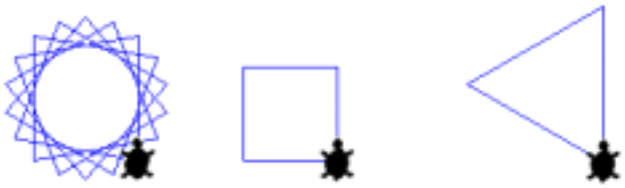
{가자 a} : (x,y,h)→(x+bsinh,y+bcosh,h)
{돌자 a} : (x,y,h)→(x,y,h+a)



즉, 가자는 거북이의 위치를 (x,y)→(x+bsinh,y+bcosh)로 변화 시키고 돌자는 거북이의 방향을 (h)→(h+a)로 변화 시키는 것을 알 수 있다. 여기서 다음과 같은 사실들을 알 수 있다.

① 거북이의 모든 상태 변화는 가자 a 와 돌자 b 로 표현할 수 있다.
② 연산에 있어서 결합 법칙이 성립한다. 여기서 연산은 {가자 A}와 {돌자 B}의 합성으로 정의했다. 예를 들어 g_1: {가자 50; 돌자 90} g_2:{가자 70; 돌자 30} g_3:{가자 -40} 이라면
$(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$ 가 성립함을 알 수 있다.
③ 항등원이 되는 원소는 가자 와 돌자를 이용해서 상태에 변화가 없도록 만드는 것이다. 간단한 예로 {가자 0} 또는 {돌자 0} 이 있다.
즉 g_1:{가자 50; 돌자 30}, g_2:{가자 0; 돌자 0} 을 하게 되면 $g_1 \circ g_2 = g_1$ 이 되므로 g_2 는 거북이의 상태변화에 영향을 주지 않는다. 역원이 되는 원소는 거북이의 상태를 처음으로 되돌려 놓을 수 있도록 만드는 것이다. 즉 내린 명령을 다시 반대로 내려주는 것이다. g_1 : {가자 50; 돌자 30} 의 경우 반대로 돌자 90을 하고 가자 50을 하면 원래 상태로 돌아오게 된다. 따라서, g_1^{-1}:{돌자 -30; 가자 -50} 이 된다는 것을 알 수 있다.{가자 a} 의 역원은 {가자 -a} 이고 {돌자 a} 의 역원은 {돌자 -a} 이다.

④ 각 명령어가 다른 그림을 그리더라도 처음과 끝의 거북이의 상태가 동일하다면 그 명령들은 같은 원소로 취급한다. 따라서 다음의 두 원소는 같게 취급된다. (실제로 하나는 정사각형을 그리는 명령이고 하나는 정삼각형을 그리는 명령이다.)
{가자 30; 돌자 90; 가자 30; 돌자 90; 가자 30; 돌자 90; 가자 30; 돌자 90}
= {가자 30; 돌자 120; 가자 30; 돌자 120; 가자 30; 돌자 120}



실제로 {가자 0; 돌자 0} 과 같은 상태로 만들어 놓는 모든 원소는 항등원과 같다. 즉 동일한 항등원을 표현하는 방법은 여러 가지인 것이다. 이것은 유리수의 덧셈에 대한 연산에서 항등원 0을 0=3-3=-5+5 등 여러가지로 표현할 수 있는 것과 같다.

【문제】
LOGO GROUP에서 {가자 0; 돌자 0} 과 다른 그림을 그리지만 같은 원소에 해당되는 명령을 찾아서 써보자.

【문제】
다음과 같은 그림을 그리려고 한다. 그런데 아래의 명령을 입력했더니 잘못된 그림이 나왔다 잘못된 곳을 찾아 고치고 올바른 그림을 그려보자.

입력한 명령

```
함수 거미줄{
반복 9{호; 돌자 40;}
}

함수 호{
반복 20{가자 4; 돌자 4;}
반복 20{가자 -4;돌자 -4; }
}

거미줄;
```

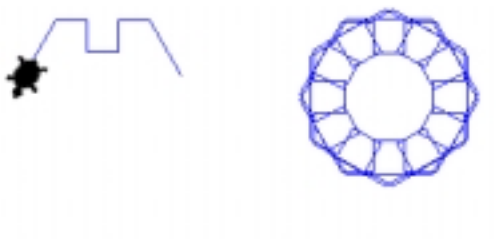
<올바른 그림>

< 잘못된 그림>

2 . 반복 명령과 위수

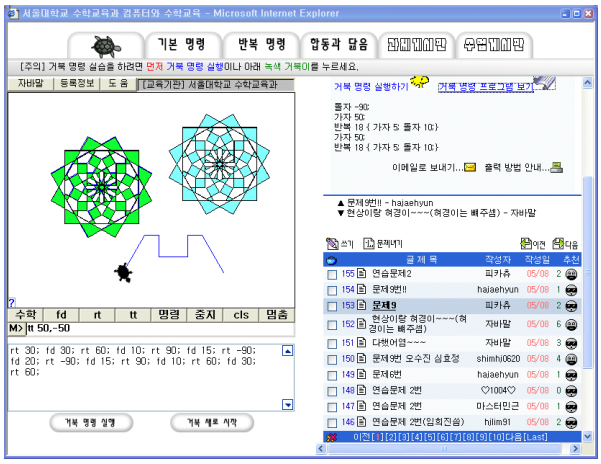
위의 과정에서 어떤 동일한 명령을 계속 사용할 때 반복을 사용할 수 있음을 배웠다. 그렇다면 어떤 동일한 명령을 몇 번 반복해서 거북이가 원래의 위치로 돌아오게 하는 것을 생각해 볼 수 있다. 여기서 원래의 위치로 돌아오기 위해 반복해야 하는 회수를 위수라고 한다. 실제로 일정한 명령을 계속 반복시키게 되는 경우 거북이는 특정한 원 상을 움직이게 된다. (이것은 뒤에서 자세히 다룰 것이다.) 따라서 명령 한번에 이동하는 각도를 t 라면 nt=360k 를 만족하는 정수 n, k 가 존재할 때 거북이는 원위치로 돌아오게 된다. 실제로 이것을 만족하는 최소의 자연수 n 이 위수가 되고 그 값은 t 와 360 의 최소공배수를 t 로 나눈 값이다. 그런데 모든 경우에 그러한 n 이 존재하는 것은 아니다. t 가 0 일 경우, 즉 방향 회전 없이 가자 명령만 나열된 경우에는 원래의 위치로 돌아올 수 없다. 즉 $n\times 0=360k$ 를 만족하는 값을 찾을 수 없다. 또한 t 가 무리수 일 경우에도 $n\times t=360k$ 를 만족하는 값을 찾을 수 없다. 즉 그 명령을 몇 번을 반복하더라도 제자리로 돌아올 수 없다. 이러한 경우 우리는 무한 위수를 갖는다고 말한다.

【문제】
왼쪽의 명령으로 오른쪽의 모양을 만들 때 명령의 위수는 얼마인지 생각해 보자.



왼쪽의 명령은 다음과 같다.

돌자 30; 가자 20; 돌자 60; 가자 10;
돌자 90; 가자 10; 돌자 -90; 가자 10;
돌자 -90; 가자 10; 돌자 90; 가자 10;
돌자 60; 가자 20;



【문제】
위에서 위수는 t 와 360 의 최소공배수를 t 로 나눈 값이라고 설명되어 있다. 그 이유가 무엇일까?

3. SUBGROUP

위에서 우리는 LOGO GROUP을 명령어 전체를 원소로 갖는 것으로 정의했다. 그 중의 특정한 명령어만을 원소로 해서 만들어지는 군을 부분군이라고 한다.

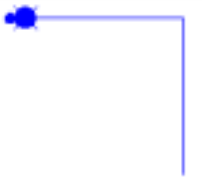
예를 들어 {가자 a}만으로 구성된 군의 경우,
① 그런 명령들을 합성해도 {가자 a} 의 꼴로 나타낼 수 있고
② 항등원은 {가자 0} 이고 {가자 a}의 역원은 {가자 -a}가 된다.

{돌자 b}만으로 구성된 경우도 부분군으로,
① 그런 명령들을 합성해도 {돌자 b} 의 꼴로 나타낼 수 있고
② 항등원은 {돌자 0} 이고 {돌자 b}의 역원은 {돌자 -b}가 된다.

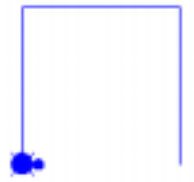
부분군 중에서 특별히 어떤 명령의 반복으로 모든 원소를 표현할 수 있을 때 그것을 순환군이라고 한다. 그리고 그 명령을 반복해서 항등원을 만들 수 있을 때 그 때의 반복회수가 위수가 된다.
예를 들어 {가자 50; 돌자 90}의 반복으로 만들어지는 군을 생각해 보자.

초기 거북 상태 (0,0,0)

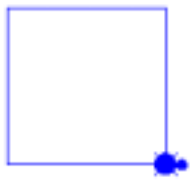
{가자 50; 돌자 90} 거북 상태 (0,50,90)



반복 2 {가자 50; 돌자 90} 거북 상태 (-50,50,180)



반복 3 {가자 50; 돌자 90} 거북 상태 (-50,0,270)



반복 4 {가자 50; 돌자 90} 거북 상태 (0,0,360)=(0,0,0)

이므로 반복 4번만에 원래 상태로 돌아오게 된다. 따라서 이 순환군의 위수는 4이다.

【문제】특정 명령어의 반복으로 모든 원소를 표현 할 수 있는 순환군을 생각해보자.(위에서 예로 들고 있는 **{가자 50; 돌자 90}의 반복으로 만들어지는 군**도 순환군중 하나이다.)

Ⅲ. LOGO 탐구

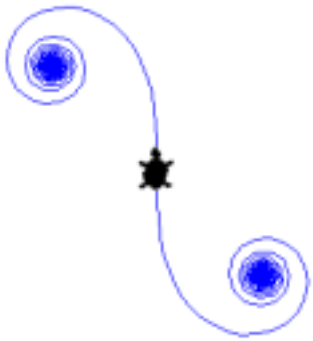
앞에서 얘기한 LOGO GROUP 의 위수에 대해 생각해 보자.
가자 x ; 돌자 y 명령을 반복하여 실행시키면 무한 위수를 갖는 경우와 유한 위수를 갖는 경우가 생긴다.
LOGO GROUP에서 성립하는 다음의 정리를 살펴보자.

(CLOSED PATH THEOREM)
모든 closed path(닫혀 있는 경로)에 대해서, 그 전체 경로를 따라 여행할 때 회전한 각도는 360의 배수이다.

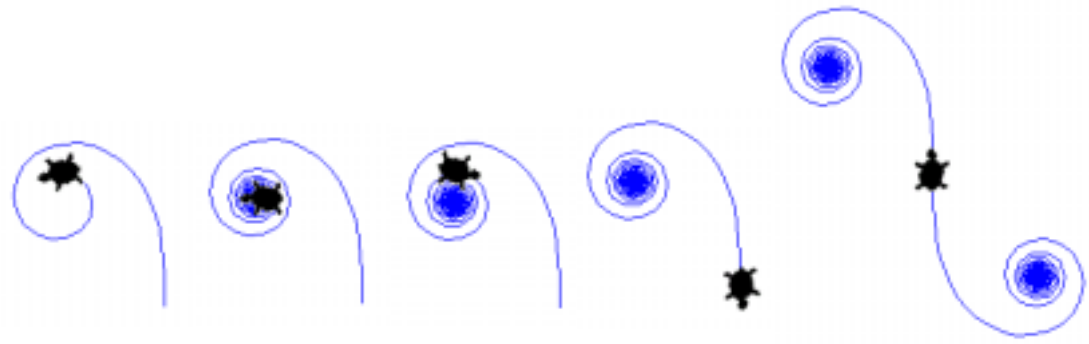
1. CLOSED PATH

【예】 다음 그림이 닫힌 그림을 그리는 이유를 생각해 보자.

```
for i=1 to 720; fd 10; rt i; next;
```



거북이의 중간 단계를 생각해 보자.

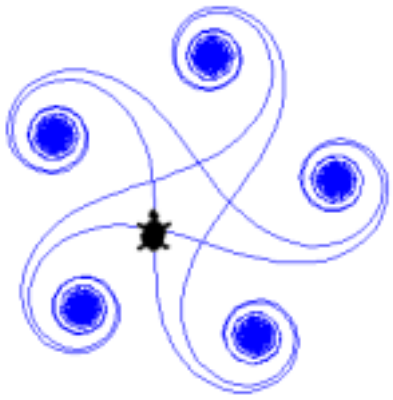


돌자 30　　돌자 180　　돌자 330　　돌자 360　　돌자 720

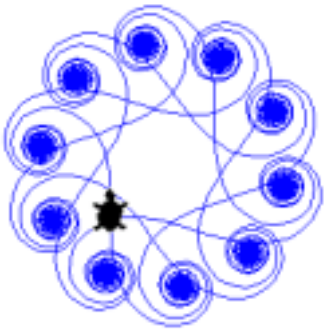
거북이의 방향을 잘 보면 반 시계 방향 회전이 양의 각, 시계 방향 회전이 음의 각이고 반 시계 방향으로 a 회전한 것은 시계방향으로 180-a 만큼 회전한 것과 같다. 즉 180도가 되기 전까지는 거북이가 도는 각도는 1,2,…178,179,180 이런 순서이지만 180을 넘어서 181,182,…359,360 이 되면 이 때의 회전은 -179,-178,…-1,0 으로 회전한 것과 같다. 즉 180도를 넘은 뒤 회전 방향은 지금까지 온 방향과 반대이다. (돌자 명령으로만 따지면 역원에 해당한다.) 따라서 거북은 지금까지 온 길을 그대로 다시 돌아 나가게 된다.(머리의 방향은 반대로 유지된다.) 360도 이후에는 다시 그 과정이 반복 된다고 할 수 있다.

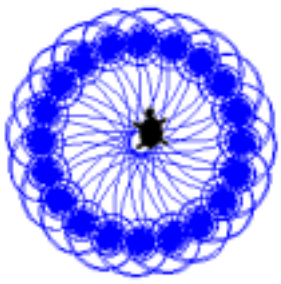
【문제】 다음의 그림이 닫힌 그림을 그리는 이유를 제시하여라. 이 그림이 위의 예제와 다른 모양으로 그려지는 이유는 무엇인가?

```
for i=1 to 1800; fd 10; rt i+0.1; next;
```



【문제】 일반적으로 이러한 형태가 n번 나타나게 하려면 명령을 어떻게 바꾸어야 하는가?

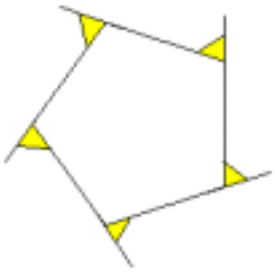




【문제】 위의 형태와 다르게 다음의 그림이 그려지는 이유를 생각해 보자.

for i=1 to 1080; fd 10; rt i+0.5; next;

2 . 도형의 외각 맞추기



왼쪽 그림은 정오각형이다.

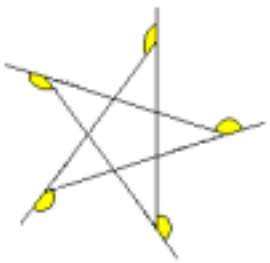
노랗게 색칠한 정오각형의 다섯 개의 외각들의 합은 360°이다. 따라서 정 n 각형의 한 외각의 크기는 360°/n 이다. 왜냐하면 거북이가 정n 각형을 그릴 때 360°의 각을 n번에 걸쳐 회전하게 되기 때문이다.

즉 거북이가 매번 회전해야 하는 각을 A 라고 하면

$n \times A = 360$ 이라는 식을 세울 수 있다.

따라서 정n 각형을 그리는 명령을 다음과 같이 생각할 수 있다.

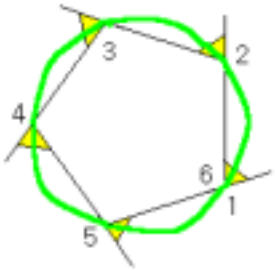
반복 n{가자 30; 돌자 360/n}

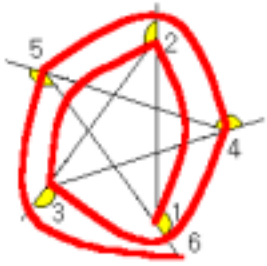


이번에는 별 모양을 관찰 해 보자.

이 그림은 위의 정오각형처럼 다섯 개의 변으로 이루어져 있지만 거북이가 한번 회전할 때 돌게 되는 각이 훨씬 크기 때문에 변과 변이 겹쳐져서 별모양이 되었다.

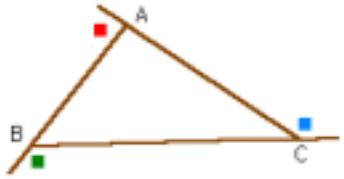
그러나 두 그림 모두 거북이는 그림을 그린 후 원래의 상태(0,0,0)로 돌아왔다. 여기서 두 그림의 회전각은 얼마일까?



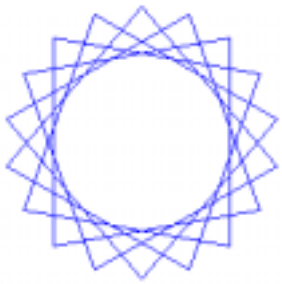


그림에서 나타난 거북이의 자취에서 보듯이 정오각형은 한바퀴를 돌았고 (360°)별은 두바퀴를 돌았다.(720°) 따라서 별의 경우 한번에는 720°/5=144°만큼을 회전하게 된다.

【문제】 삼각형의 내각의 합은 왜 180도 인가?

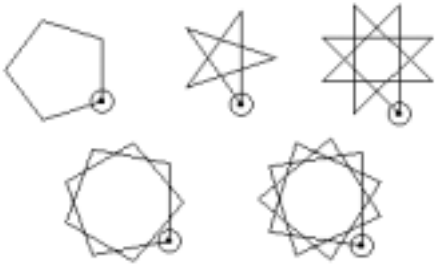


【문제】다음 도형을 그리기 위해서 거북이는 몇도 회전을 했을까?
한번 돌 때 명령에서 몇 도씩 돌도록 만들어주어야 할까?

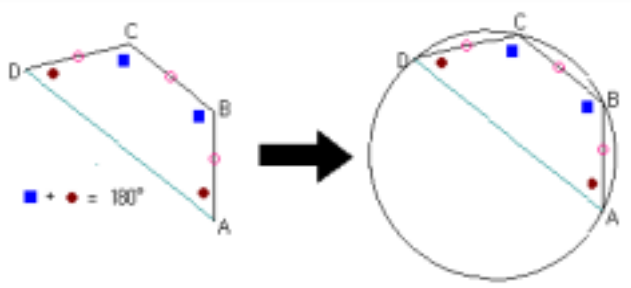


3. 거북이는 동일한 원주 상을 벗어나지 못한다.

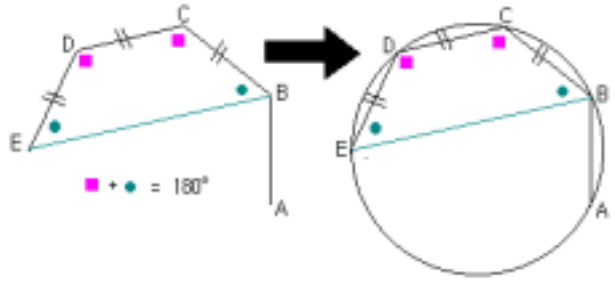
여러 가지 정 다각형을 그려보고 그 꼭지점의 자취를 생각해보자. 이 경우 꼭지점들은 모두 한 원 위에 위치함을 알게 된다. 거북이가 원을 이탈하지 못하는 이유는 무엇인가?



정다각형을 그리는 명령을 생각해 보자. 거북이가 A를 출발한다. 아래의 그림은 거북이가 A에서 B로,B에서 C로,C에서 D로, 이동한 것이다. 이때 A 와 D를 잇는 보조선을 그으면 아래와 같은 사각형이 나온다.



사각형 ABCD 는 마주보는 내각의 합이 180°이므로 이 사각형은 어떤 원에 내접한다. 마찬가지로 사각형 BCDE 도 어떤 원에 내접한다.

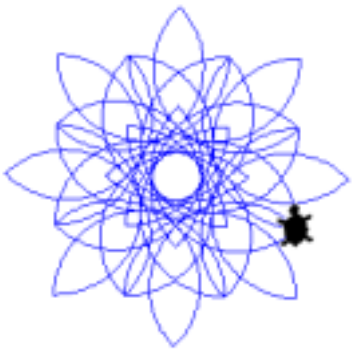


이 두 원이 일치함을 보인다면 거북이가 하나의 원 위를 움직인다는 주장을 할 수 있다. 두 사각형은 세 점 B,C,D를 공유하고 있으므로 그들의 외접원은 일치하게 된다.(왜냐하면 세점은 한 원을 결정하는 요소이기 때문이다.) 따라서 거북이는 하나의 원을 따라 움직이게 된다.

4. 반복되는 명령들은 동일 원주상에 위치한다.

그럼 정다각형을 그리는 명령이 아니라 복잡한 명령을 반복할 경우에는 어떻게 될까? 다음의 예를 보자.

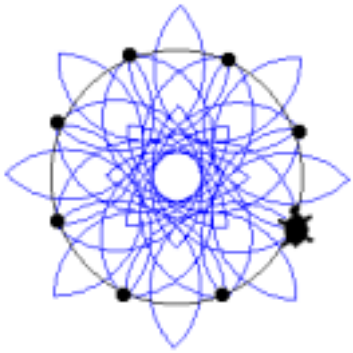
【예】



```
repeat 8 { rt 45;
repeat 6 {
repeat 90 { fd 2; rt 2;} ; rt 90 ;
}
}
```

얼핏 보기에 이 경우에는 동일 거북이가 동일 원주 상에서 이탈하는 것처럼 보인다.

그러나 거북이의 **상태**에 초점을 두면 역시 마찬가지임을 알 수 있다.



즉 **하나의 명령이 끝났을 때의 거북의 상태들을 단위** 로 보면 그 때는 단순히 가자, 돌자로 그 상태를 표현할 수 있게 되므로 위의 논리가 동일하게 적용된다.

【문제】
다음과 같은 그림을 그리게 하는 명령이 있다.
이것을 반복해서 시행할 경우 거북이의 상태 변화를 관찰해 보자.

이 명령은 다음과 같다.

```
함수 fdrtone(x,y,w) { fd x; rt y;fdrtone(x+w,y,w);}
 fdrtone(2,70,0.1)
```

【문제】
명령에서 변수의 값이 변하게 되는 경우에는 어떤가?
다음 두함수의 경우를 생각해 보자.

```
함수 one(x,y,w) { 가자 x; 돌자 y; one(x+w,y,w);}

함수 two(x,y,w) { 가자 x; 돌자 y; fdrt(x,y+w,w);}
```

5. 가자 돌자 명령의 연산이 항등원이 되는 경우의 문제

앞에서 설명했듯이 거북이의 상태 (state) 란 몸의 위치와 머리 방향 두 가지로 이루어져 있다.
다음 함수를 보자.

함수 fdrt(x,y) { fd x; rt y; fdrt(x,y);}

바로 전에 어떤 명령을 계속 반복해 나갈 경우 거북이의 상태를 살펴보면 동일 원 상에 위치한다는 것을 배웠다. 그렇다면 동일 원상일 뿐만 아니라 원래 있던 그 자리로 돌아가기 위해서는 어떤 조건이 만족 되어야 할까?

위에 나온 식에서 nt=360k를 기억해 보자. 즉 한번에 t 만큼씩 회전하는 거북이가 n번을 돌았을 때 완전한 원한바퀴의 배수 만큼을 돌게 된다는 것이다. 따라서 거북이가 도는 각도가 유리수여야 함을 알 수 있다.
여기서도 그대로 적용된다. 즉, fdrt 함수는 y가 유리수일때 출발점 상태 (위치및 머리 방향이 출발점과 같음)로 돌아오게 된다. (이 때, 거북은 출발점에 다시 오기까지 360의 배수 만큼 몸을 회전시킨다.) 또한 역으로, fdrt 함수에 의해 거북이가 360의 배수 만큼 몸을 돌리는 그 순간 거북이는 출발점에 와 있게 된다.

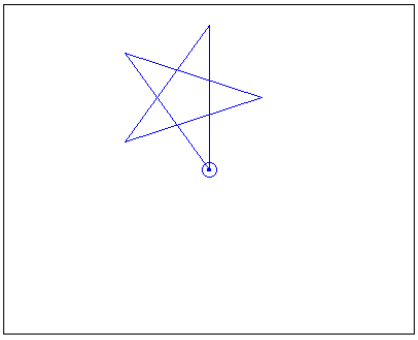
fdrt(10,30*pi) : 무리수일 경우 반복하면 할수록 빈 곳이 채워진다.

fdrt(10,70) : 유리수일 경우 빈 곳을 다 채우지 않고 일정 궤도로만 이동한다.

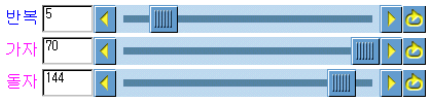
반복 a { 가자 b ; 돌자 c } 탐구 : 중학교 수학에서 이차함수 ax^2+bx+c 의 그래프가 주요한 학습 대상물이다. 학생들은 여기서 각각의 계수와 방정식의 근과의 관계 및 그래프의 모양 등을 학습하게 된다.

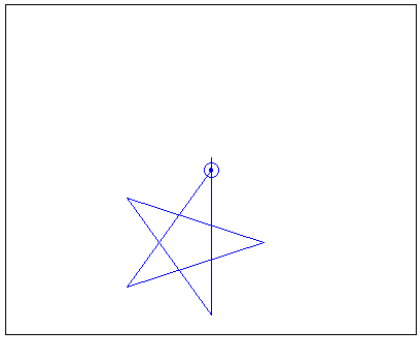
LOGO에서도 동일한 구조로 학습할 수 있다.

반복 a { 가자 b ; 돌자 c }

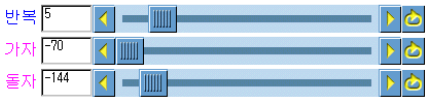


반복 5 { 가자 70 ; 돌자 144 }





반복 5 { 가자 -70 ; 돌자 -144 }

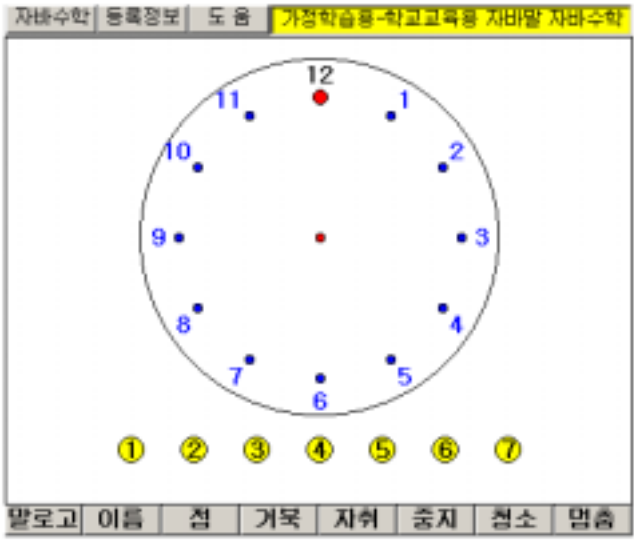


6. 시계대수

· 거북이는 0으로 다시 돌아올 수 있는가?
· 돌아올 때까지 거북이는 몇 개의 변을 생성하는가?
· 거북이는 돌아오기까지 시계를 몇바퀴 돌았는가?

거북이는 아래 그림과 같은 시계에서 12시 위에 앉아 있다. 거북이는 12시 지점을 출발해 여행을 하려고 한다. 12시 위에서만 쉴 수 있으며 아른 시각에 도착하면 곧바로 떠나야 한다.

거북이는 2시간씩 건너 뛰며 돌 때에는 6번만에 12시에 돌아온다. 4시간씩 건너 뛸때에는 얼마만에 돌아올 수 있을까? 아래의 애플릿의 실험을 통해 확인해보자.



건너뛴 시계의 눈금과 12시에 올때까지 그려지는 변의 개수의 관계를 확인해 보자.
예를 들어 건너뛰는 눈금이 2시간 일 때에는 돌아오기 전까지 6개의 변이 생긴다.

간격	1	2	3	4	5	6	7	8	9	10	11
변의 수		6									

어떤 규칙을 발견할 수 있는가?

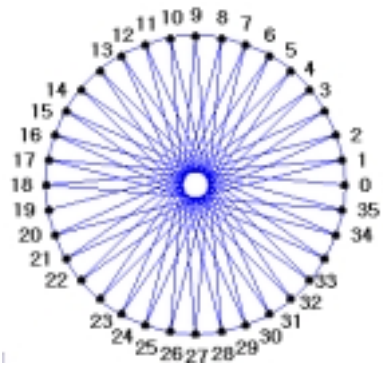
모든 눈금을 다 거쳐서 눈금 12로 돌아오려면 간격을 얼마씩 해야 하는가?

12로 다시 돌아가기 위해서 돌아야 하는 바퀴의 수와 변의 개수,시간 간격 사이에는

어떤 관계가 있을까? 관계식을 생각해 보자.

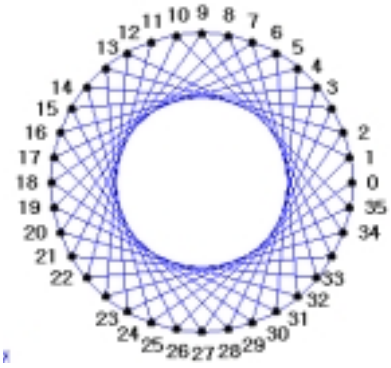
```
CLS; tt; rt 5;
for i=0 to 35 ; point v_i; fd 300/36; rt 10; next
```

명령을 하여 화면에 36개의 점으로 이루어진 시계를 만들어 보자.



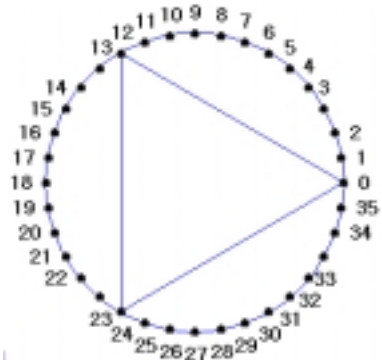
i=17%36;
for j=0 to 35 ;rt v_(i*j)%36; fd v_(i*j)%36 ; next

이 그림의 경우 17칸씩 건너 뛰어가게 된다.(i=17%36 가 의미 하는 것은 17을 36으로 나눈 나머지이다.) 이 경우에 그림으로 보는 것처럼 원래의 지점으로 돌아가기 위해서는 모든 지점을 다 거쳐가야 함을 알 수 있다.



i=61%36;
for j=0 to 35 ;rt v_(i*j)%36; fd v_(i*j)%36 ; next

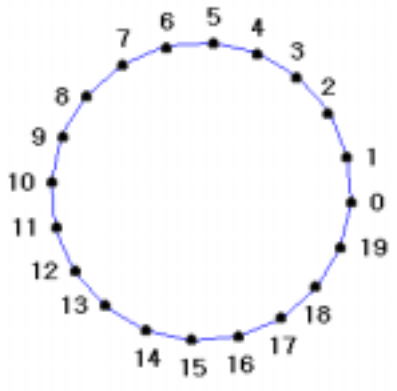
이 그림의 경우에는 61칸씩 건너 뛰어가게 된다. 그러나 눈금이 36개 단위로 반복이 되고 있으므로 이것은 61-36=25 칸 씩 건너뛰어 가는 것과 동일하다. 이 경우에 위의 경우와 그림을 다르지만 역시 모든 지점을 거쳐야 원래의 위치로 되돌아 갈 수 있음을 알 수 있다.



i=12%36;
for j=0 to 35 ;rt v_(i*j)%36; fd v_(i*j)%36 ; next

마지막의 경우에는 위의 두 개와 달리 단 세 번만에 원래의 위치로 도착했다.
앞의 두가지 경우와 다른 점이 무엇일까?
다른 숫자들에 대해서는 어떻게 될지 생각해 보고 법칙을 일반화 시켜보자

【문제】
눈금의 단위가 다른 시계의 경우를 생각해 보자. 모든 지점을 거쳐서 0 으로 도달하려면 몇 칸씩 건너 뛰어야 하는가? 8번 만에 원래 지점으로 도달하게 되는 경우는 어떤 경우인가?



위에서 계속 설명했던 것처럼 원래 지점으로 도달하기 위해서는 거북이가 한번에 도는 각의 배수가 360의 배수가 되어야 한다. 즉 nt=360k(t는 거북이가 한번에 도는 각, n:원래자리로 가기위해 돌아야 하는 횟수, k : 원래지점에 도달할 때까지 회전하는 총 바퀴 수) 가 성립해야 한다.

첫 번째의 경우 17과 36은 서로 소이다. 따라서 k 가 17의 배수가 되어야 하고 n 은 36 의 배수가 되어야 한다. 그러므로 최소의 k=17, n=36 이 된다. 즉 36번 이동하게 되면 원래의 지점에 도달하게 되고 그때까지 거북이는 시계를 총 17바퀴 돌게 된다. 두 번째의 경우도 61과 26은 서로 소이므로 마찬가지이다.

그러나 마지막의 경우는 다르다. 12 는 이미 36의 배수이다. 식에 의해 n=3k를 만족하면 되므로 최소의 k=1, n=3 이다. 즉 3번 이동 만에 원래 자리로 돌아오게 되며 이때 거북이는 시계를 한바퀴만 돌면 되는 것이다.

일반적으로는 거북이가 이동해야 하는 총 회수는 t(거북이가 한번에 건너 뛰는 크기)와 360의 최소공배수를 t 로 나눈 값이 됨을 알 수 있다.