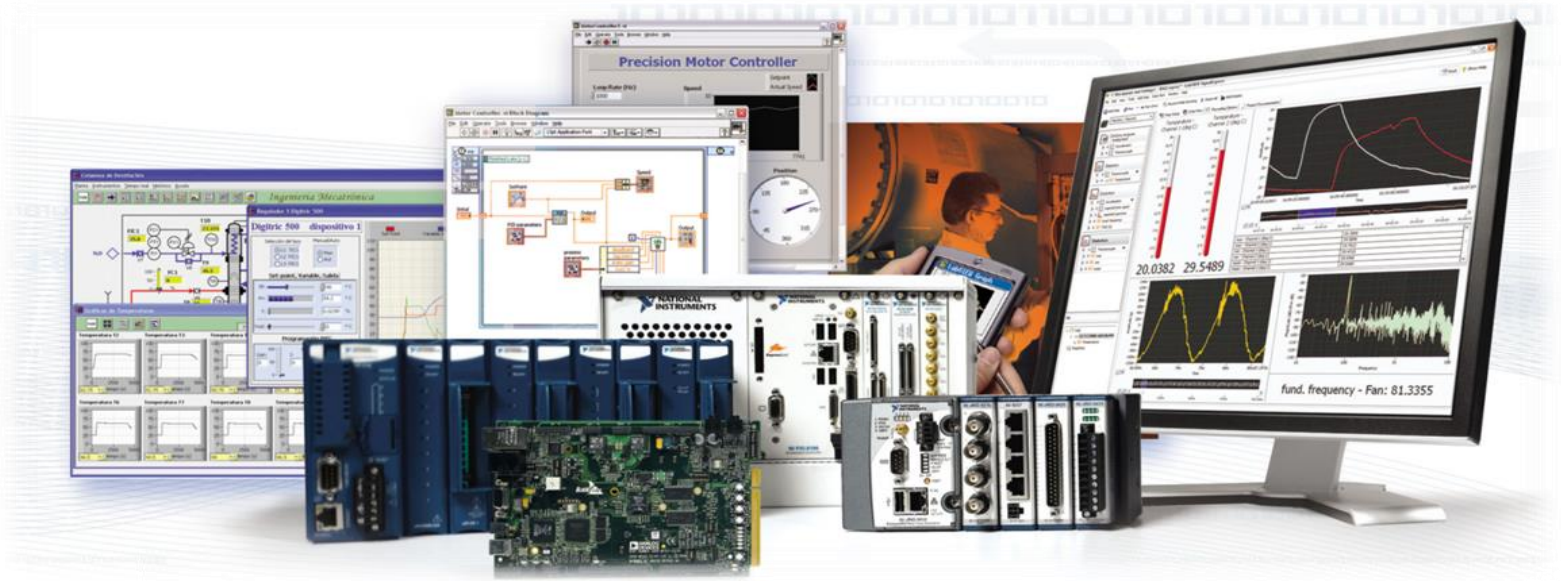
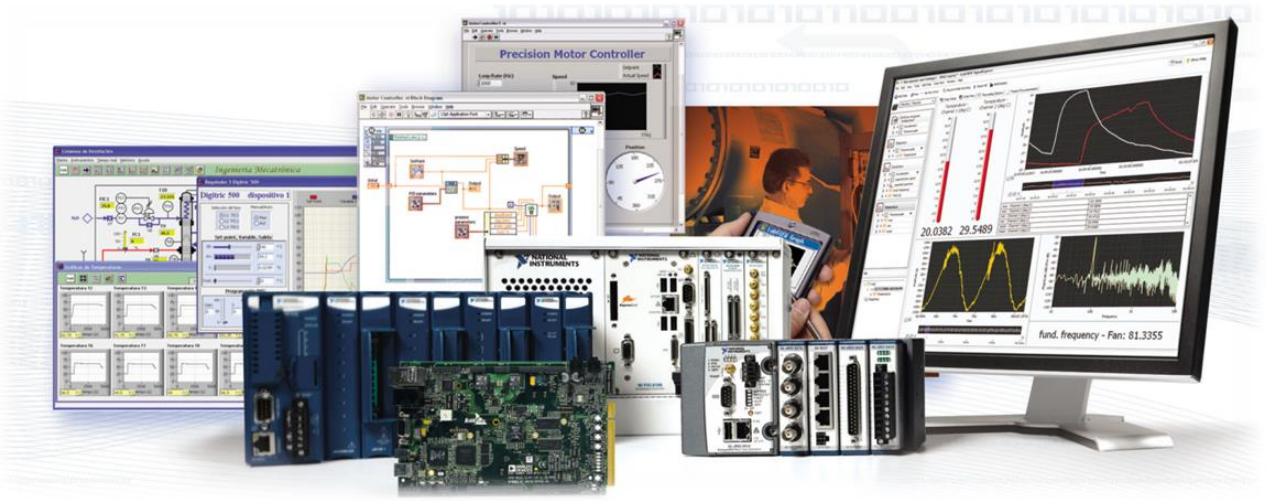
LabVIEW

LabVIEW의 정석 기본편

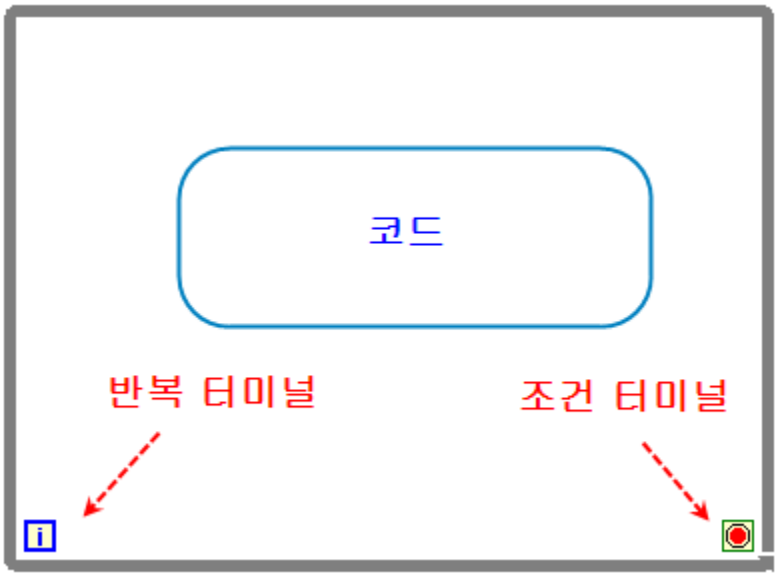


INFINITYBOOKS
인 피 니 티 북 스

4. 구조


Precision Motor Controller
Digitric 500 dispositivo 1
20.0382 29.5489
fund. frequency - Fan: 81.3355

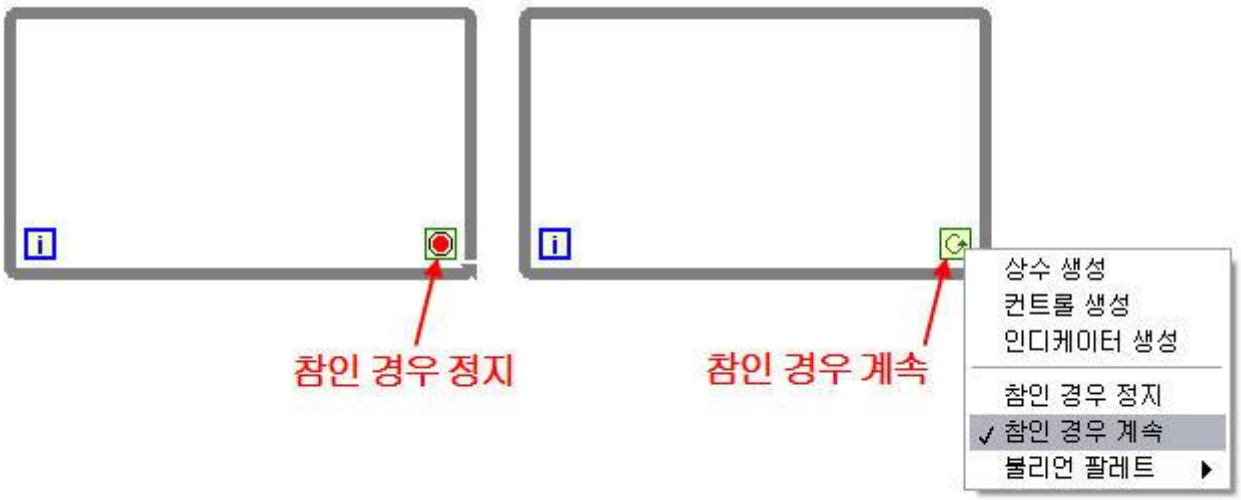
While 루프



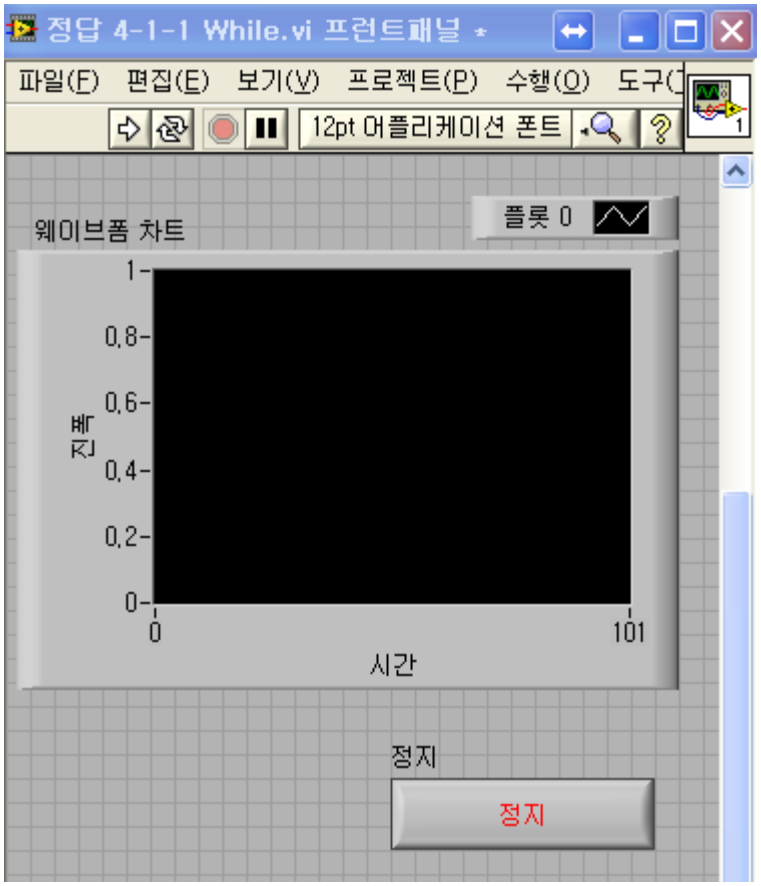
•**While** 루프 **:** 반복을 위한 구조

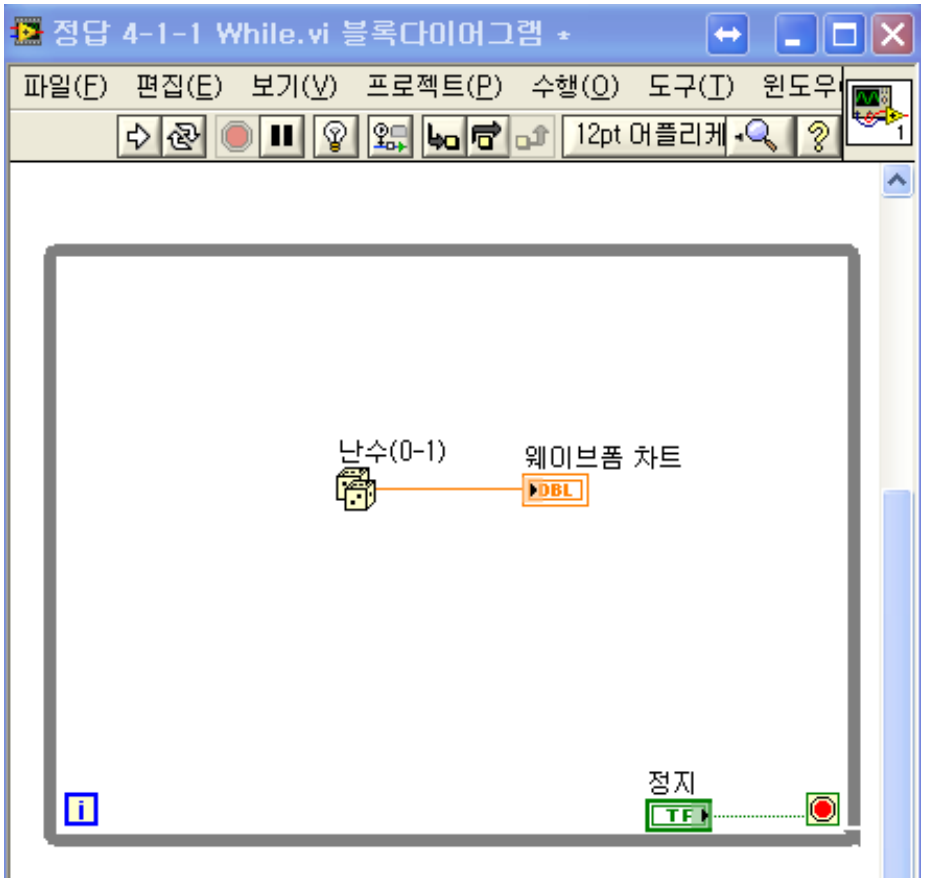
반복 터미널은 현재 몇 번 반복이 이뤄지 고 있는지를 표시
조건 터미널은 While 루프가 정지하는 조건을 정의

조건 터미널

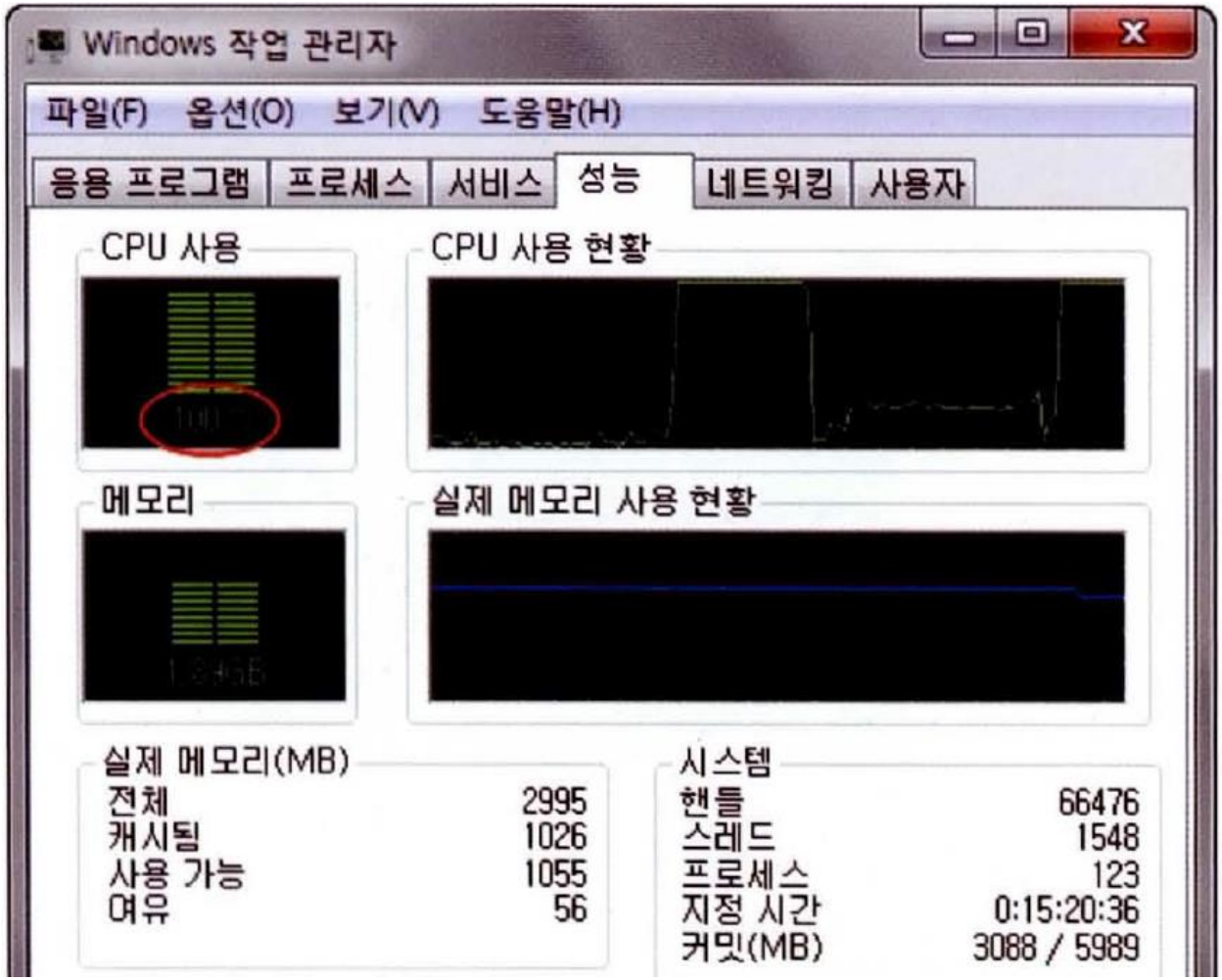


실습4-1-1) While 루프


정답 4-1-1 While.vi 프런트패널 *
파일(F) 편집(E) 보기(V) 프로젝트(P) 수행(O) 도구(
12pt 어플리케이션 폰트
웨이브폼 차트
플롯 0
1
0,8
0,6
0,4
0,2
0
진폭
0
101
시간
정지
정지

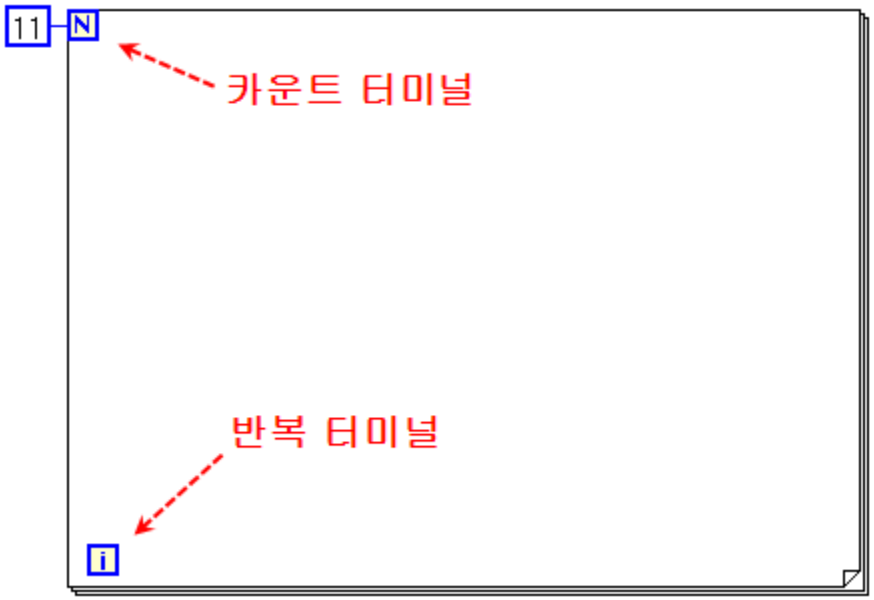

정답 4-1-1 While.vi 블록다이어그램 *
파일(F) 편집(E) 보기(V) 프로젝트(P) 수행(O) 도구(T) 윈도우
12pt 어플리케
난수(0-1)
웨이브폼 차트
DBL
정지
TF

❖ 윈도우의 작업 관리자 창을 띄우고 실행



프로그램 실행 중에
CPU 점유율이 현저히
올라가는 것을 확인

For 루프

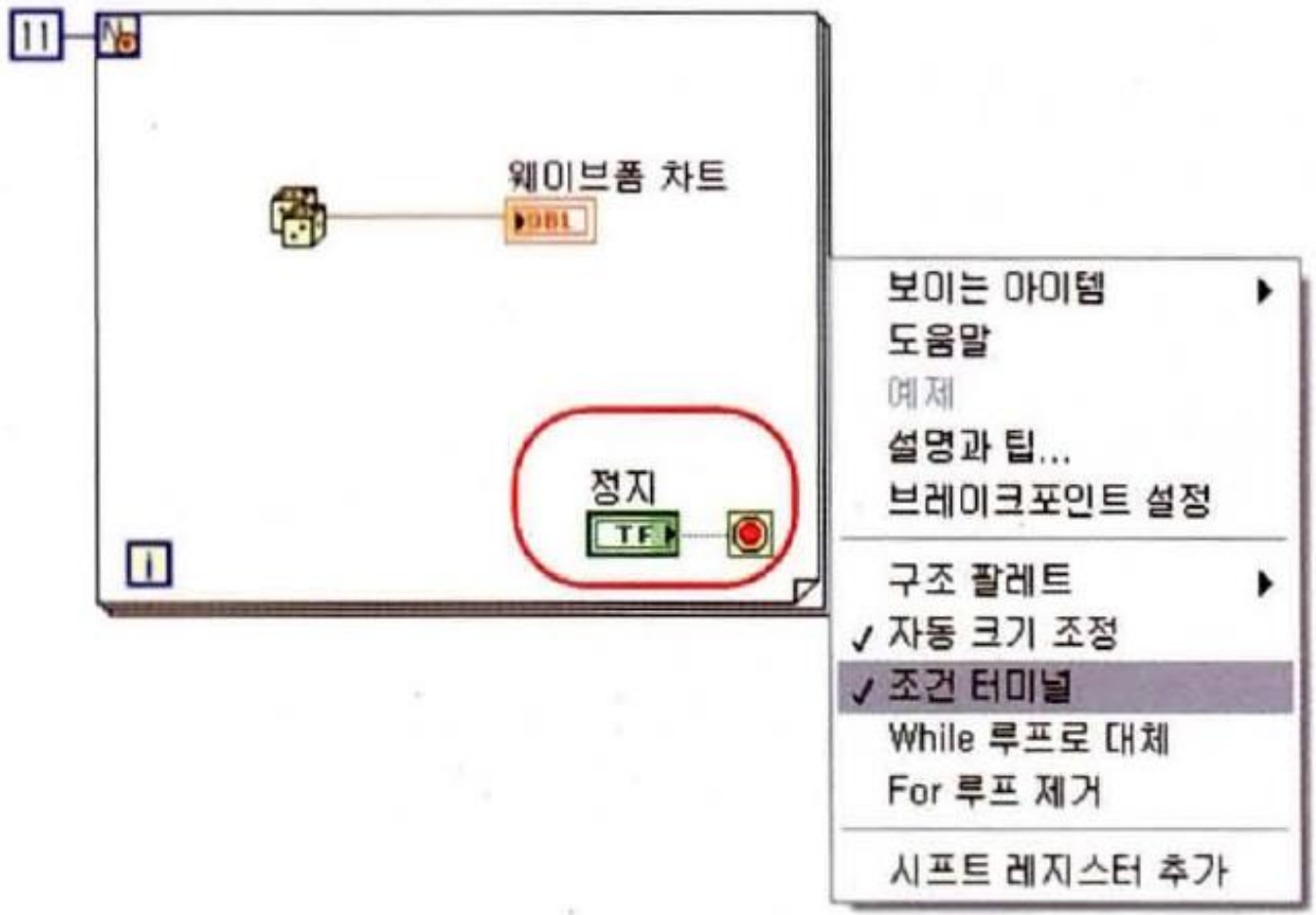


•**For** 루프 **:** 반복 횟수를 지정하여 사용**.** 카운트 터미널에 반복횟수를 지정함**.**

For 루프

- For 루프가 While 루프처럼 조건 터미널을 가지고 있는 경우가 있습니다.
- 조건 터미널을 만들려면 For 루프 경계선에서 마우스 오른쪽 버튼을 클릭하여 바로가기메뉴 >조건 터미널을 선택하면 나타납니다.
- 이는 프로그래밍 할 때 지정해 놓은 반복 횟수를 다 실행하지 않고 '정지 ' 버튼을 이용하여 정지시킬 수 있습니다.
- For 루프는 카운트 터미널에 '0'을 연결하게 되면 반복을 하지 않습니다.
- 즉 For 루프 안에 있는 코드를 실행하지 않습니다.

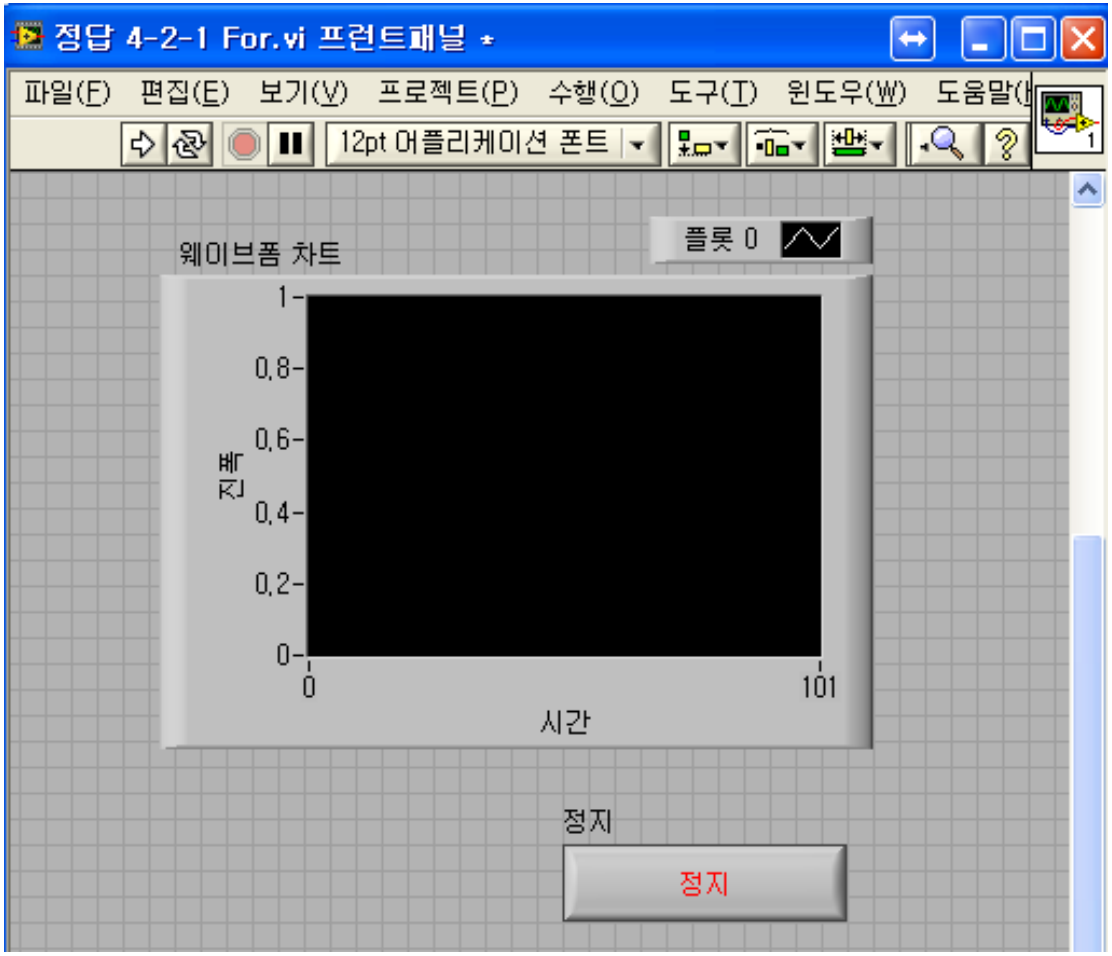
For 루프


웨이브폼 차트
DBL
정지
TF
보이는 아이템
도움말
예제
설명과 팁...
브레이크포인트 설정
구조 팔레트
✓ 자동 크기 조정
✓ 조건 터미널
While 루프로 대체
For 루프 제거
시프트 레지스터 추가

실습4-2-1) For 루프

실습 4-2-1

❖ 새 VI를 열고 그림과 같이 웨이브폼 차트와 정지 버튼으로 프런트패널을 구성



실습 4-2-1

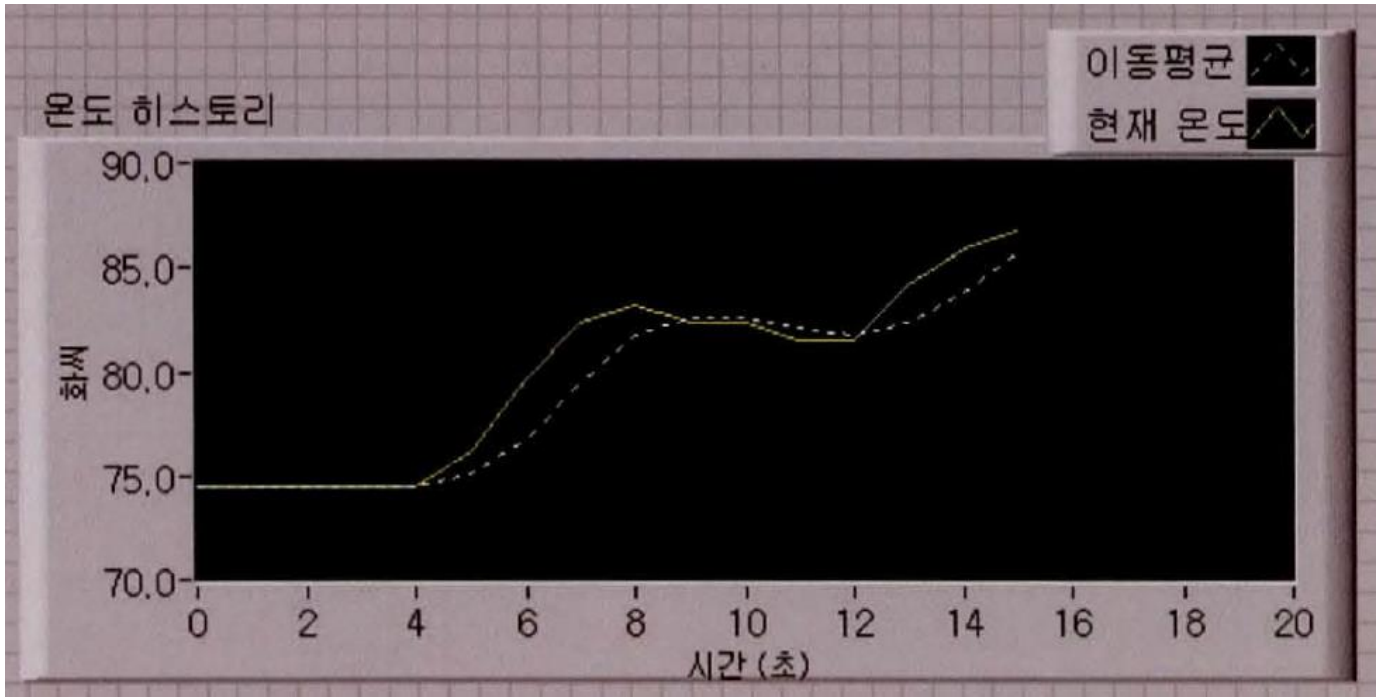
- 블록다이어그램에서 난수(0-1).vi와 For 루프를 추가해 그림과 같이 코딩
- For 루프의 바로가기메뉴 > 조건 터미널을 선택해서 조건 터미널이 나타나게 합니다

실습 4-2-1

- 실행하면 0에서 1사이의 임의의 숫자가 웨이브폼 차트에 100개가 디스플레이됩니다.
- 100번이 실행되기 전에 정지 버튼을 누르면 100번 반복하지 않고 바로 정지하게 됩니다.
- 이를 시도해 보려면 4.5장의 타이밍 노드를 For 루프에 추가하는 것이 더 편리합니다.

시프트 레지스터

- 1초에 한 번씩 온도를 측정하여 현재 측정값을 기준으로 두 개 이전 값과 함께 평균 계산해서 웨이브폼 차트에 디스플레이 할 경우 과거 측정값을 기억해 두어야 합니다.
- 그림을 보면 '현재 온도' (실선)와 '평균'(접선)을 함께 웨이브폼 차트에 디스플레이 했습니다.

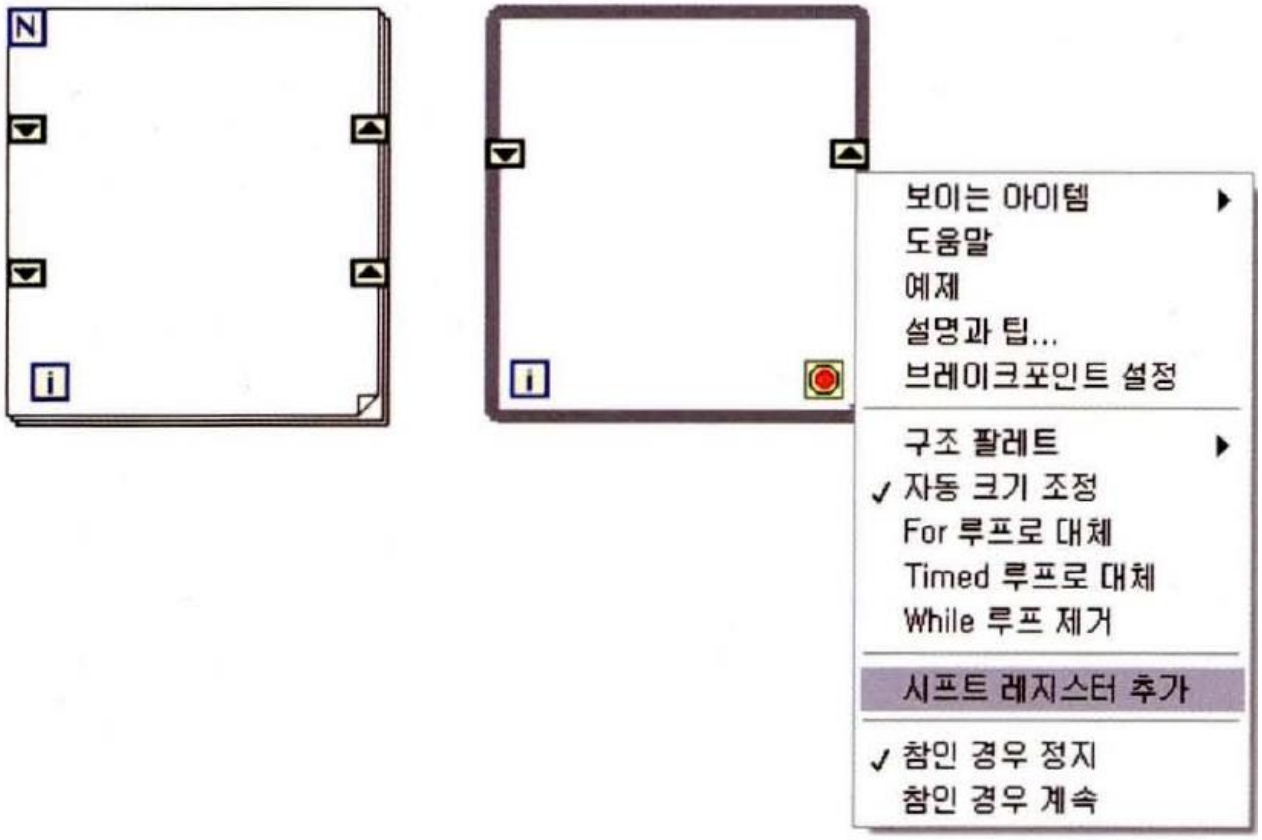


시프트 레지스터

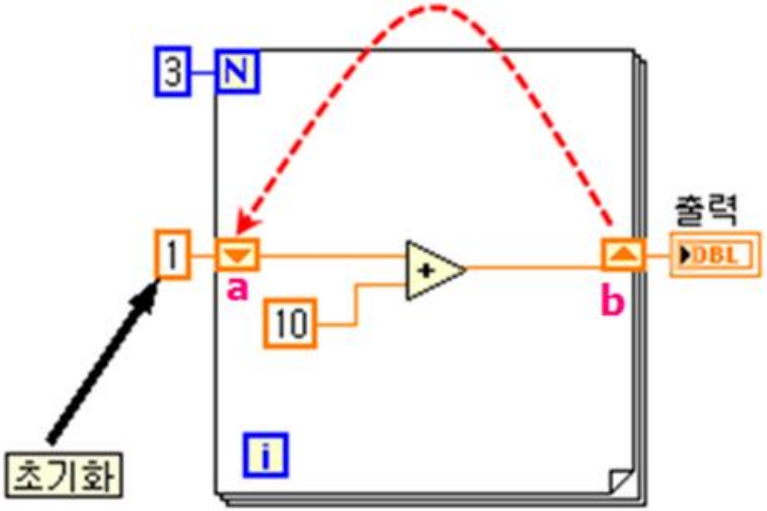
- 여기서 평균은 현재 측정된 값과 이를 기준해서 바로 전에 측정한 두 개의 온도값을 합쳐서 평균내어 디스플레이 했습니다.
- 이때 사용한 것이 시프트 레지스터입니다.
- 시프트 레지스터는 반복구문, 즉 **While/For** 루프에서 사용가능

시프트 레지스터

- 시프트 레지스터를 만드는 방법은 **While/For** 루프의 경계선에서 마우스 오른쪽 버튼을 클릭하여 바로가기 메뉴 > 시프트 레지스터 추가를 선택



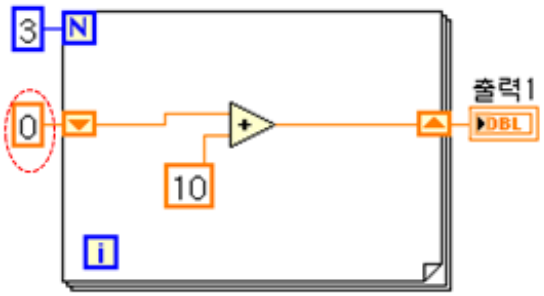
시프트 레지스터



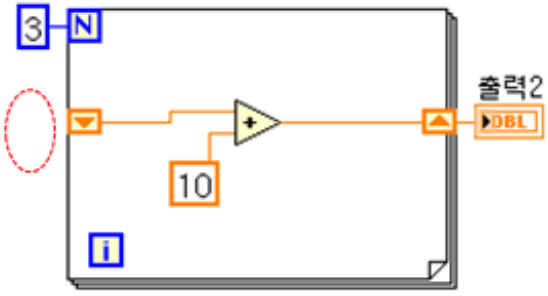
•시프트 레지스터 : 과거값을 저장

반복 횟수	지점 a 의 값	지점 b 의 값
1st	1	11
2nd	11	21
3rd	21	31

시프트 레지스터 초기화 유무



초기화 된 시프트 레지스터



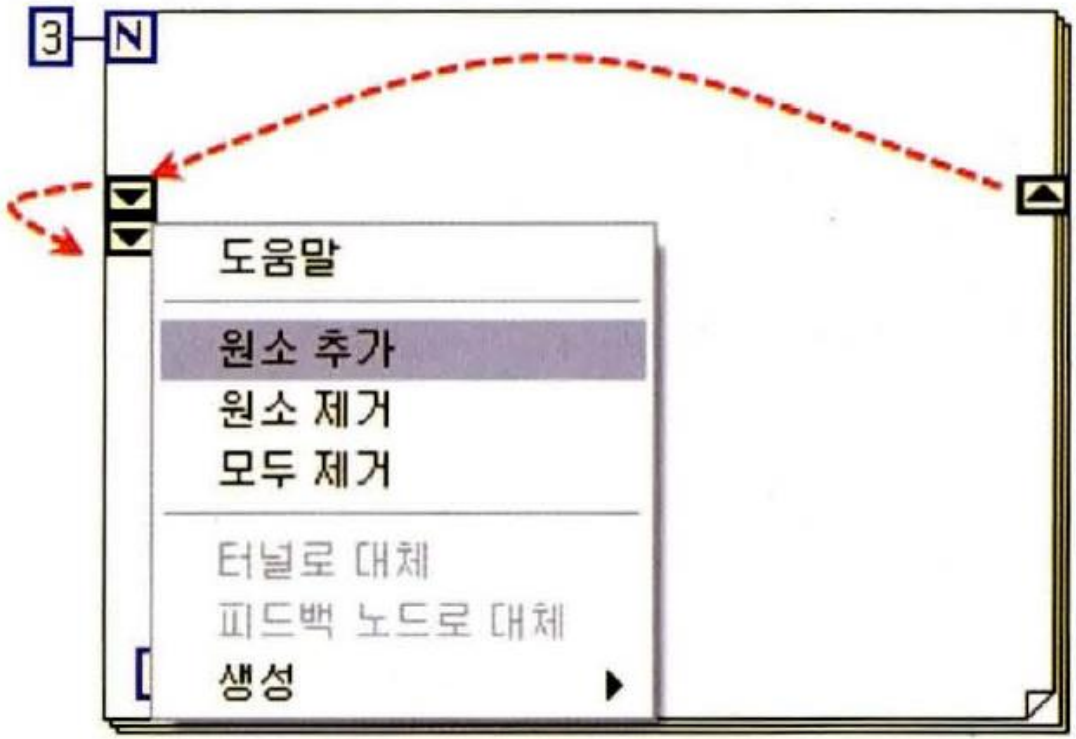
초기화 안된 시프트 레지스터

실행 횟수	초기화	초기화 안 함
1st 실행	출력1 : 30	출력2 : 30
2nd 실행	출력1 : 30	출력2 : 60
3rd 실행	출력1 : 30	출력2 : 90

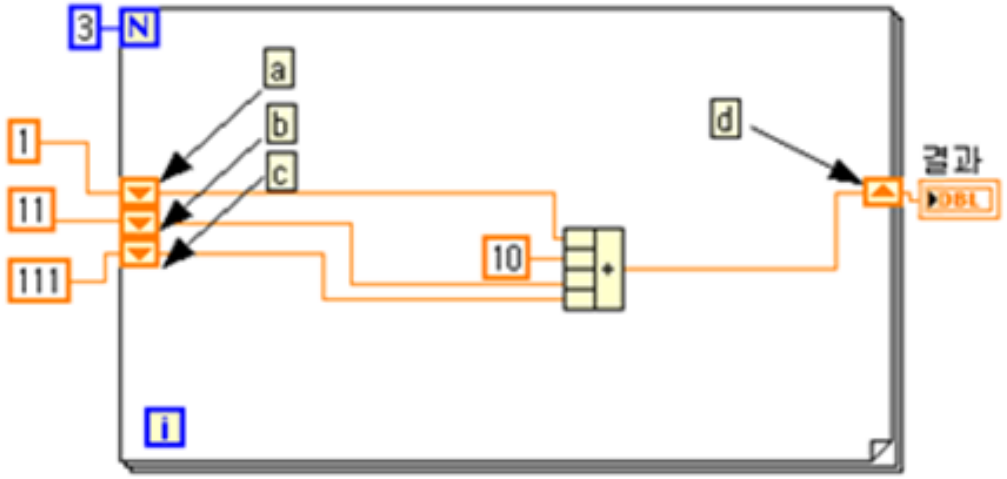
•시프트 레지스터의 초기화 유무에 따라 결과값이 달라짐.

다층 레지스터

- 그림은 시프트 레지스터의 왼쪽에 원소를 추가한 모습
- 원소는 왼쪽에서만 원하는 만큼 추가 가능
- 원소를 추가하면, 두 번째 생긴 원소는 첫 번째 시프트 레지스터의 값을 받아옵니다

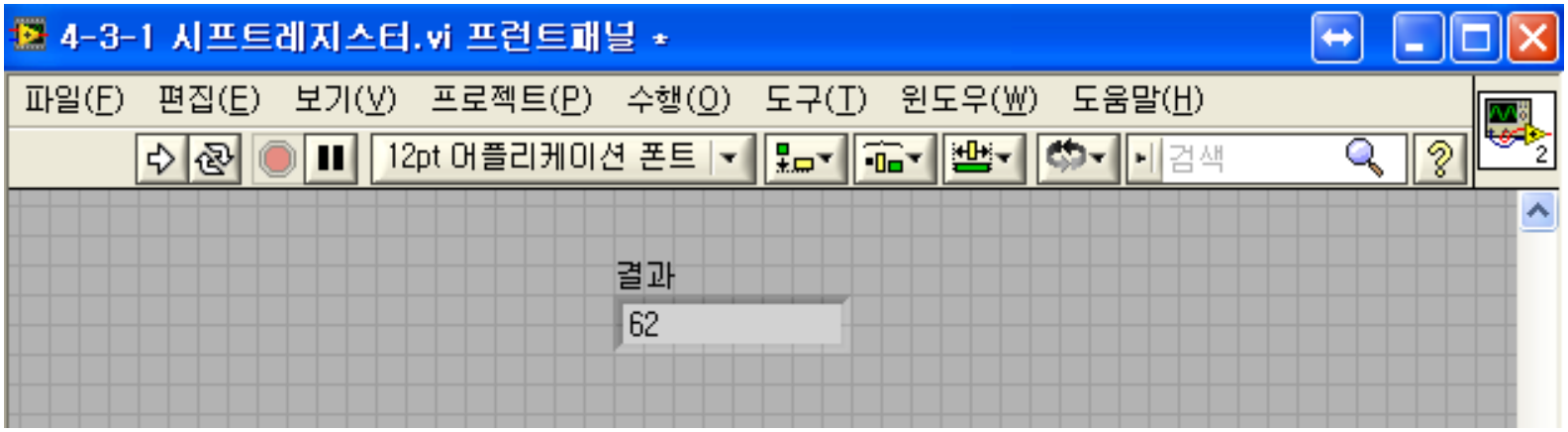


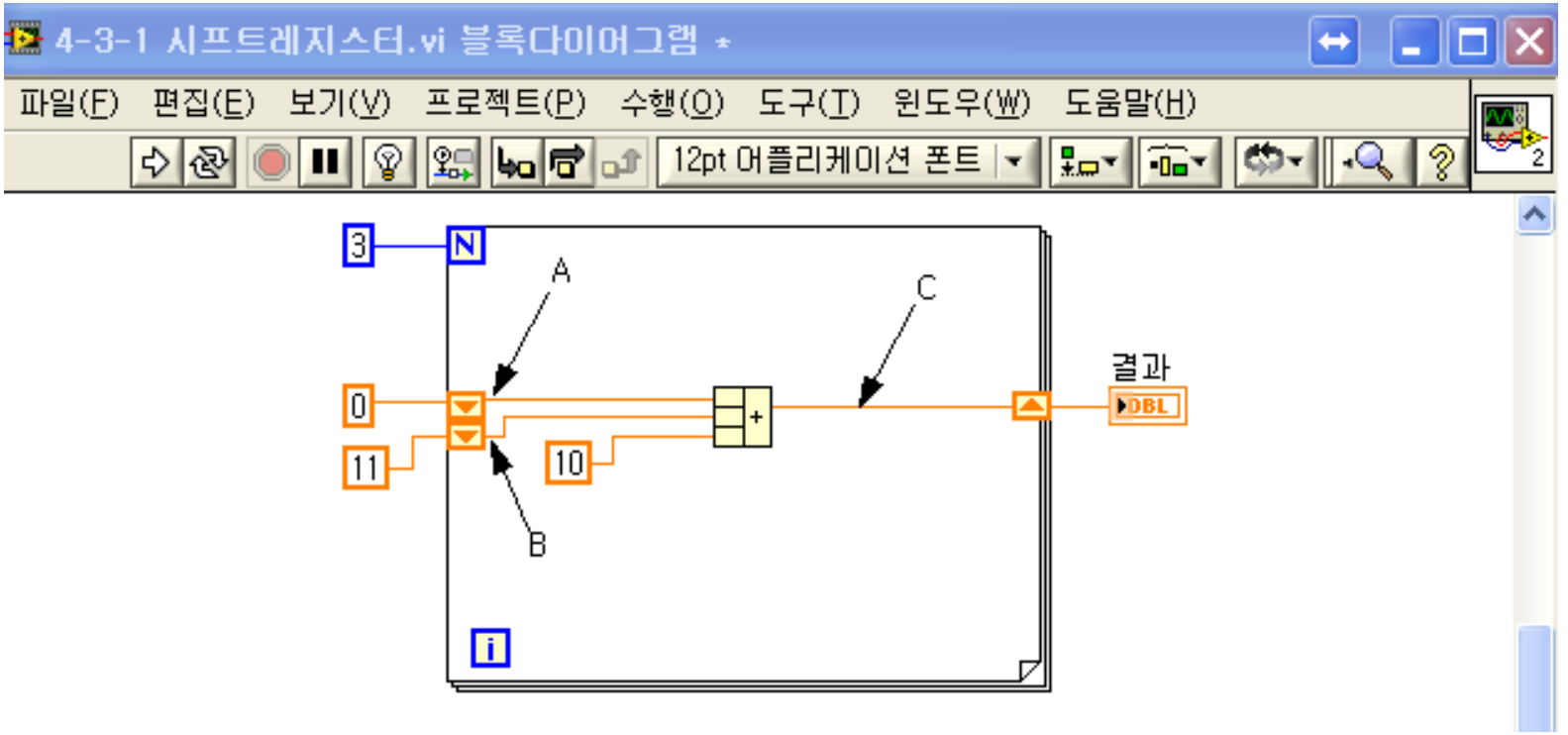
다층 레지스터



반복 횟수	지점 a 의 값	지점 b 의 값	지점 c 의 값	지점 d 의 값
1st	1	11	111	133
2nd	133	1	11	155
3rd	155	133	1	299

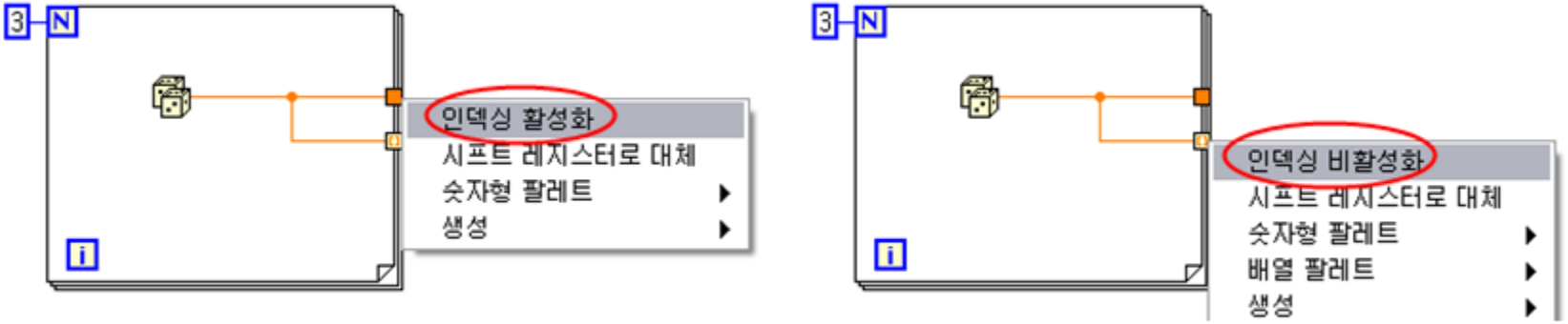
실습4-3-1) 시프트 레지스터


4-3-1 시프트레지스터.vi 프런트패널 *
파일(F) 편집(E) 보기(V) 프로젝트(P) 수행(O) 도구(T) 윈도우(W) 도움말(H)
12pt 어플리케이션 폰트
검색
결과
62


4-3-1 시프트레지스터.vi 블록다이어그램 *
파일(F) 편집(E) 보기(V) 프로젝트(P) 수행(O) 도구(T) 윈도우(W) 도움말(H)
12pt 어플리케이션 폰트
3
N
A
C
결과
0
DBL
11
10
B
i

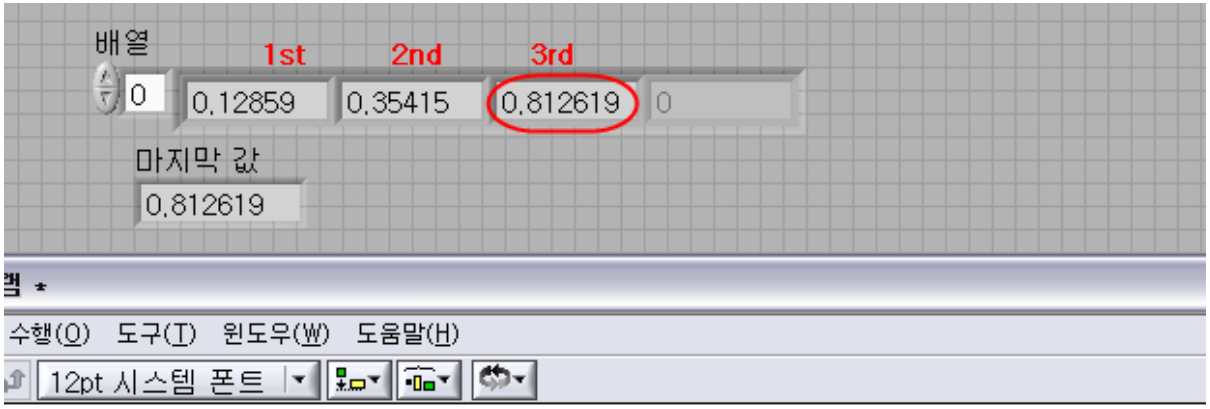
인덱싱 활성화/비활성화

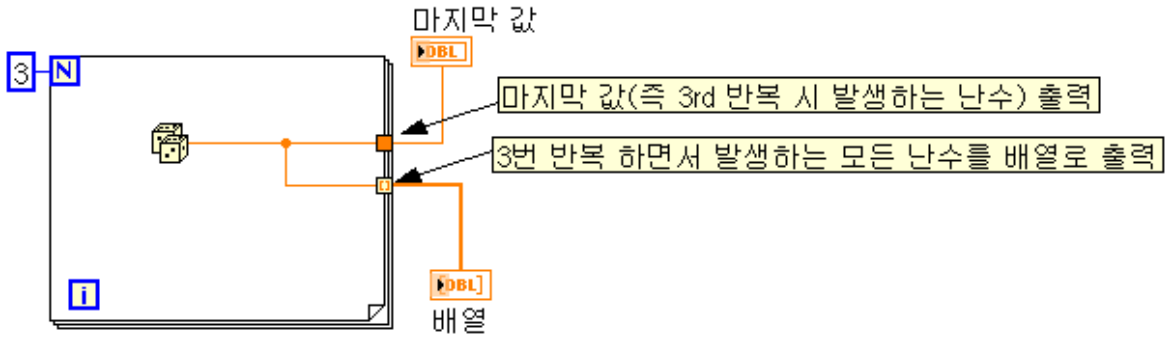
- ❖ 반복 구조(While/For)는 두 가지 출력 모드를 가지고 있습니다.
- ❖ 즉 마지막 값만 출력하든지 또는 실행 중 발생하는 모든 값을 출력하든지 두 가지 중 선택하도록 되어 있습니다.



•인덱싱 활성화/비활성화 유무에 따라 출력이 달라짐.

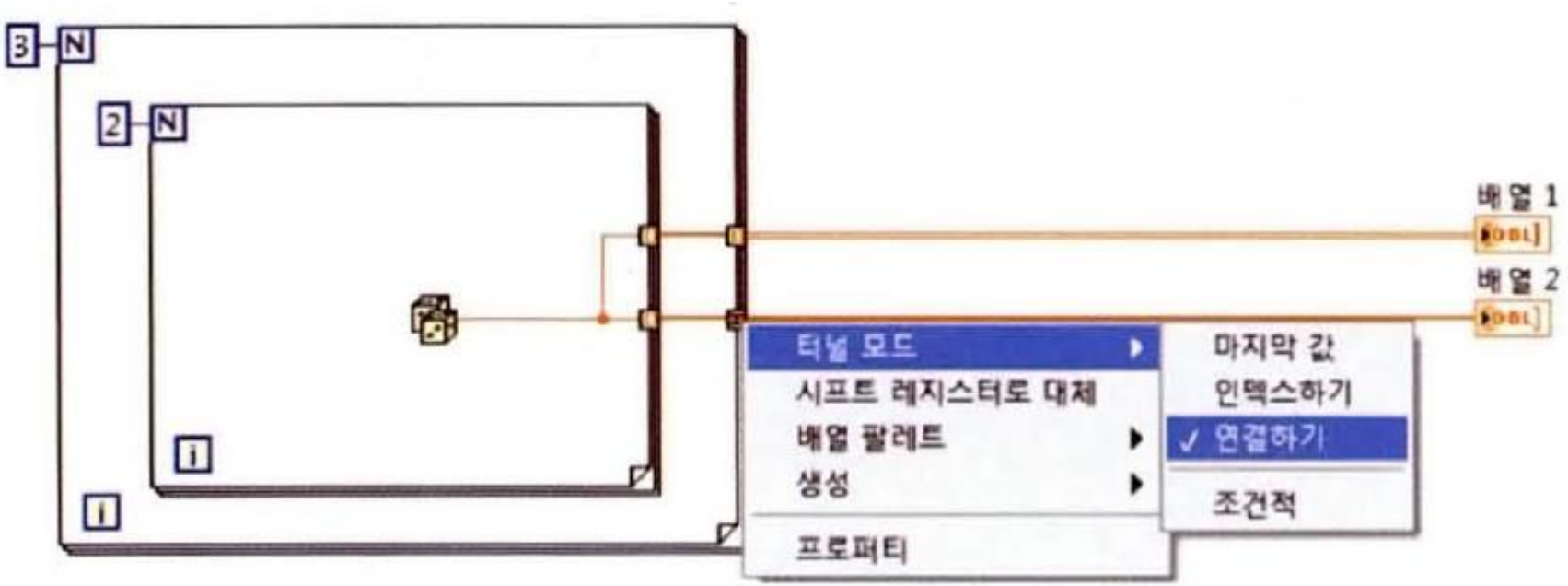
인덱싱 활성화/비활성화 출력





인덱싱 활성화/비활성화 출력

❖ 출력 터널에서 '연결하기' 기능이 있는데 이것은 모든 입력이 순서대로 추가되어, 터널 입력에 연결된 배열과 같은 차원의 출력 배열이 생성



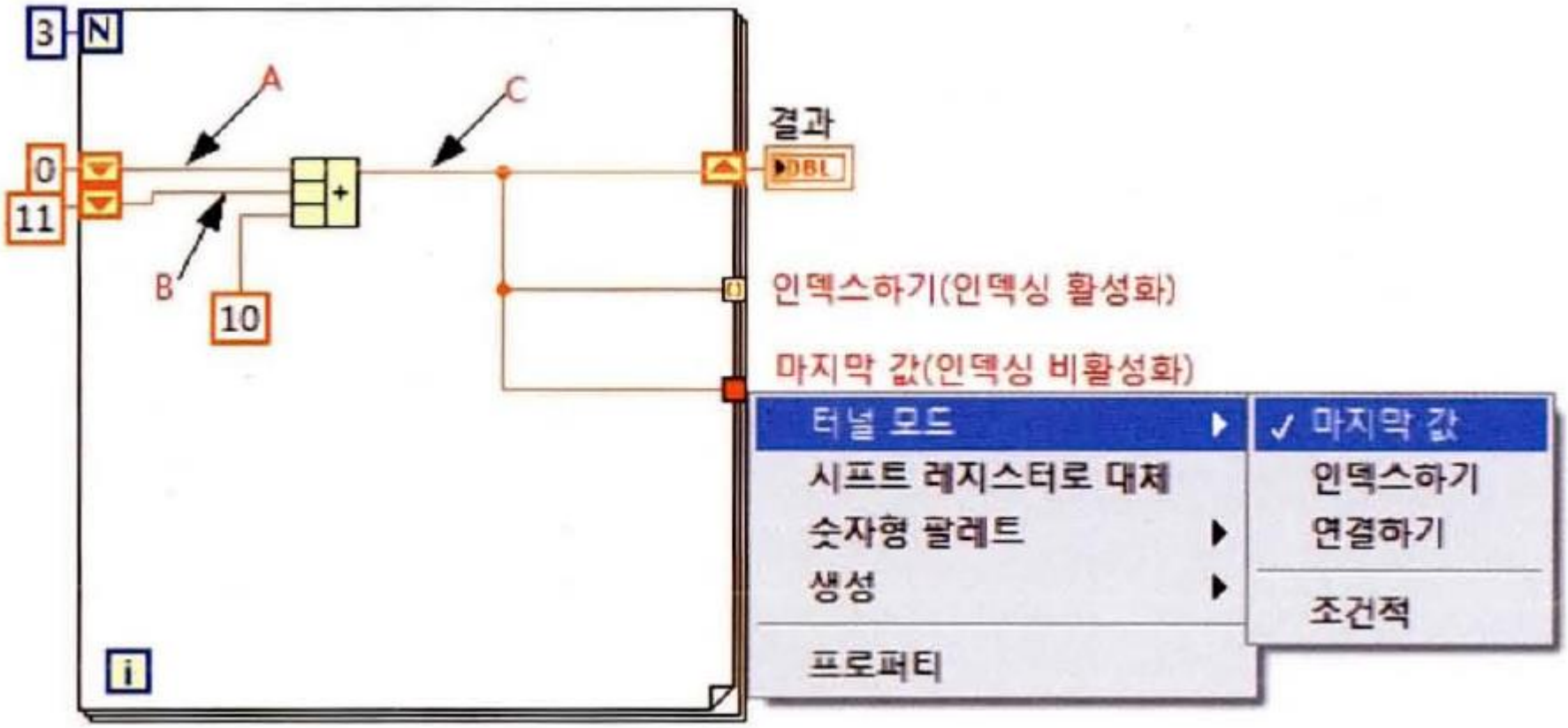
인덱싱 활성화/비활성화 출력

❖ 마지막으로 '조건적' 기능은 출력 터널에서 출력되는 값을 조건에 맞는 값만 출력할 수 있도록 하는 기능입니다.

실습4-4-1)
인덱싱 활성화/비활성화

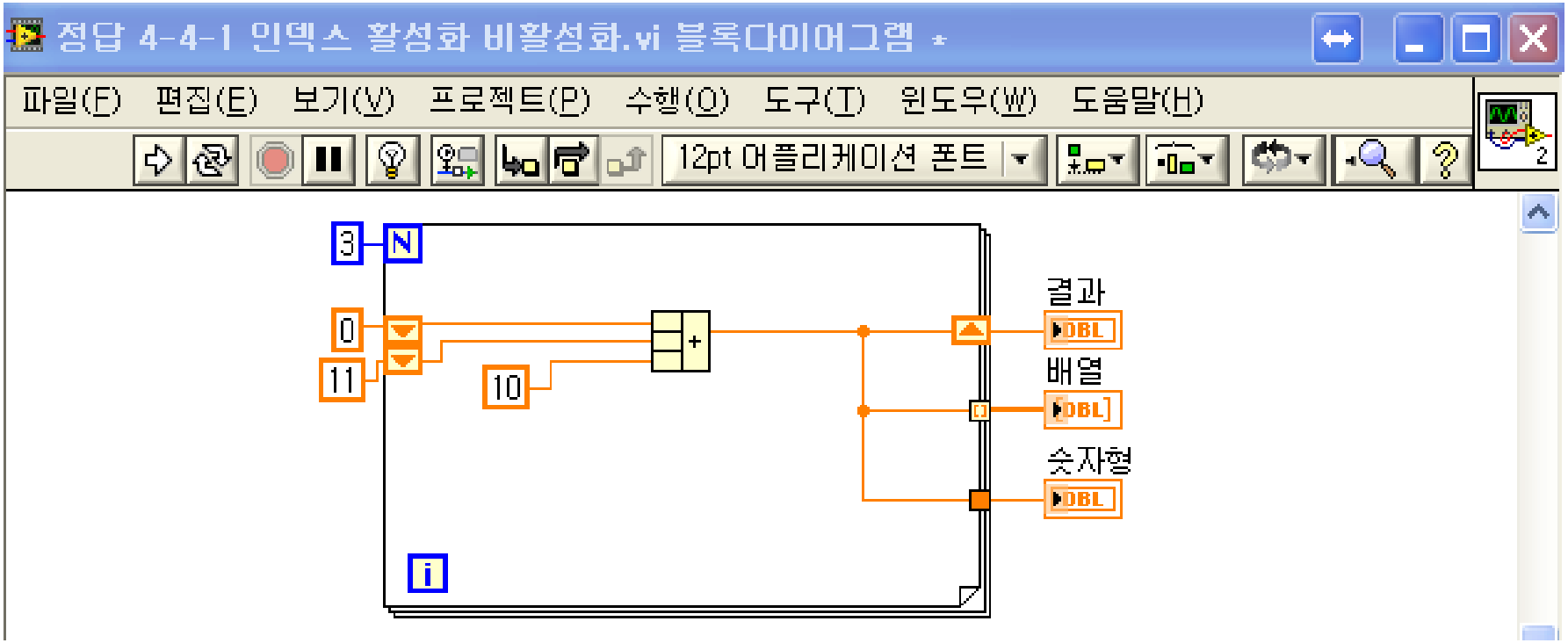
실습 4-4-1

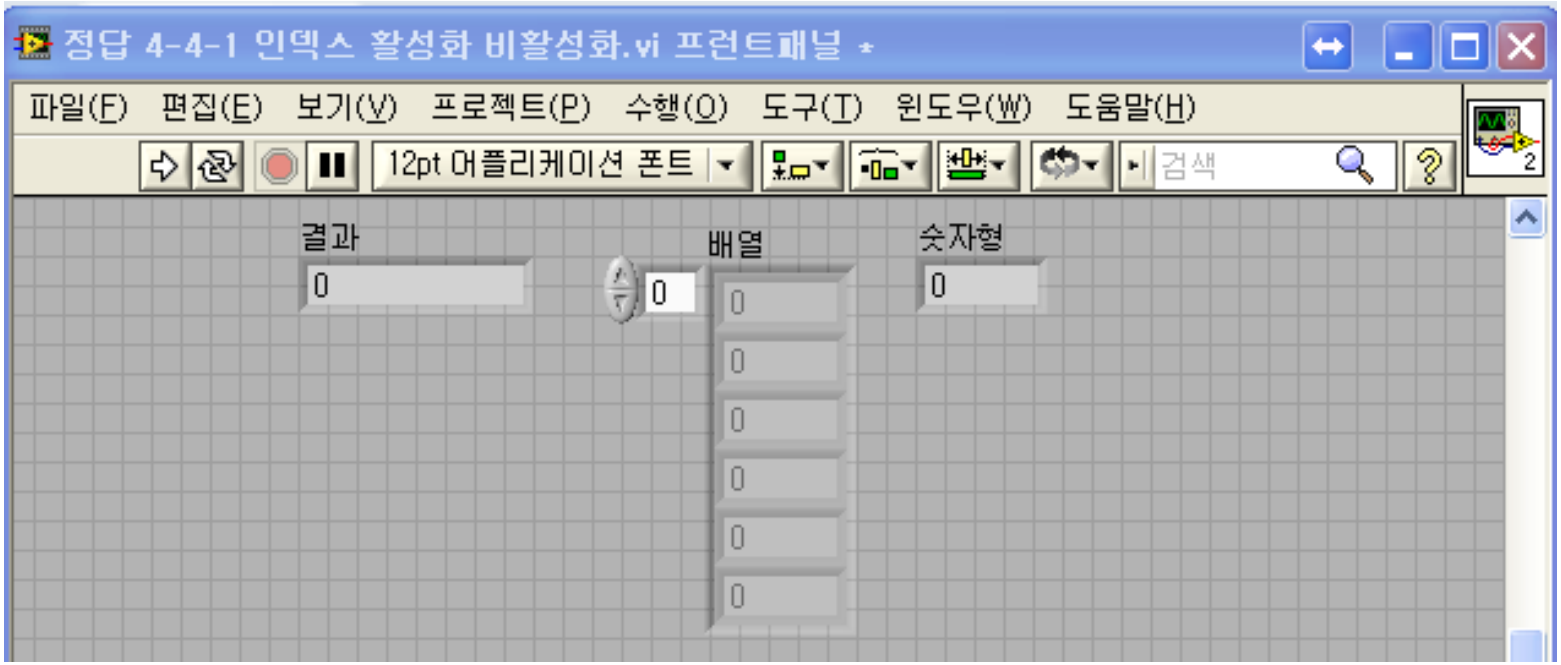
❖ 터널에서 바로가기메뉴 〉터널모드〉 마지막 값/인덱스하기를 선택하여 블록다이어그램을 그림과 같이 수정합니다



실습 4-4-1

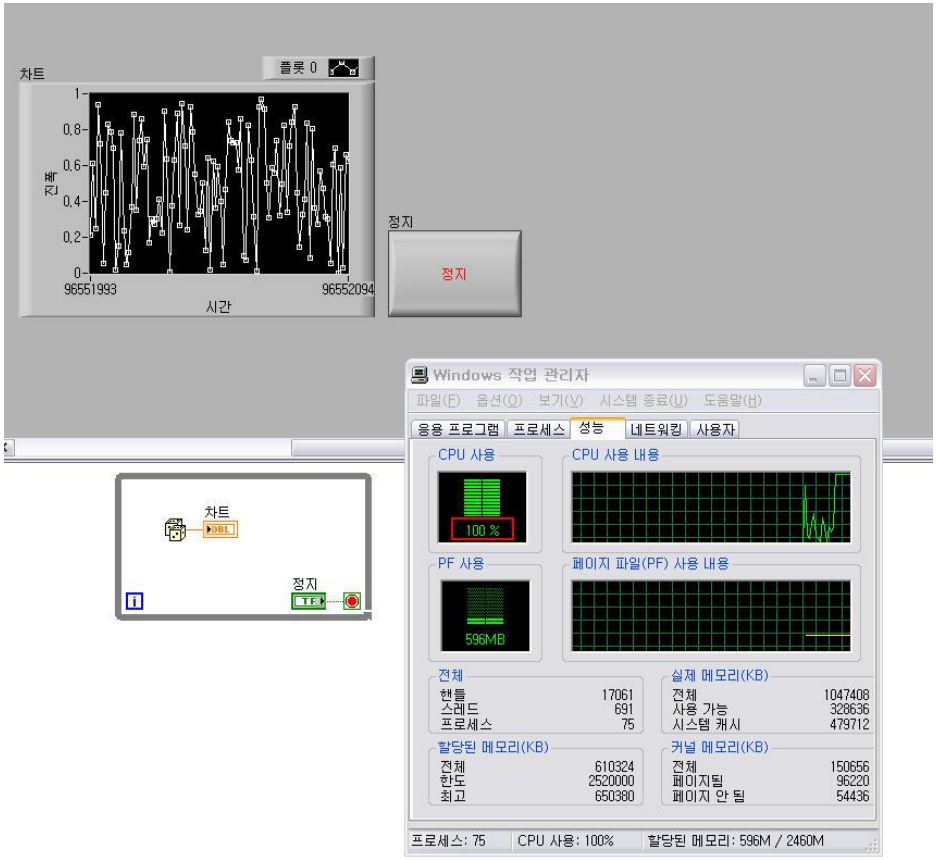
❖ 각 출력 터널의 바로가기메뉴 > 생성 > 인디케이터를 선택합니다



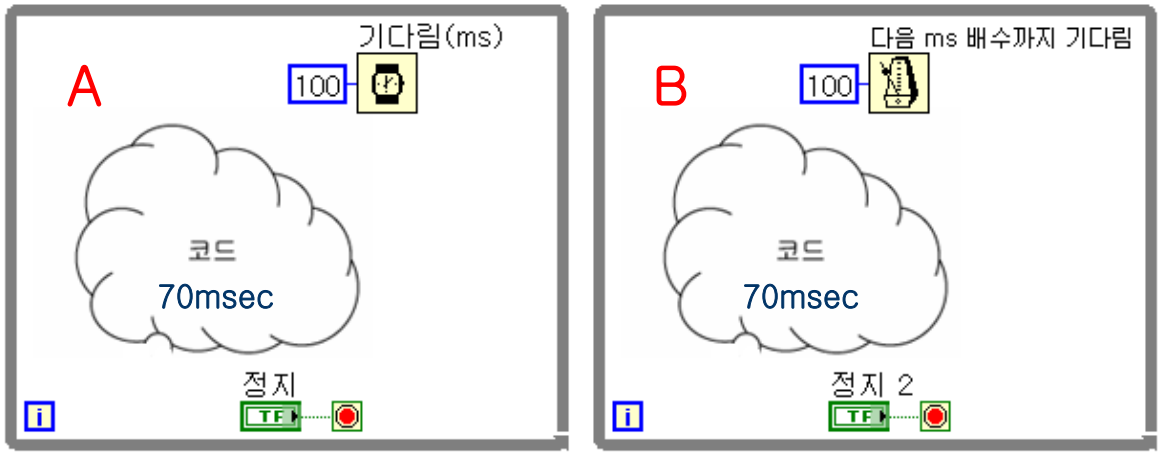

정답 4-4-1 인덱스 활성화 비활성화.vi 프런트패널 *
파일(F) 편집(E) 보기(V) 프로젝트(P) 수행(O) 도구(T) 윈도우(W) 도움말(H)
12pt 어플리케이션 폰트
검색
결과
0
배열
0
0
0
0
0
0
0
숫자형
0

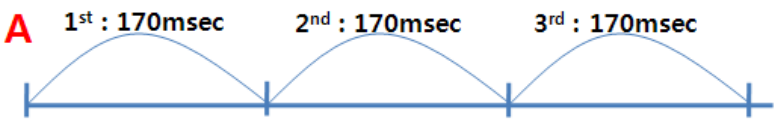
타이밍 노드 필요성

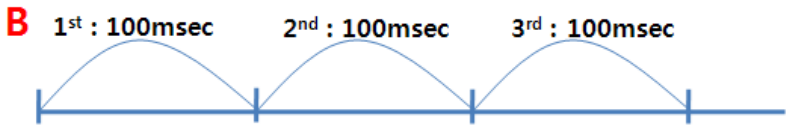
•반복 구조에 타이밍 노드가 없으면 **CPU** 점유율이 최대로 상승함.



타이밍 노드







- 기다림 : 절대적 기다림.
- 다음**ms**배수까지 기다림 : 상대적 기다림.

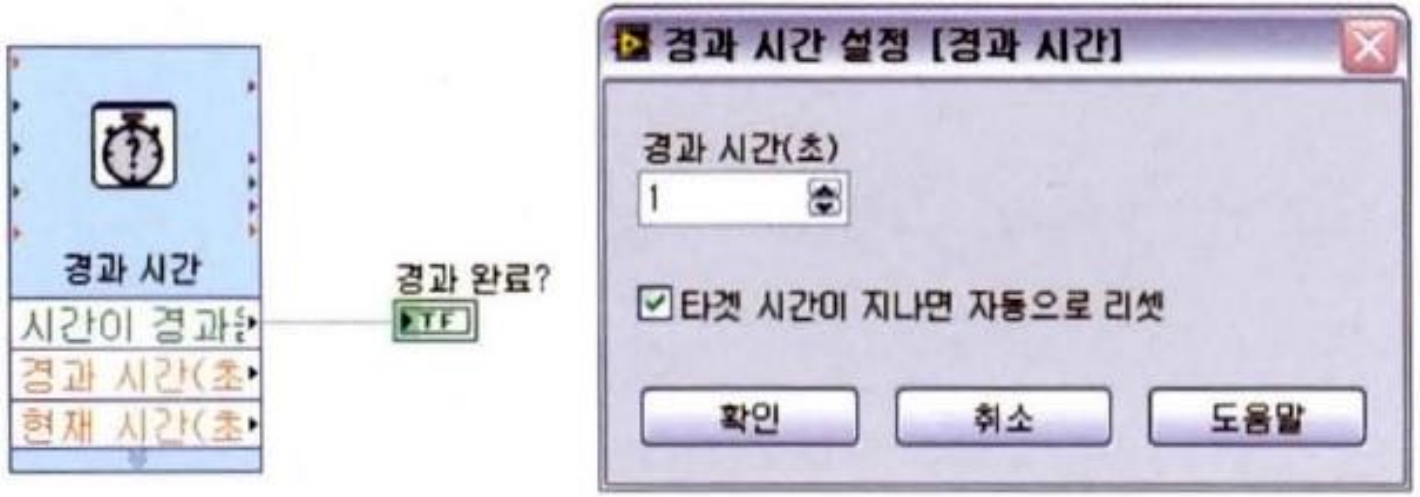
*타이밍 노드와 나머지 노드들간의 순서를 결정해줘야지만 위 그림과 같이 정확하게 동작함.

다양한 타이밍 노드들

- 시간 지연 : 기다림 노드와 기능이 같음.
- 경과 시간 : 설정한 시간을 알려줌.
- 틱 카운트 : 초시계와 같은 같은 기능.

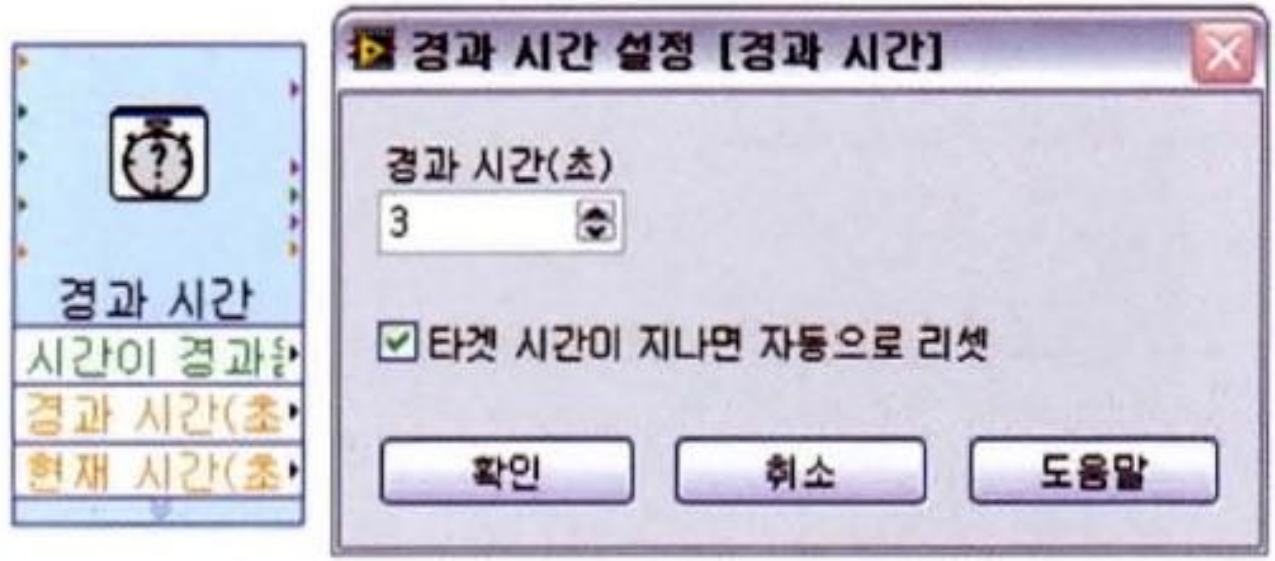
다양한 타이밍 노드들

•경과 시간 : 그림에서 **1**초를 설정하였고 **1**초가 경과하게 되면 '경과 완료**?**'에 '참' 값을 출력.

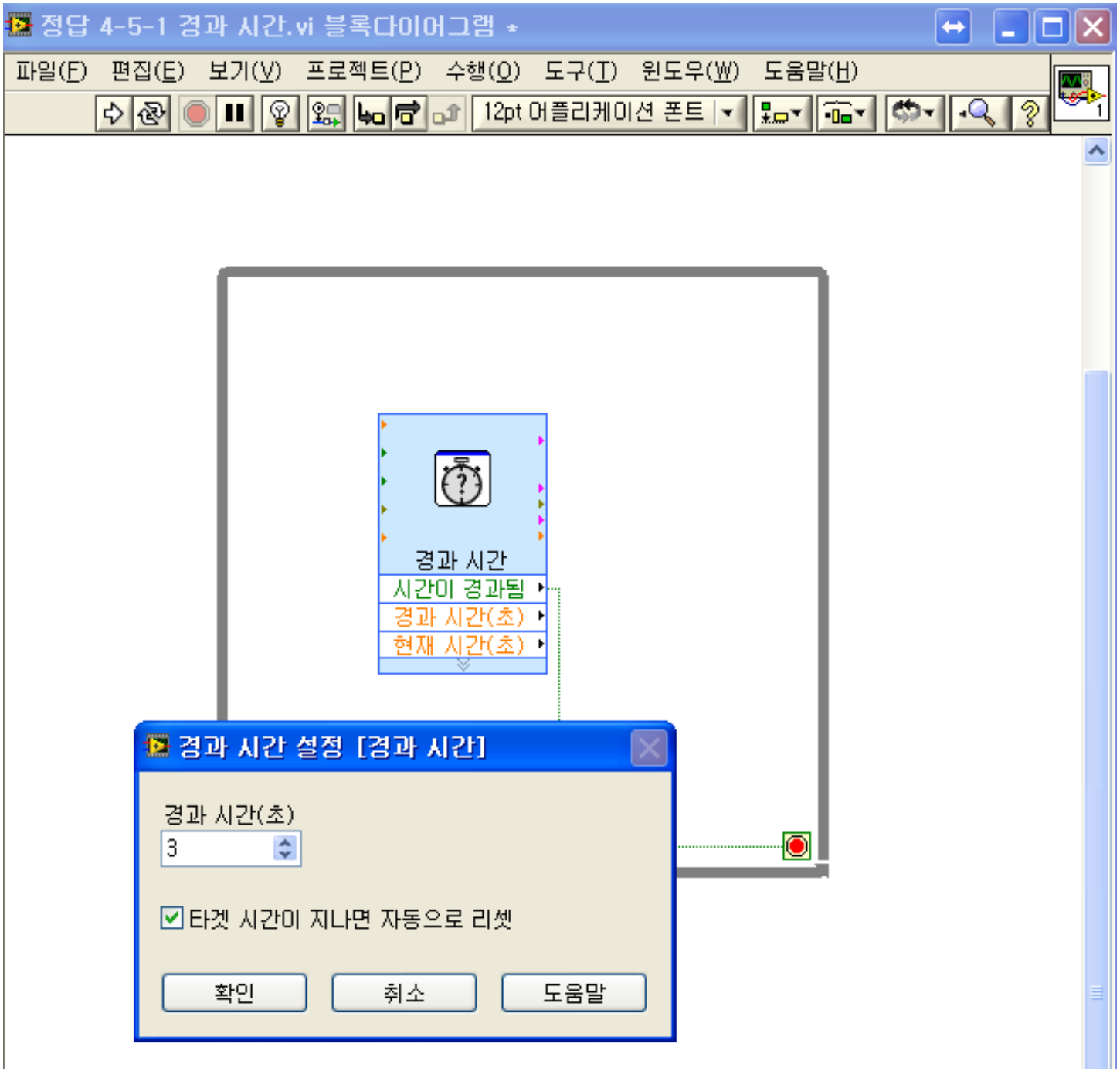


실습4-5-1) 경과 시간 노드

❖ 새 VI를 열어 블록다이어그램에 경과시간.VI를 놓고 경과 시간을 '3'초로 설정하고 확인을 누른다.



- 블록다이어그램에서 **While** 루프를 이용하여 그림과 같이 구성
- 프로그램을 실행하여 프로그램이 **3**초만에 정지하는지 확인

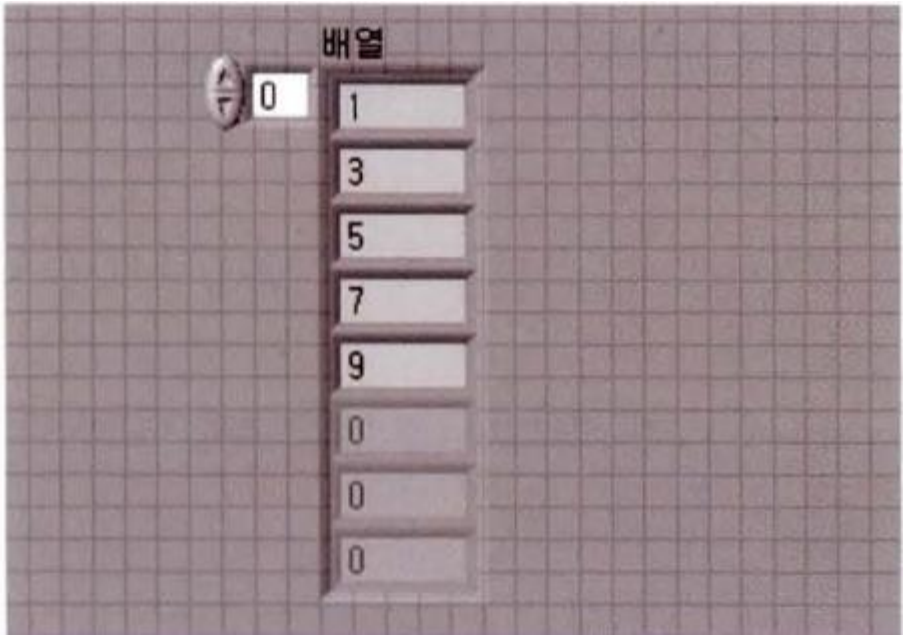

정답 4-5-1 경과 시간.vi 블록다이어그램 *
파일(F) 편집(E) 보기(V) 프로젝트(P) 수행(O) 도구(T) 윈도우(W) 도움말(H)
12pt 어플리케이션 폰트
경과 시간
시간이 경과됨
경과 시간(초)
현재 시간(초)
경과 시간 설정 [경과 시간]
경과 시간(초)
3
타겟 시간이 지나면 자동으로 리셋
확인
취소
도움말

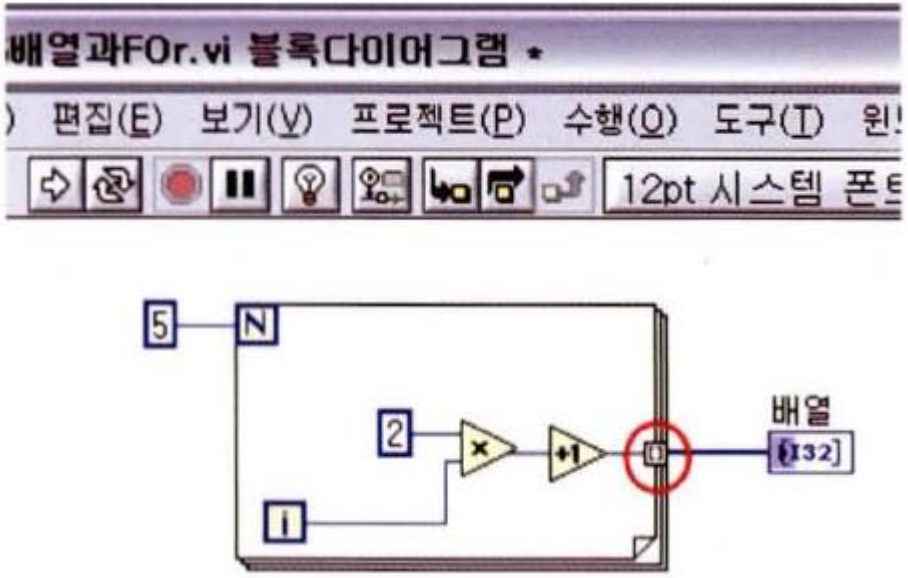
배열과 For루프

- 배열의 원소를 개별적으로 풀고, 낱개의 원소들을 묶어서 배열로 만드는 법을 배워보자.
- 이를 위해 **For** 루프의 인덱싱 기능을 사용
- **For** 루프의 출력 터널에 인덱싱 활성화를 이용하여 배열을 생성

배열과 For루프

❖ 1, 3. 5, 7, 9 의 원소를 가진 1차원 배열을 For 루프르 이용하여 배열을 생성하는 예



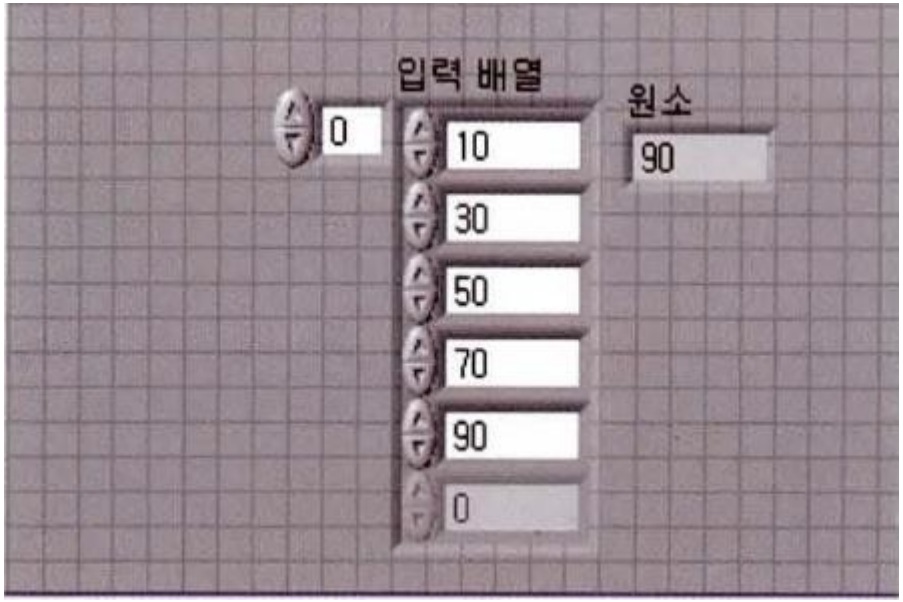


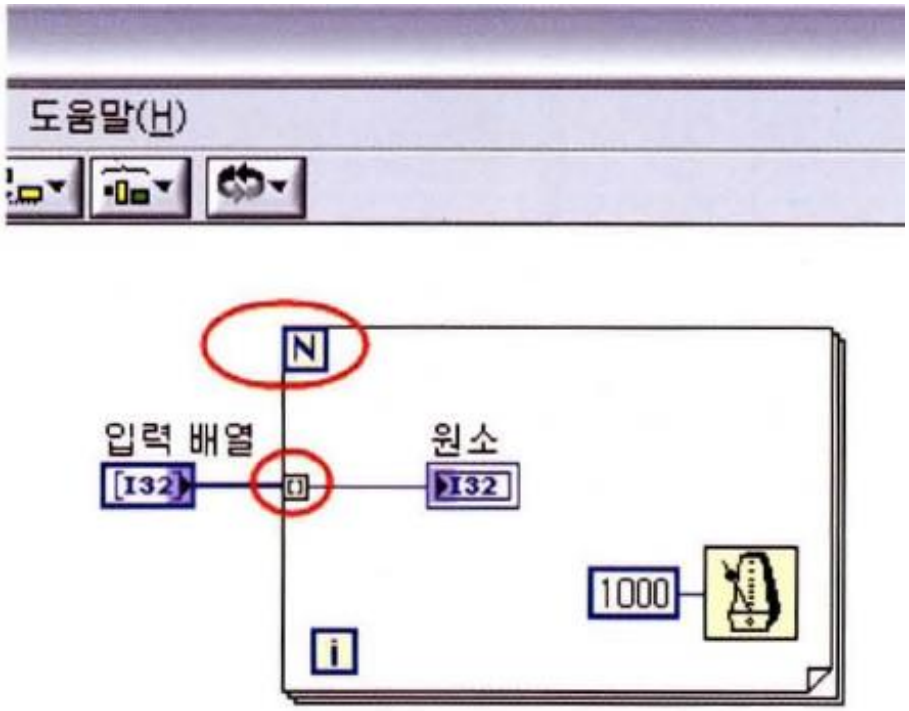
배열과 For루프

- 이번엔 반대로 배열의 원소를 하나하나 푸는 방법을 살펴보자.
- 10, 30, 50, 70, 90의 원소를 가진 1차원 배열에서 각 원소를 하나씩 풀어내려면 For 루프의 압력 터널에서 인덱싱 활성화하면 For 루프의 N(카운트 터미널) 에 값을 주지 않아도 배열의 원소 개수만큼 반복하게 된다.
- 즉 현재 1차원 배열의 원소의 개수가 5개이므로 이 프로그램을 실행하면 5번 반복을 하면서 각 반복 때마다 차례로 원소의 값들이 하나씩 '원소'에서 모니터링된다.

배열과 For루프

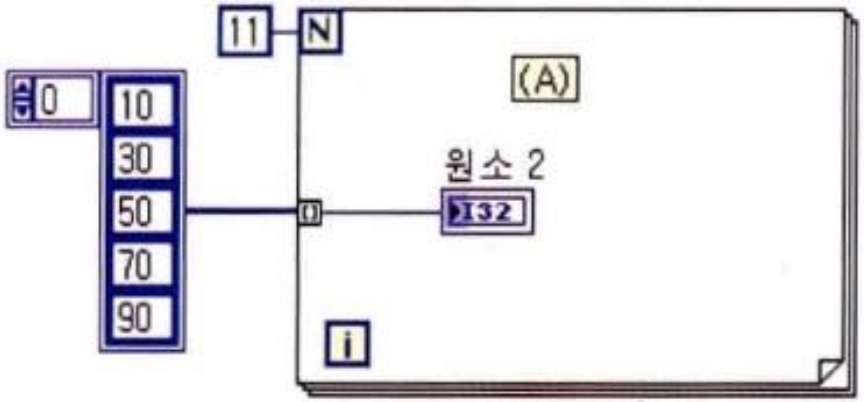
❖ For 루프를 이용하여 배열의 원소를 푸는 예

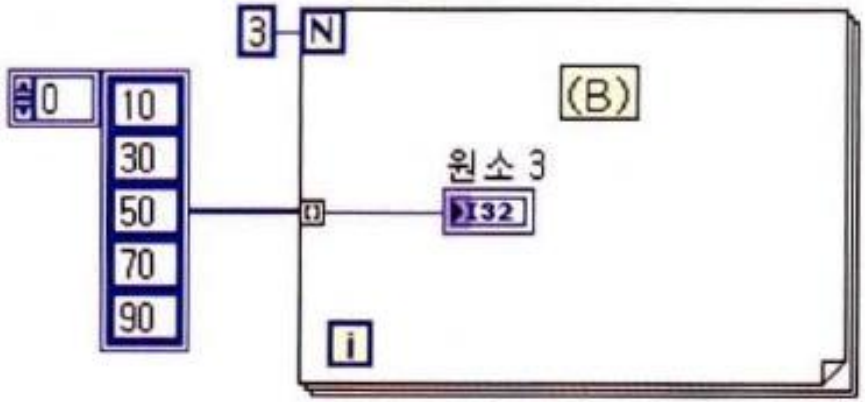




배열과 For루프

❖ 그림에서 (A)와 (B)는 각각 For 루프가 몇 번 반복할까?





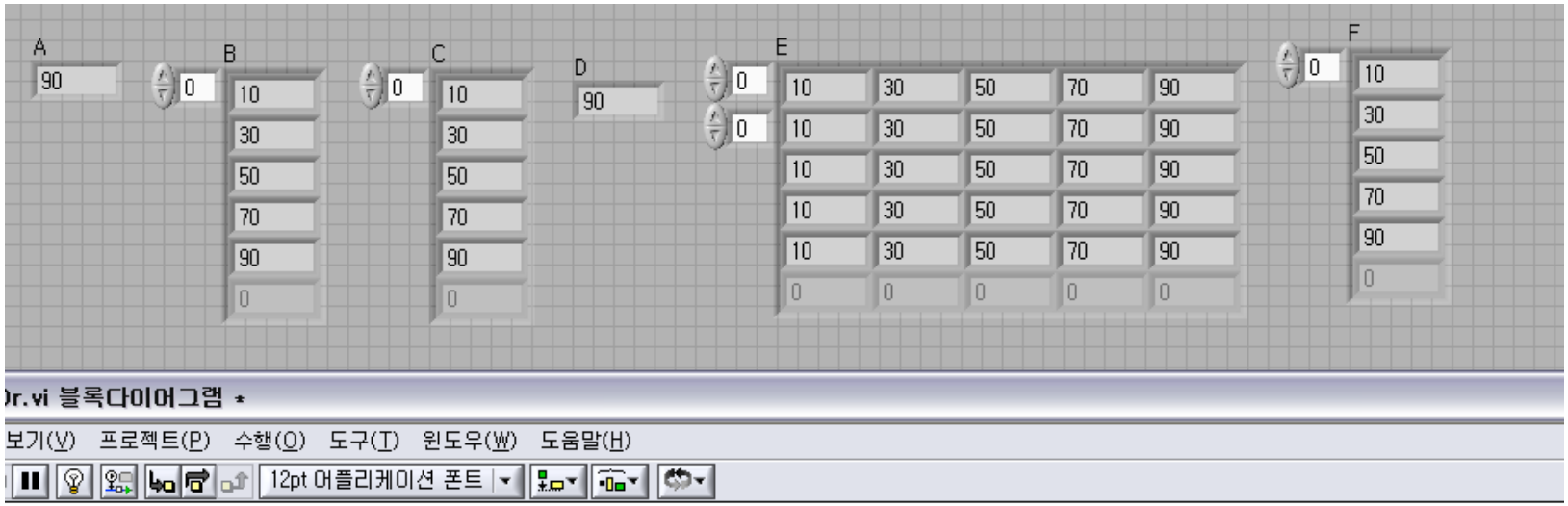
❖ N에 연결된 값과 배열의 원소 개수 중 작은쪽에 영향을 받는다.

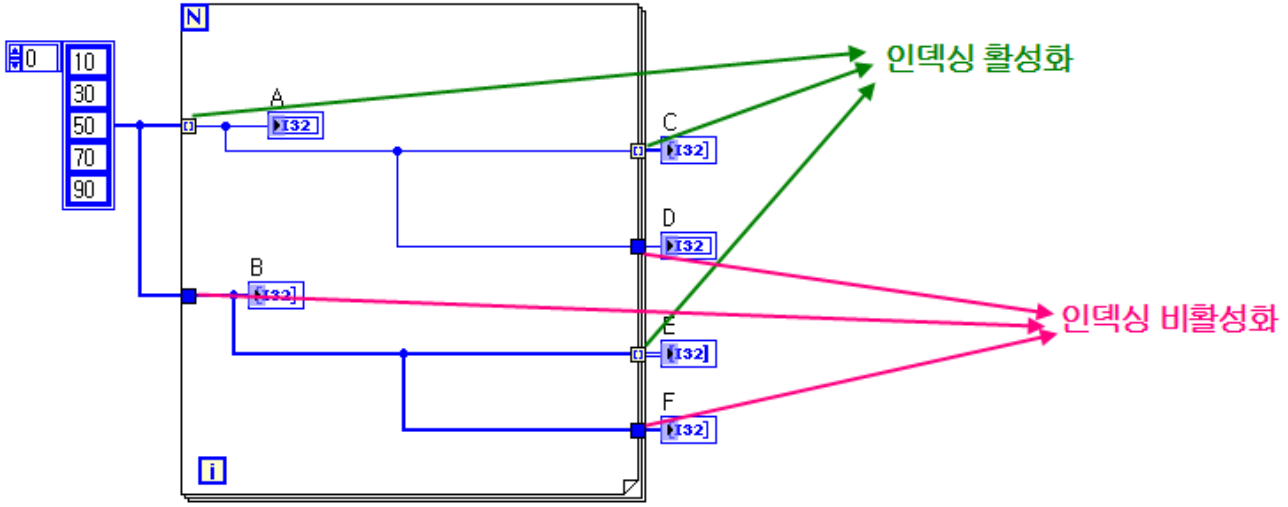
❖ 즉 (A)는 5번, (B)는 3번 반복

배열과 For루프

- 아래 그림에서 입출력 터널에서 인덱싱 활성화/비활성화 설정에 따라서 결과값이 달라지는 것을 확인해 볼 수 있다.
- A는 For 루프가 5번 반복하면서 차례차례 원소값이 입력될 것이며, B는 5번 반복하는 동안 입력 배열 통째로 입력될 것이고, C는 5번 반복을 다하고 나서 10, 30, 50, 70, 90을 원소로 가진 배열이 출력될 것이며, D는 마지막 출력 때 발생한 90만 출력된다.
- 그리고 E는 10, 30, 50, 70, 90 이 5 번 반복이 되어 2 차원 배열로 출력되고 마지막으로 F는 마지막 반복, 즉 5 번째에 B에서 나온 값이 그대로 출력이 되어 10, 30, 50, 70, 90이 출력된다.

배열과 For루프

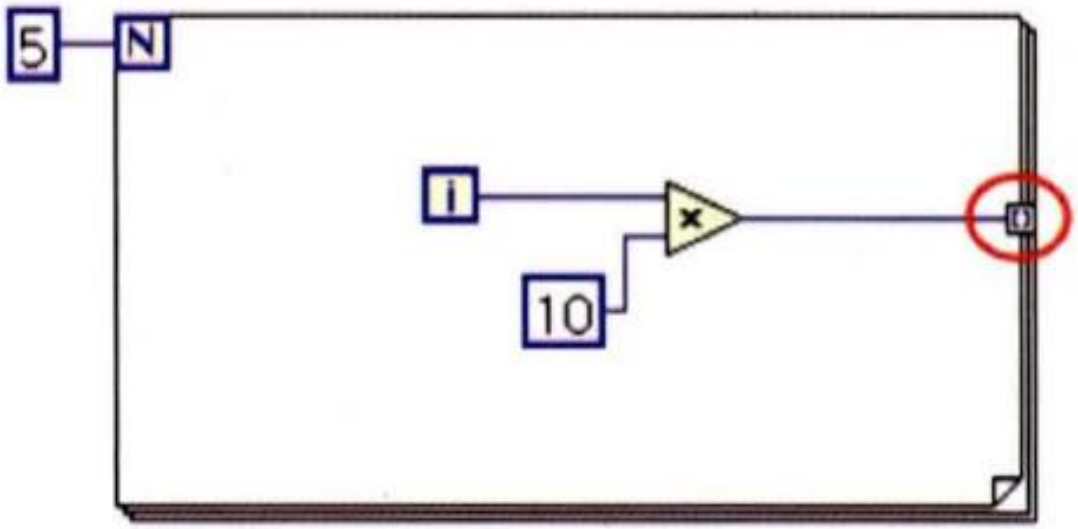

A
90
B
0
10
30
50
70
90
0
C
0
10
30
50
70
90
0
D
90
E
0
0
10 30 50 70 90
F
0
10
30
50
70
90
0
r.vi 블록다이어그램 *
보기(V) 프로젝트(P) 수행(O) 도구(T) 윈도우(W) 도움말(H)
12pt 어플리케이션 폰트


N
0
10
30
50
70
90
A
I32
B
I32
C
I32
D
I32
E
I32
F
I32
i
인덱싱 활성화
인덱싱 비활성화

실습4-6-1) 배열과 FOR 루프

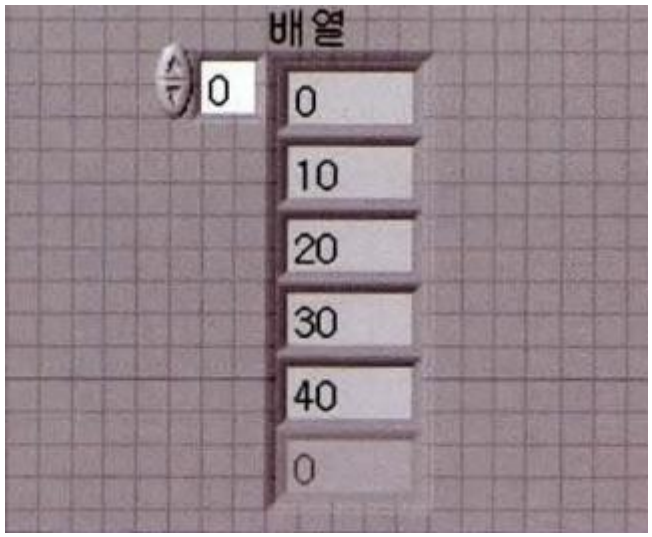
실습4-6-1) 배열과 FOR 루프

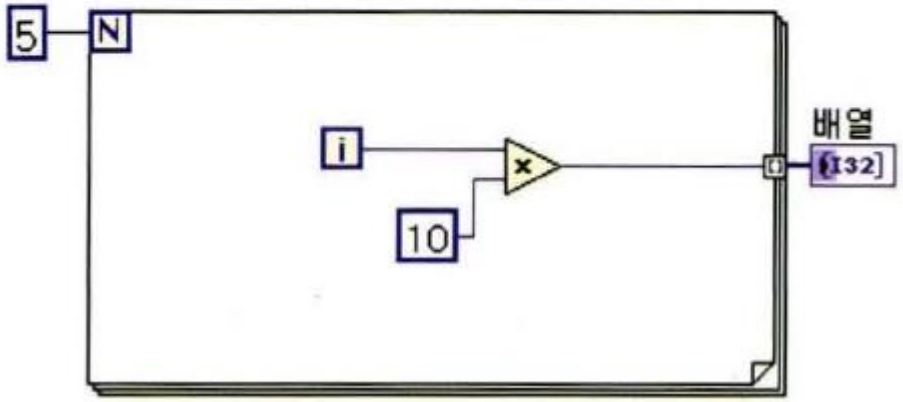
- 새 VI를 열어 블록다이어그램을 그림과 같이 구성
- 출력 터널은 인덱싱 활성화로 설정



실습4-6-1) 배열과 FOR 루프

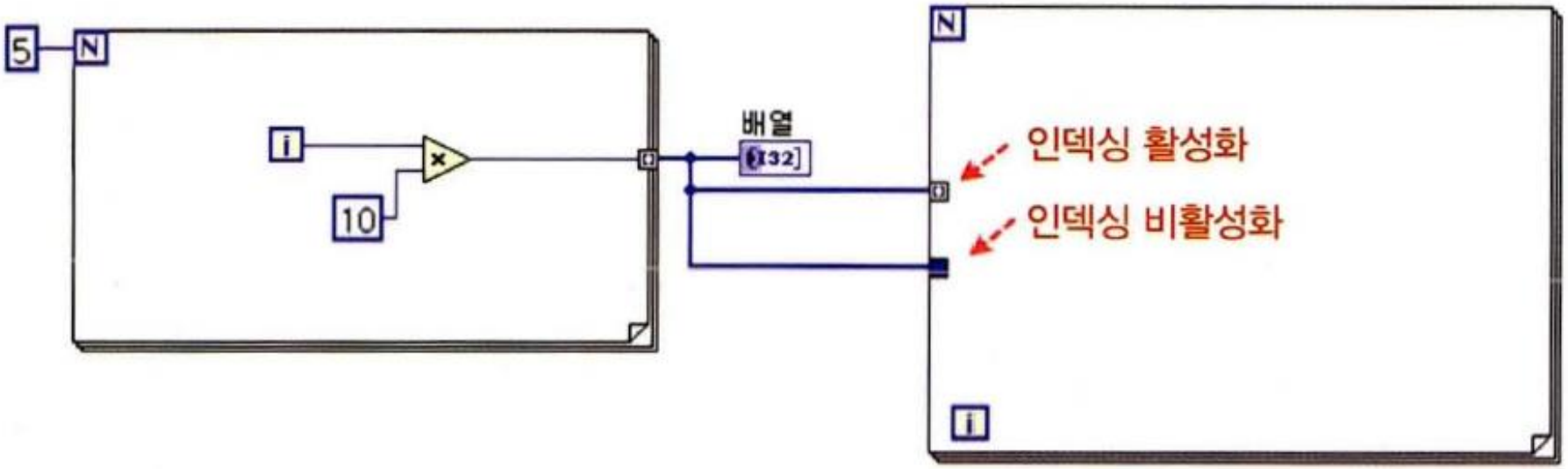
- 출력 터널에서 바로가기메뉴 > 생성 > 인디케이터를 선택.
- 프런트패널로 가서 실행을 하고 결과를 확인





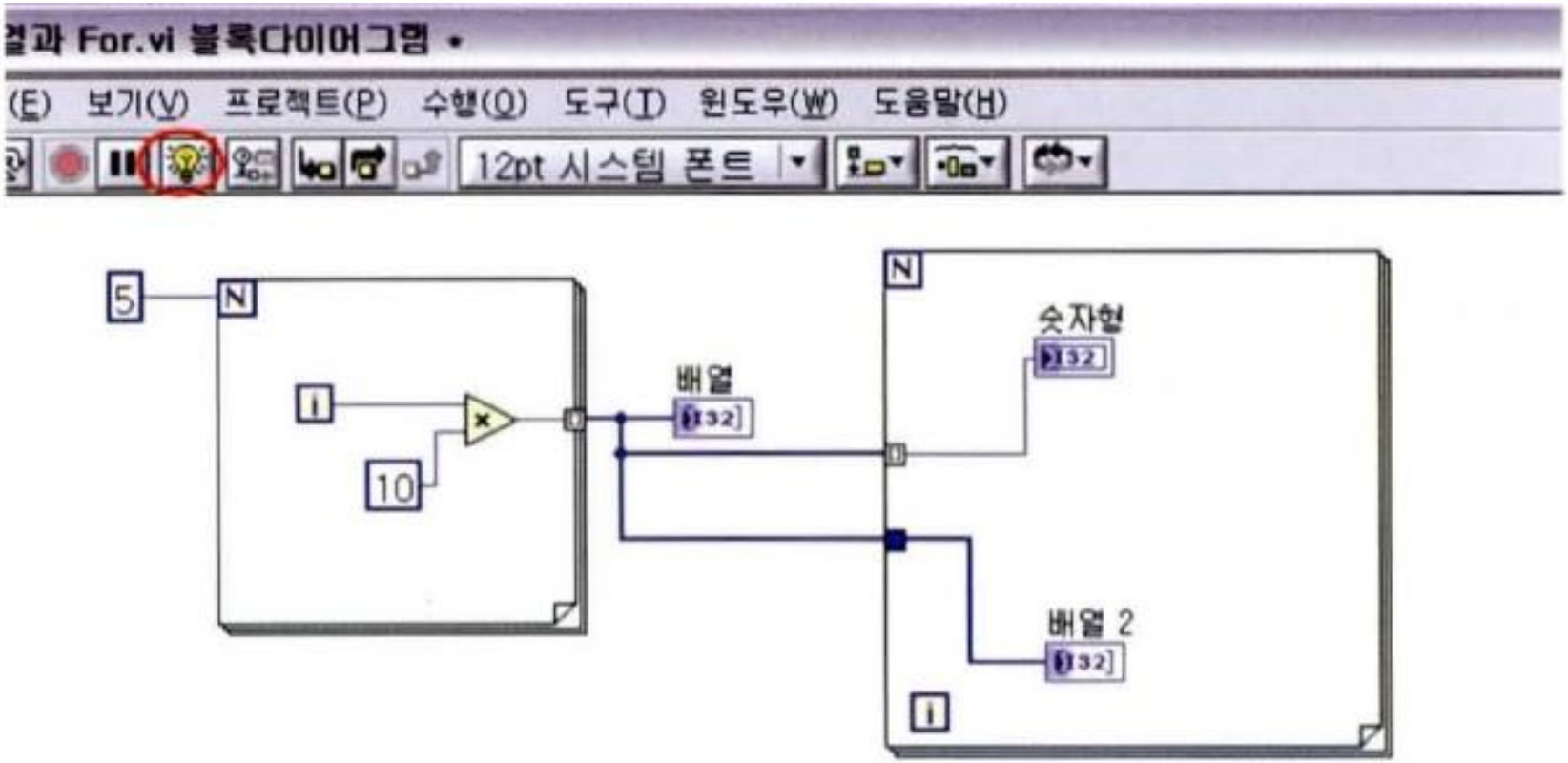
실습4-6-1) 배열과 FOR 루프

- 블록다이어그램에서 **For** 루프를 추가하여 그림처럼 와이어링
- 두 개의 입력 터널을 하나는 인덱싱 활성화, 나머지 하나는 인덱싱 비활성화를 선택



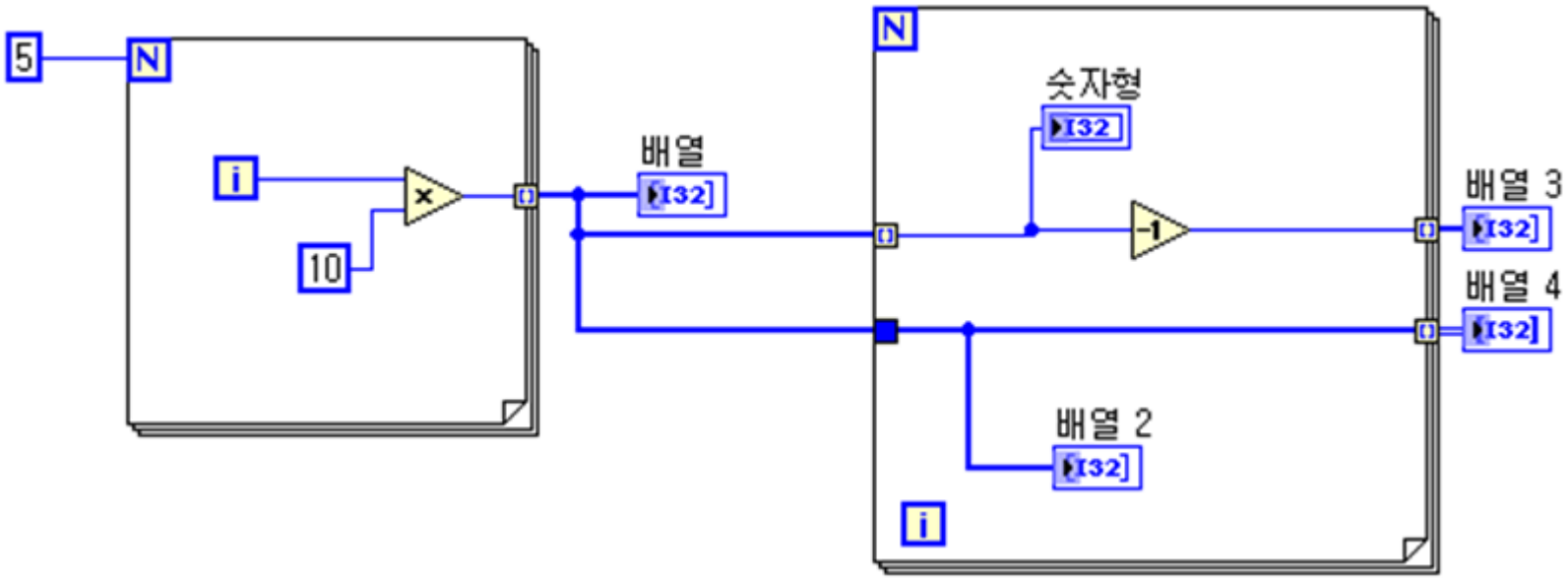
실습4-6-1) 배열과 FOR 루프

❖ 두 개의 입력 터널에서 바로가기메뉴 > 생성 > 인디케이터를 선택한 후 실행 하이라이트를 켜고 실행



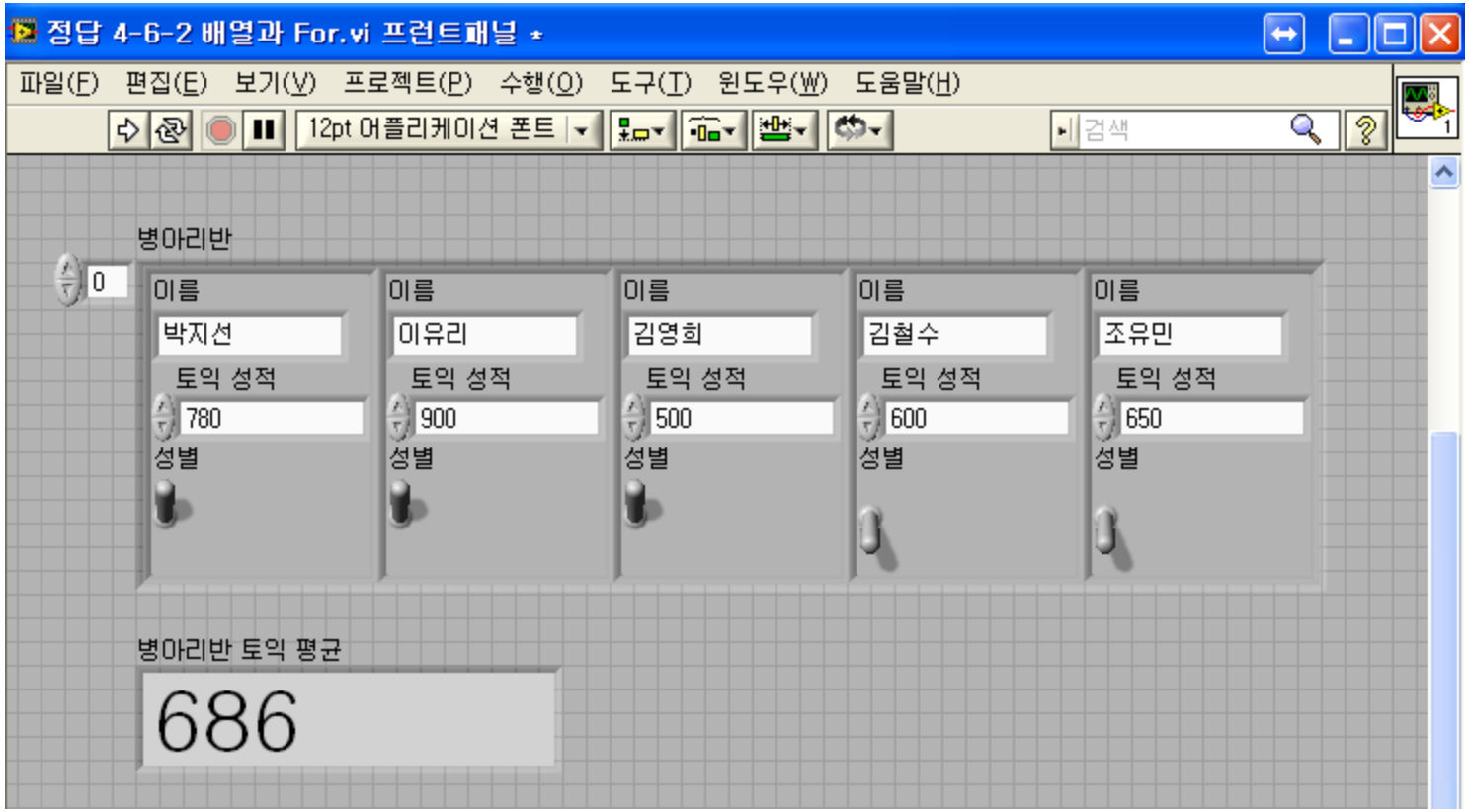
실습4-6-1) 배열과 FOR 루프

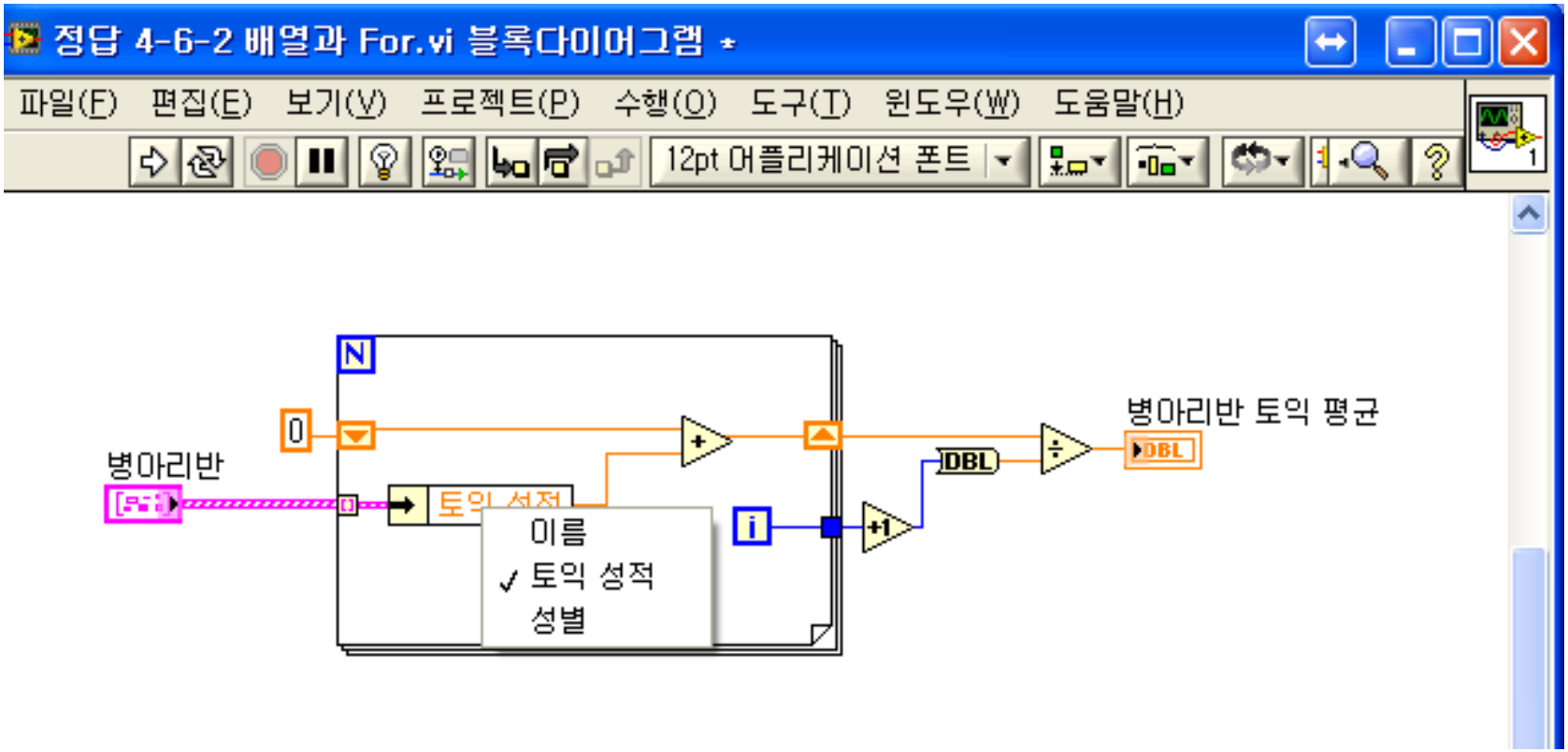
- 블록다이어그램을 그림과 같이 수정하여 두 번째 For 루프의 출력 터널에서 각각 바로가기메뉴 >생성 > 인디케이터를 선택
- 실행하고 프런트패널에서 ' 배열 3'과 ' 배열 4'의 결과 값을 확인




정답 4-6-1 배열과 For.vi 프런트패널 *
파일(F) 편집(E) 보기(V) 프로젝트(P) 수행(O) 도구(T) 윈도우(W) 도움말(H)
12pt 어플리케이션 폰트
검색
배열
숫자형
40
배열 2
배열 3
배열 4
0
10
20
30
40
-1
9
19
29
39

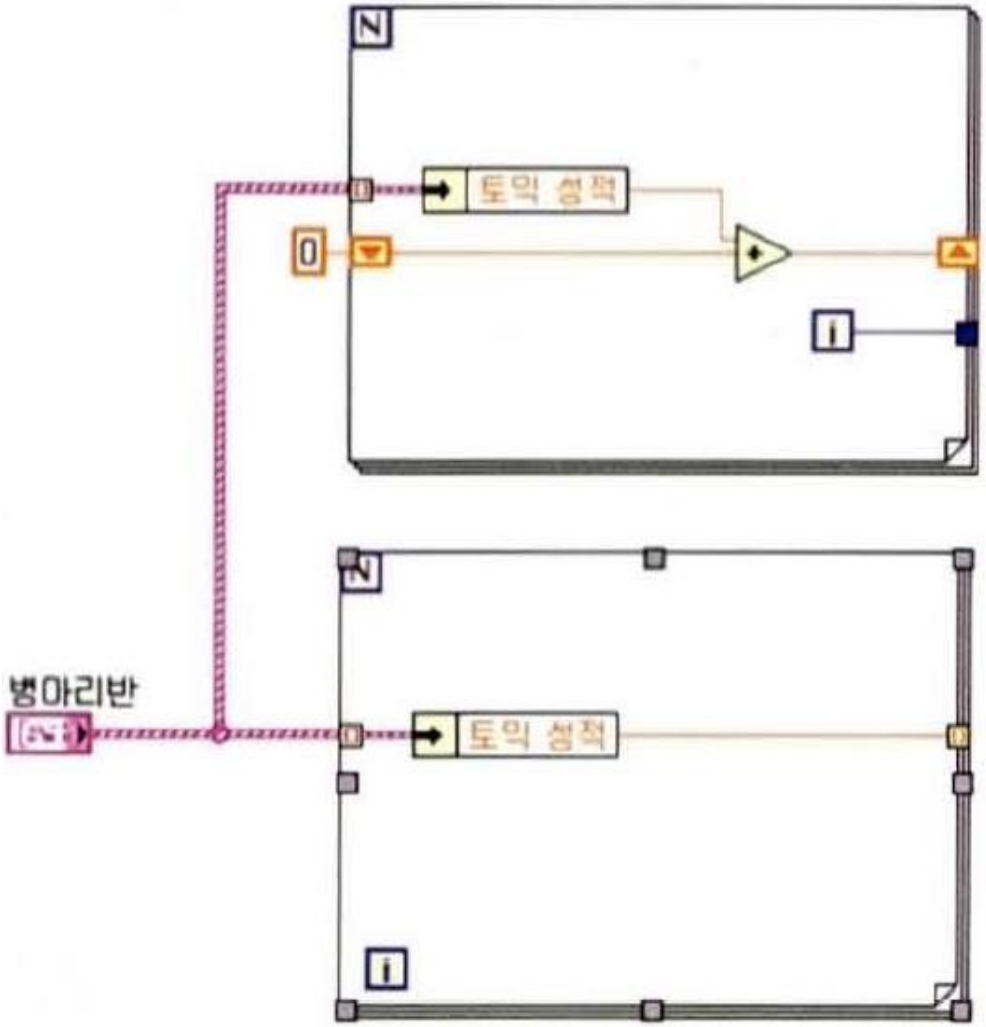
실습4-6-2) 배열과 FOR 루프


정답 4-6-2 배열과 For.vi 프런트패널 *
파일(F) 편집(E) 보기(V) 프로젝트(P) 수행(O) 도구(T) 윈도우(W) 도움말(H)
12pt 어플리케이션 폰트
검색
병아리반
0
이름
박지선
토익 성적
780
성별
이름
이유리
토익 성적
900
성별
이름
김영희
토익 성적
500
성별
이름
김철수
토익 성적
600
성별
이름
조유민
토익 성적
650
성별
병아리반 토익 평균
686


정답 4-6-2 배열과 For.vi 블록다이어그램 *
파일(F) 편집(E) 보기(V) 프로젝트(P) 수행(O) 도구(T) 윈도우(W) 도움말(H)
12pt 어플리케이션 폰트
N
0
병아리반
토익 성적
이름
토익 성적
성별
i
+1
DBL
DBL
병아리반 토익 평균

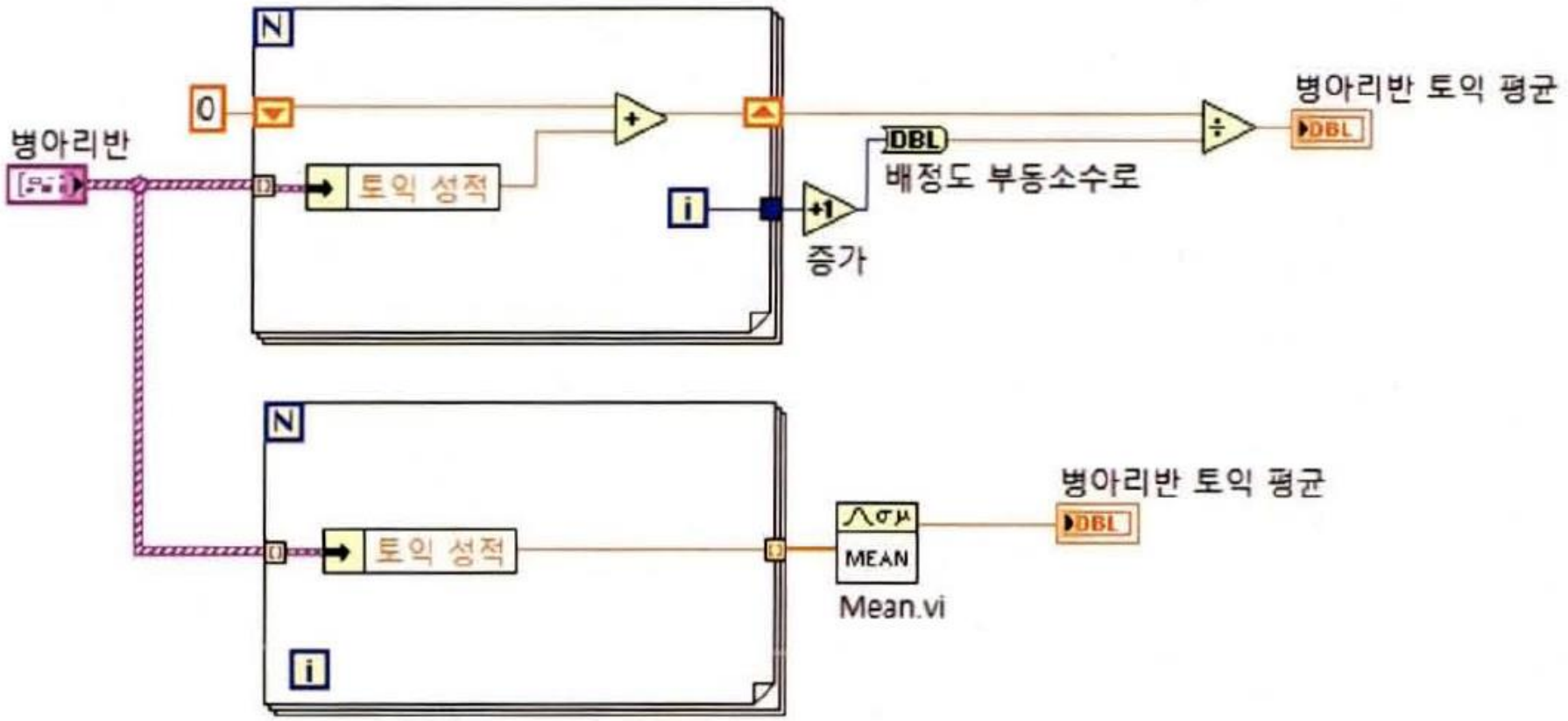
실습4-6-2) 배열과 FOR 루프

❖ 블록다이어그램을 그림과 같이 구성



실습4-6-2) 배열과 FOR 루프

❖ 병아리 반의 토익 성적의 총합계 점수를 학생 수로 나누어 평균을 구하거나 평균.VI를 사용

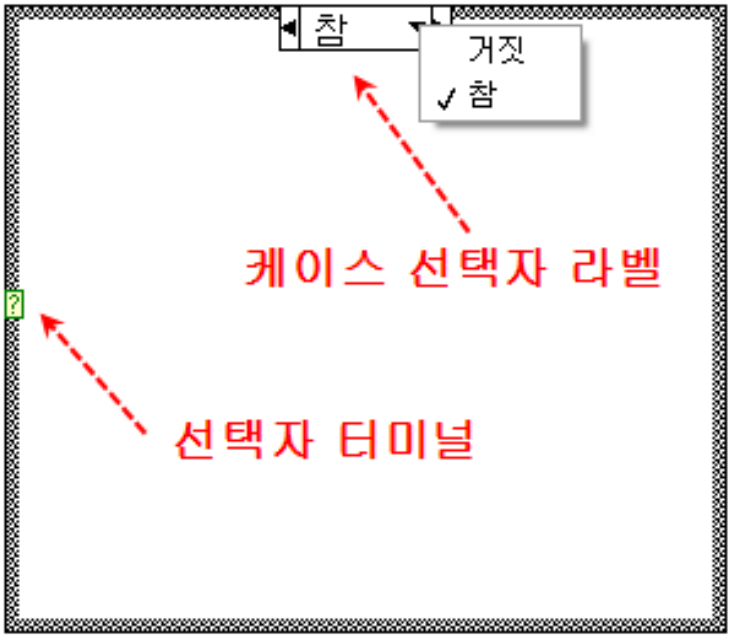


케이스 구조

- 프로그램에서 '+'가 입력되면 더하기 하는 코드를 실행하고 '-'가 입력되면 빼기하는 코드가 실행하게 하고 싶은 경우 어떻게 하면 좋을까?

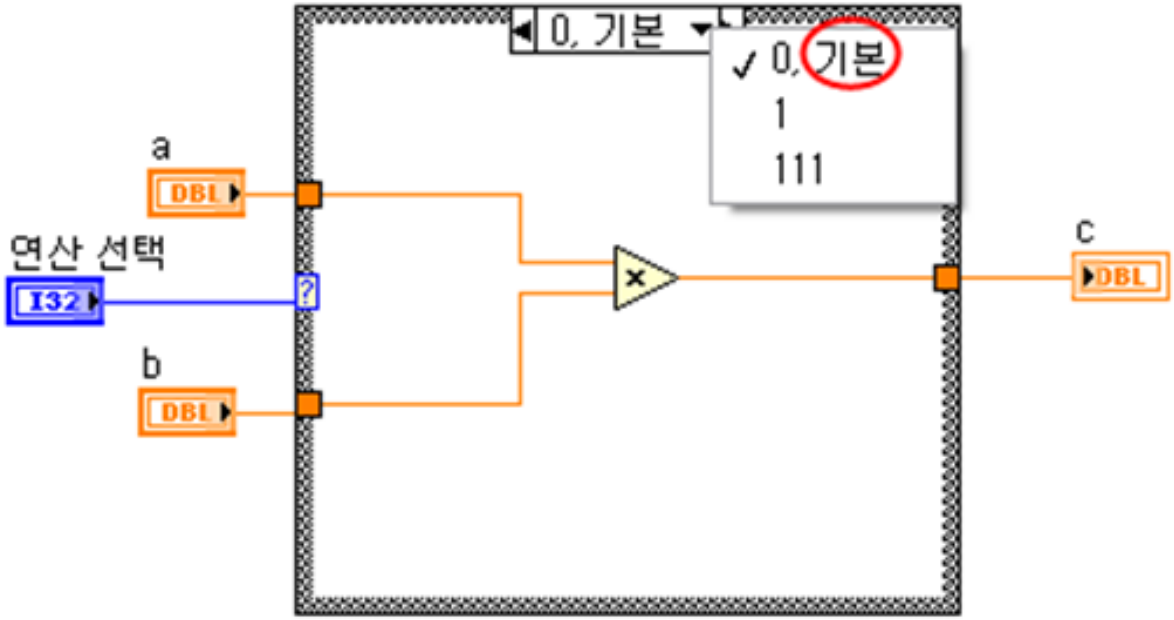
- 이처럼 입력하는 값에 따라 실행하는 코드의 내용이 달라질 경우 케이스 구조를 사용

케이스 구조



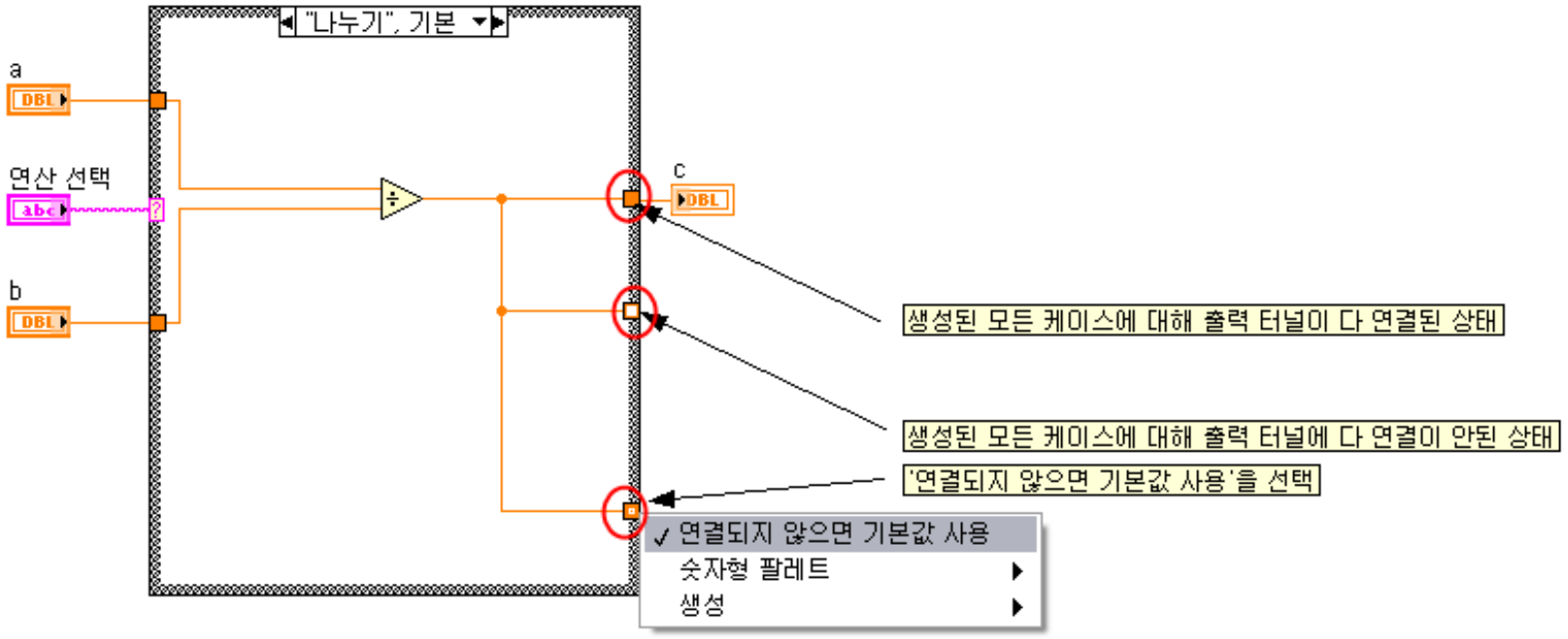
•케이스 구조 : 선택자 터미널 입력에 따라 실행되는 코드가 달라짐.

기본 케이스



•기본 케이스 : 예외 입력 때 실행되는 코드.

케이스 구조의 출력 터널

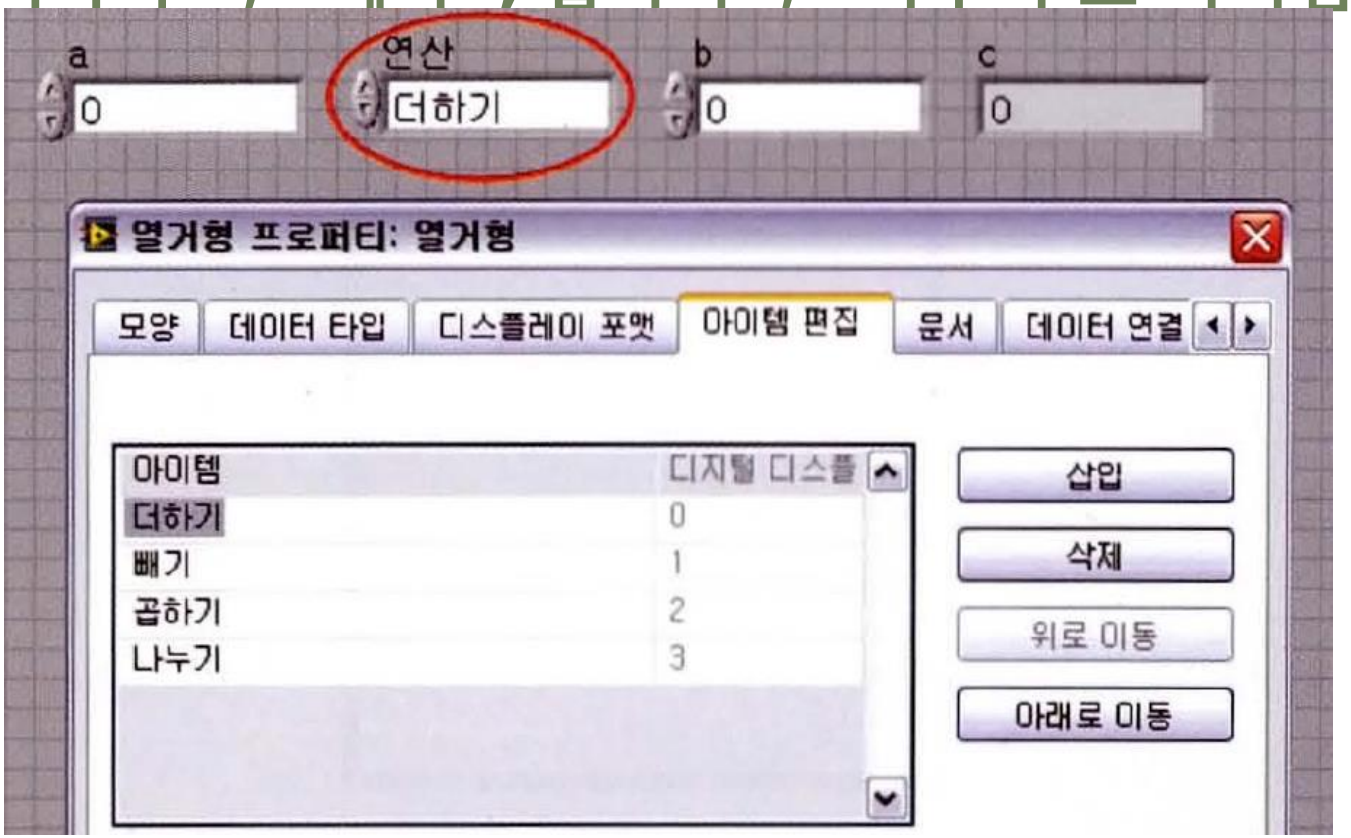


•케이스 출려 터널은 항상 막혀 있어야 함.

실습4-7-1) 케이스 구조 사용법

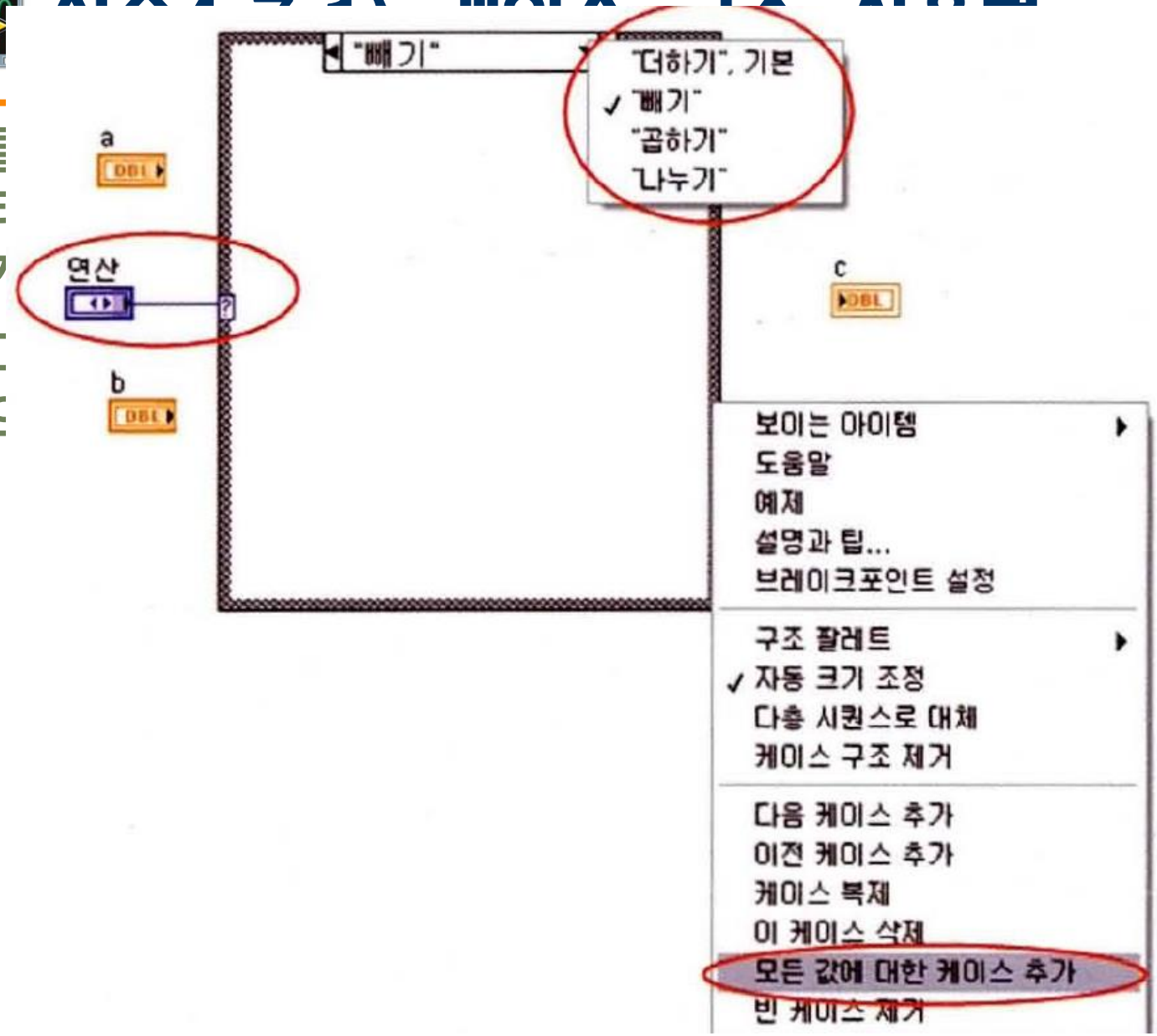
실습4-7-1) 케이스 구조 사용법

- 새 VI를 열어 프런트 패널에 숫자형 컨트롤 2개, 열거형 1개, 숫자형 인디케이터를 그림과 같이 구성
- 이 때 열거형은 바로가기 메뉴 > 아이템 편집 ... 을 이용하여 ‘더하기’, ‘빼기’, 곱하기’, ‘나누기’로 아이템을 편집



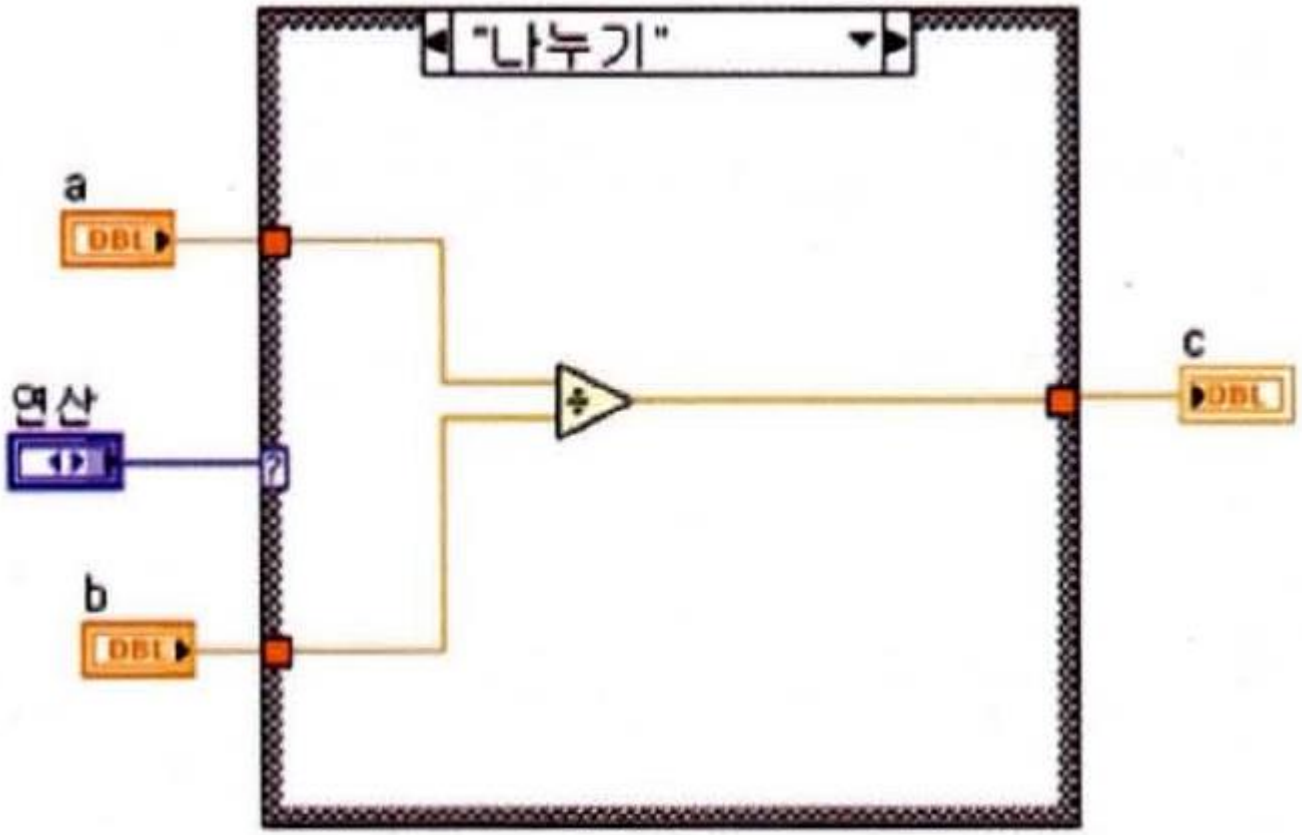
실습4-7-1) 케이스 구조 사용법

- 블록다이어그램에서 케이스 구조를 가져다 놓고 선택자 터미널에 '연산 ' 터미널을 연결한 후 케이스 구조의 바로가기메뉴 > 모든 값에 대한 케이스 추가를 선택.
- 그러면 열거형이 가진 아이템에 대한 모든 케이스가 자동으로 생성


"빼기"
"더하기", 기본
✓ "빼기"
"곱하기"
"나누기"
a
DBL
연산
b
DBL
c
DBL
보이는 아이템
도움말
예제
설명과 팁...
브레이크포인트 설정
구조 팔레트
✓ 자동 크기 조정
다층 시퀀스로 대체
케이스 구조 제거
다음 케이스 추가
이전 케이스 추가
케이스 복제
이 케이스 삭제
모든 값에 대한 케이스 추가
빈 케이스 제거

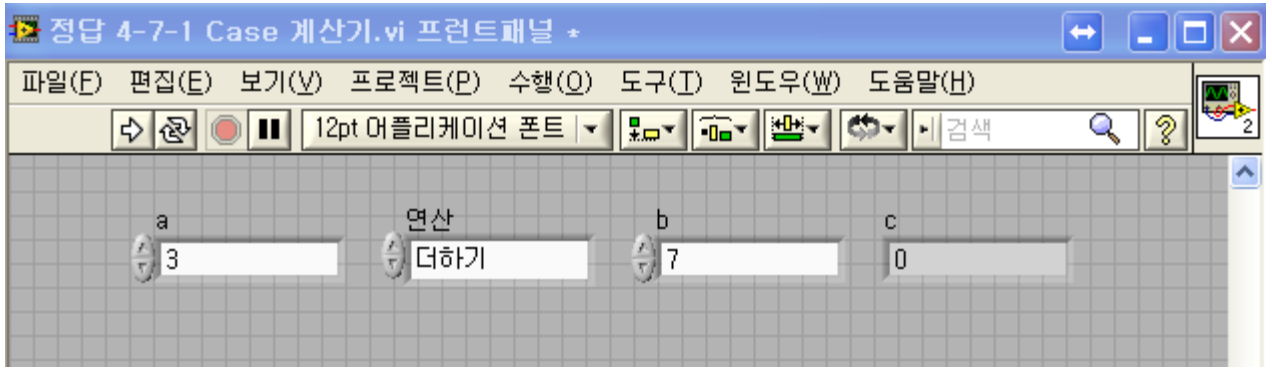
실습4-7-1) 케이스 구조 사용법

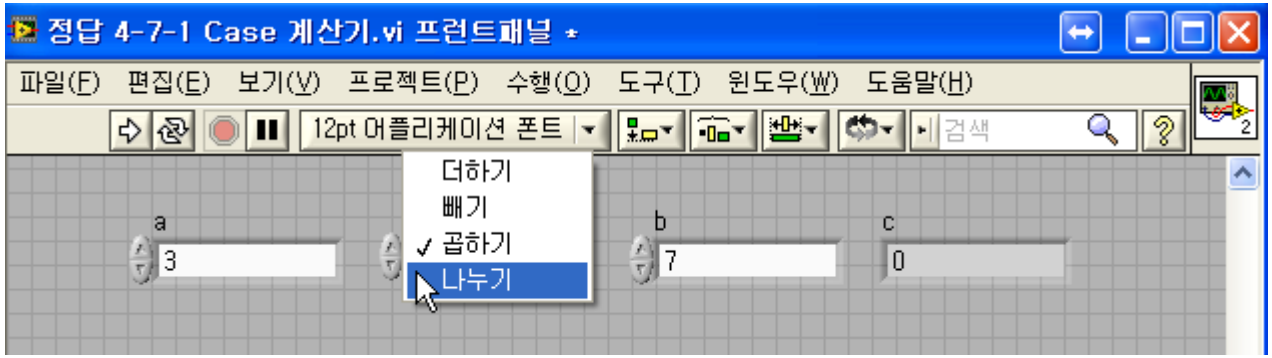
❖ 각 해당 케이스에 더하기.vi, 빼기.vi, 곱하기.VI, 나누기.VI를 넣어서 그림과 같이 와이어링

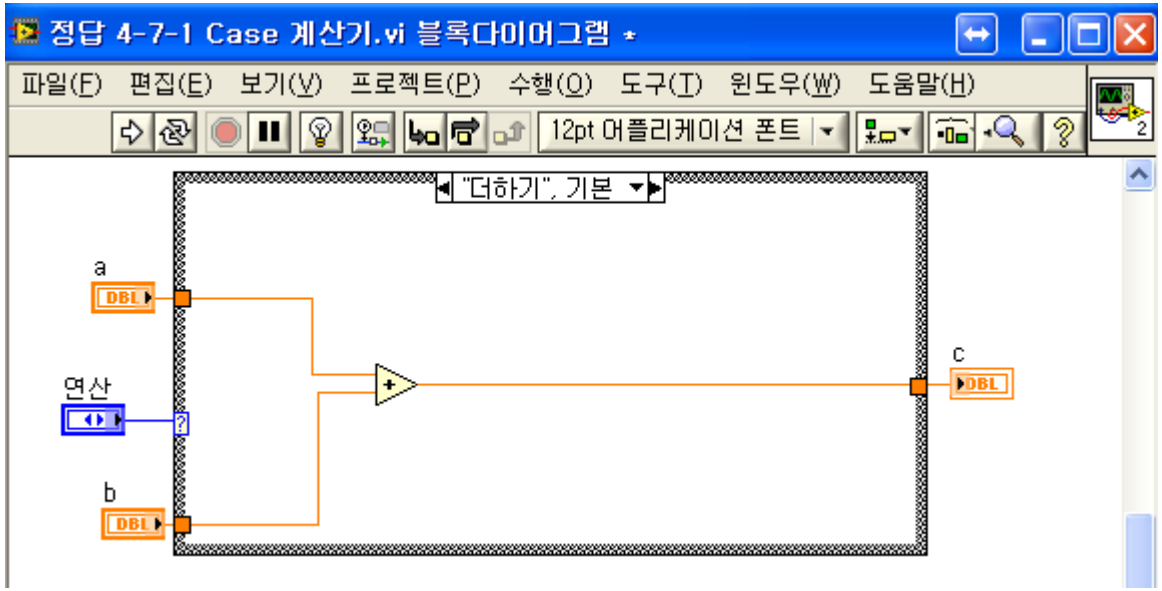


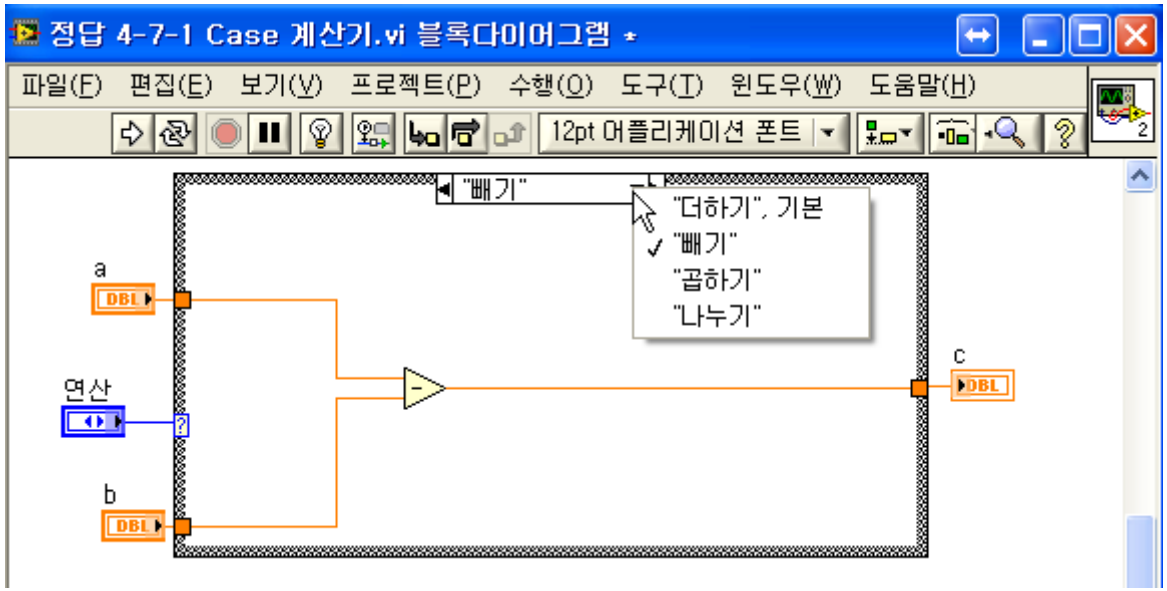
실습4-7-1) 케이스 구조 사용법

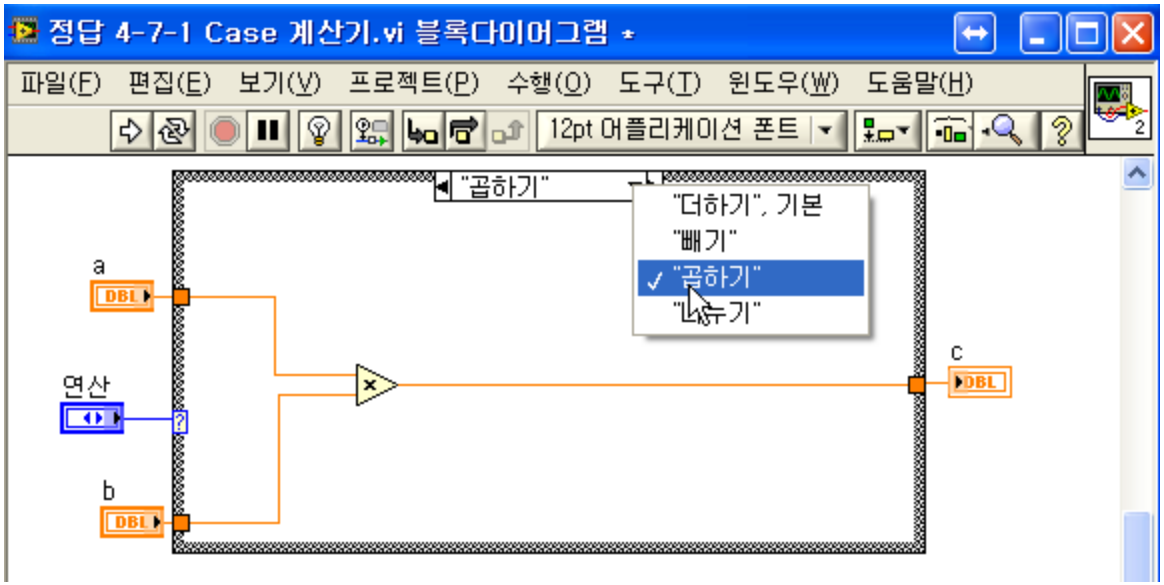
❖ 프런트패널 에서 'a'와 'b'에 적당한 값을 입력하고 실행하여 사칙연산이 되는지 확인

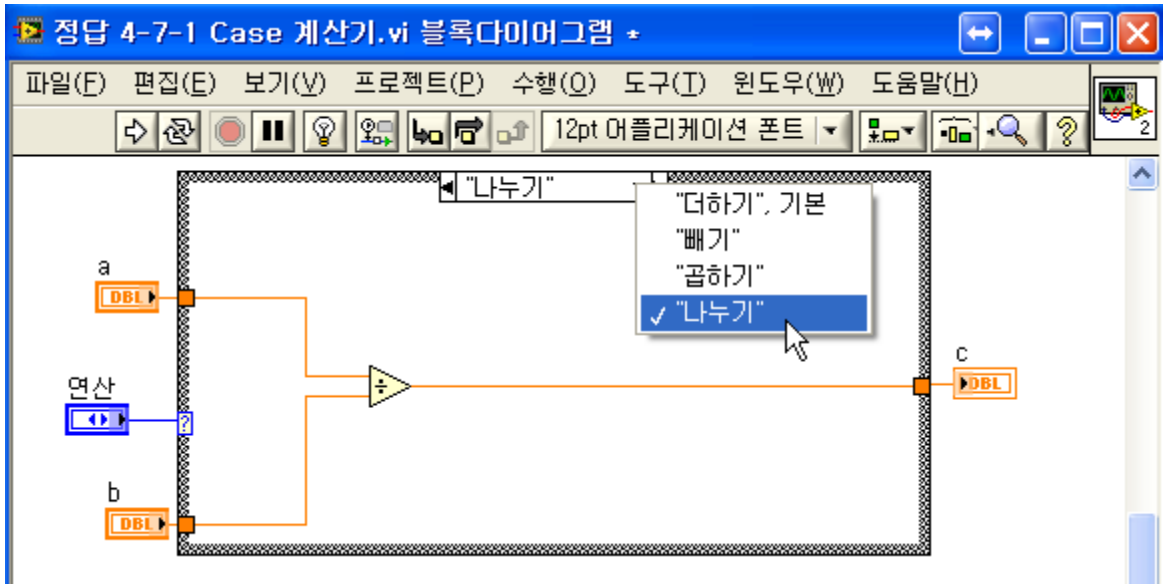




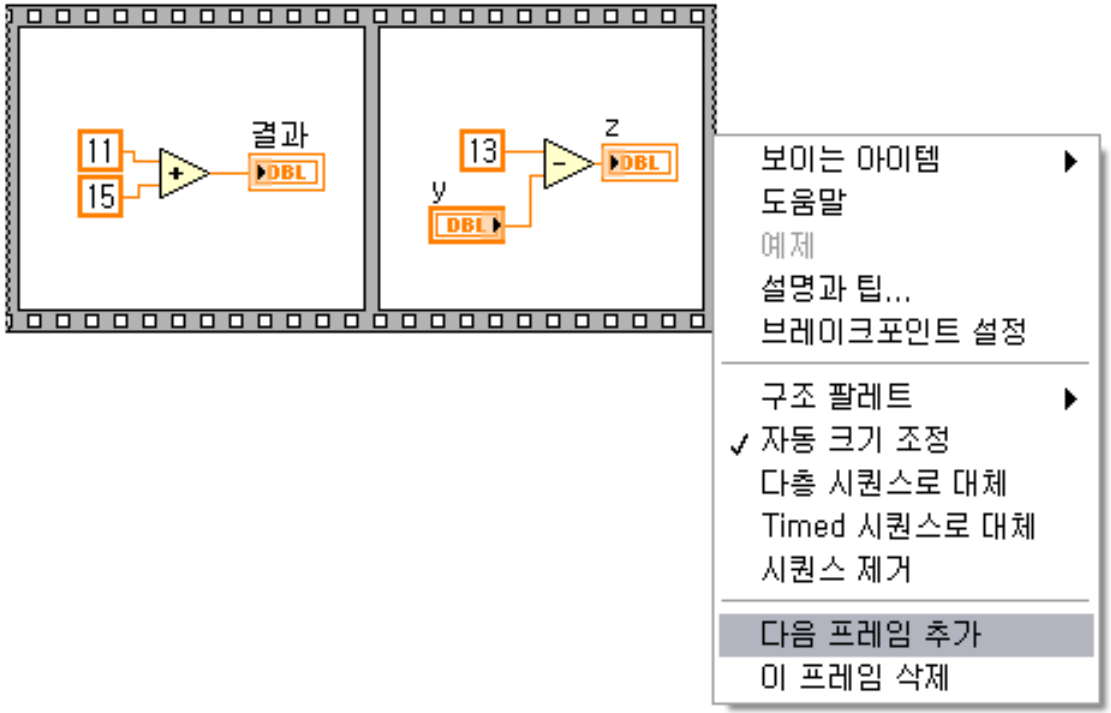

정답 4-7-1 Case 계산기.vi 블록다이어그램 *
파일(F) 편집(E) 보기(V) 프로젝트(P) 수행(O) 도구(T) 윈도우(W) 도움말(H)
12pt 어플리케이션 폰트
"더하기", 기본
a
DBL
연산
b
DBL
c
DBL


정답 4-7-1 Case 계산기.vi 블록다이어그램 *
파일(F) 편집(E) 보기(V) 프로젝트(P) 수행(O) 도구(T) 윈도우(W) 도움말(H)
12pt 어플리케이션 폰트
"빼기"
"더하기", 기본
"빼기"
"곱하기"
"나누기"
a
DBL
연산
b
DBL
c
DBL


정답 4-7-1 Case 계산기.vi 블록다이어그램 *
파일(F) 편집(E) 보기(V) 프로젝트(P) 수행(O) 도구(T) 윈도우(W) 도움말(H)
12pt 어플리케이션 폰트
"곱하기"
"더하기", 기본
"빼기"
"곱하기"
"나누기"
a
연산
b
c
DBL


정답 4-7-1 Case 계산기.vi 블록다이어그램 *
파일(F) 편집(E) 보기(V) 프로젝트(P) 수행(O) 도구(T) 윈도우(W) 도움말(H)
12pt 어플리케이션 폰트
"나누기"
"더하기", 기본
"빼기"
"곱하기"
"나누기"
a
연산
b
c
DBL

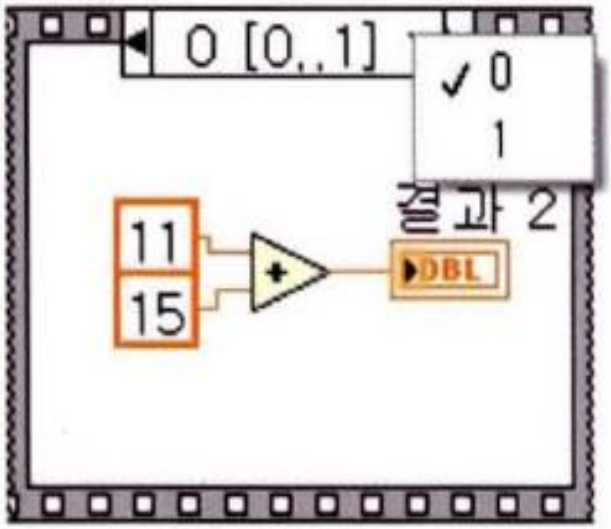
시퀀스 구조



•시퀀스 구조 : 강제적으로 실행 순서를 결정함.

시퀀스 구조

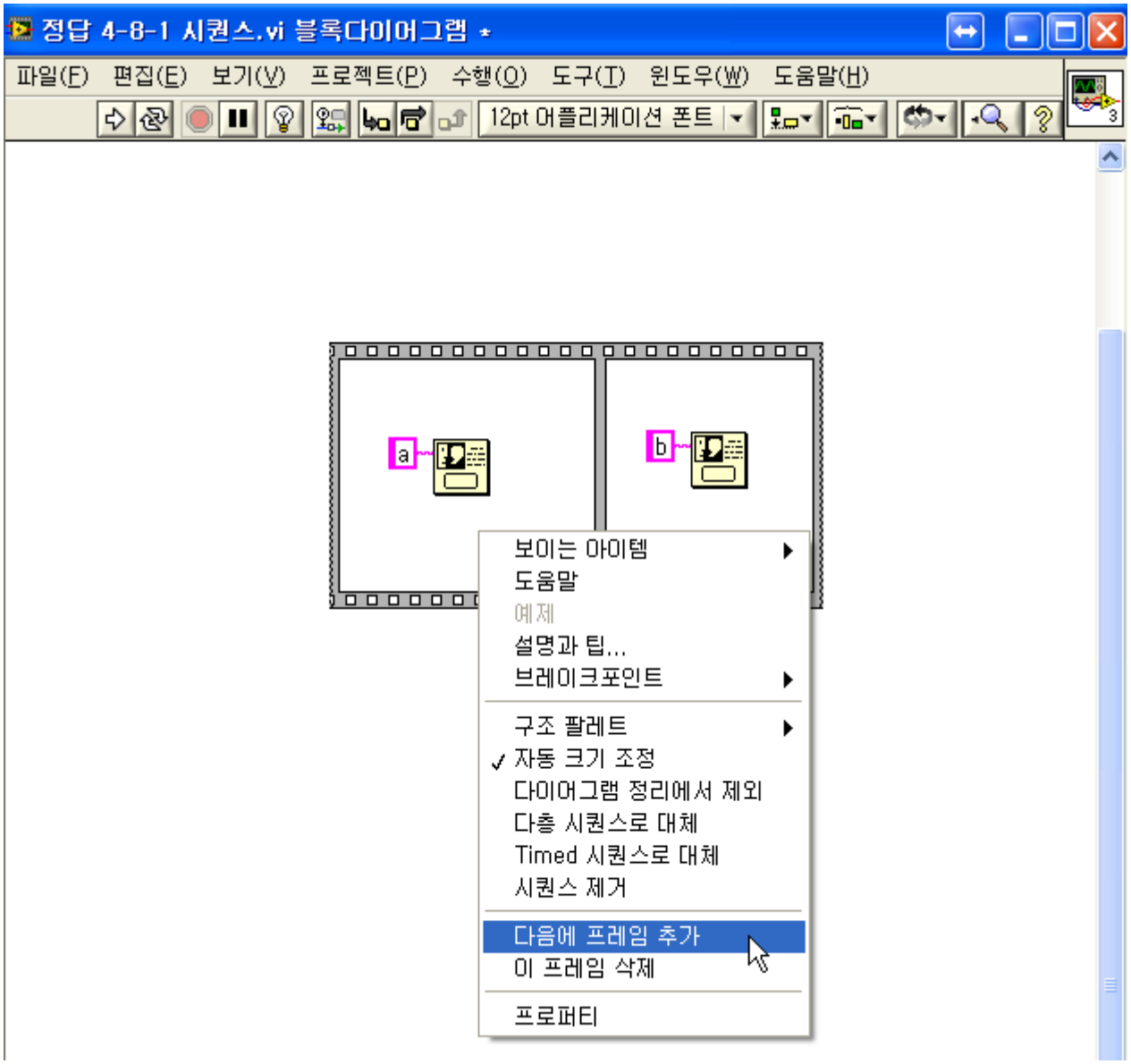
- 플랫 시퀀스 구조는 프레임이 다 펼쳐져 있어 코드를 한눈에 확인할 수 있어 좋긴 하지만 공간을 많이 차지
- 반면에 다층 시퀀스 구조는 그림처럼 프레임이 차곡차곡 쌓이게 된다.
- 다층 시퀀스 구조는 공간을 조금 차지하지만 코드가 한 눈에 보이지 않는 단점이 있다.



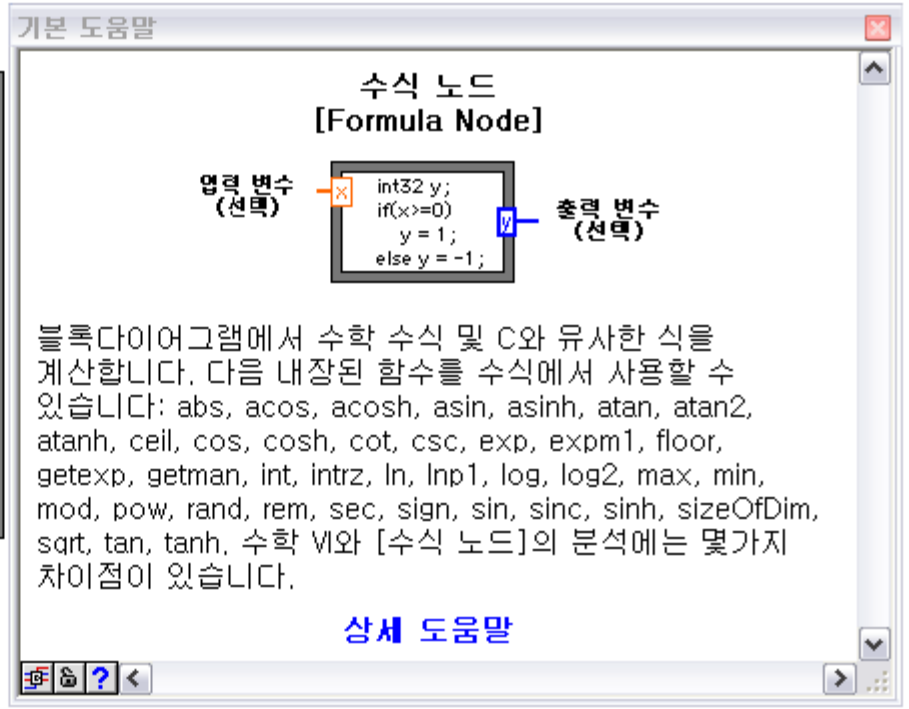
시퀀스 구조

- 시퀀스 구조를 시용할 때는 각 프레임들을 실행 하는 중에 에러가 발생할 경우 시퀀스를 중간에 빠져나올 수가 없다.
- 즉 가지고 있는 모든 프레임을 다 실행 해야지만 시퀀스 구조를 빠져나가게 된다.
- 시퀀스 구조를 사용할 때는 실행 중에 에러가 발생했을 경우를 대비하여 프로그램해야 한다.

실습4-8-1) 시퀀스 구조 사용법


정답 4-8-1 시퀀스.vi 블록다이어그램 *
파일(F) 편집(E) 보기(V) 프로젝트(P) 수행(O) 도구(T) 윈도우(W) 도움말(H)
12pt 어플리케이션 폰트
a
b
보이는 아이템
도움말
예제
설명과 팁...
브레이크포인트
구조 팔레트
자동 크기 조정
다이어그램 정리에서 제외
다층 시퀀스로 대체
Timed 시퀀스로 대체
시퀀스 제거
다음에 프레임 추가
이 프레임 삭제
프로퍼티

수식 노드



•수식 노드 : 복잡한 수식을 **C** 프로그래밍과 비슷한 기법으로 구현.

수식 노드

- 출력을 정의하는 방법은 수식 노드의 바로가기메뉴 > 입력 추가/출력 추가를 하면 원하는 개수만큼 입력과 출력을 정의할 수 있다.
- 수식 노드와 비슷한 기능을 하는 노드 중 수식 익스프레스 노드가 있다.

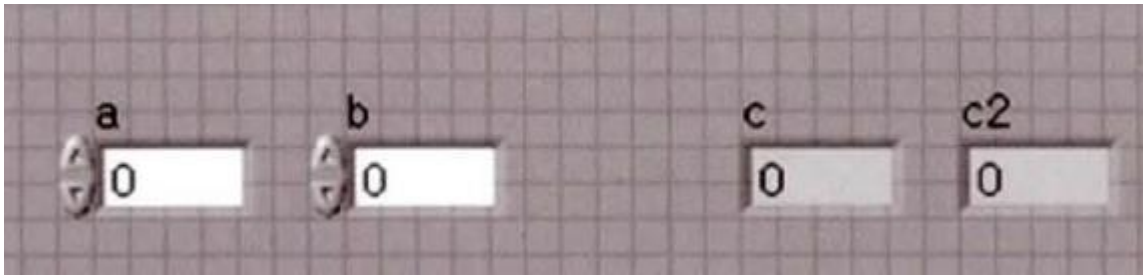
수식 노드

- 수식 익스프레스 노드는 그림과 같은 대화창을 통해 수식을 구현할 수 있다.
- 수식 노드와 가장 큰 차이점은 사용할 수 있는 입력은 최대 8개 출력은 1개로 정해져 있다는 점이다.

실습4-9-1) 수식 노드 사용법

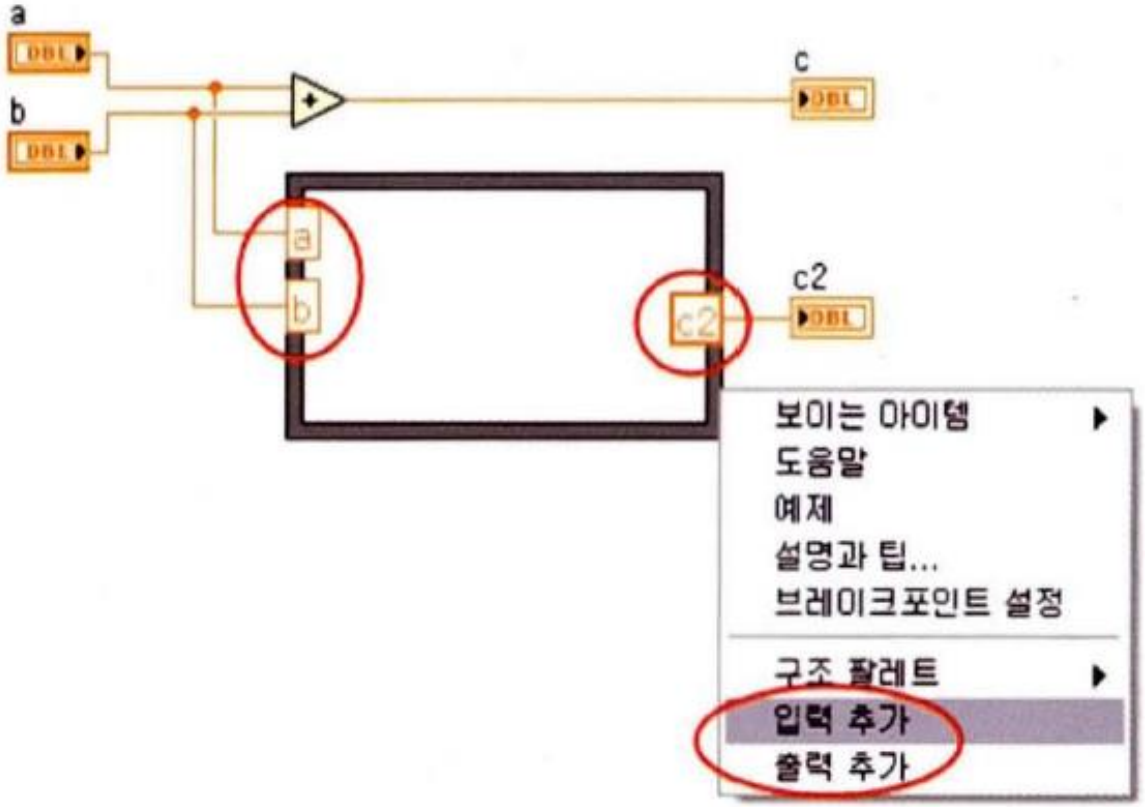
실습4-9-1) 수식 노드 사용법

❖ 새 VI를 열어서 그림과 같이 숫자형 컨트롤 2개와 숫자형 인디케이터 2개를 만든다.



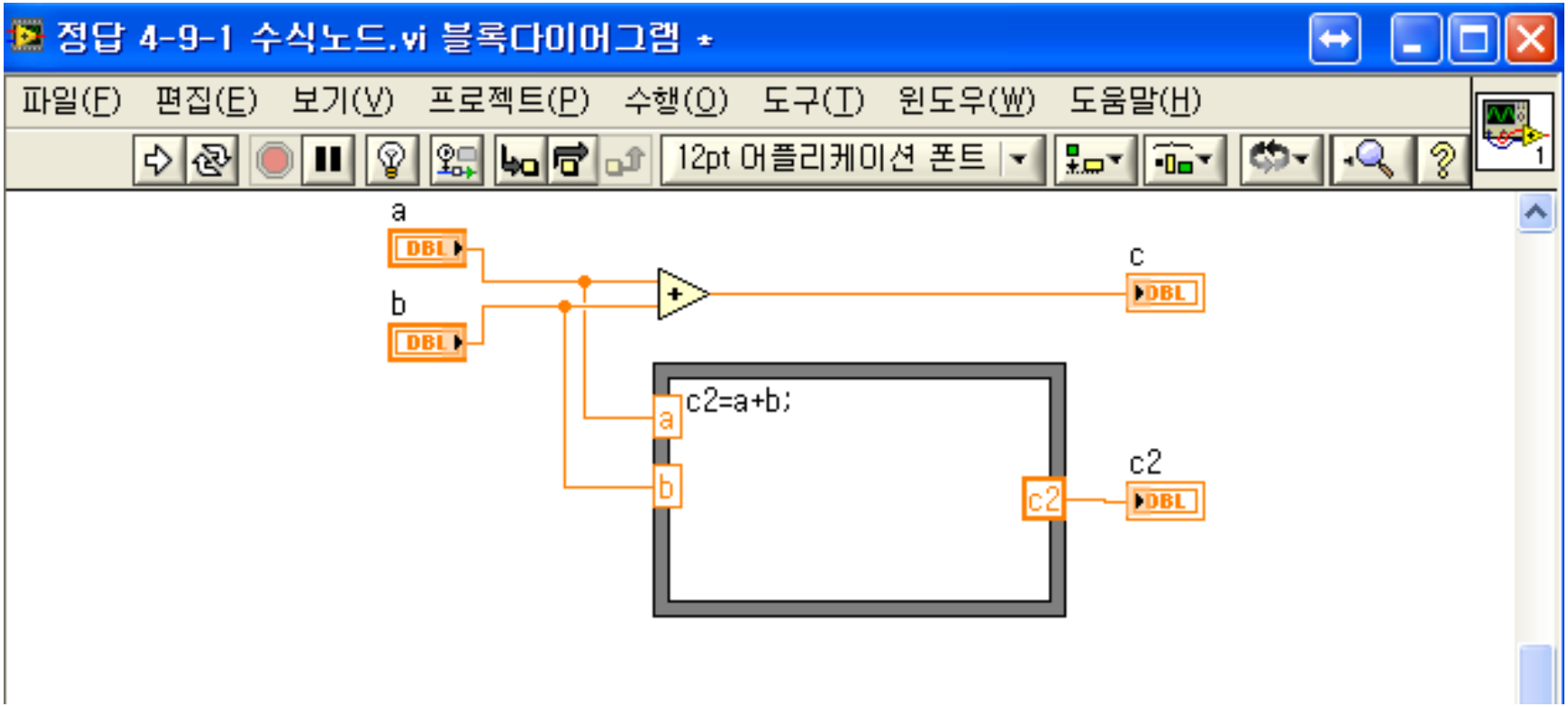
실습4-9-1) 수식 노드 사용법

- 블록다이어그램에서 더하기.VI와 수식 노드를 추가한다.
- 수식 노드의 바로가기메뉴 > 입력 추가/출력 추가를 선택하여 그림과 같이 와이어링한다.



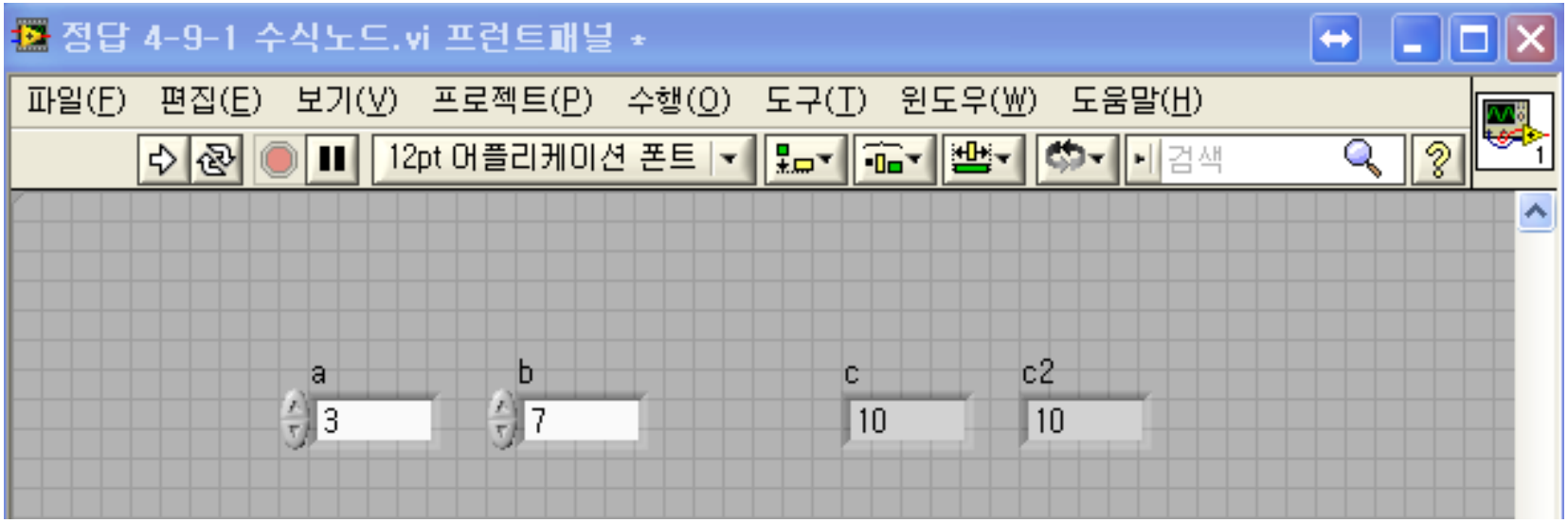
실습4-9-1) 수식 노드 사용법

❖ 수식 노드에 그림과 같이 수식을 구현한다

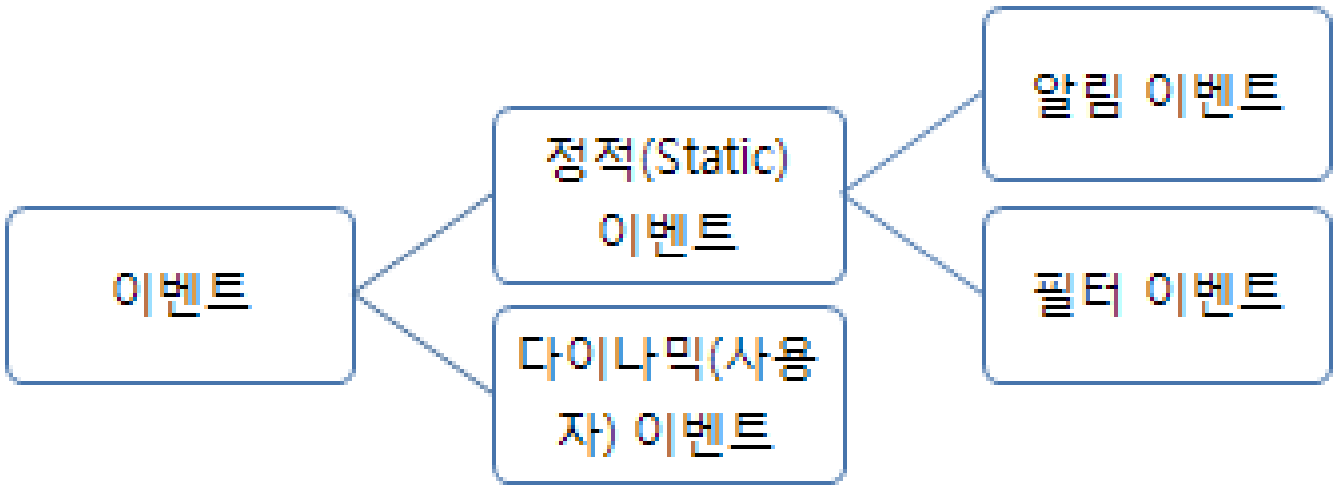


실습4-9-1) 수식 노드 사용법

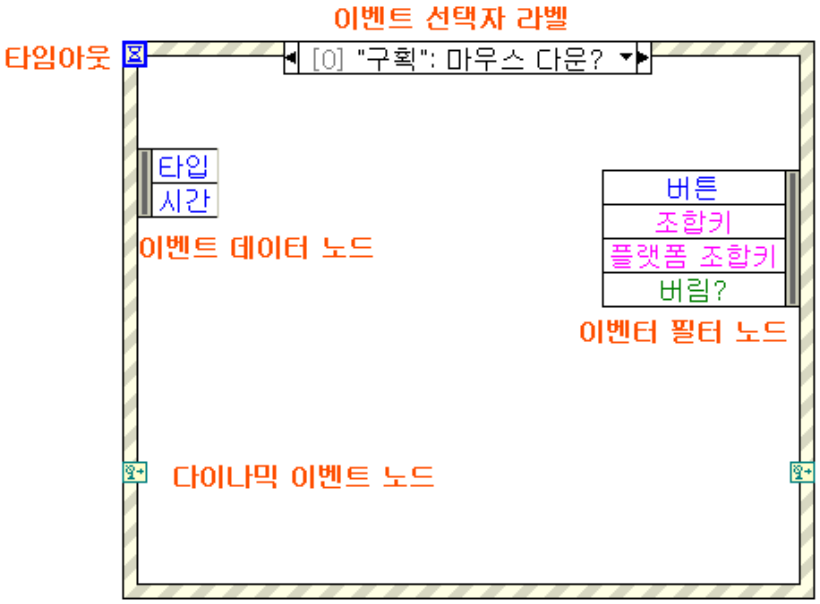
❖ ‘a’와 ‘ b’에 적당한 값을 입력하고 실행한 후, ‘c’와 ‘c2’의 값이 같은지 확인한다.



이벤트 구조 종류



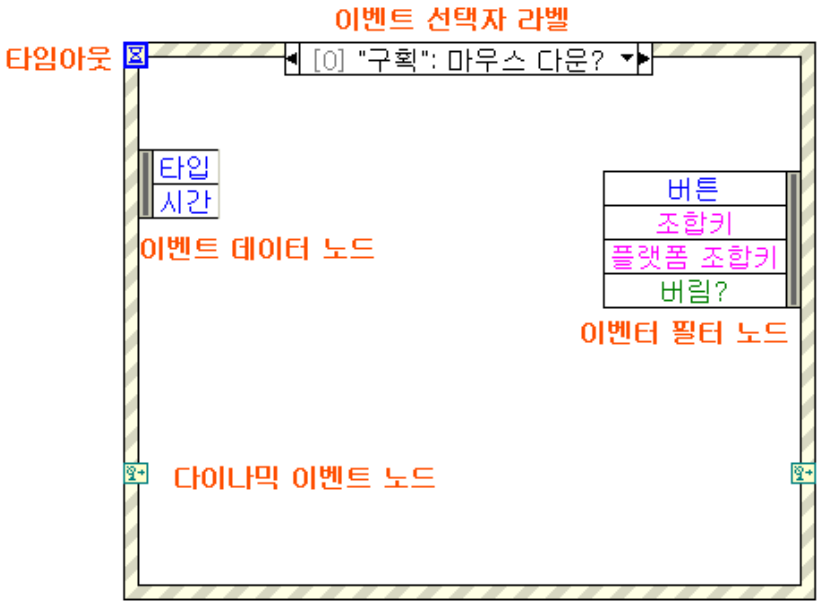
이벤트 구조



•이벤트 구조 : 마우스나 키보드를 통해 발생하는 이벤트에 동기화 하여 코드 실행.

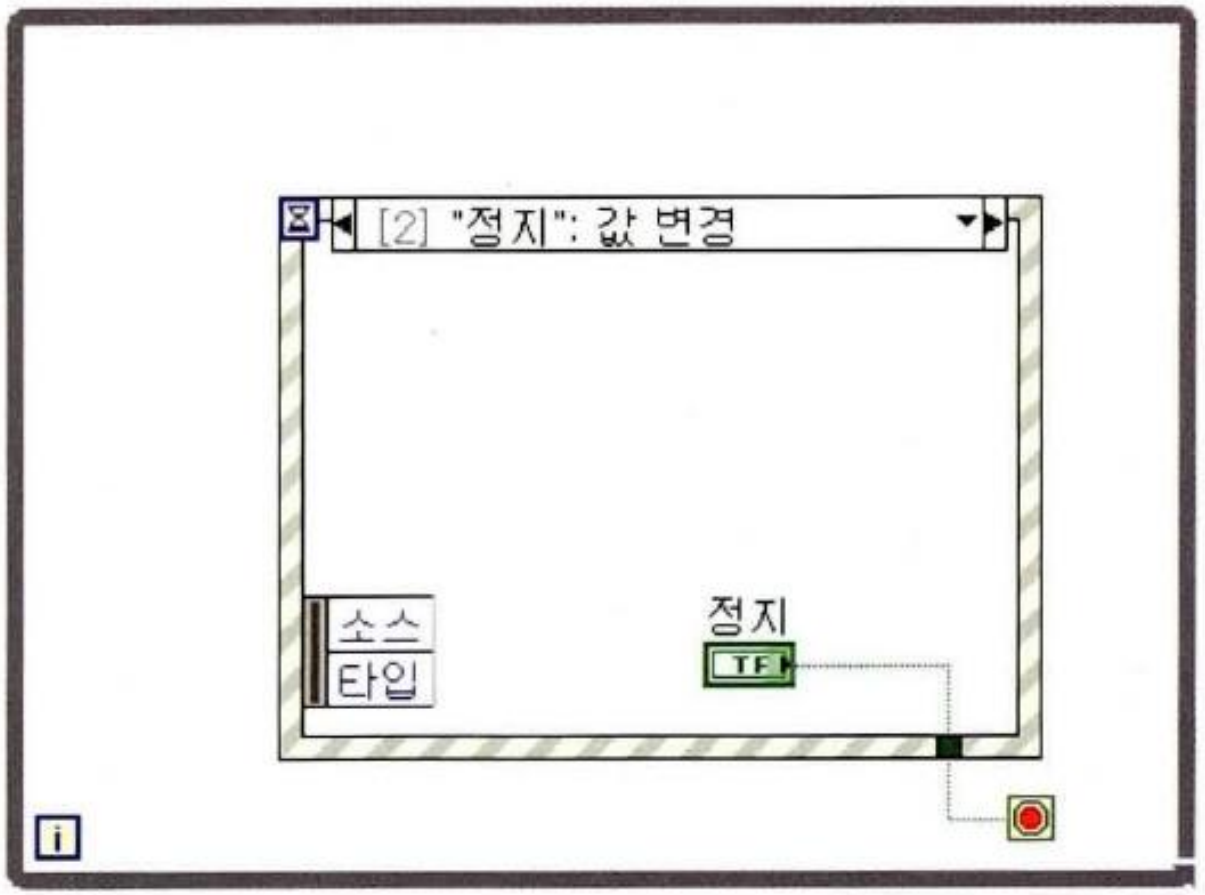
이벤트 구조

- 이벤트 선택자 라벨 : 어떤 이벤트인지를 구별
- 타임아웃 : 이벤트를 기다리는 시간(msec)
- 이벤트 데이터 노드 : 이벤트가 가지고 있는 속성들
- 이벤트 필터 노드 : 필터 이벤트가 가지고 있는 속성들. 필터 이벤트에서만 사용 가능



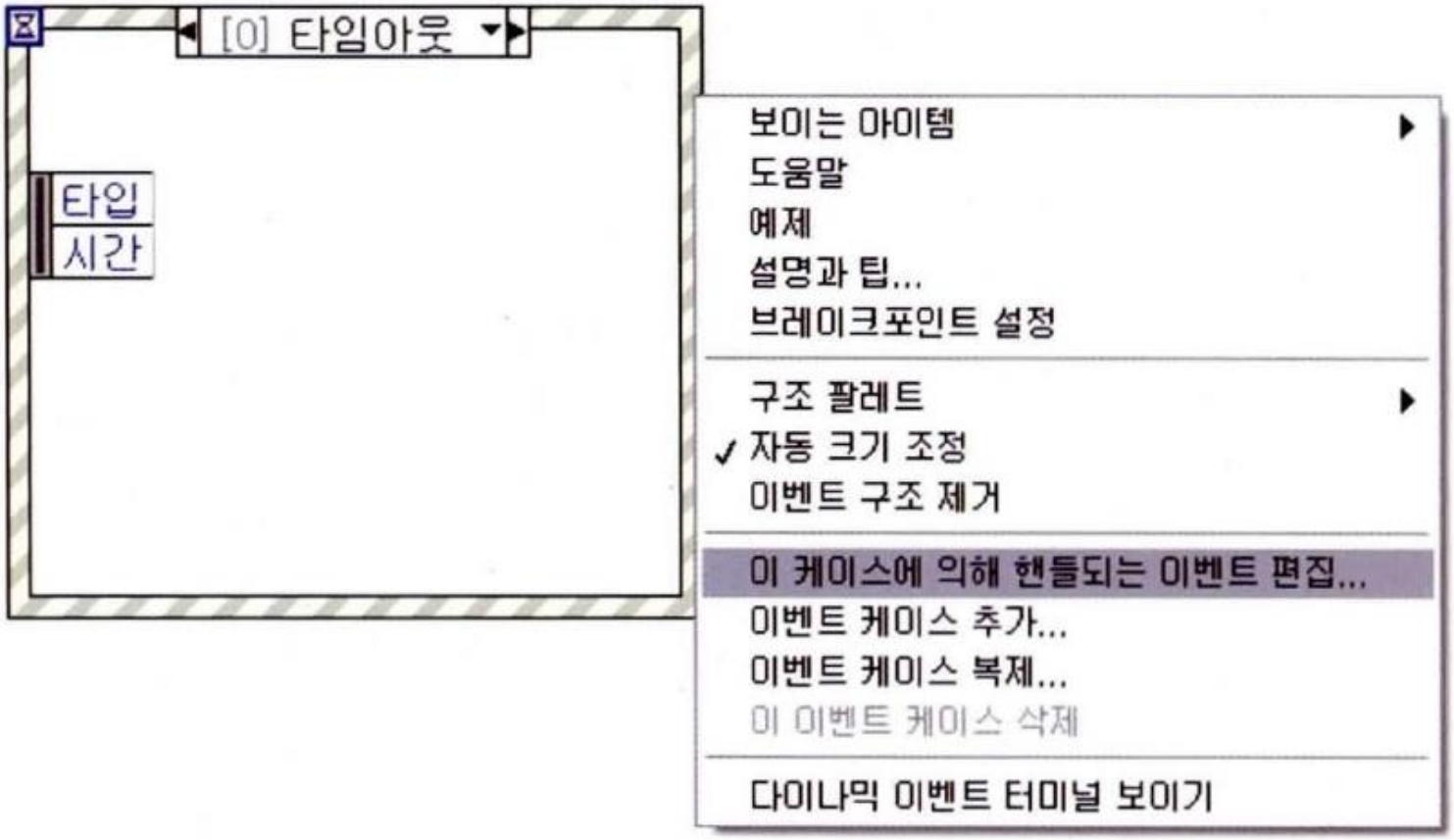
이벤트 구조

❖ 이벤트 구조는 일반적으로 While 루프와 함께 사용

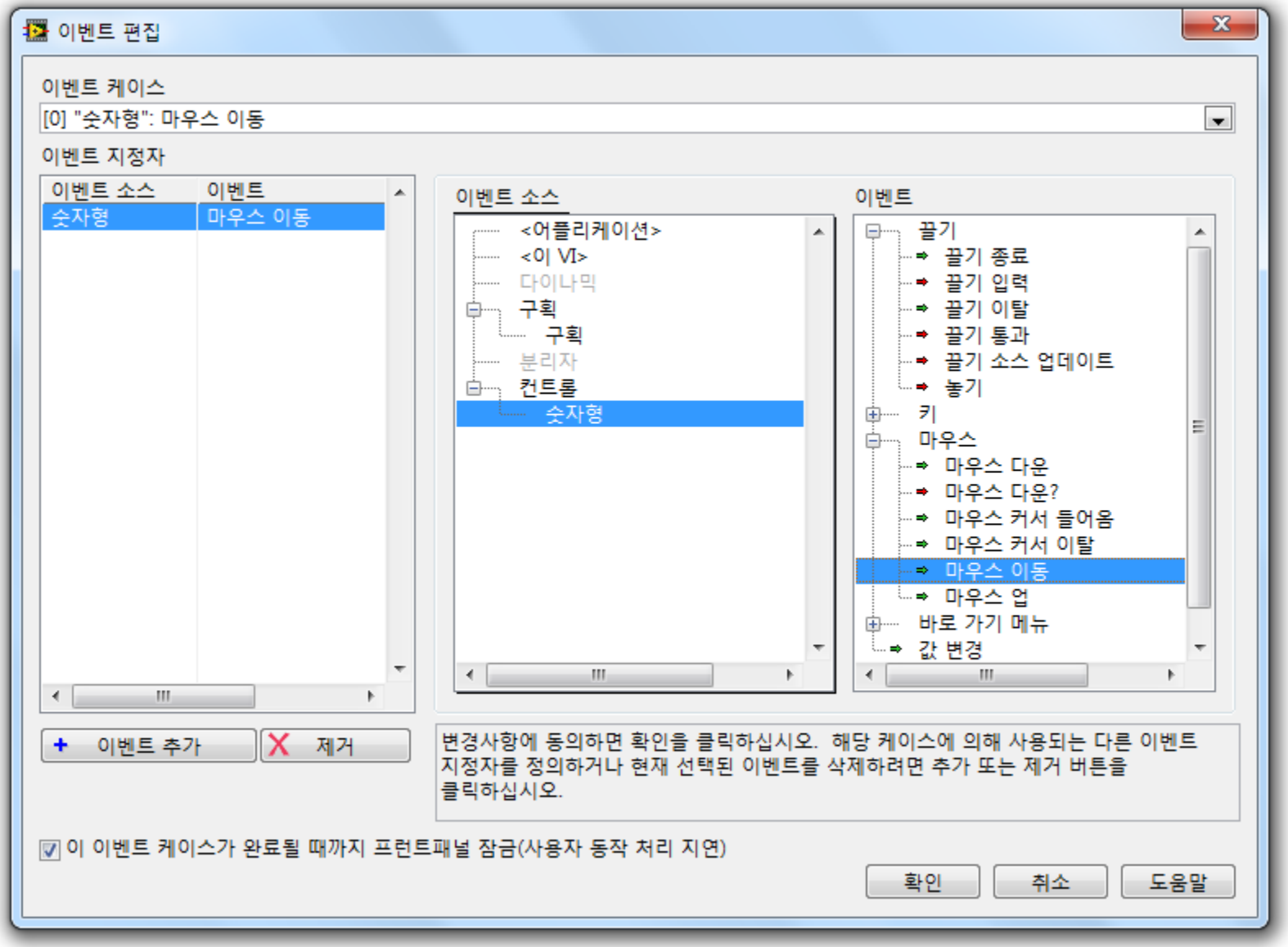


이벤트 구조

❖ 이벤트 구조의 바로가기메뉴 > 이 케이스에 의해 핸들되는 이벤트 편집 ... 을 선택하게 되면 이벤트 편집창이 나타난다.

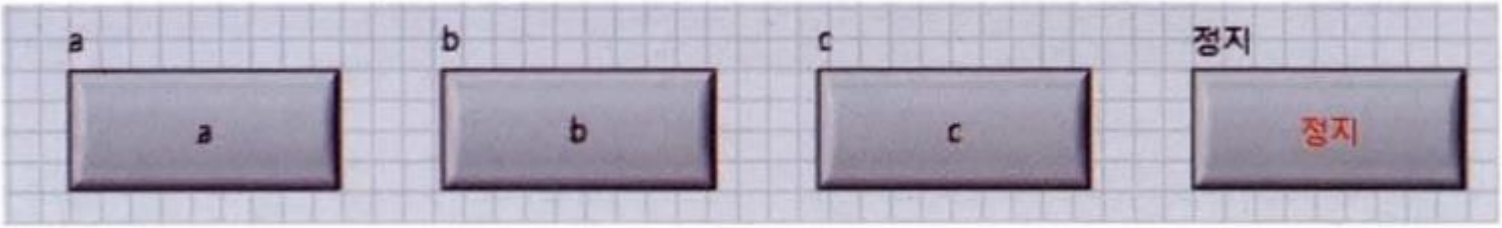


이벤트 편집


이벤트 편집
이벤트 케이스
[0] "숫자형": 마우스 이동
이벤트 지정자
이벤트 소스
이벤트
숫자형
마우스 이동
이벤트 소스
<어플리케이션>
<이 VI>
다이나믹
구획
구획
분리자
컨트롤
숫자형
이벤트
끌기
끌기 종료
끌기 입력
끌기 이탈
끌기 통과
끌기 소스 업데이트
놓기
키
마우스
마우스 다운
마우스 다운?
마우스 커서 들어옴
마우스 커서 이탈
마우스 이동
마우스 업
바로 가기 메뉴
값 변경
+ 이벤트 추가
제거
변경사항에 동의하면 확인을 클릭하십시오. 해당 케이스에 의해 사용되는 다른 이벤트 지정자를 정의하거나 현재 선택된 이벤트를 삭제하려면 추가 또는 제거 버튼을 클릭하십시오.
이 이벤트 케이스가 완료될 때까지 프런트패널 잠금(사용자 동작 처리 지연)
확인
취소
도움말

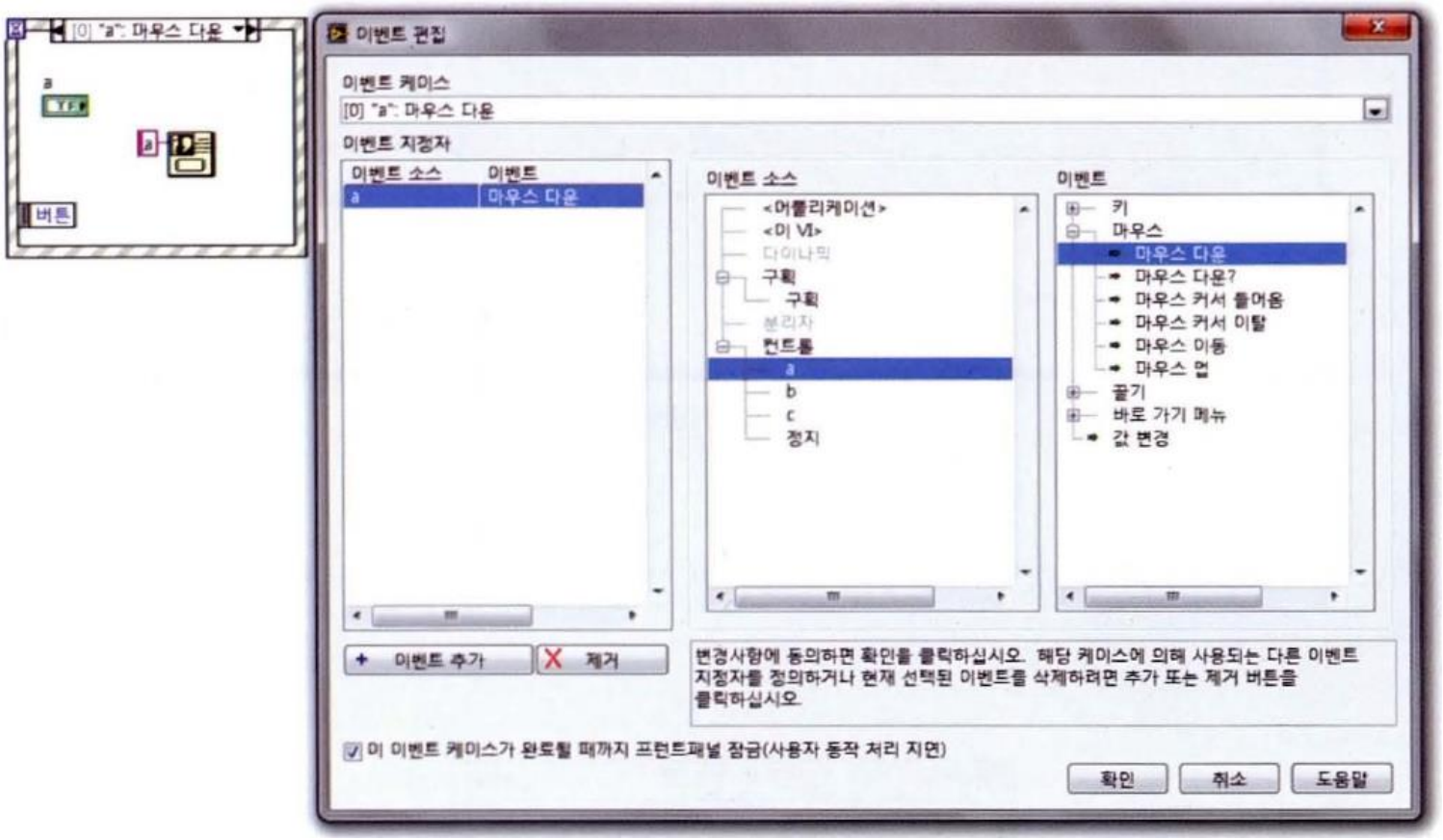
이벤트 구조

❖ 프론트패널에 'a', 'b', 'c', '정지' 라벨을 가진 불리언 버튼 4개를 만든다.



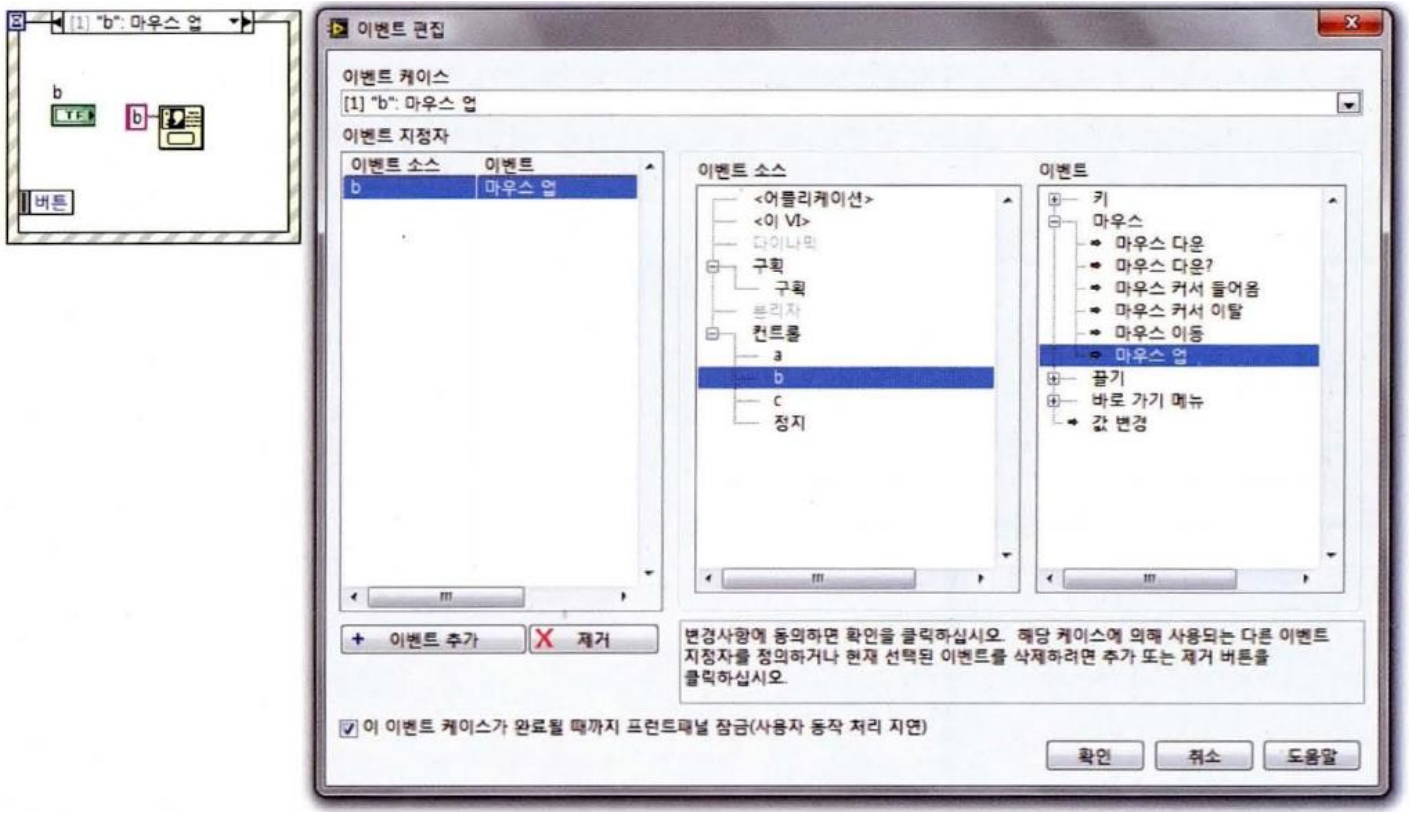
이벤트 구조

- 그림과 같이 'a' 이벤트를 만든다.
- 이벤트 편집창을 통해 이벤트 소스에서 'a'를 선택하고, 이벤트에서 '마우스 다운'을 선택한다.



이벤트 구조

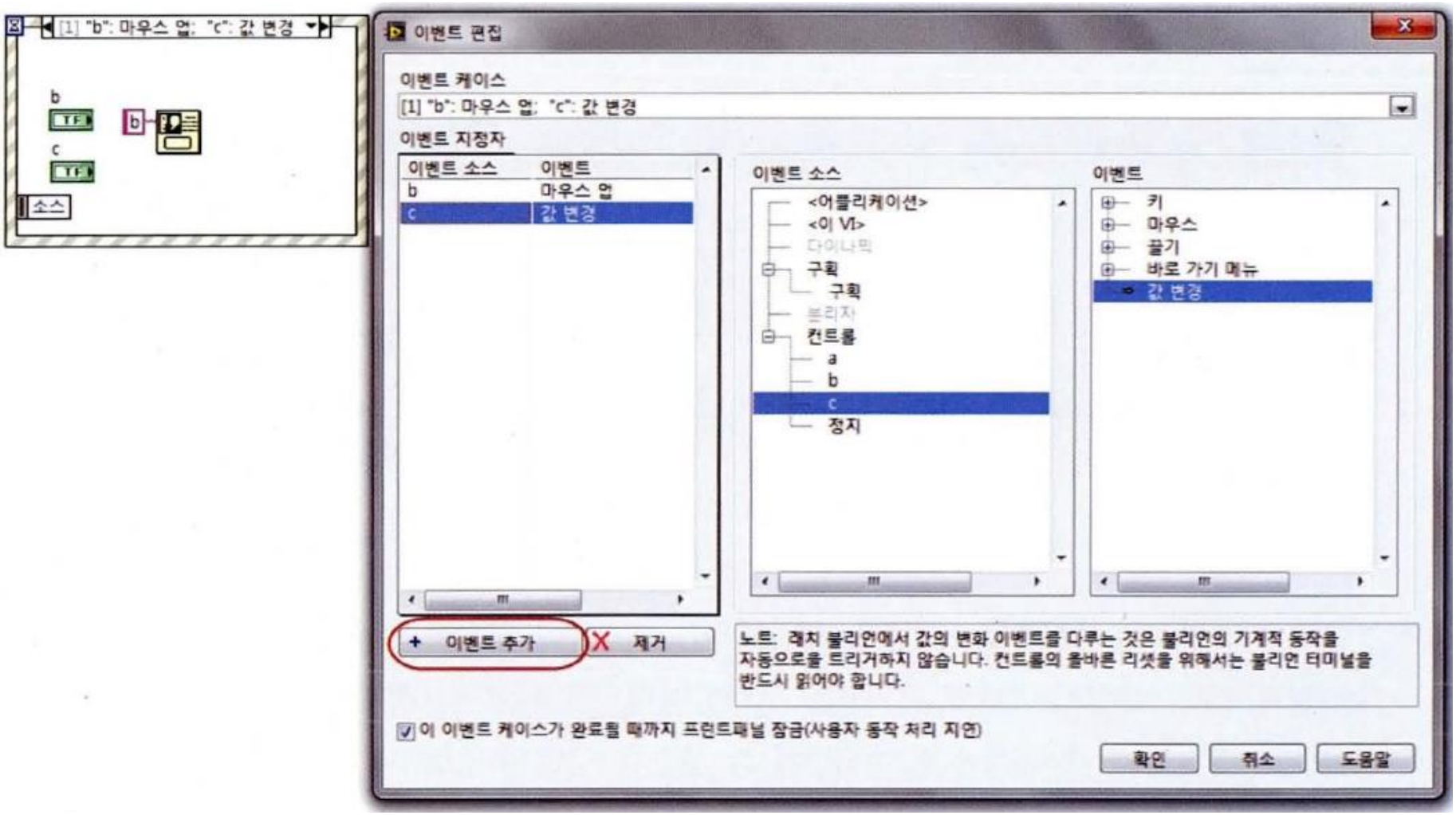
- 다음으로 이벤트 구조의 바로가기메뉴 > 이벤트 케이스 추가...하여 그림과 같이 'b' 이벤트를 만든다.
- 이벤트 편집창을 통해 이벤트 소스에서 ' b'를 선택하고, 이벤트에서 ' 마우스 업'을 선택한다.



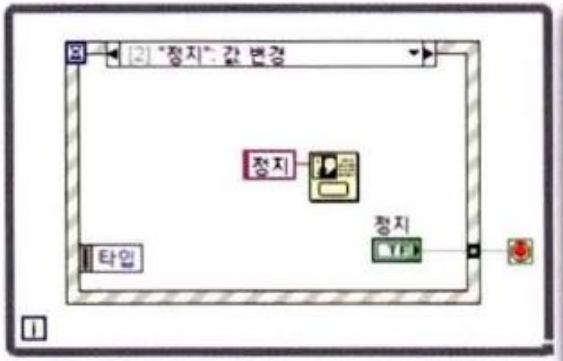
이벤트 구조

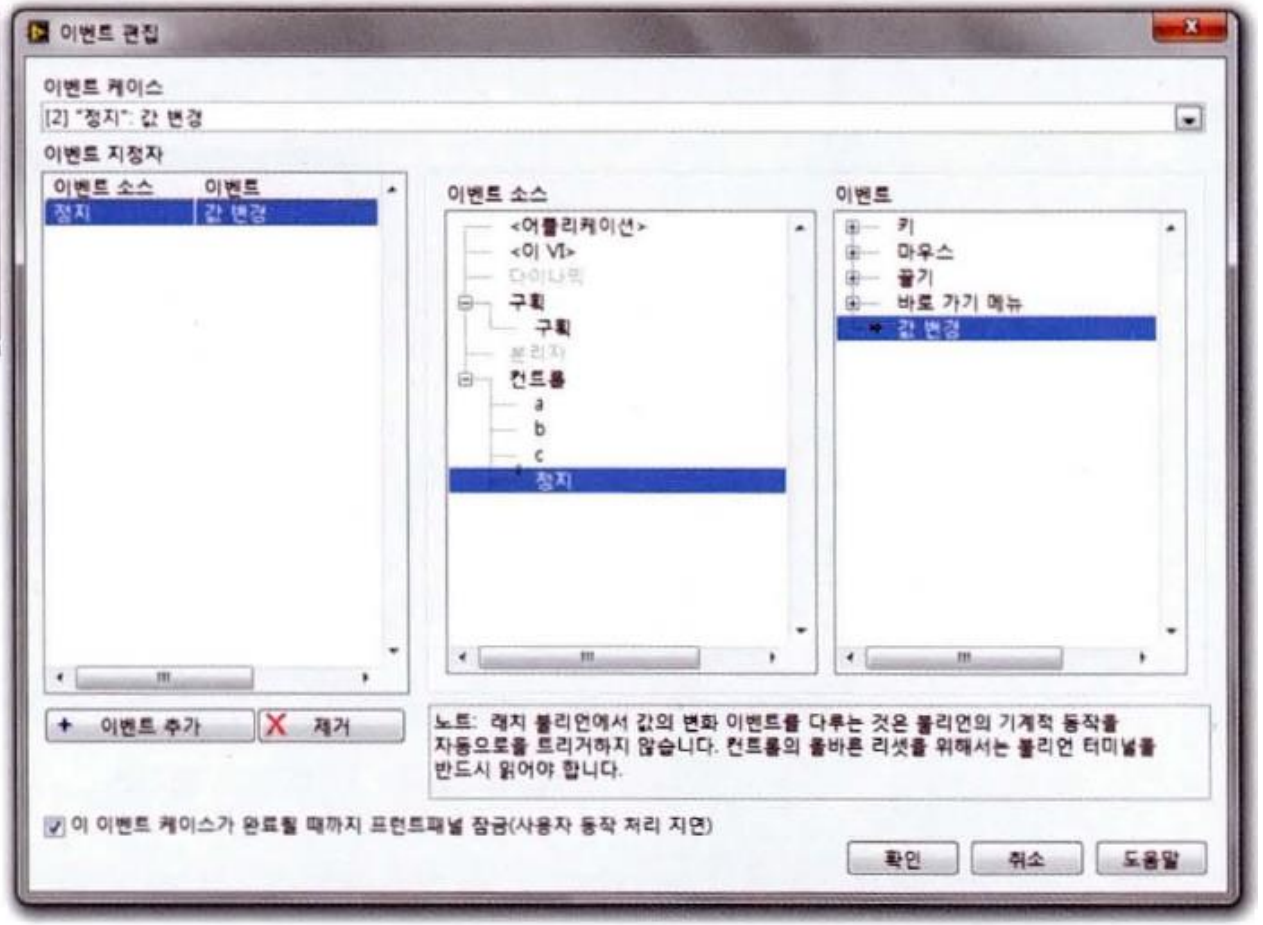
- ‘c’ 이벤트는 기존 ‘b’ 케이스에서 이벤트 편집창에서 ‘+ 이벤트 추가’ 버튼을 이용하여 이벤트 소스는 ‘c’, 이벤트는 ‘값 변경 ’을 선택하여 이벤트 지정자에 추가하면 이벤트 선택자 라벨에 두 개가 나타난다.
- 즉 ‘b’에 ‘마우스 업 ’이 발생되거나 'c'에 ‘값 변경’ 이 발생하면 ‘b’라고 적힌 단일 버튼 대화상자가 나타난다.

이벤트 구조


[1] "b": 마우스 업; "c": 값 변경
b
c
소스
이벤트 편집
이벤트 케이스
[1] "b": 마우스 업; "c": 값 변경
이벤트 지정자
이벤트 소스
이벤트
b
마우스 업
c
값 변경
이벤트 소스
<어플리케이션>
<이 VI>
다이나믹
구획
구획
프리자
컨트롤
a
b
c
정지
이벤트
키
마우스
끌기
바로 가기 메뉴
값 변경
+ 이벤트 추가
X 제거
노트: 래치 불리언에서 값의 변화 이벤트를 다루는 것은 불리언의 기계적 동작을 자동으로을 트리거하지 않습니다. 컨트롤의 올바른 리셋을 위해서는 불리언 터미널을 반드시 읽어야 합니다.
이 이벤트 케이스가 완료될 때까지 프런트패널 잠금(사용자 동작 처리 지연)
확인
취소
도움말

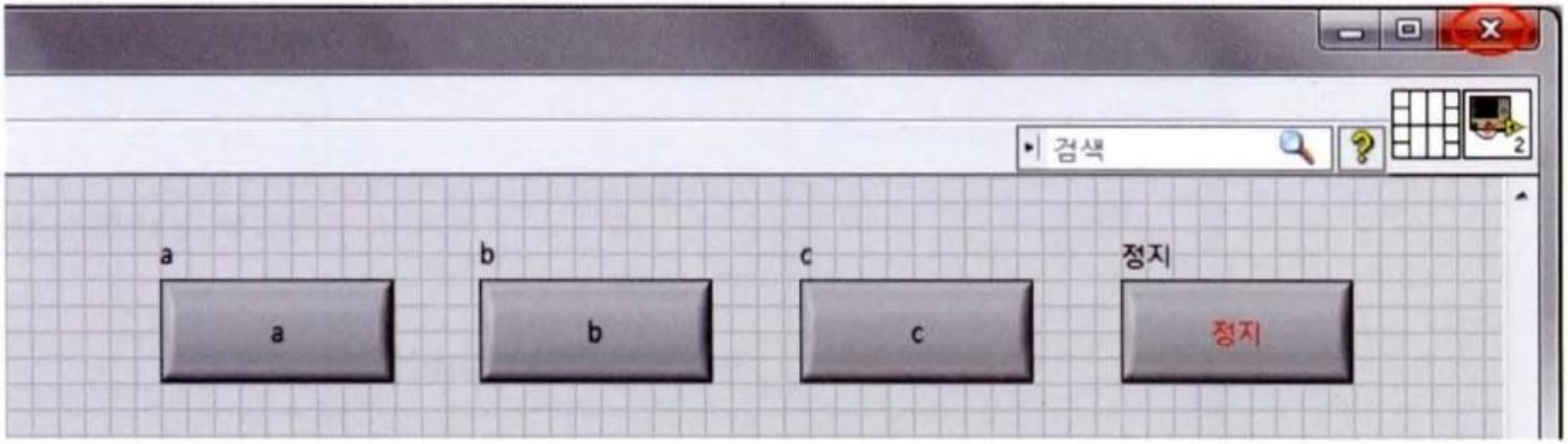
이벤트 구조


[2] "정지": 값 변경
정지
정지
타입


이벤트 편집
이벤트 케이스
[2] "정지": 값 변경
이벤트 지정자
이벤트 소스
이벤트
정지
값 변경
이벤트 소스
<어플리케이션>
<이 VI>
구획
구획
컨트롤
a
b
c
정지
이벤트
키
마우스
끌기
바로 가기 메뉴
값 변경
+ 이벤트 추가
X 제거
노트: 래치 불리언에서 값의 변화 이벤트를 다루는 것은 불리언의 기계적 동작을 자동으로를 트리거하지 않습니다. 컨트롤의 올바른 리셋을 위해서는 불리언 터미널을 반드시 읽어야 합니다.
이 이벤트 케이스가 완료될 때까지 프런트패널 잠금(사용자 동작 처리 지연)
확인
취소
도움말

이벤트 구조

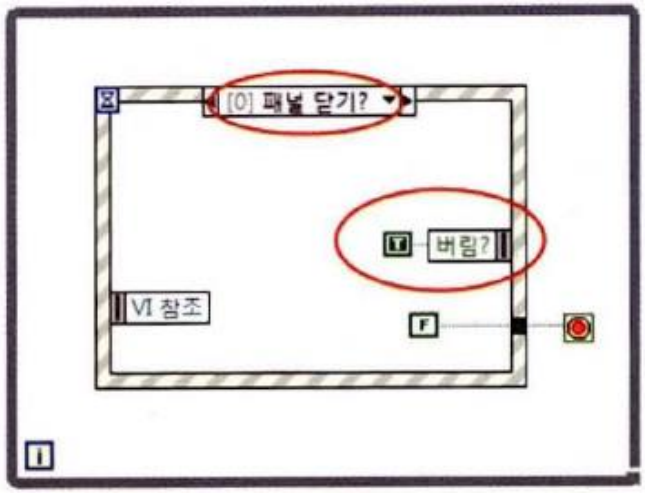
- 위의 코드에 필터 이벤트와 타임 아웃 이벤트를 추가해보자.
- 그림에서 프로그램 실행 중 우측 상단에 'X' 버튼을 누르면 프로그램이 닫히게 된다.
- 필터 이벤트로 'X' 버튼을 눌러도 닫히지 않고 정상적으로 실행되도록 코드를 만들어보자.

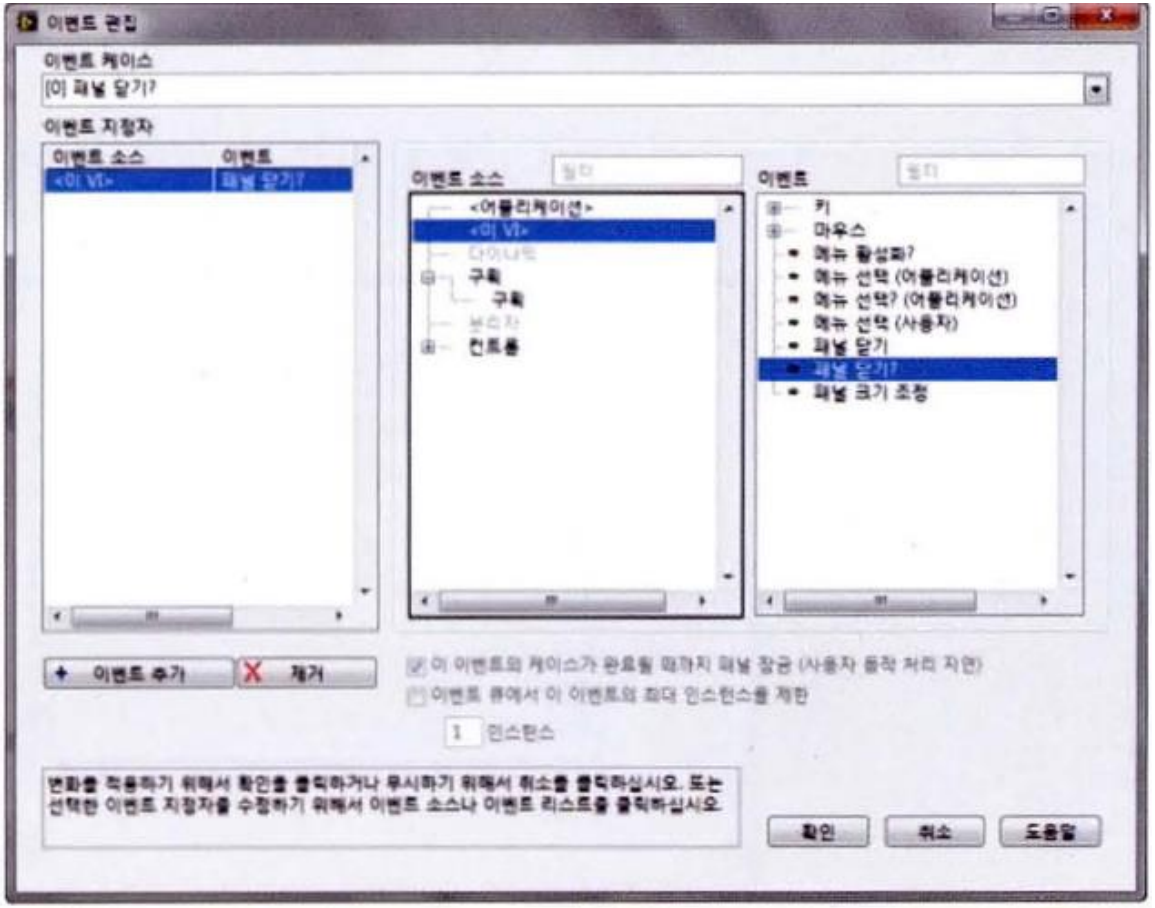


이벤트 구조

- 먼저 이벤트 구조의 바로가기메뉴 > 이벤트 케이스 추가 ... 하여 이벤트 편집창을 연다.
- 이벤트 소스는 '이 VI'를 선택하고, 이벤트는 '패널 닫기?'를 선택한다.
- 필터 이벤트를 선택하였기 때문에 이벤트 구조에 이벤트 필터 노드가 나타나고 '버림?'에 참 상수를 연결해 주면 '패널 닫기' 버튼이 눌러진 이벤트를 버리게 된다.
- 즉 필터링하여 실행 중인 프로그램은 'X' 버튼을 눌러도 닫히지 않고 정상적으로 실행하도록 하기 위하여 거짓 상수를 While 루프의 조건 터미널에 와이어링한다.

이벤트 구조

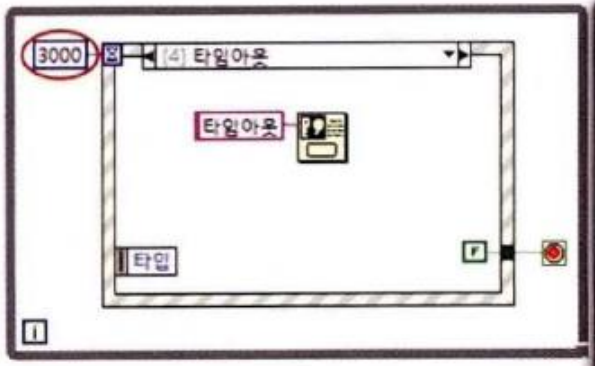

[0] 패널 닫기?
버림?
VI 참조

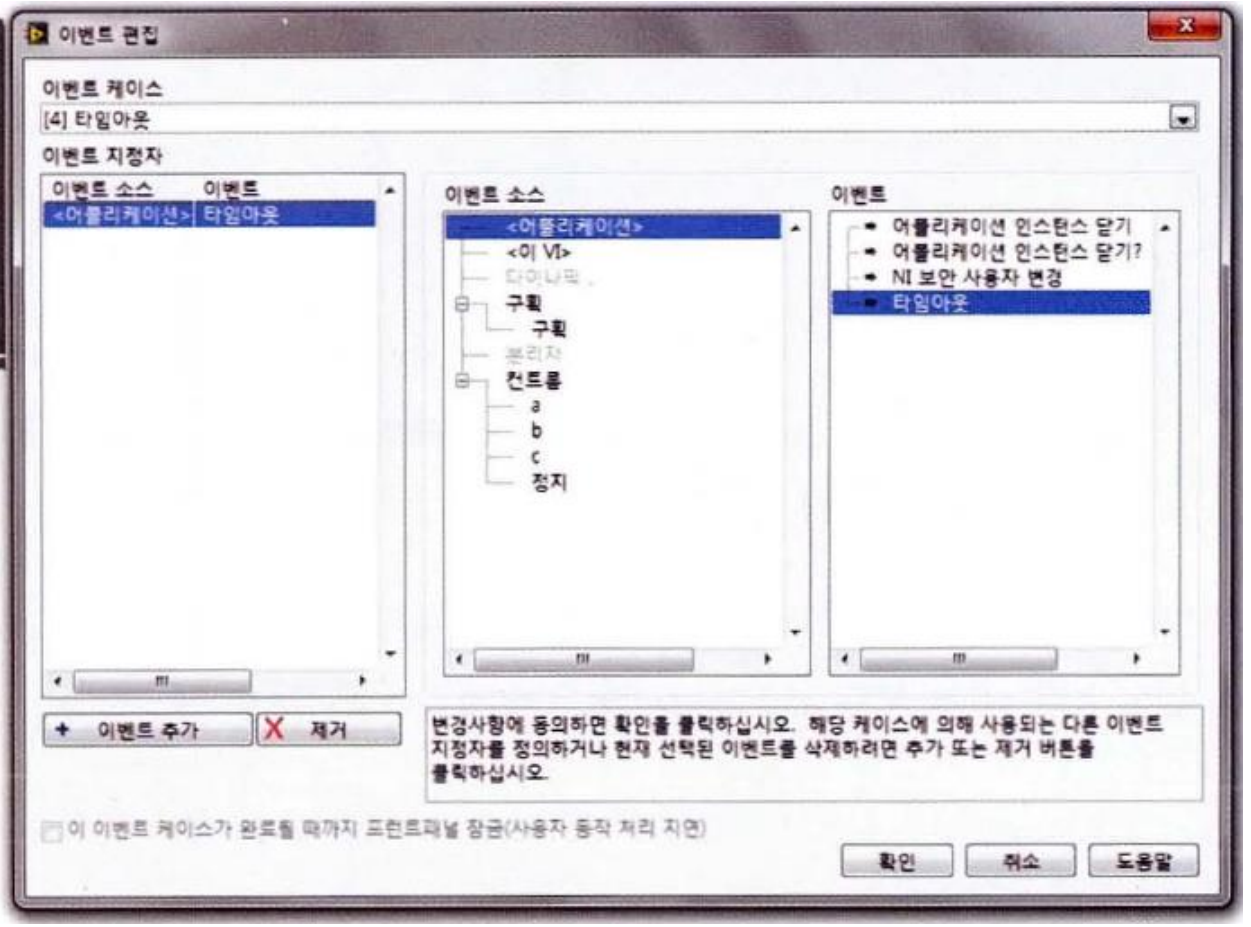

이벤트 편집
이벤트 케이스
[0] 패널 닫기?
이벤트 지정자
이벤트 소스
이벤트
<이 VI>
패널 닫기?
이벤트 소스
<어플리케이션>
<이 VI>
구획
구획
컨트롤
이벤트
키
마우스
메뉴 활성화?
메뉴 선택 (어플리케이션)
메뉴 선택? (어플리케이션)
메뉴 선택 (사용자)
패널 닫기
패널 닫기?
패널 크기 조정
이벤트 추가
제거
이 이벤트의 케이스가 완료될 때까지 패널 잠금 (사용자 동작 처리 지연)
이벤트 큐에서 이 이벤트의 최대 인스턴스를 제한
1
인스턴스
변화를 적용하기 위해서 확인을 클릭하거나 무시하기 위해서 취소를 클릭하십시오. 또는 선택한 이벤트 지정자를 수정하기 위해서 이벤트 소스나 이벤트 리스트를 클릭하십시오.
확인
취소
도움말

이벤트 구조

- 마지막으로 타임 아웃 이벤트를 설정해보자.
- 그림에서 이벤트 구조의 좌측 상단에 있는 모래시계 모양에 '3000' (msec) 의 값을 연결한다.
- 아무 값도 주지 않으면 사용자가 정의해 놓은 이벤트가 발생하기를 무한 대기하는 반면에, 구체적인 시간 값을 주고 주어진 시간 내에 이벤트가 발생하지 않게 되면 타임아웃 이벤트 케이스 안에 구현된 코드가 실행된다.

이벤트 구조

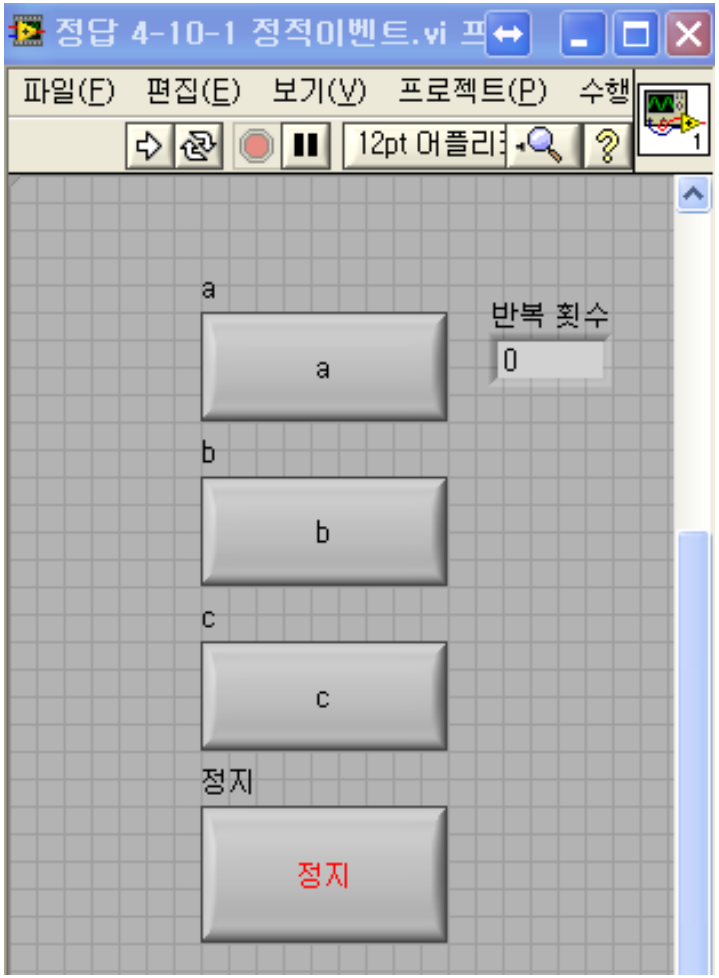

3000
[4] 타임아웃
타임아웃
타임
i


이벤트 편집
이벤트 케이스
[4] 타임아웃
이벤트 지정자
이벤트 소스
이벤트
<어플리케이션>
타임아웃
이벤트 소스
<어플리케이션>
<이 VI>
구획
구획
컨트롤
a
b
c
정지
이벤트
어플리케이션 인스턴스 닫기
어플리케이션 인스턴스 닫기?
NI 보안 사용자 변경
타임아웃
+ 이벤트 추가
X 제거
변경사항에 동의하면 확인을 클릭하십시오. 해당 케이스에 의해 사용되는 다른 이벤트 지정자를 정의하거나 현재 선택된 이벤트를 삭제하려면 추가 또는 제거 버튼을 클릭하십시오.
이 이벤트 케이스가 완료될 때까지 프런트패널 잠금(사용자 동작 처리 지연)
확인
취소
도움말

실습4-10-1) 정적 이벤트 사용법

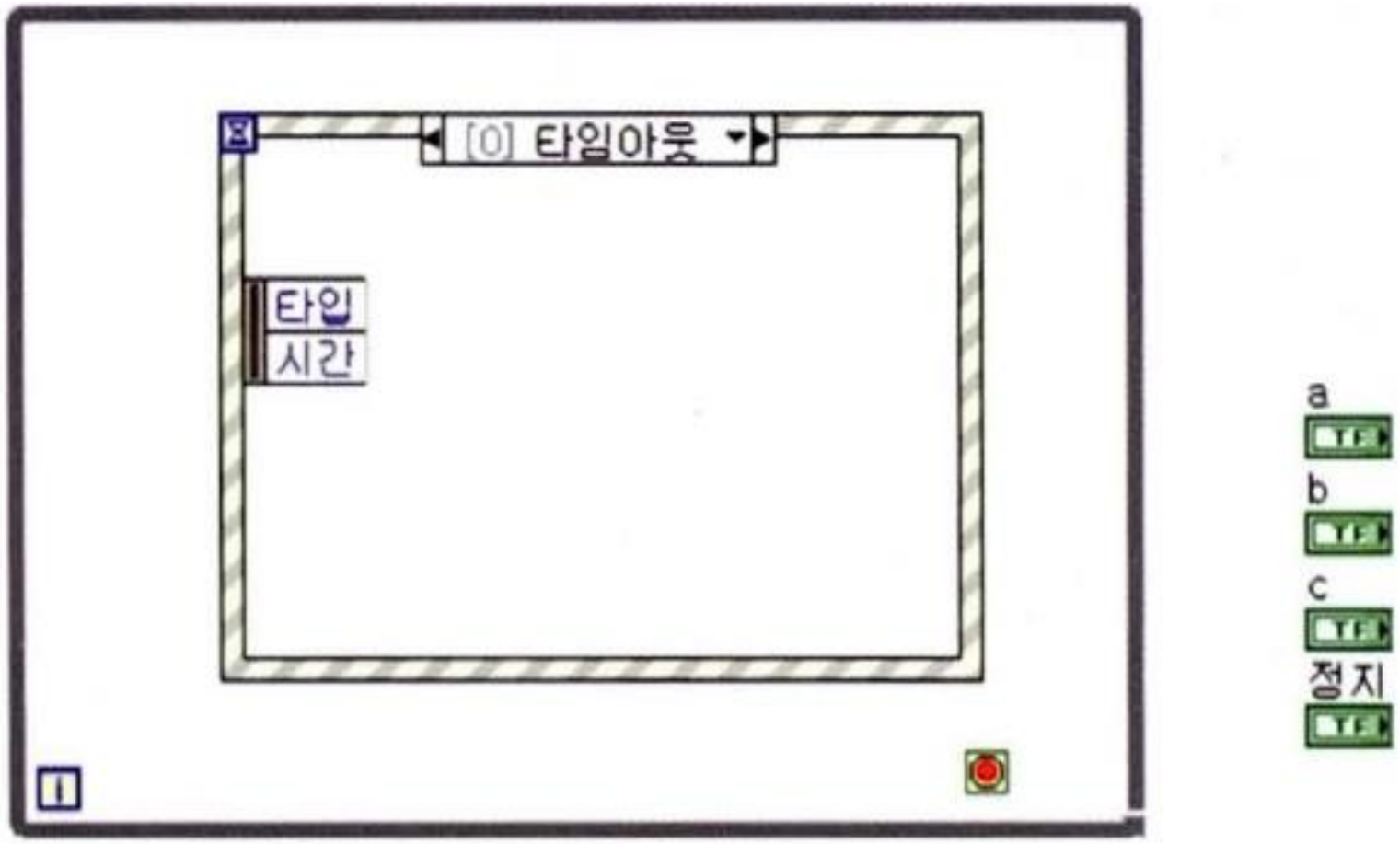
실습4-10-1) 정적 이벤트 사용법

❖ 프론트패널에 'a', 'b', 'c', '정지' 라벨을 가진 불리언 버튼 4개를 만든다.



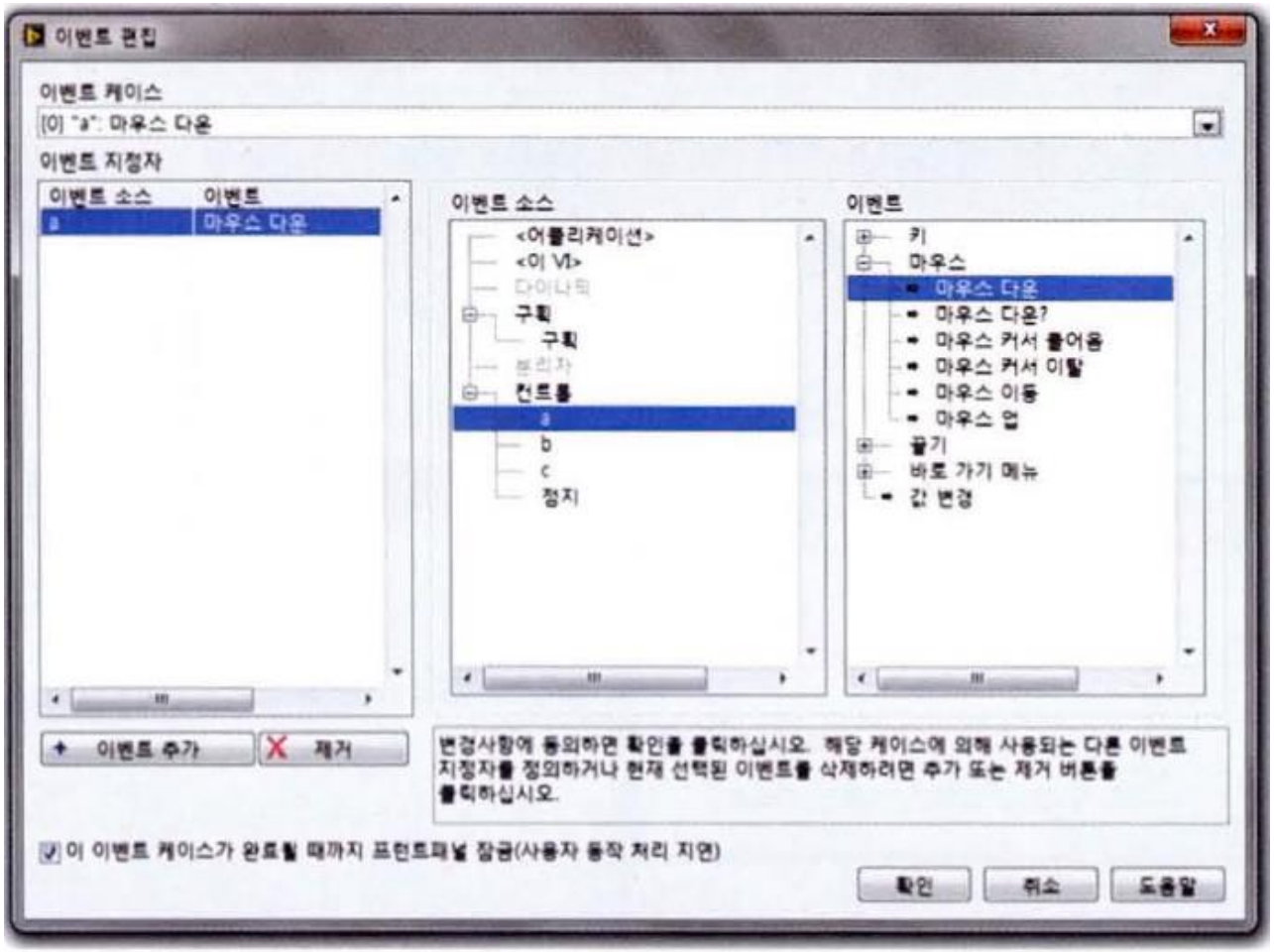
실습4-10-1) 정적 이벤트 사용법

❖ 블록다이어그램에서 이벤트 구조를 추가



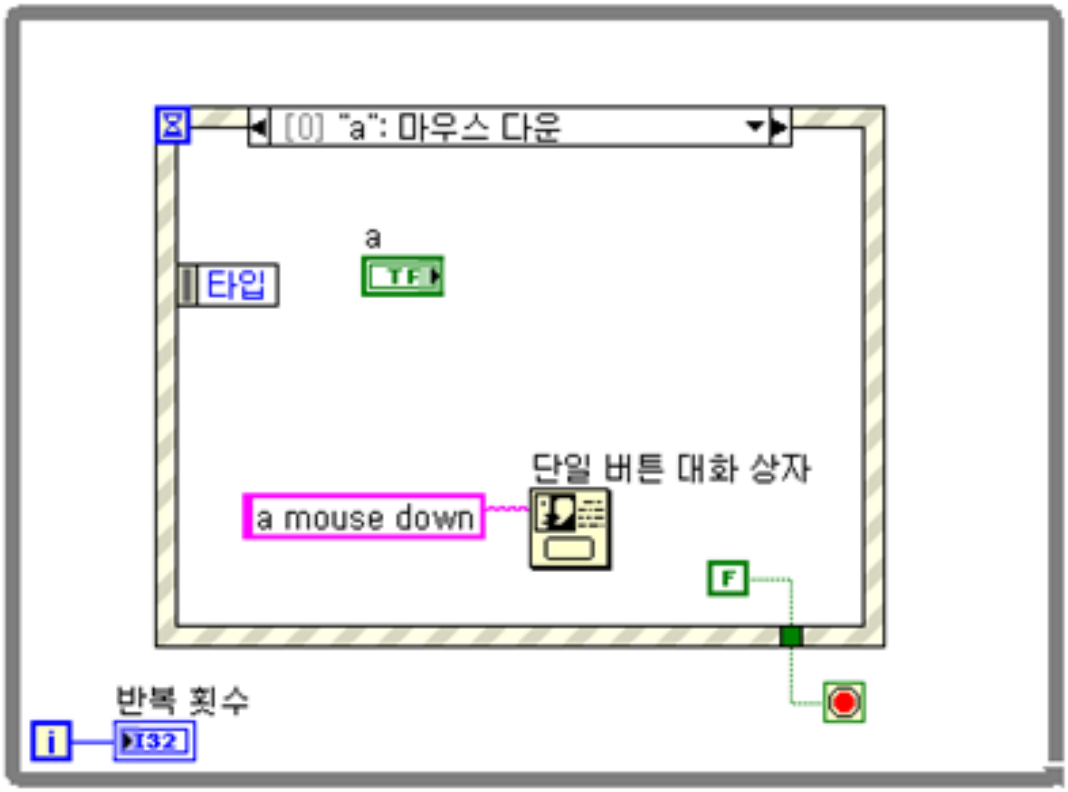
실습4-10-1) 정적 이벤트 사용법

❖ 이벤트 구조의 바로가기메뉴 > 이 케이스에 의해 핸들되는 이벤트 편집 ... 을 선택하여 그림과 같이 설정



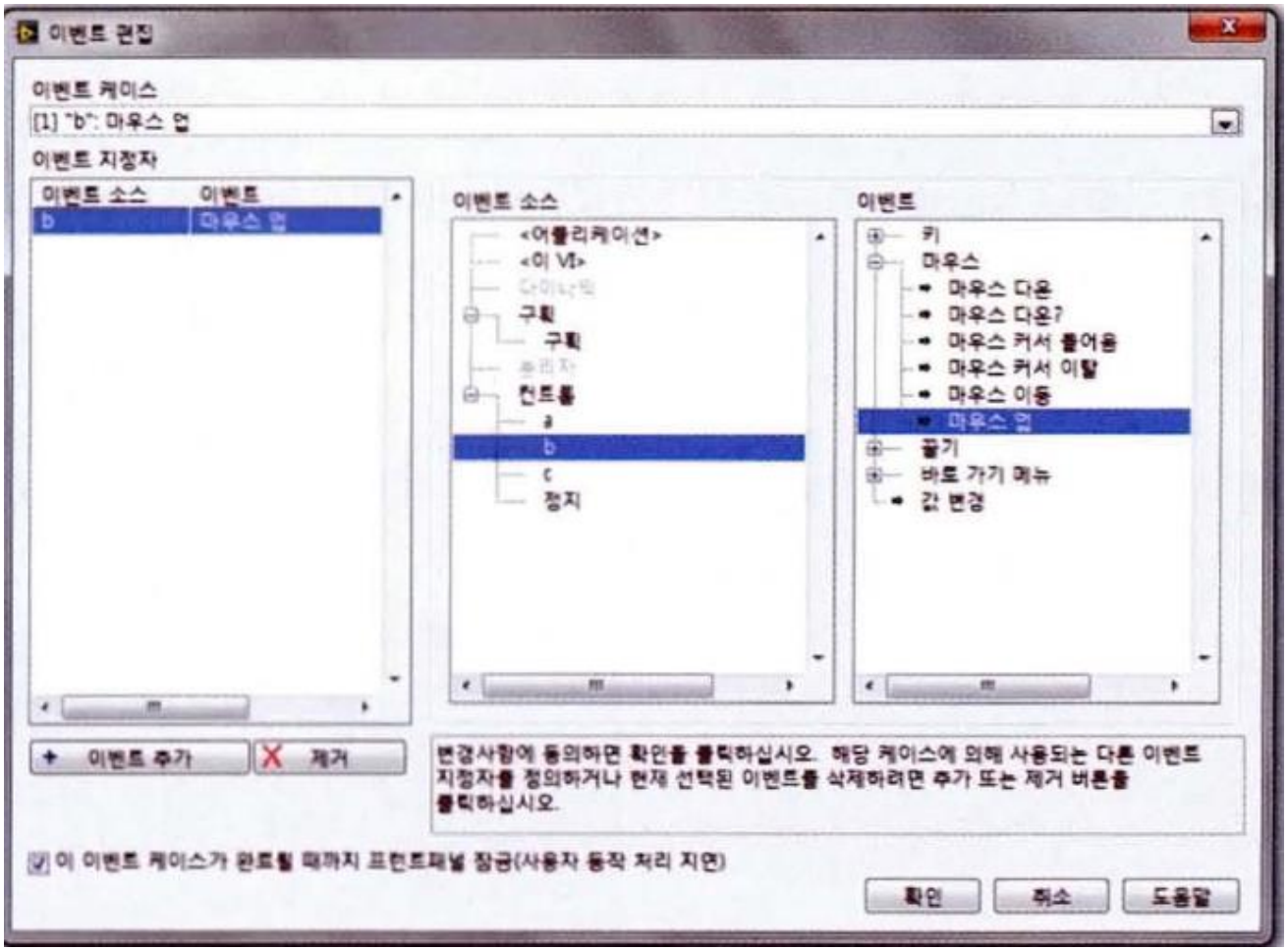
실습4-10-1) 정적 이벤트 사용법

❖ 그리고 이벤트 케이스 안에 단일 버튼 대화 상자.VI, 거짓 상수를 가져다 놓고 그림과 같이 구성한다.



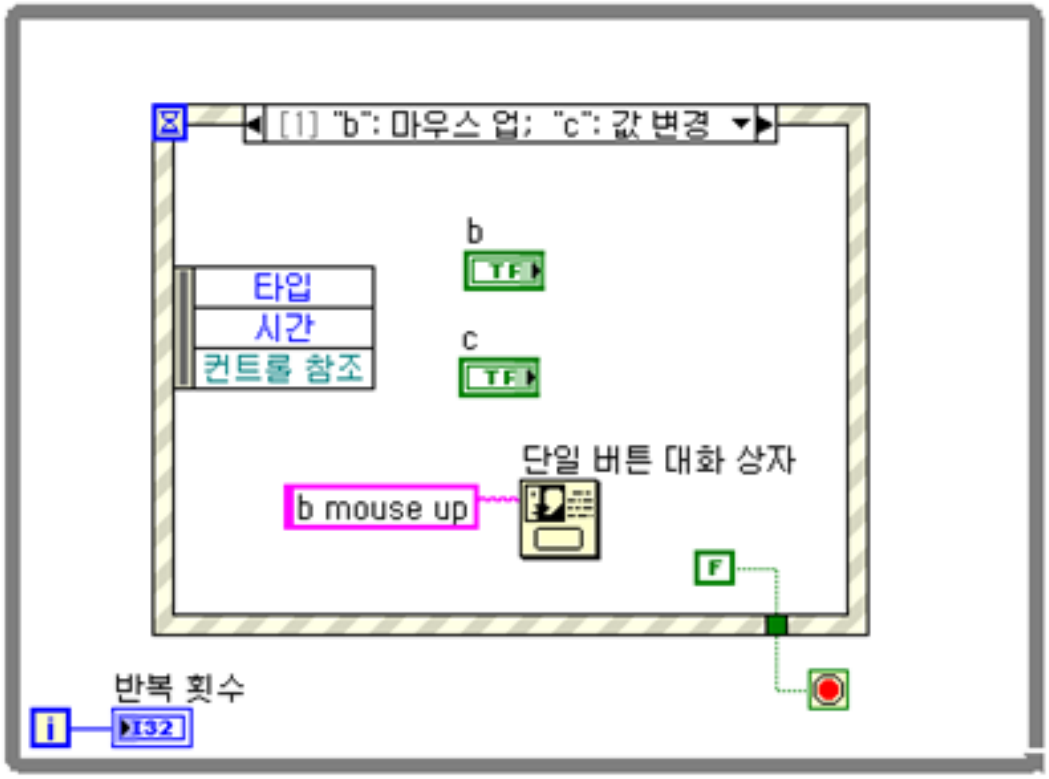
실습4-10-1) 정적 이벤트 사용법

❖ 이벤트 구조의 바로가기메뉴 > 이벤트 케이스 복제 .. 를 선택하여 그림과 같이 설정하고 '확인' 버튼을 누른다.



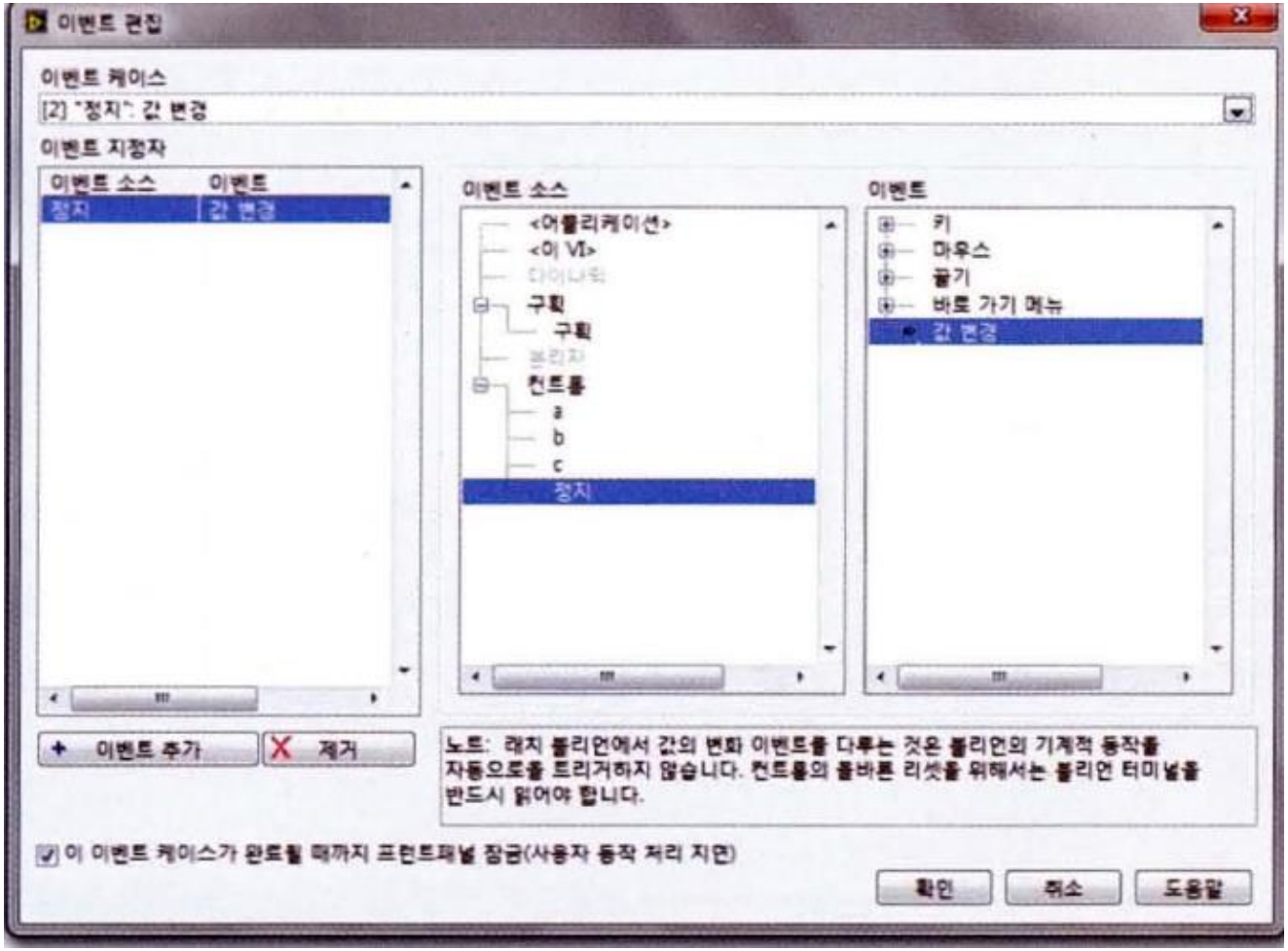
실습4-10-1) 정적 이벤트 사용법

❖ 단일 버튼 대화 상자.VI의 메시지를 다음과 같이 변경



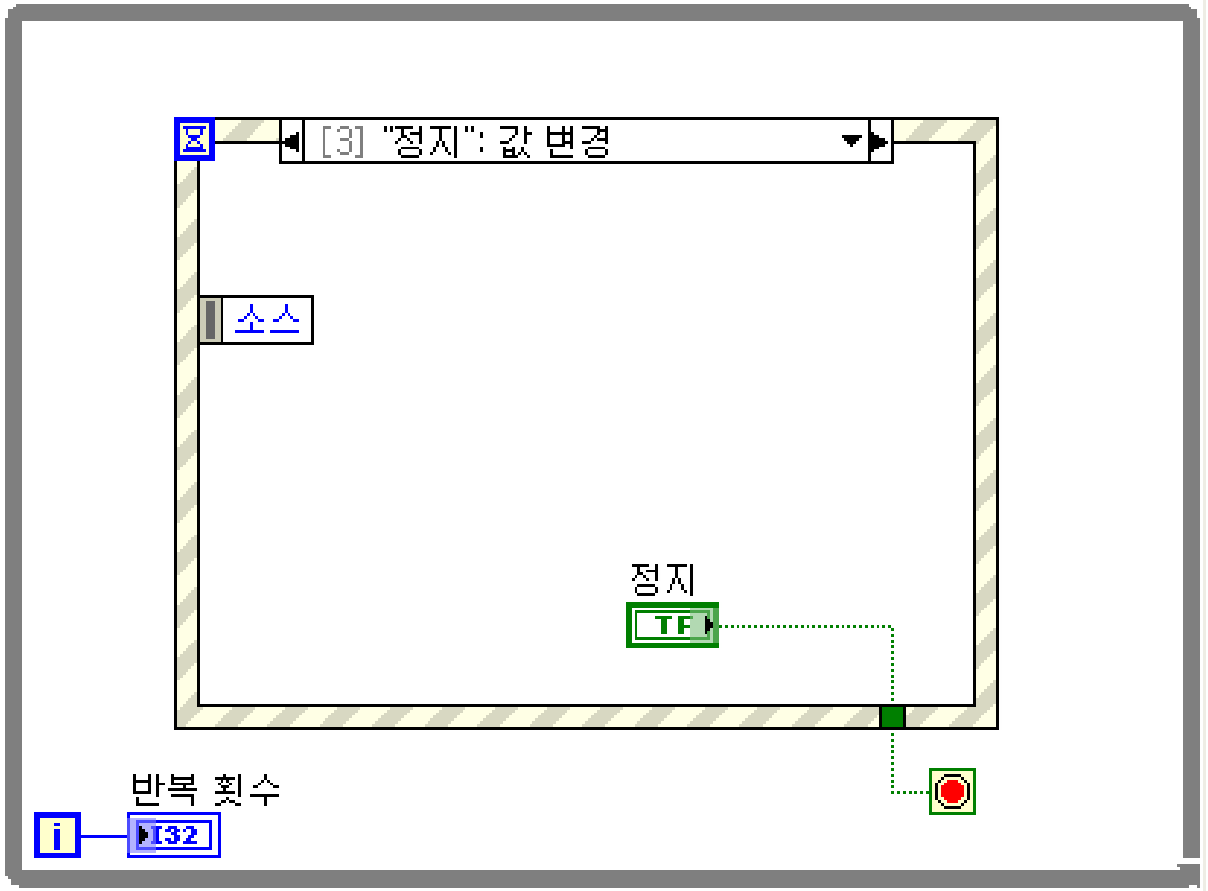
실습4-10-1) 정적 이벤트 사용법

❖ 이벤트 구조의 바로가기메뉴 > 이벤트 케이스 추가. 를 선택하여 그림과 같이 설정하고 '확인' 버튼을 누른다.



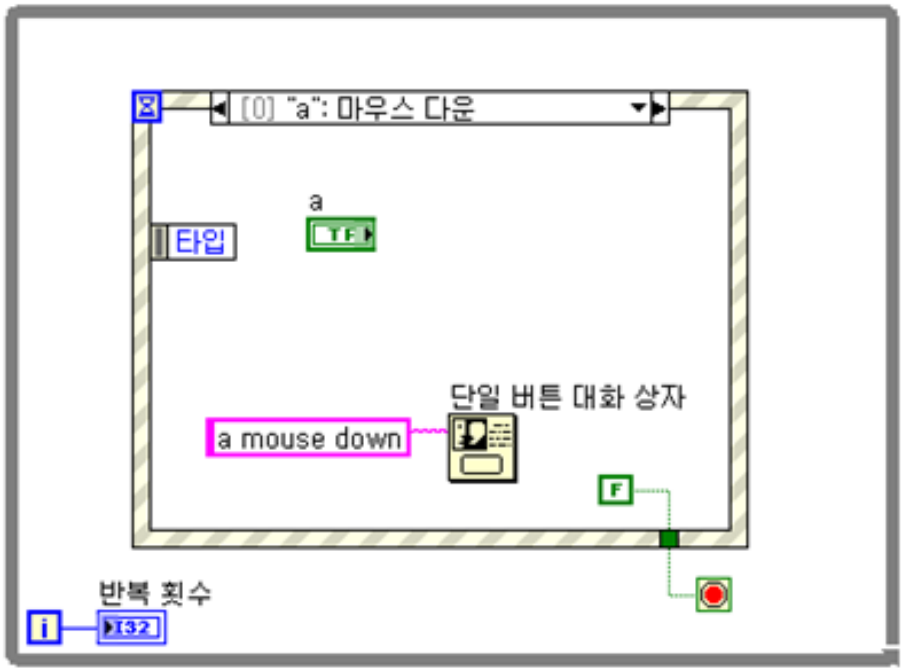
실습4-10-1) 정적 이벤트 사용법

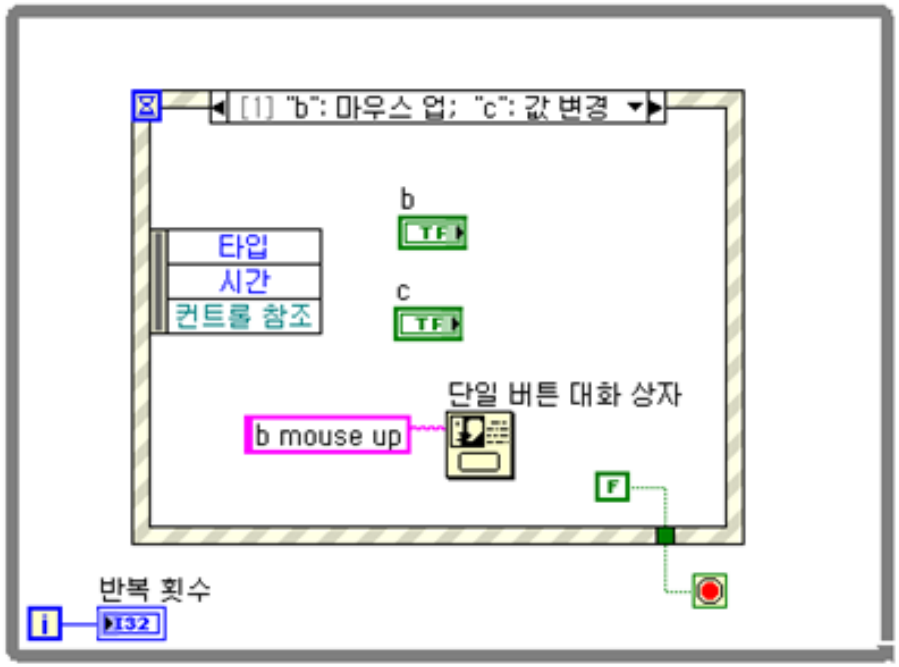
❖ '정지' 터미널을 다음과 같이 연결.



실습4-10-1) 정적 이벤트 사용법

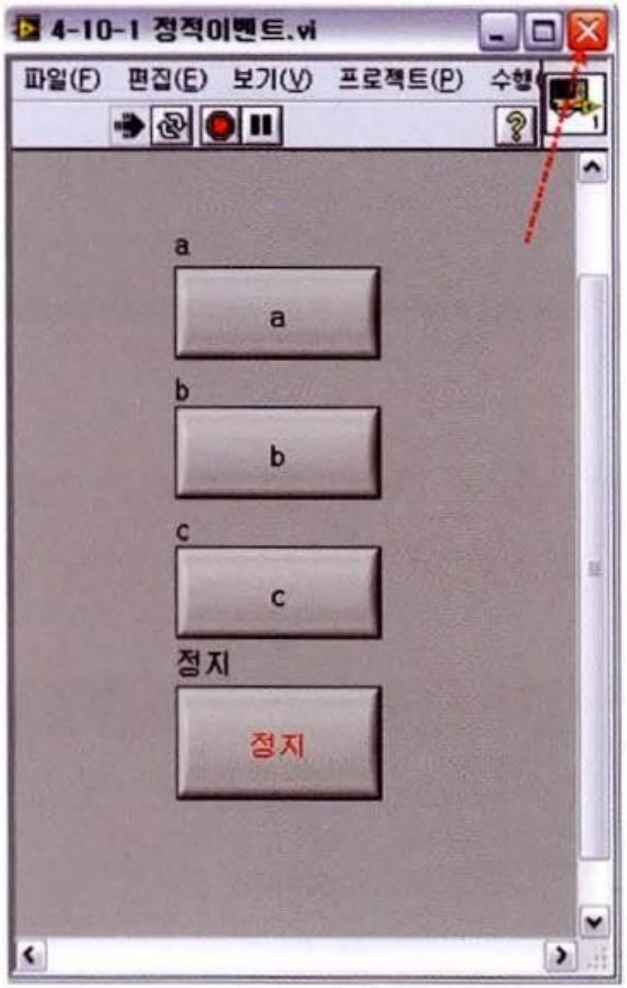
❖ 'a'와 'b' 터미널을 각각 이벤트 구조 안에 넣어준다.





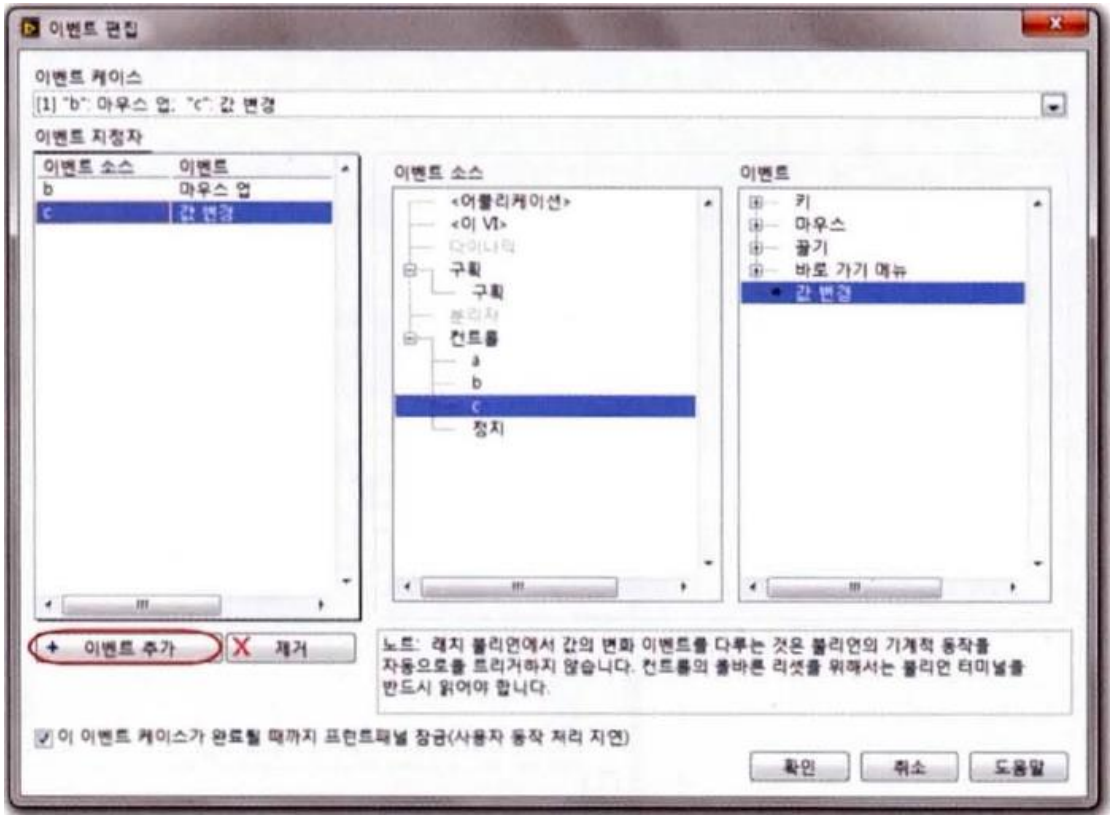
실습4-10-1) 정적 이벤트 사용법

❖ VI를 저장한 다음 다시 실행하여 그림과 같이 상단에 있는 'x' 버튼을 누르면 창이 닫히는 것을 확인



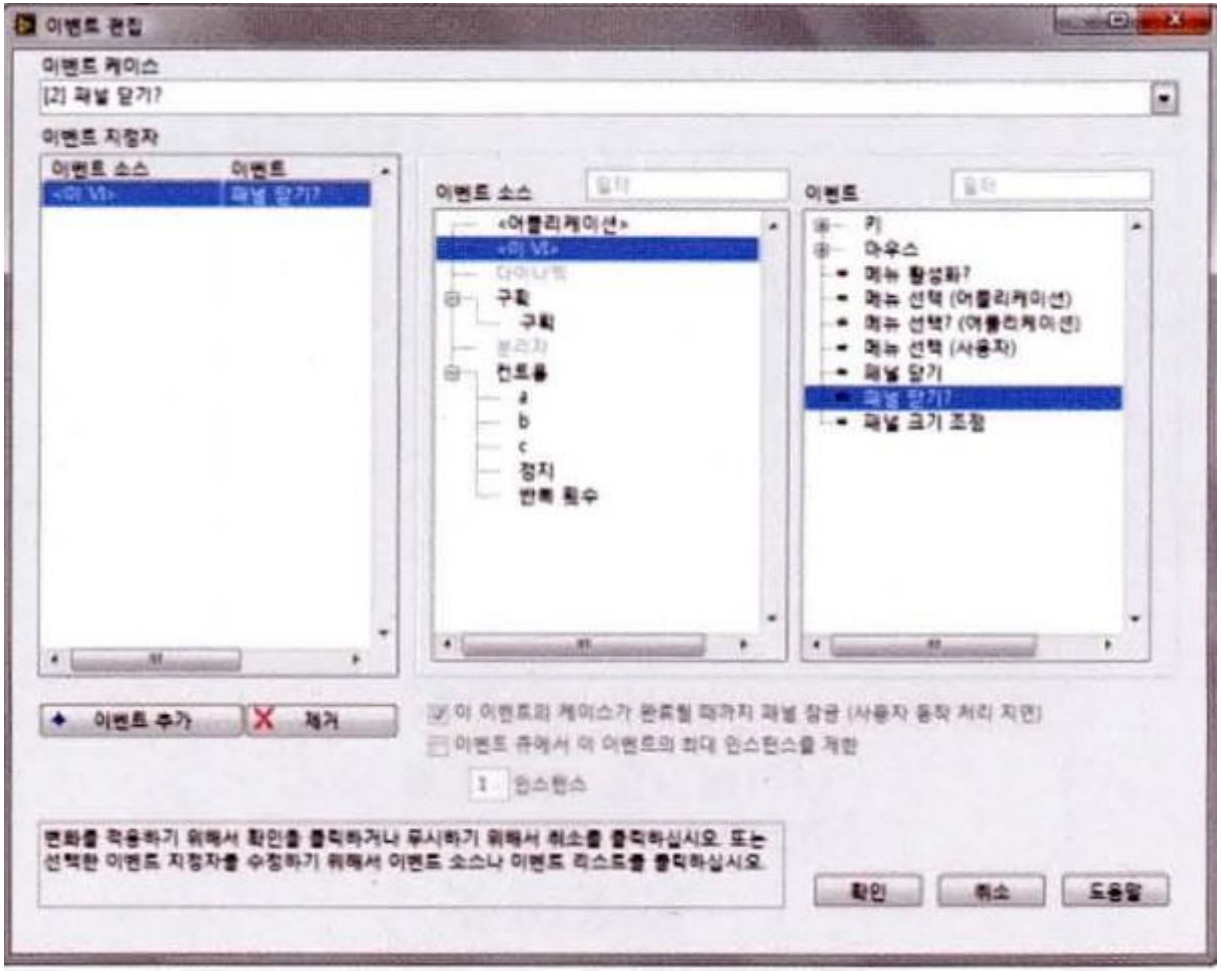
실습4-10-1) 정적 이벤트 사용법

❖ 다음과 같이 'b' 이벤트 케이스를 보이게 한 뒤 바로가기 메뉴 > 이 케이스에 의해 핸들되는 이벤트 편집 ... 을 선택하여 그림과 같이 파란색 '+ 이벤트 추가' 버튼을 눌러 기존 설정에 'c'의 '값 변경'을 추가



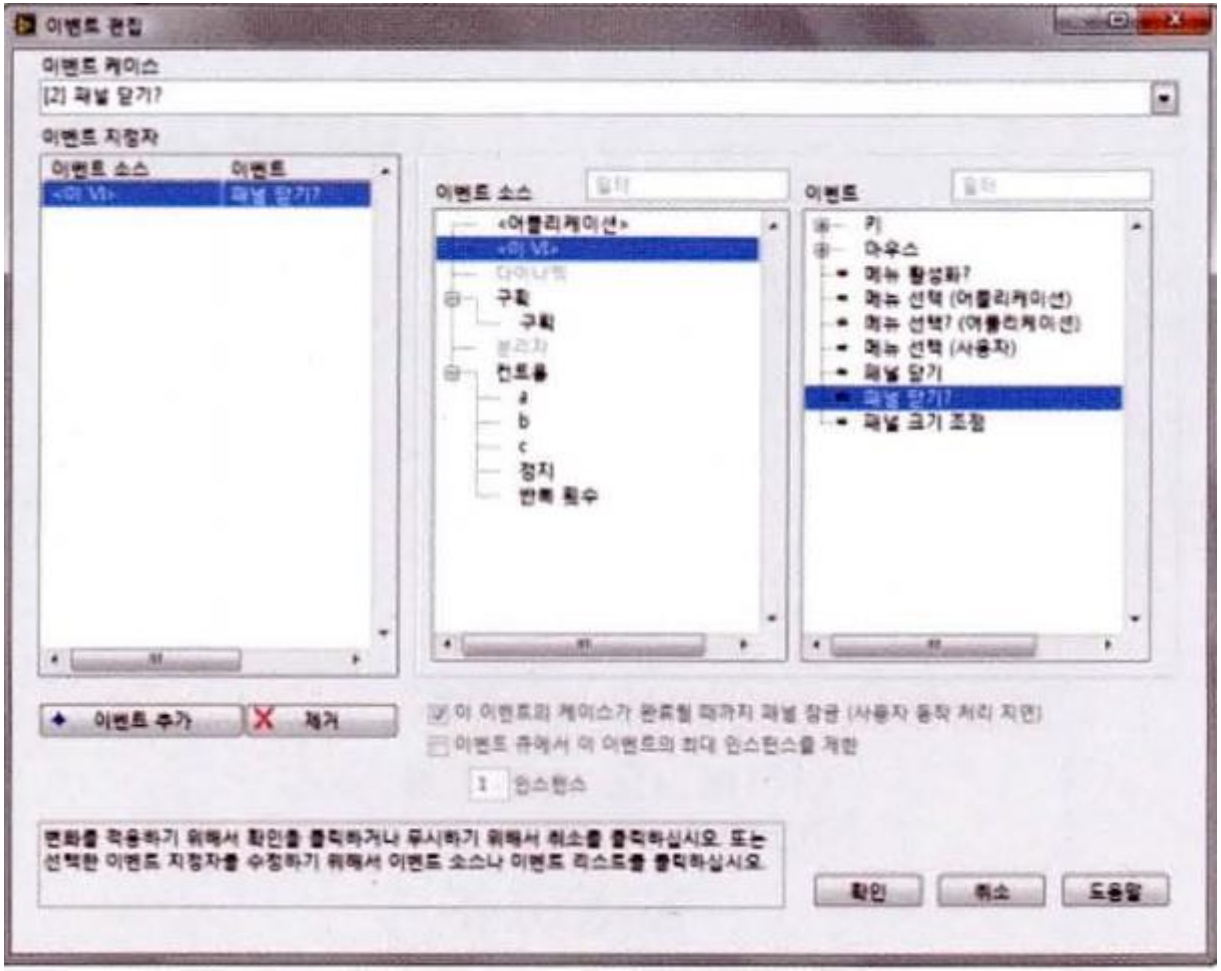
실습4-10-1) 정적 이벤트 사용법

❖ 이벤트의 바로가기메뉴 > 이벤트 케이스 추가 ... 를 선택하여 그림과 같이 설정하고 '확인' 버튼을 누른다

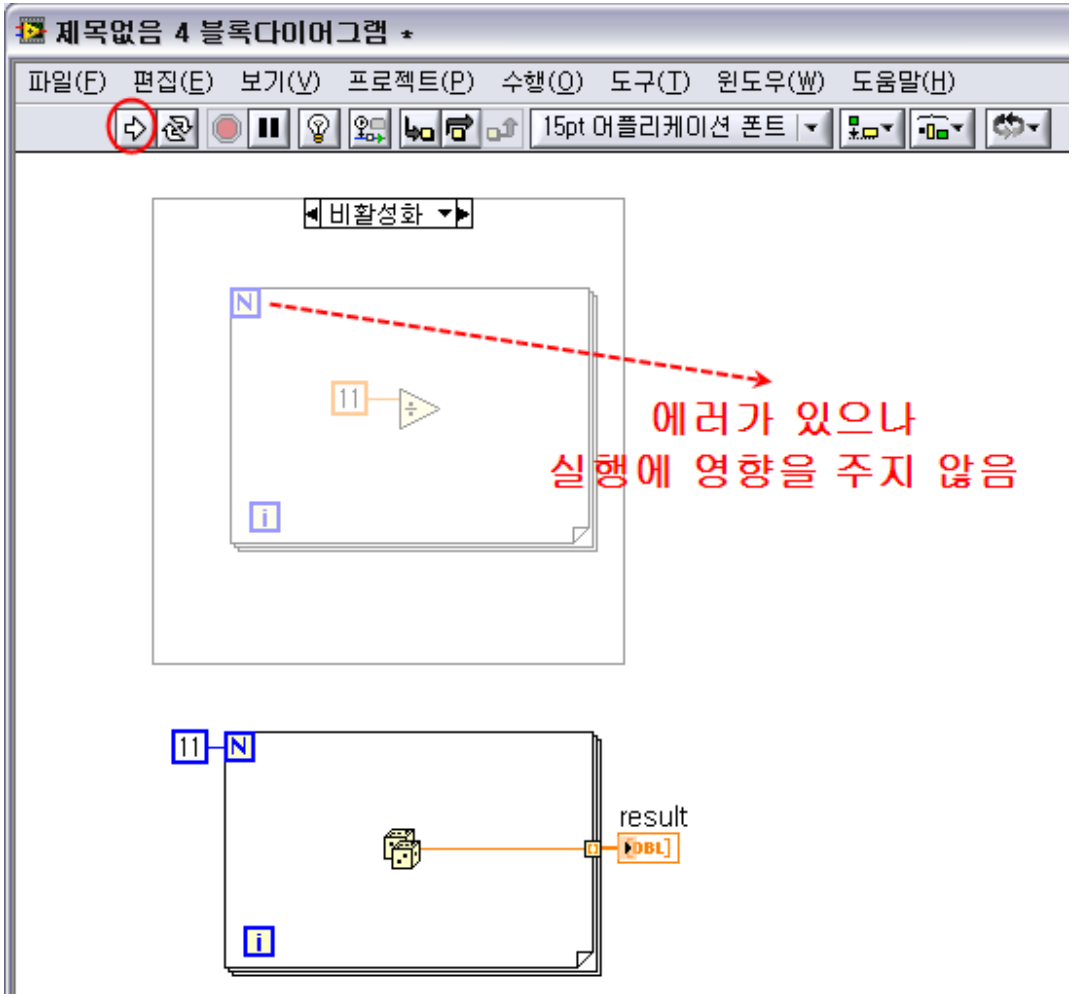


실습4-10-1) 정적 이벤트 사용법

❖ 이벤트의 바로가기메뉴 > 이벤트 케이스 추가 ... 를 선택하여 그림과 같이 설정하고 '확인' 버튼을 누른다

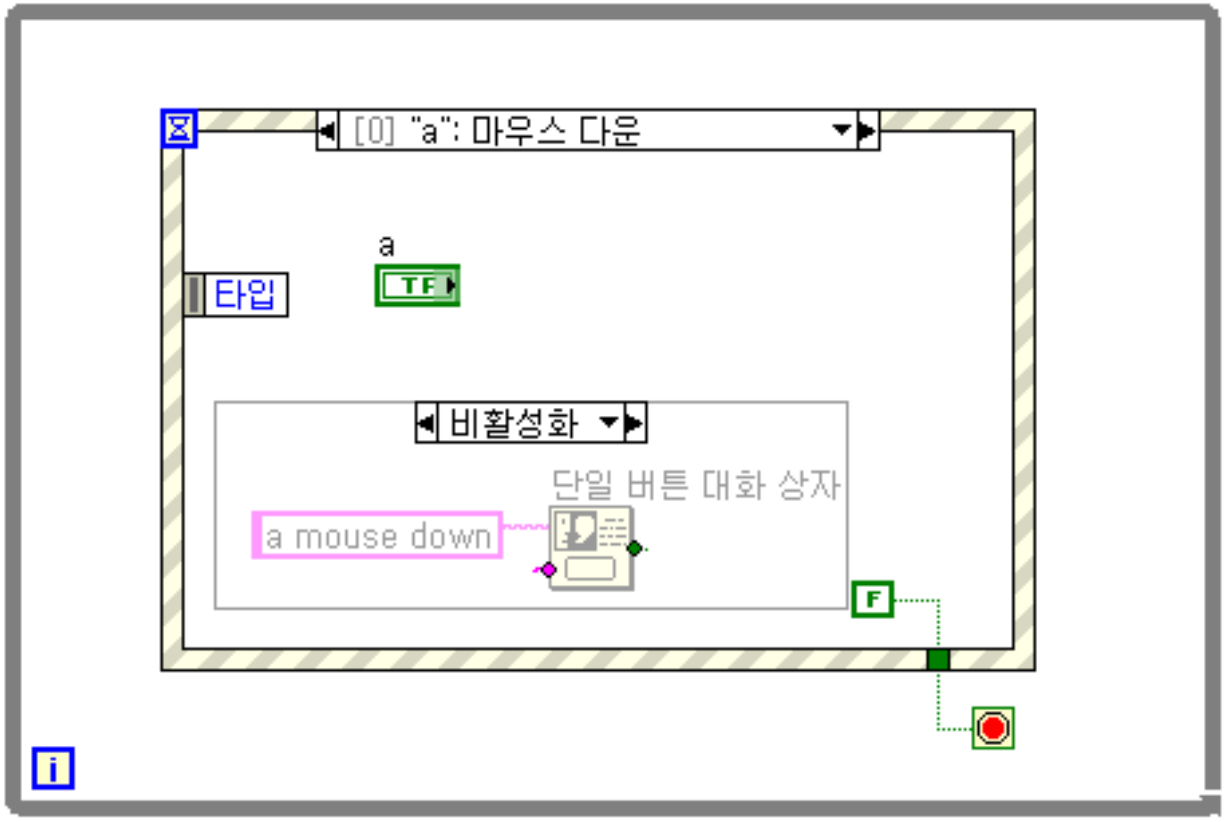


다이어그램 비활성화 구조

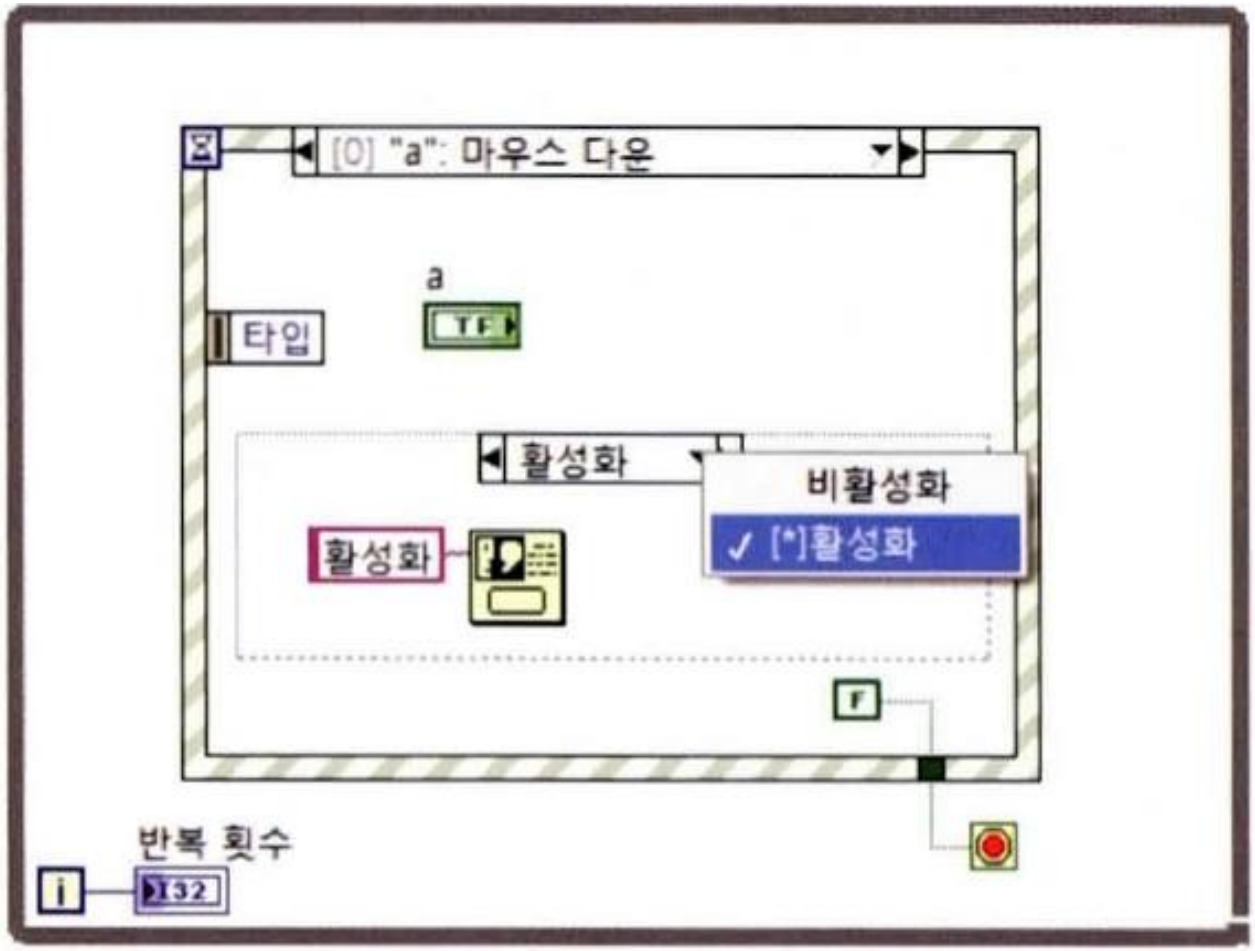


실습4-11-1)
다이어그램 비활성화 구조 사용법

❖ 블록다이어그램에서 다이어그램 비활성화 구조를 다음과 같이 추가



❖ 다이어그램 비활성화 구조에서 활성화를 선택하여 다음과 같이 단일 버튼 대화 상자 **VI** 가져다 놓는다.



❖ VI를 실행한 뒤 'a ' 버튼을 눌렀을 때. '활성화'라는 대화 상자가 나타나는 것을 확인