Intro to Labview

Contents

프로그래밍이란?

- 프로그래밍
  - 사용자의 입력(Input)을 받아 연산한 결과를 출력(Output)하는 과정
- 변수
  - 입력 또는 출력을 표현
  - 숫자형/문자열/불리언 등
- 데이터의 최소 단위
  - 비트(bit) : 0 또는 1의 두 가지 표현
  - 바이트(byte) : 컴퓨터 데이터 표현의 최소단위(8 bits)

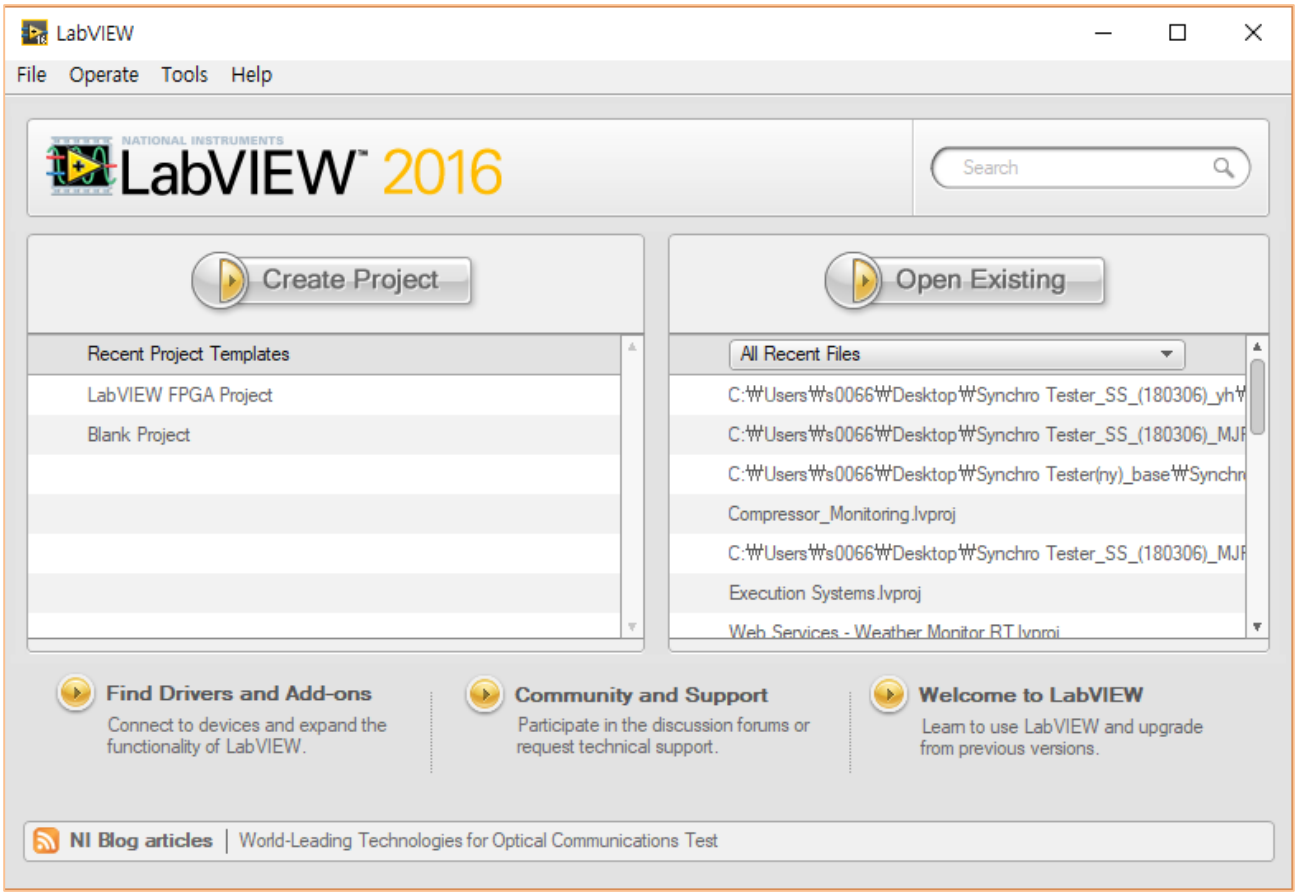
프로그래밍이란?

- 비트의 데이터 표현
  - 예) 8비트 정수의 표현

0	1 1 1 1 1 1 1	= 127
0	1 1 1 1 1 1 0	= 126
0	0 0 0 0 0 1 0	= 2
0	0 0 0 0 0 0 1	= 1
0	0 0 0 0 0 0 0	= 0
1	1 1 1 1 1 1 1	= -1
1	1 1 1 1 1 1 0	= -2
1	0 0 0 0 0 0 1	= -127
1	0 0 0 0 0 0 0	= -128

2의 보수 표현을 이용한 정수와 비트의 관계

랩뷰 시작하기


LabVIEW
File Operate Tools Help
NATIONAL INSTRUMENTS
LabVIEW 2016
Search
Create Project
Open Existing
Recent Project Templates
LabVIEW FPGA Project
Blank Project
All Recent Files
C:₩Users₩s0066₩Desktop₩Synchro Tester_SS_(180306)_yh₩
C:₩Users₩s0066₩Desktop₩Synchro Tester_SS_(180306)_MJ
C:₩Users₩s0066₩Desktop₩Synchro Tester(ny)_base₩Synchr
Compressor_Monitoring.lvproj
C:₩Users₩s0066₩Desktop₩Synchro Tester_SS_(180306)_MJ
Execution Systems.lvproj
Web Services - Weather Monitor RT.lvproj
Find Drivers and Add-ons
Connect to devices and expand the functionality of LabVIEW.
Community and Support
Participate in the discussion forums or request technical support.
Welcome to LabVIEW
Learn to use LabVIEW and upgrade from previous versions.
NI Blog articles | World-Leading Technologies for Optical Communications Test

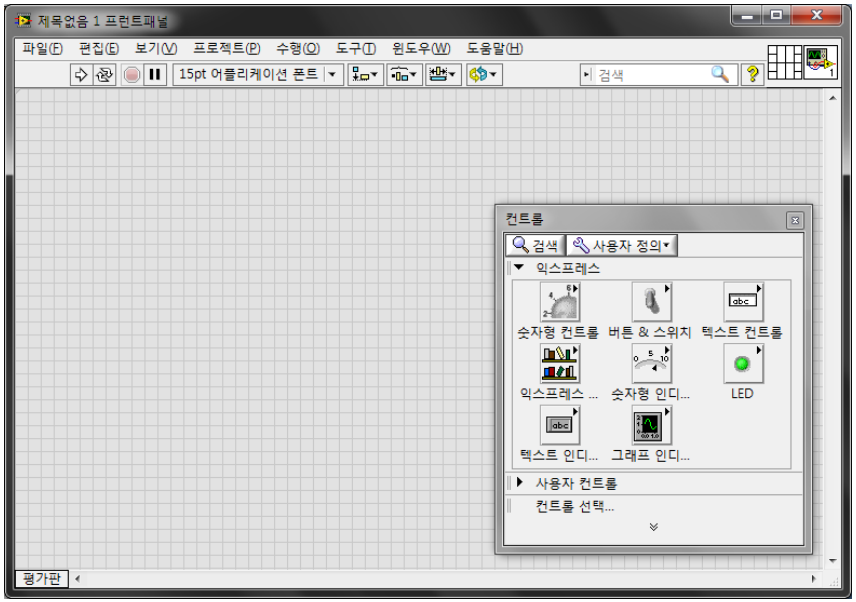
랩뷰의 구성 요소

- 랩뷰 파일 확장자
  - VI(Virtual Instrument)
- 프런트패널/블록다이어그램
  - 사용자 인터페이스(User Interface)
  - 소스 코드(Source Code)

랩뷰의 구성 요소

- 프런트패널(Front Panel)
  - 사용자 인터페이스(UI) 구성하는 부분

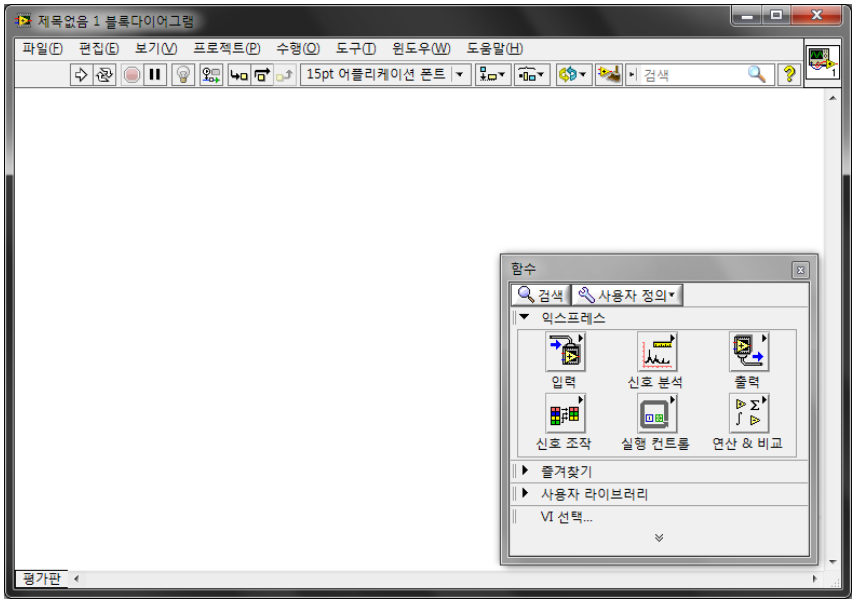
- 컨트롤 팔레트
  - UI 구성용 아이콘 포함



랩뷰의 구성 요소

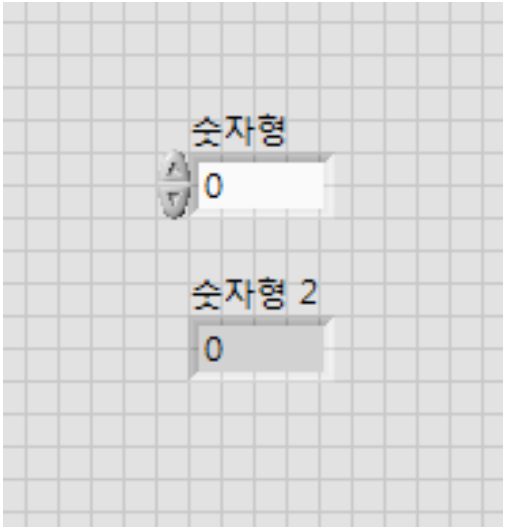
- 블록다이어그램(Block Diagram)
  - 프로그램의 기능을 구성하는 소스코드에 해당

- 함수 팔레트
  - 연산을 위한 함수 포함



랩뷰의 구성 요소

- 컨트롤(Control)/인디케이터(Indicator)
  - 각각 입력 및 출력에 해당
  - 프런트패널의 컨트롤 팔레트에서 생성
  - 화살표 키 또는 마우스 드래그로 위치 변경



연습문제 1

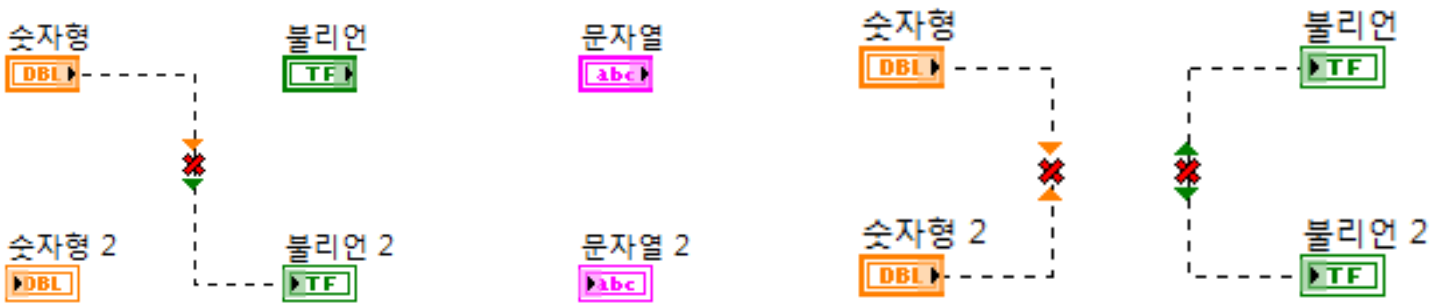
- [숫자형]/[문자열]/[불리언] 각각의 컨트롤 및 인디케이터 만들기

랩뷰 데이터의 전달

- 와이어(Wire)
  - 랩뷰 데이터를 전달하는 통로
  - 데이터의 타입에 따라 다른 색깔

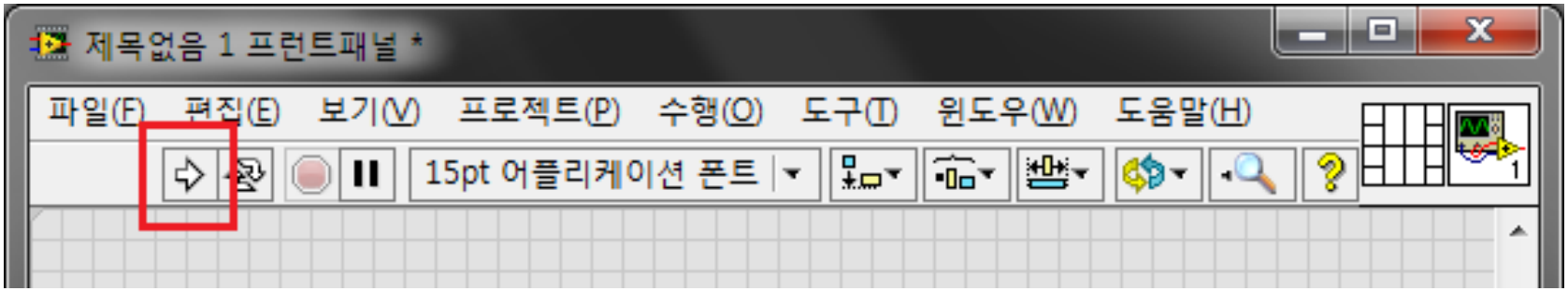
랩뷰 데이터의 전달

- 깨진 와이어(Broken Wire)
  - 잘못된 와이어의 연결
  - 서로 다른 데이터 타입 연결
  - 같은 컨트롤 또는 인디케이터 끼리 연결



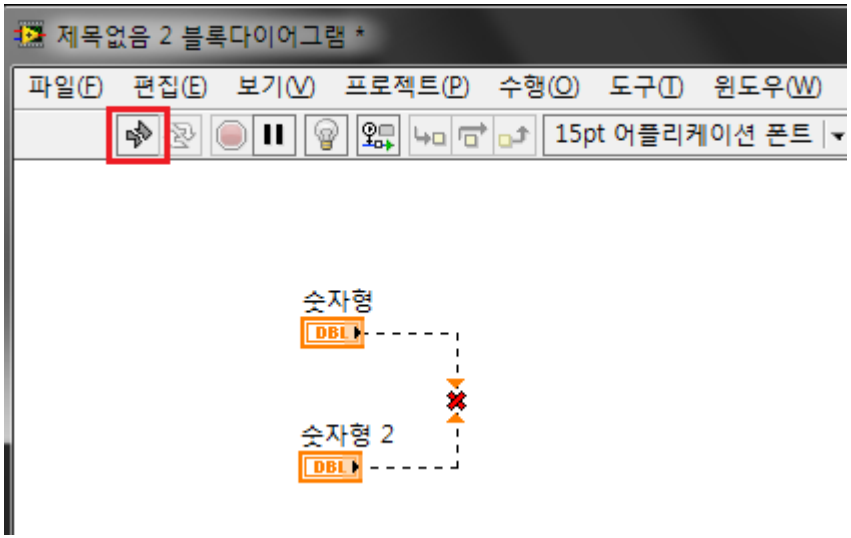
랩뷰의 실행

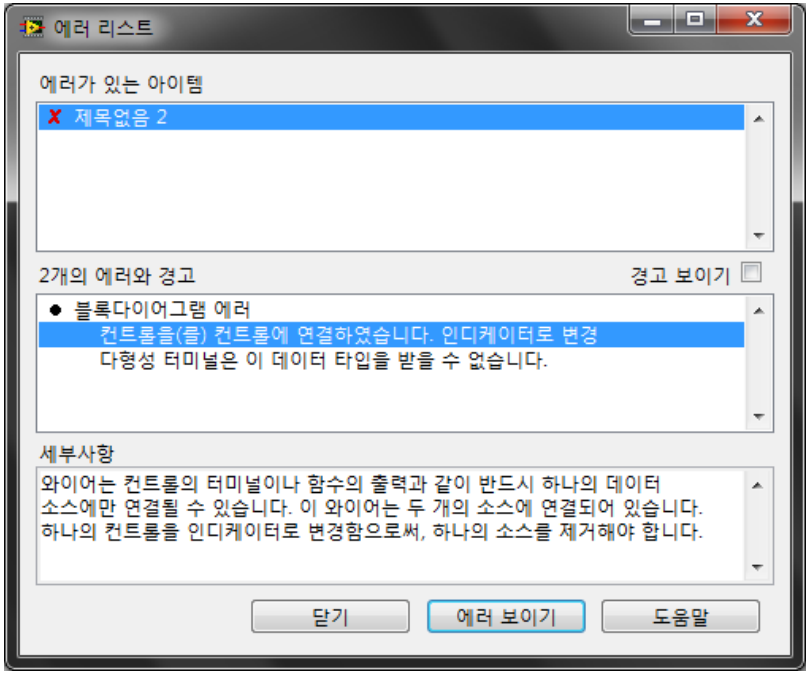
- 도구 아이콘 중 화살표 모양 버튼 이용
- 단축키 **Ctrl+R**



랩뷰의 실행

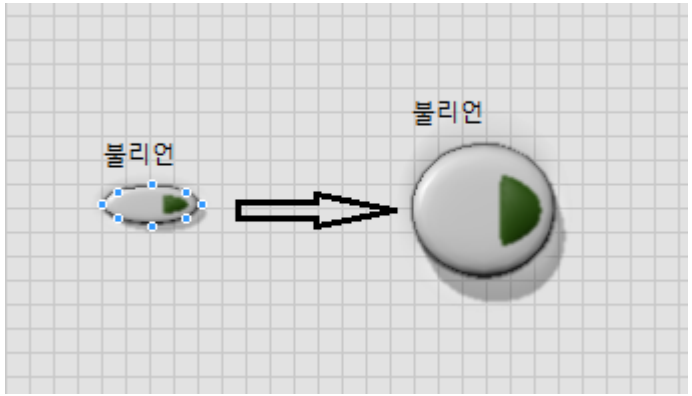
- 깨진 실행 버튼
  - VI의 문법적 오류로 인한 실행 불가 상태
  - 깨진 와이어 발생 등





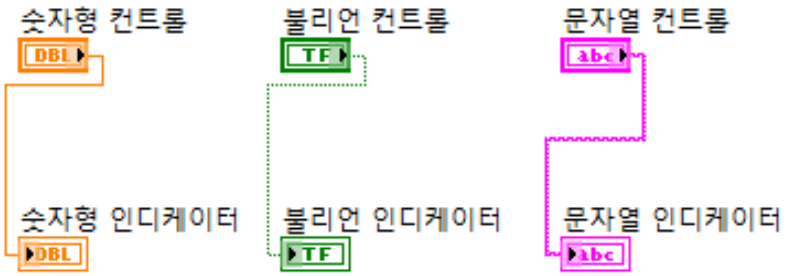
입출력 아이콘의 크기 및 이름 변경

- 아이콘의 크기 변경
  - 크기 조절점 이용
- 이름(라벨,Label) 변경
  - 텍스트 부분 더블 클릭 후 수정



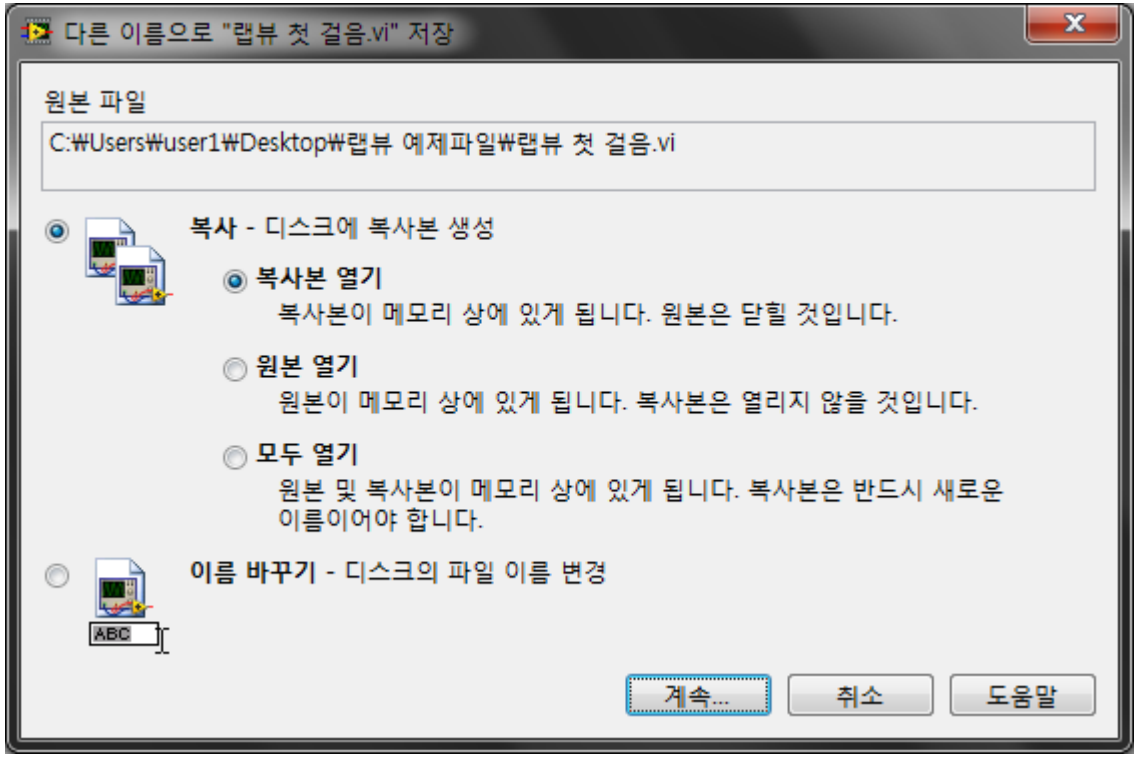
연습문제 2

- 컨트롤 및 인디케이터의 라벨 변경하기
- 와이어의 연결 및 실행하기



VI 저장하기

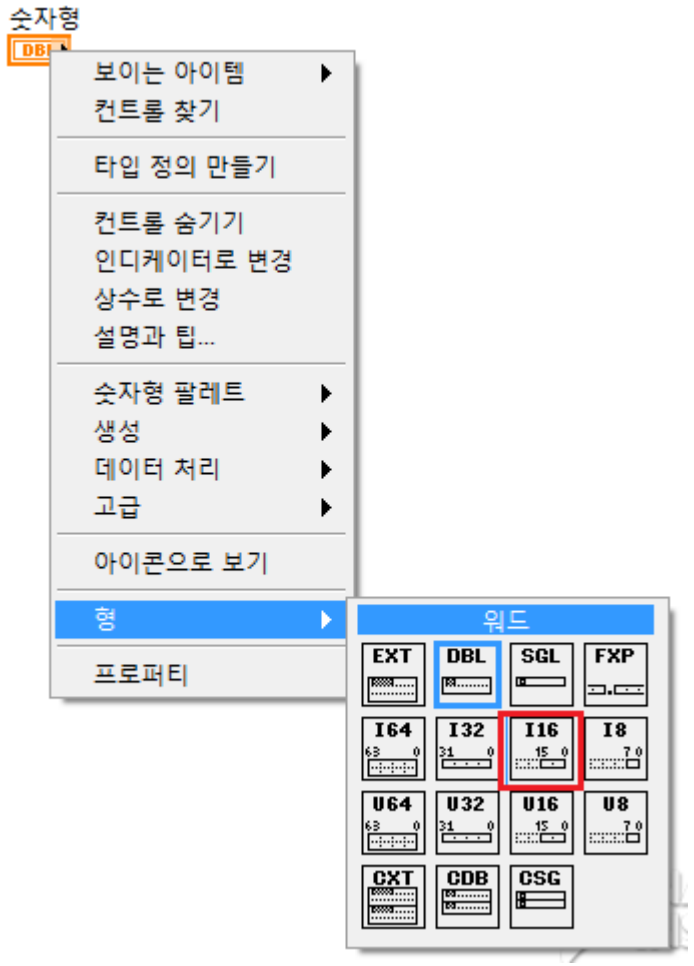
- 복사본 열기 / 원본 열기 / 모두 열기 옵션
- 단축키 **Ctrl+S**



랩뷰의 데이터 형식

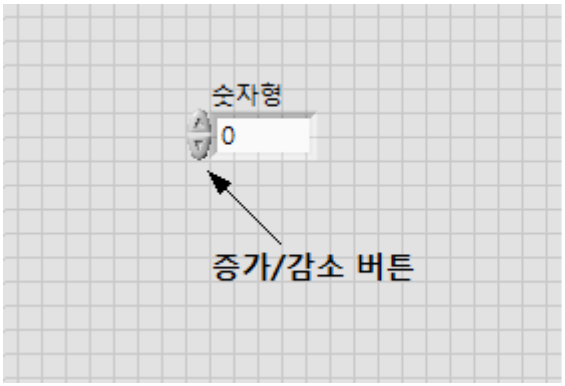
- 숫자형(Numeric)
  - 부동소수점형 실수
  - 고정소수점형 실수
  - 정수
  - 부호없는 정수

- 형(type)에 따라 다른 터미널 색깔



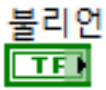
연습문제 3

- 숫자형 컨트롤 및 인디케이터의 데이터 타입 변환하기



랩뷰의 데이터 형식

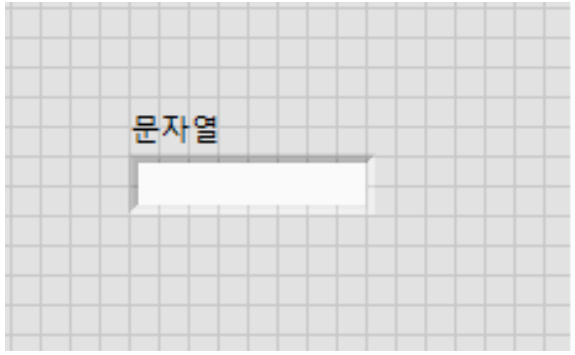
- 불리언(Boolean)
  - 참 또는 거짓
  - 0 또는 1
- 초록색 터미널

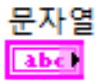


랩뷰의 데이터 형식

- 문자열(String)
  - ASCII 방식
- 분홍색 터미널

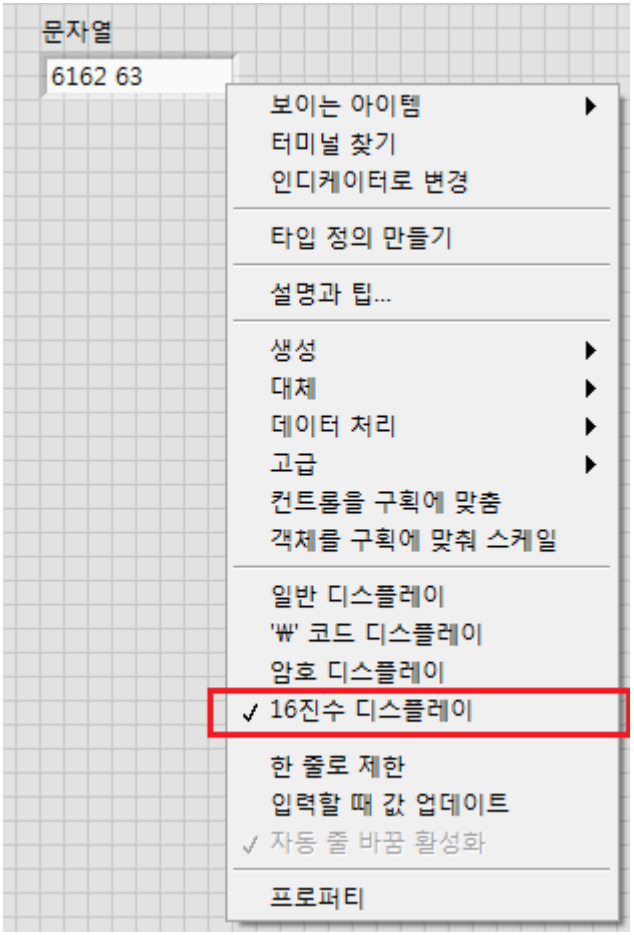
2진수	10진수	16진수	문자
0100 0001	65	41	A
0100 0010	66	42	B
0100 0011	67	43	C
0100 0100	68	44	D
0100 0101	69	45	E





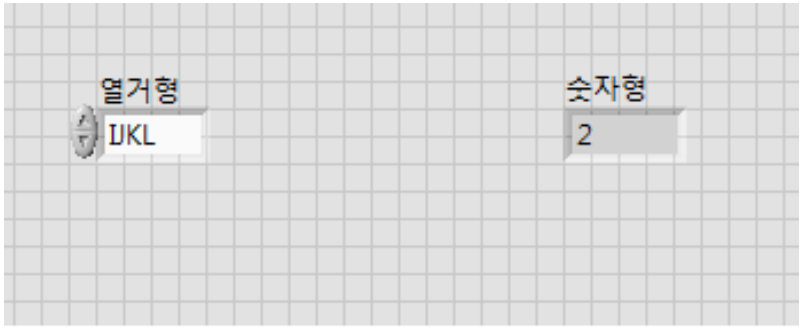
랩뷰의 데이터 형식

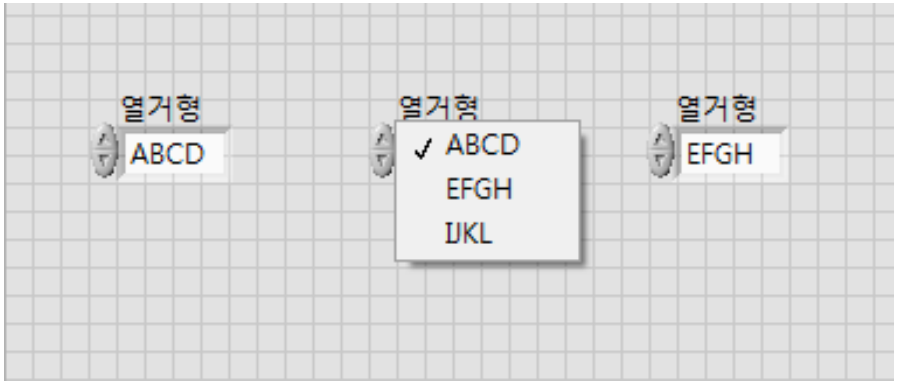
- 문자열(String)
  - 보기 옵션 변경 가능
  - 일반
  - 16진수
  - 암호
  - '₩' 코드



랩뷰의 데이터 형식

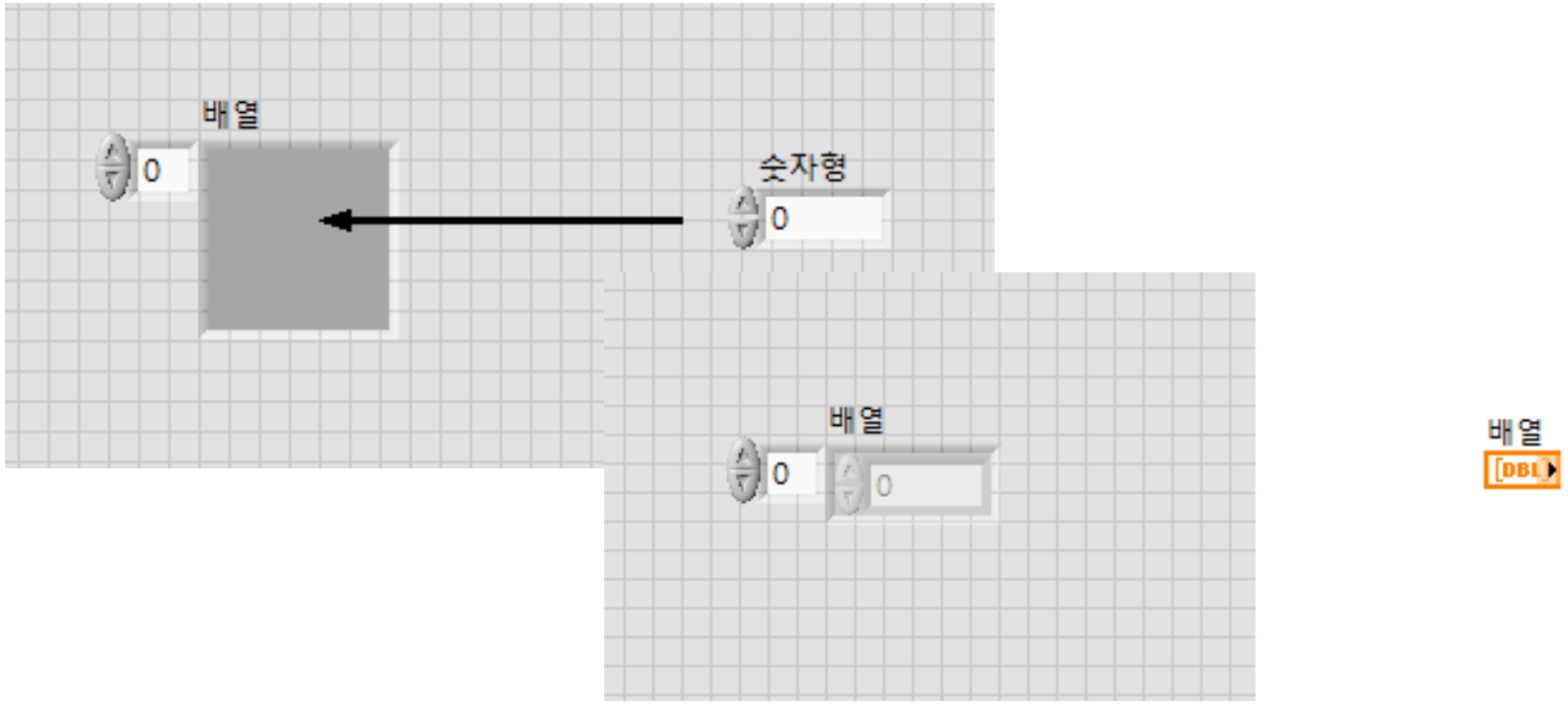
- 열거형(Enumeration)
  - 문자처럼 보이지만 실제로는 숫자
  - 아이템 편집 옵션 이용





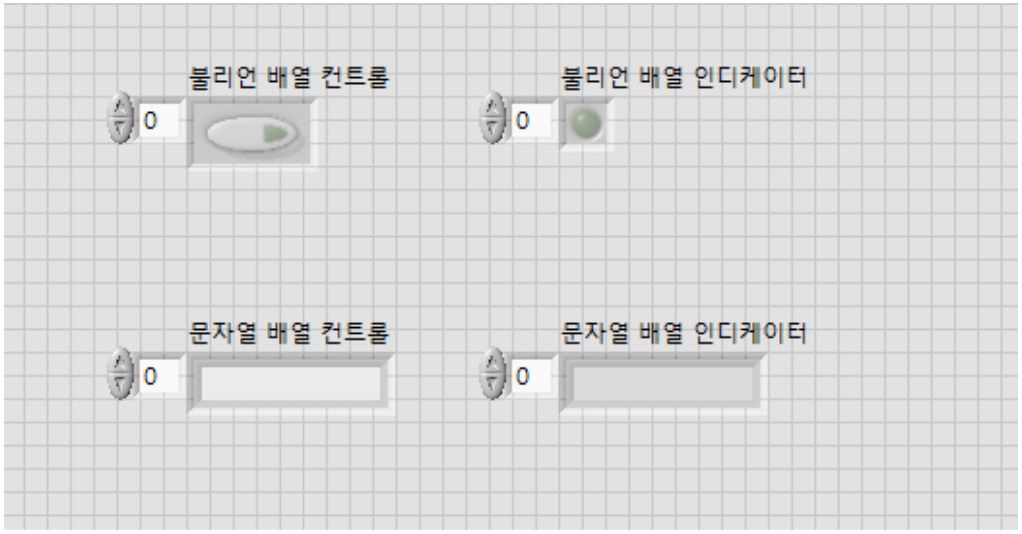
랩뷰의 데이터 형식

- 배열(Array)
  - 배열 껍데기에 원하는 데이터 형태 끌어 놓아 만듦



연습문제 4

- 여러 가지 데이터 형태의 배열 만들기



불리언 배열 컨트롤

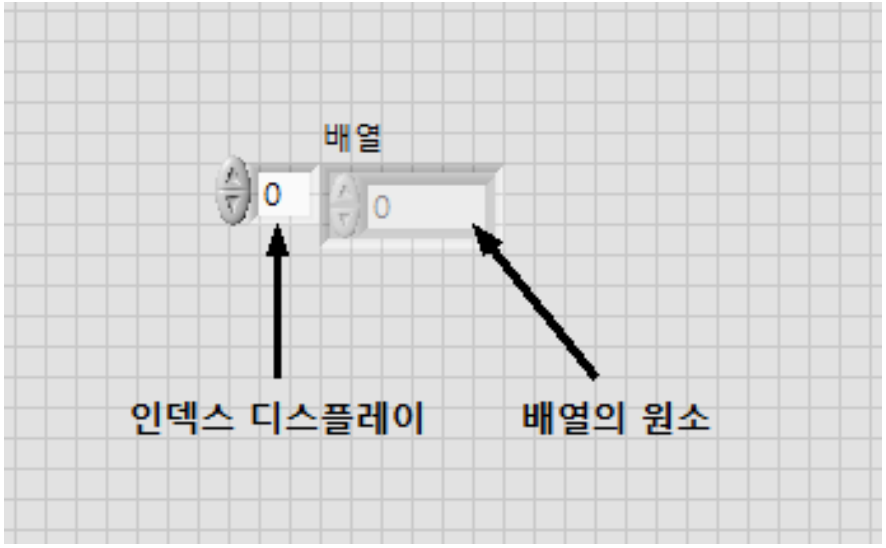
불리언 배열 인디케이터

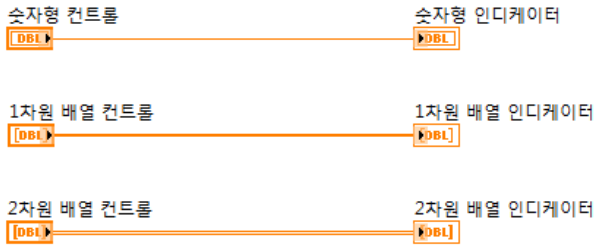
문자열 배열 컨트롤

문자열 배열 인디케이터

랩뷰의 데이터 형식

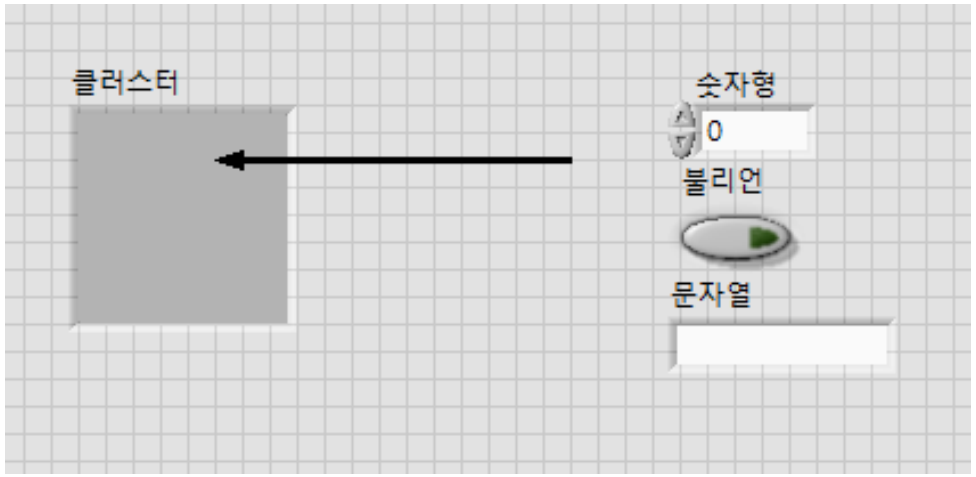
- 배열(Array)
  - 원소와 차원으로 구성(단, 원소는 같은 종류)
  - 원소의 순서: 인덱스(Index)
  - 차원에 따라 와이어 굵기 변화

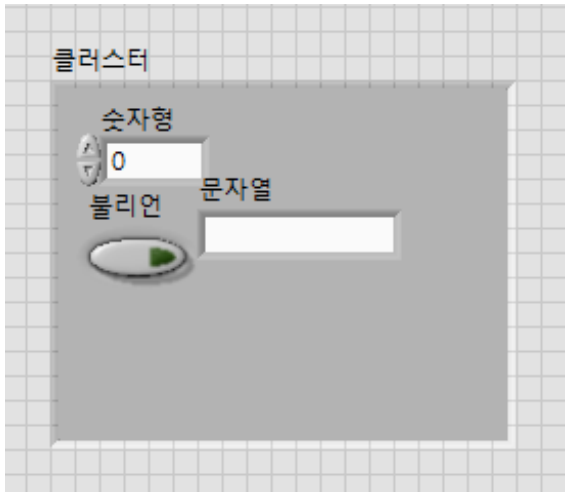




랩뷰의 데이터 형식

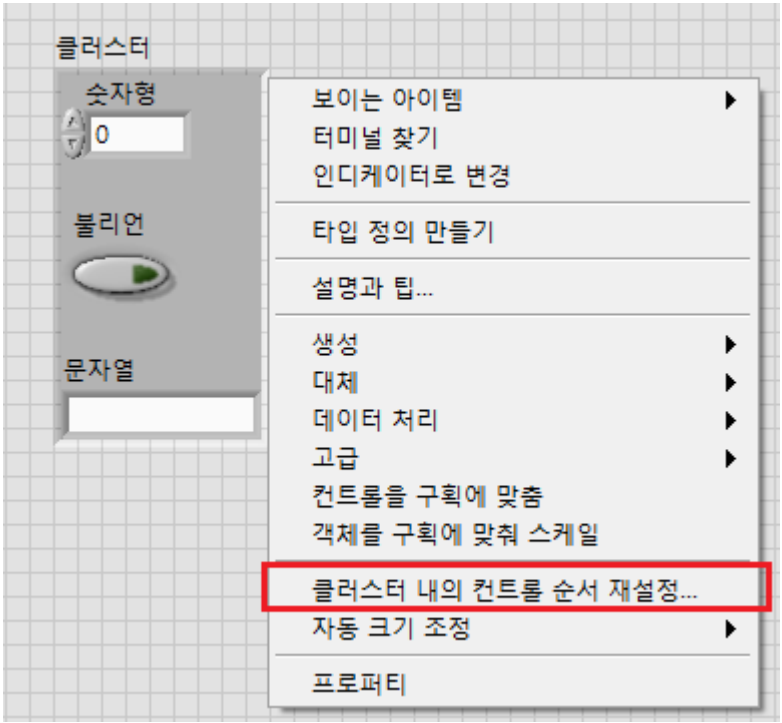
- 클러스터(Cluster)
  - 여러 가지 형태 변수들의 집합
  - 클러스터 껍데기에 필요한 변수 형태를 끌어 놓아 만듦

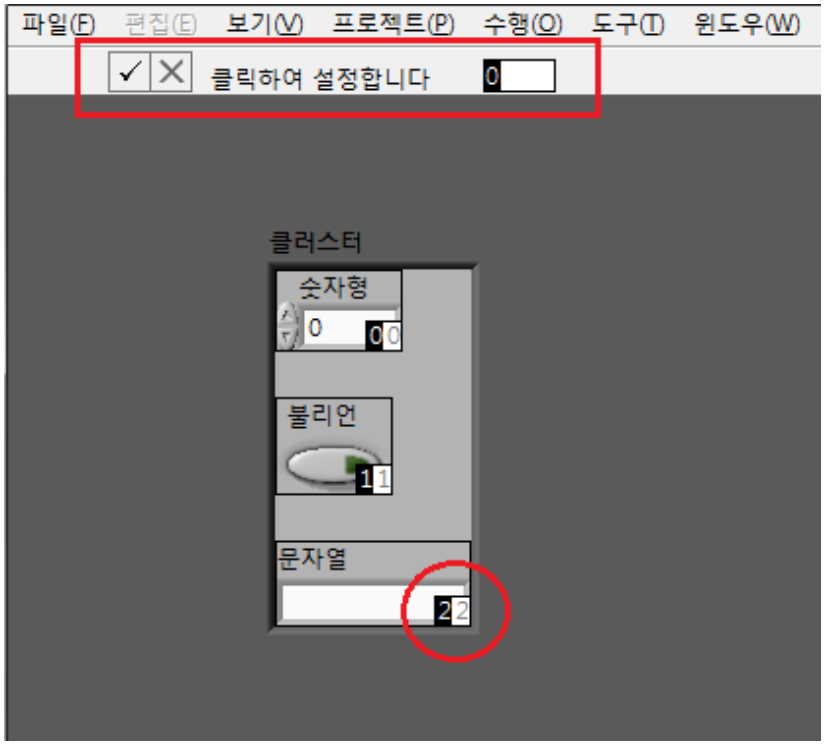




랩뷰의 데이터 형식

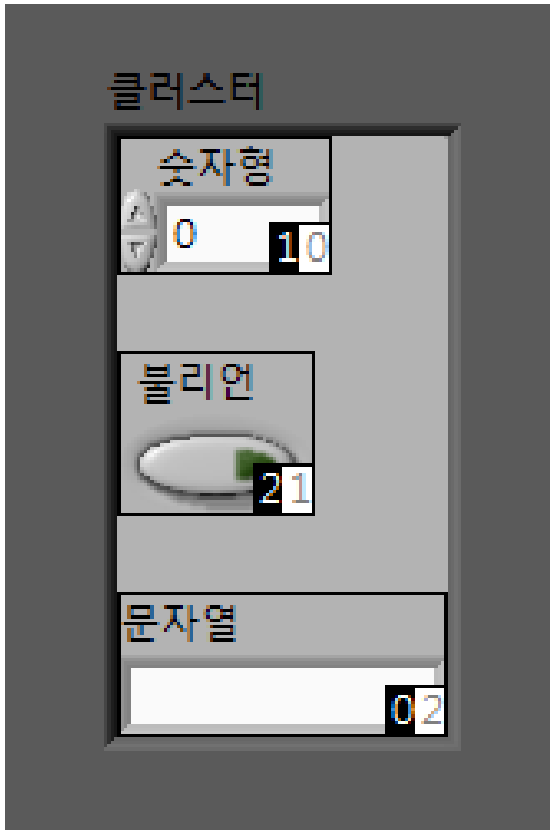
- 클러스터(Cluster)
  - 각각의 원소는 순서를 가짐
  - 클러스터 내의 컨트롤 순서 재설정 옵션 이용





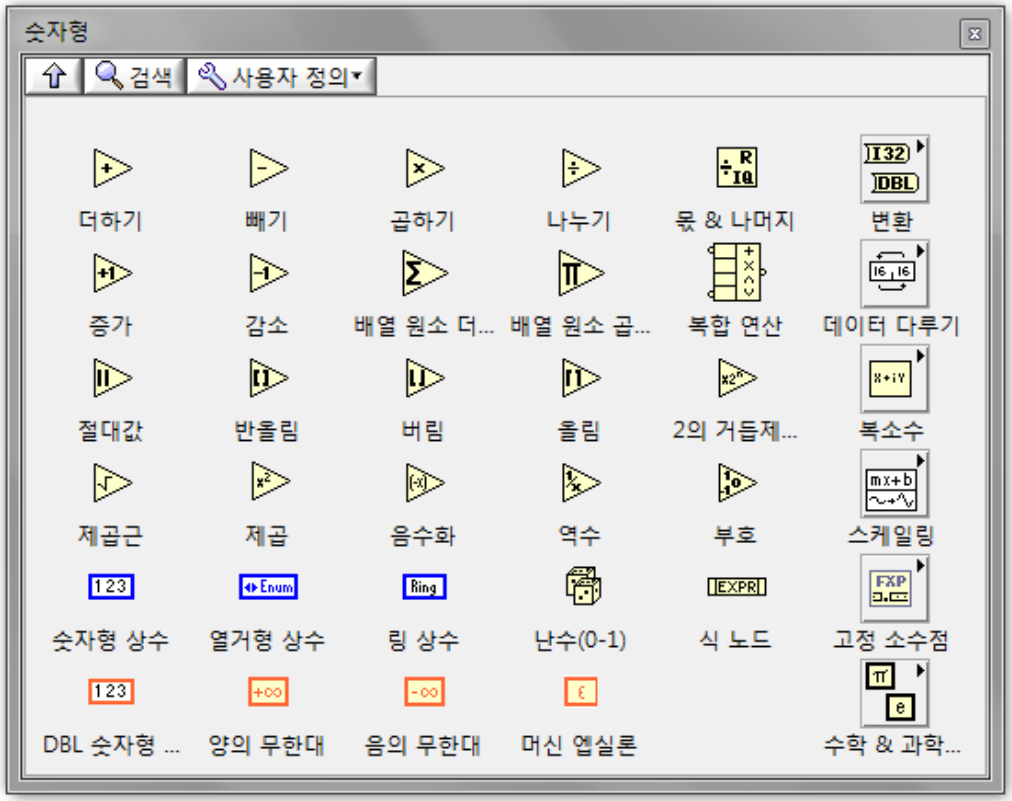
연습문제 5

- 클러스터 내의 원소 순서 변경하기



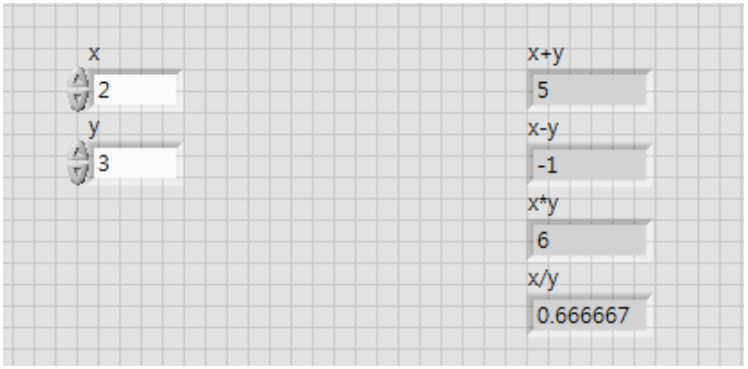
숫자형 함수 팔레트

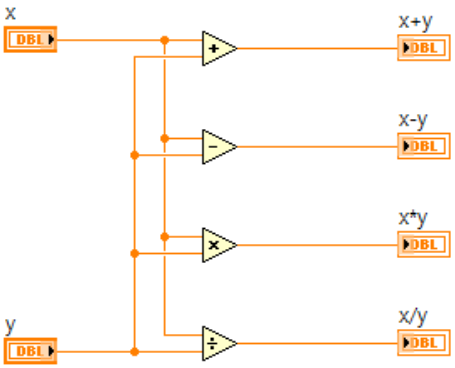
- 숫자형 함수 팔레트
  - 사칙연산 / 숫자형 상수 / 절대값 함수 등

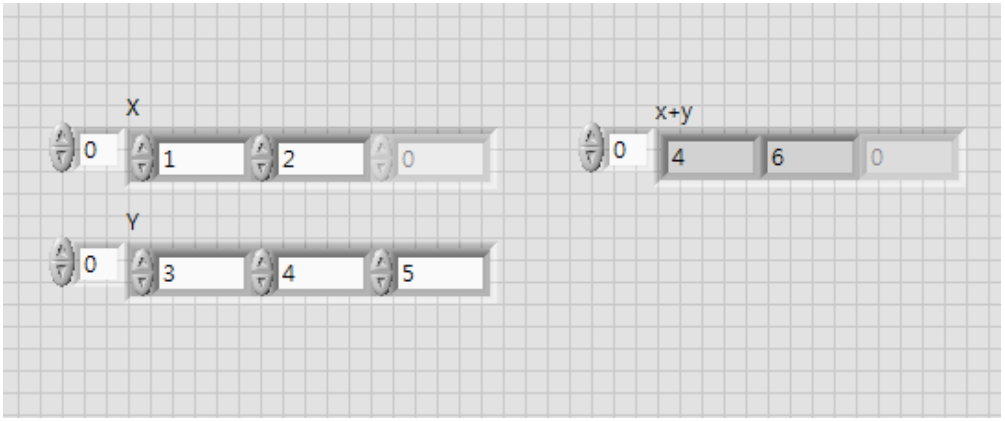


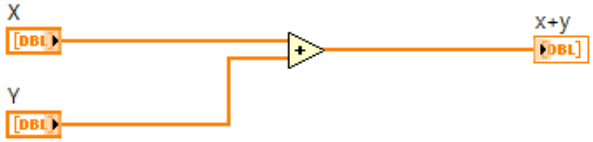
숫자형 함수 팔레트

- 사칙 연산 함수의 사용
  - 스칼라 또는 배열 원소 연산 가능



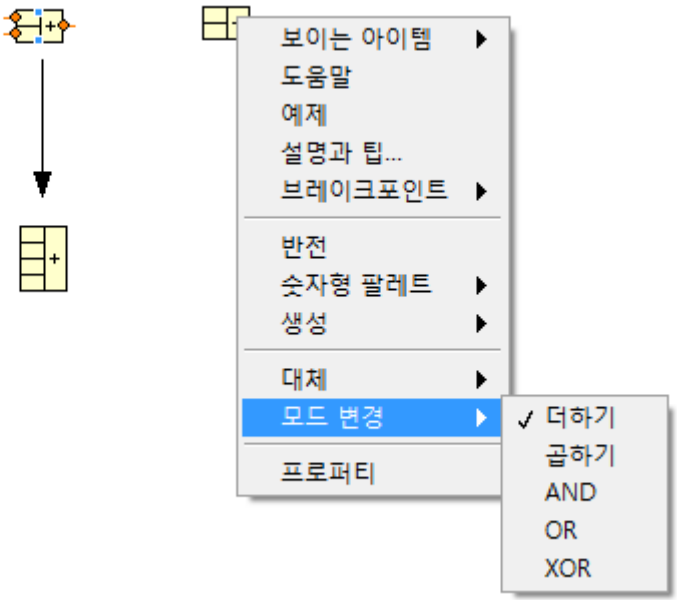






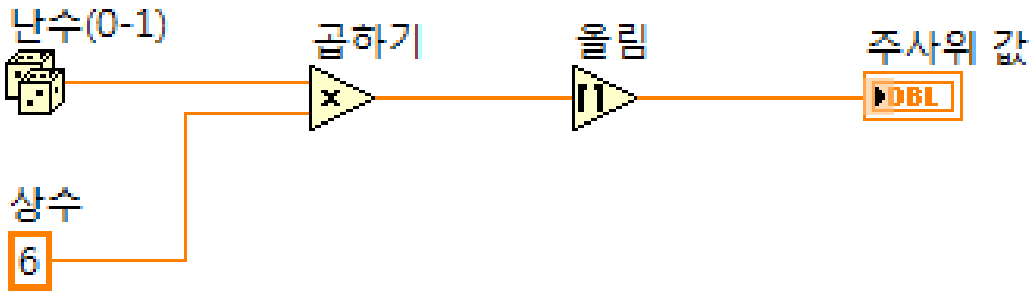
숫자형 함수 팔레트

- 복합 연산 함수
- 숫자형 상수



연습문제 6

- 주사위 던지기 프로그램 만들기

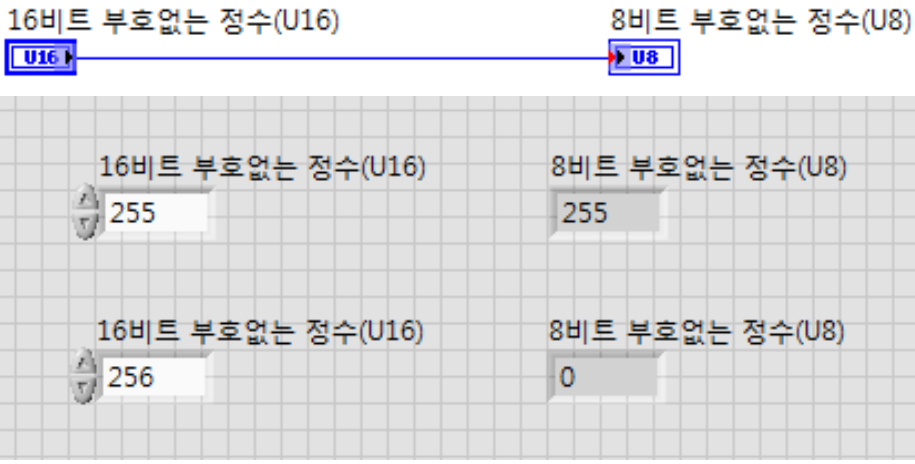


숫자형 함수 팔레트

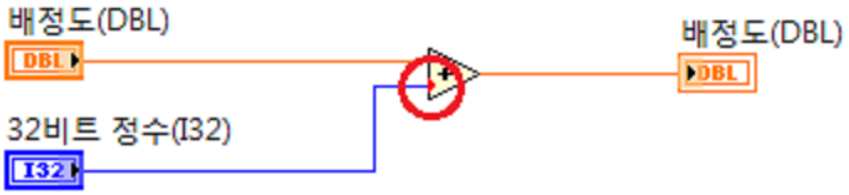
- 숫자형 데이터의 변환
  - 서로 다른 숫자형 데이터 타입 연결

숫자형 함수 팔레트

- 숫자형 데이터의 변환
  - 서로 다른 크기의 숫자형 연결

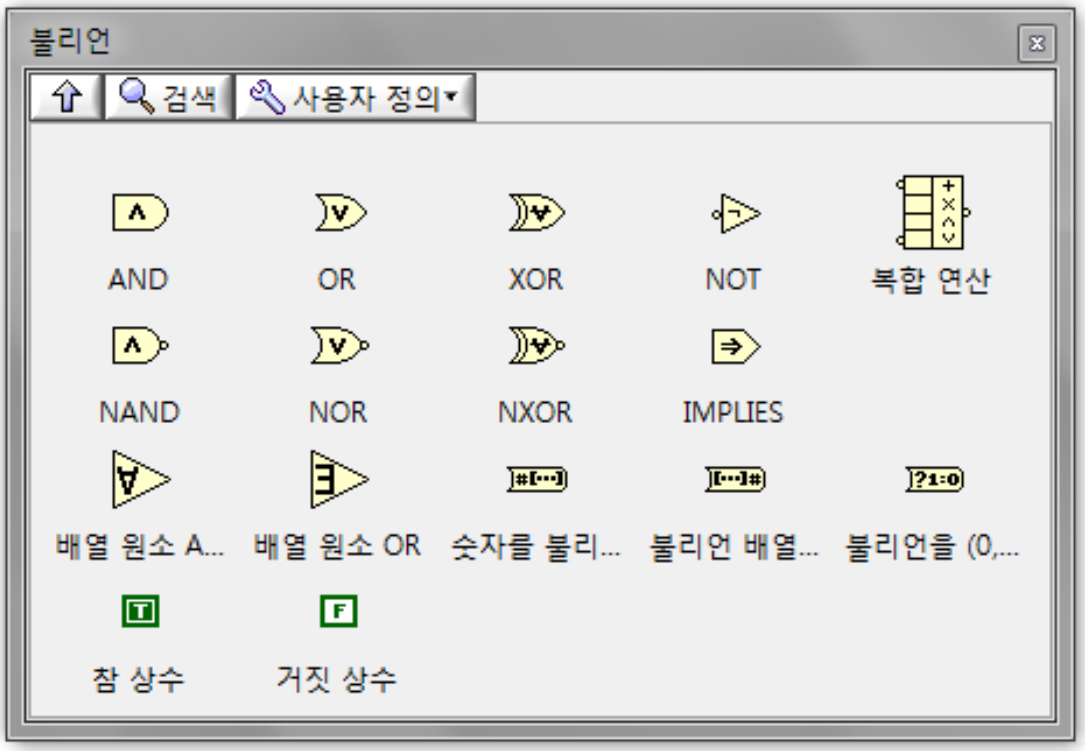


  - 강제



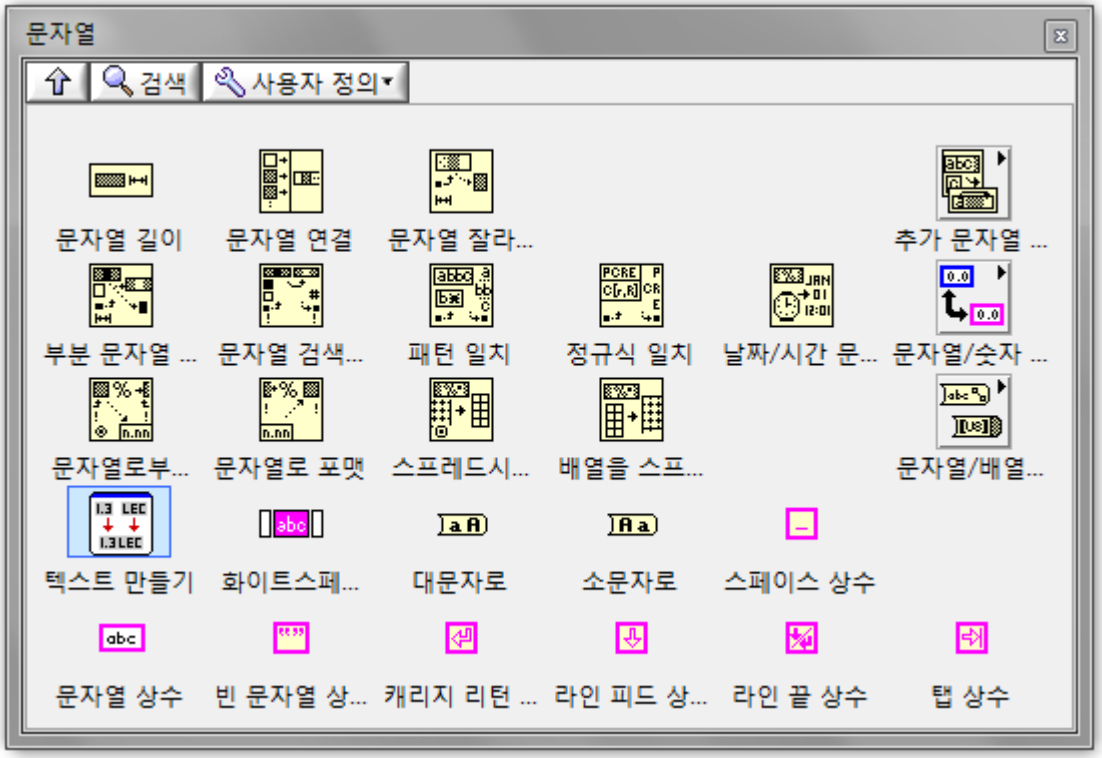
불리언 함수 팔레트

- 불리언 함수 팔레트
  - AND/OR/NOT 등의 불리언 연산 함수



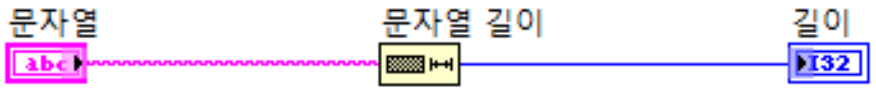
문자열 함수 팔레트

- 문자열 함수 팔레트
  - 문자열 편집 함수의 모음



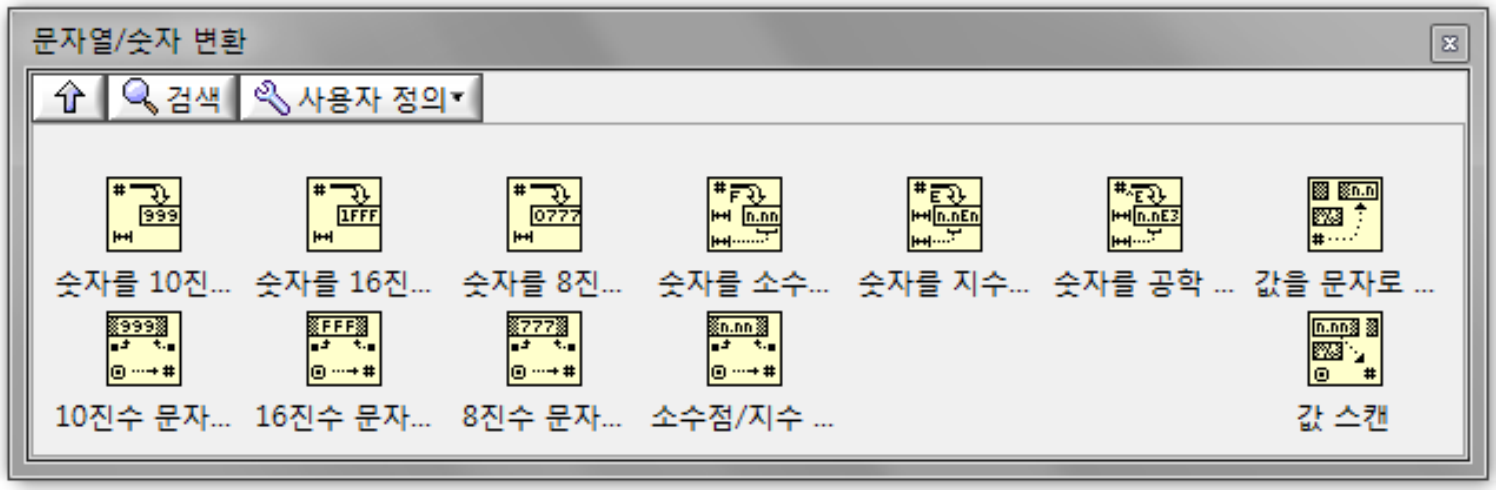
연습문제 7

- 문자열의 길이 구하기



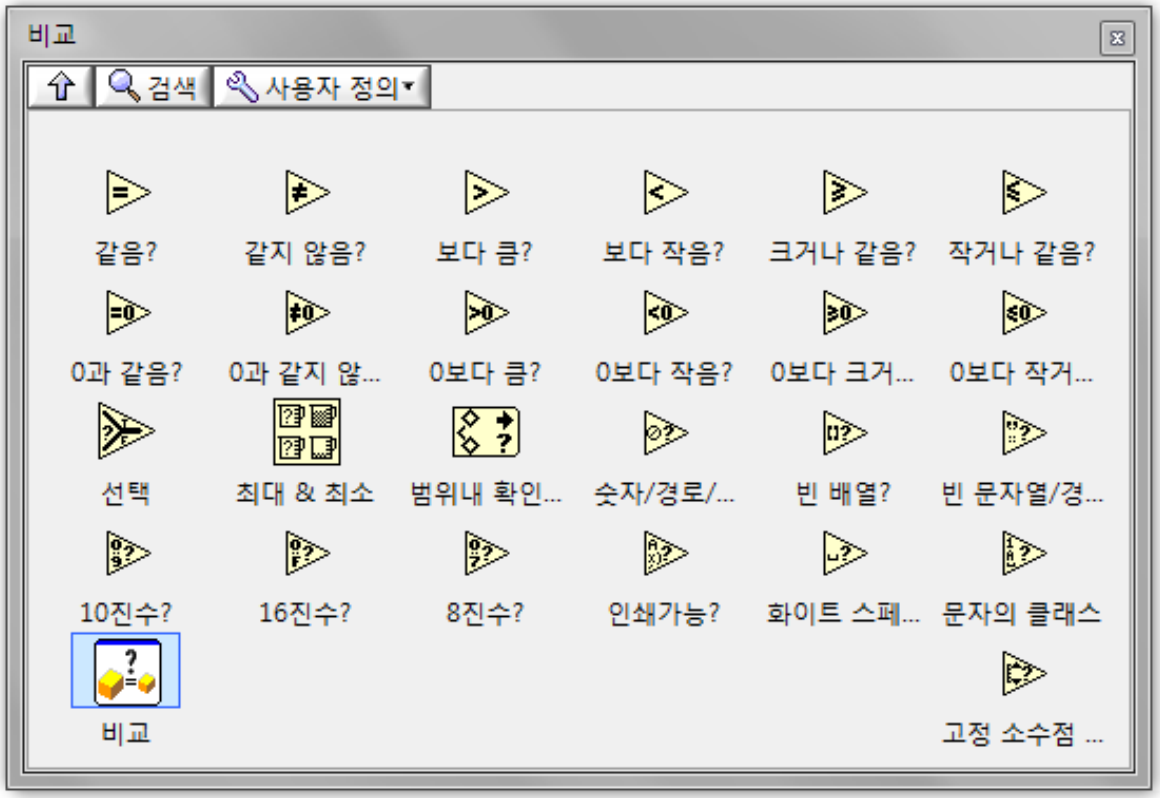
문자열 함수 팔레트

- 문자열/숫자 변환 함수
  - 10진수/16진수/소수점/지수 숫자의 문자 변환
  - 반대의 경우 문자의 숫자 변환



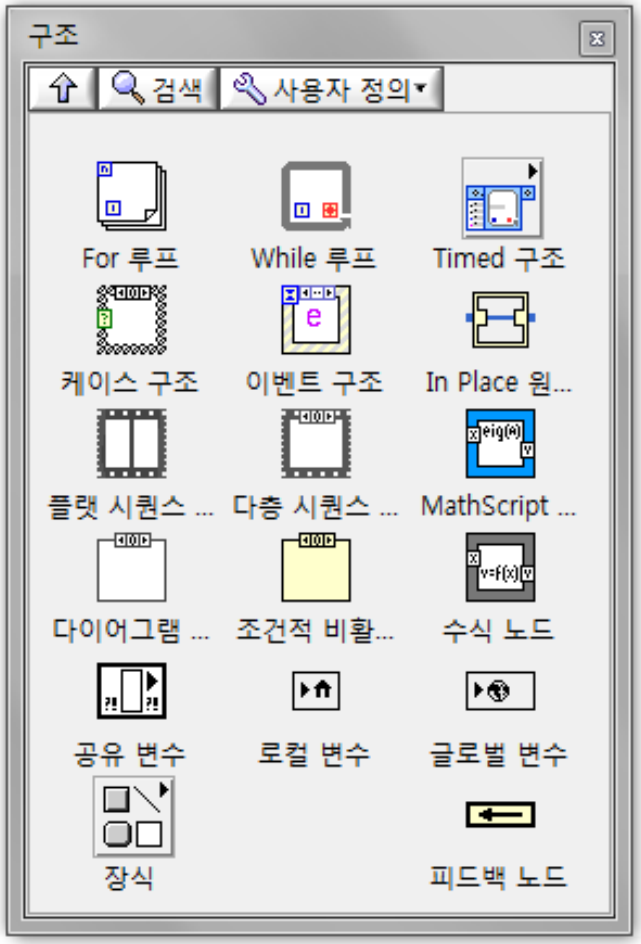
비교 함수 팔레트

- 비교 함수 팔레트
  - 크다/작다 등의 비교 함수 포함



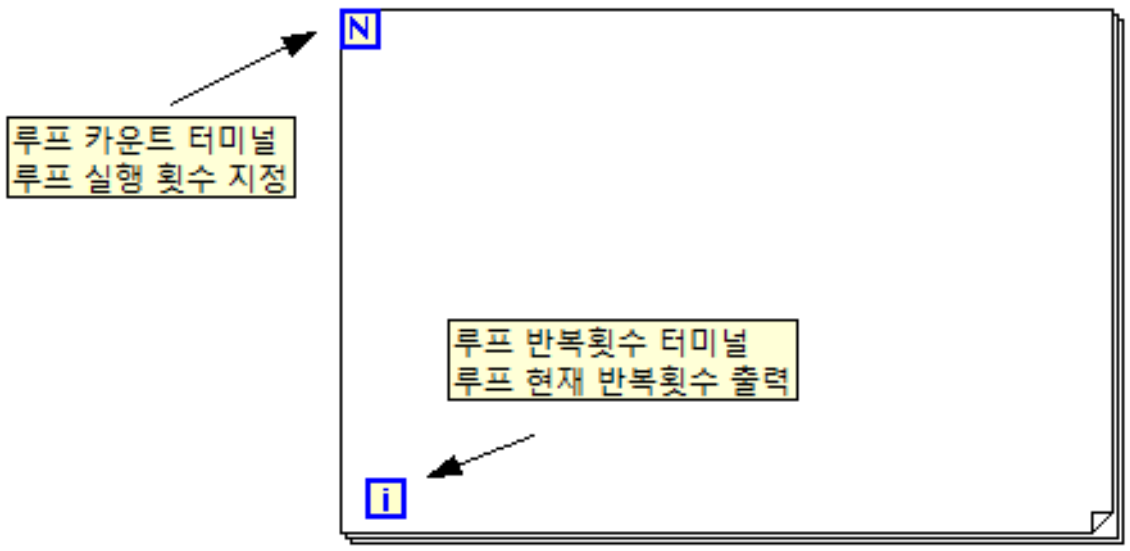
구조 함수 팔레트

- 구조 함수 팔레트
  - For/While 의 루프 함수
  - 시퀀스 구조 함수
  - 케이스 구조 함수



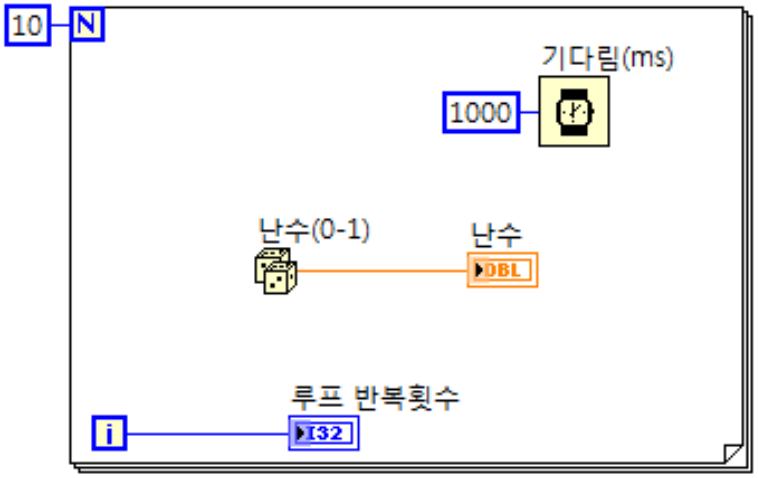
구조 함수 팔레트

- For 루프 함수
  - 지정된 횟수 만큼 루프 내의 코드를 실행
  - 루프 카운트 터미널 / 루프 반복횟수 터미널



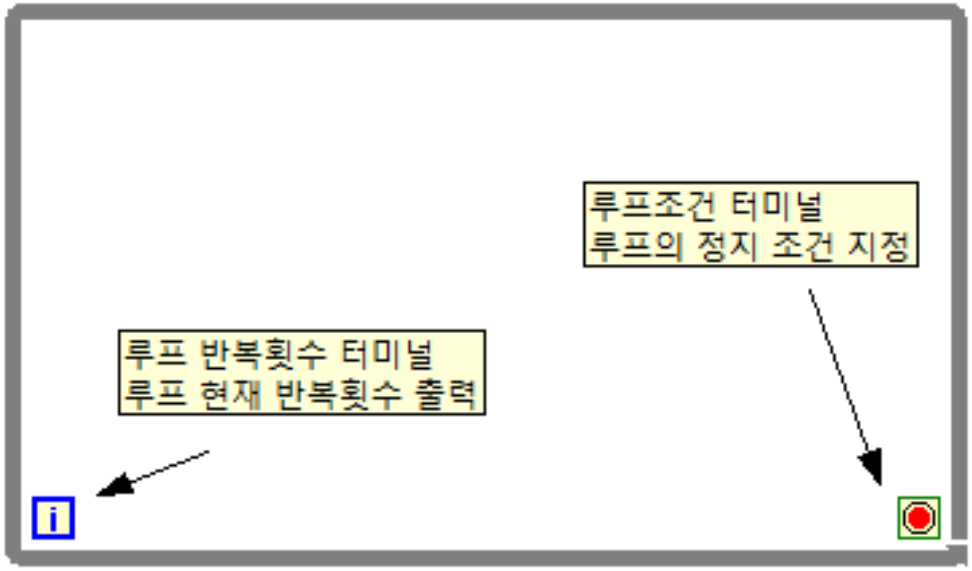
연습문제 2-3

- For 루프를 사용해 난수를 10번 생성하기



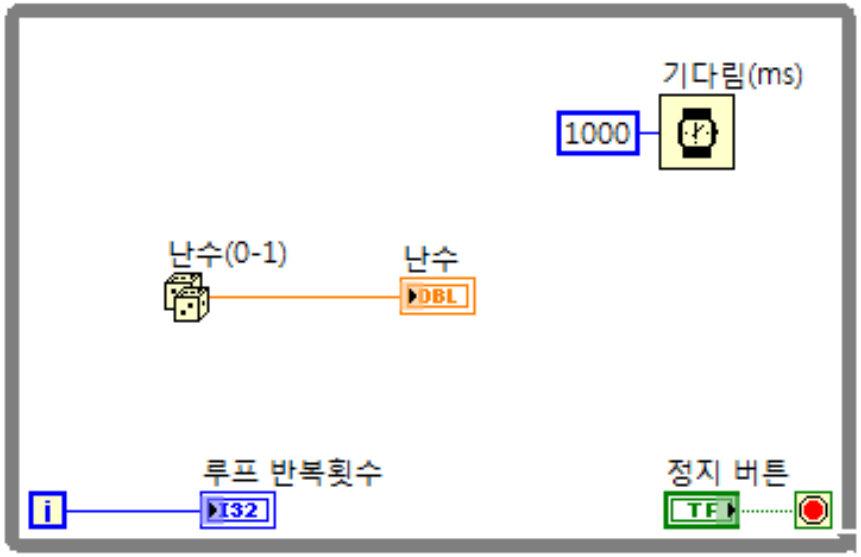
구조 함수 팔레트

- While 루프 함수
  - 정해진 조건을 만족할 때까지 실행 반복
  - 루프조건 터미널 / 루프 반복횟수 터미널



연습문제 8

- 사용자가 정지 버튼을 누를 때 까지 난수를 생성하기



구조 함수 팔레트

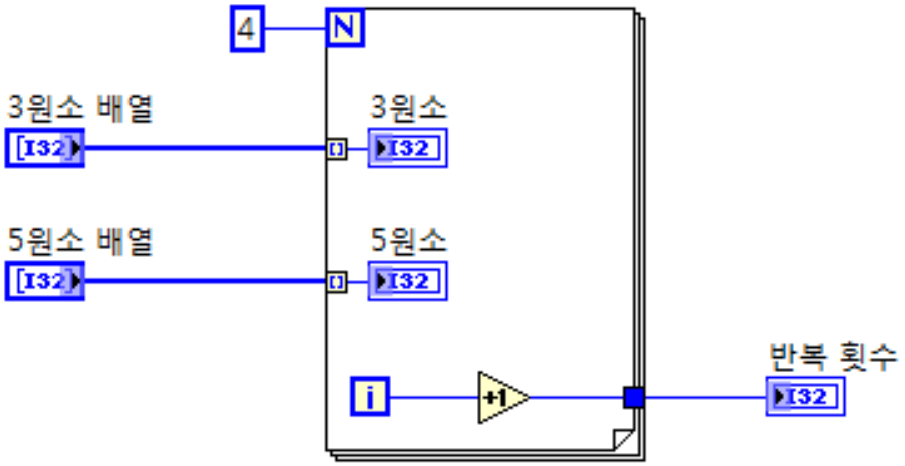
- For 루프 함수 vs. While 루프 함수
  - 최소 실행 횟수
  - 루프 반복 조건 유무
  - 알고리즘에 따라서 유리한 함수 선택 사용

구조 함수 팔레트

- 오토 인덱싱 기능
  - 루프의 입력으로 배열이 연결된 경우
  - 루프의 출력을 배열 형태로 출력할 때

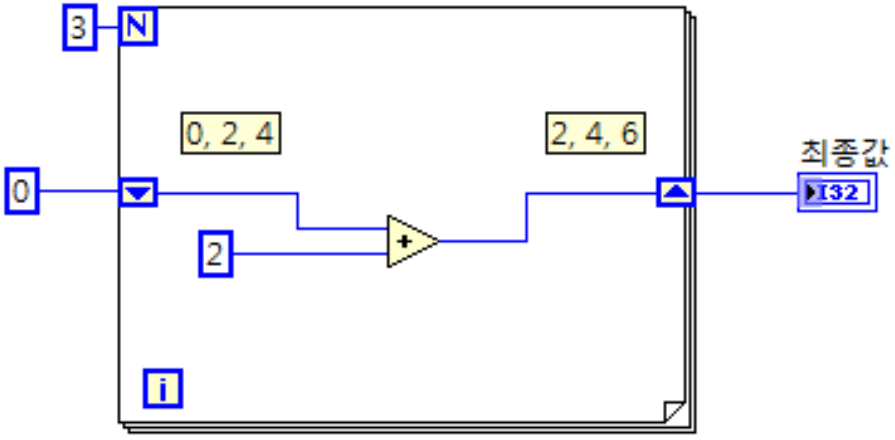
퀴즈

- 아래와 같이 For 루프의 입력에 배열이 입력된 경우 반복 횟수 인디케이터의 최종 값은?



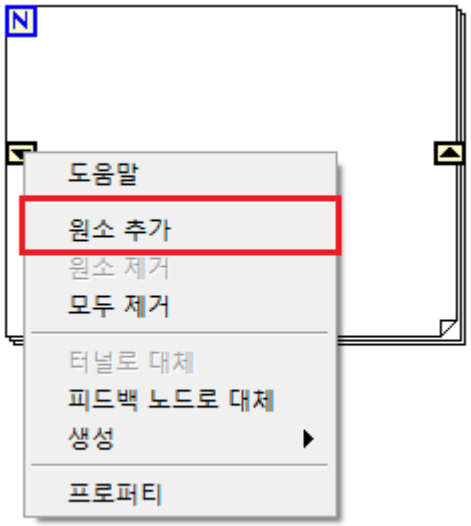
구조 함수 팔레트

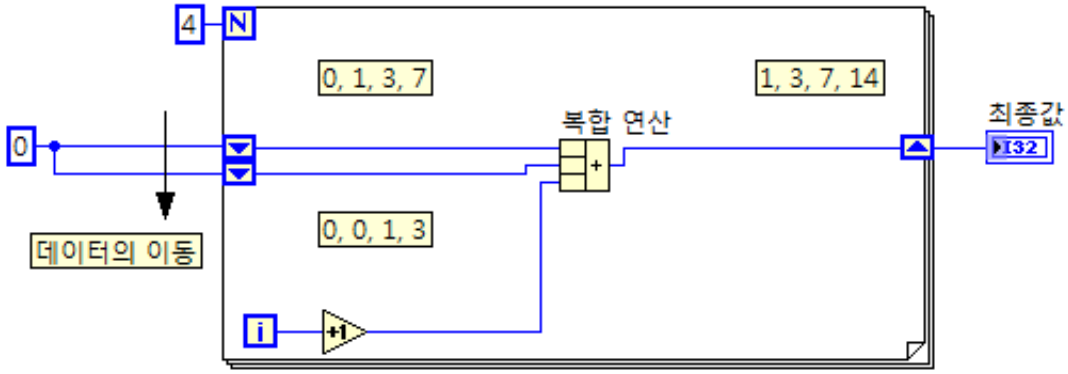
- 시프트 레지스터
  - 이전의 루프에서 연산한 결과를 현재의 루프에서 사용
  - 루프 내에서 사용하는 임시 저장 공간



구조 함수 팔레트

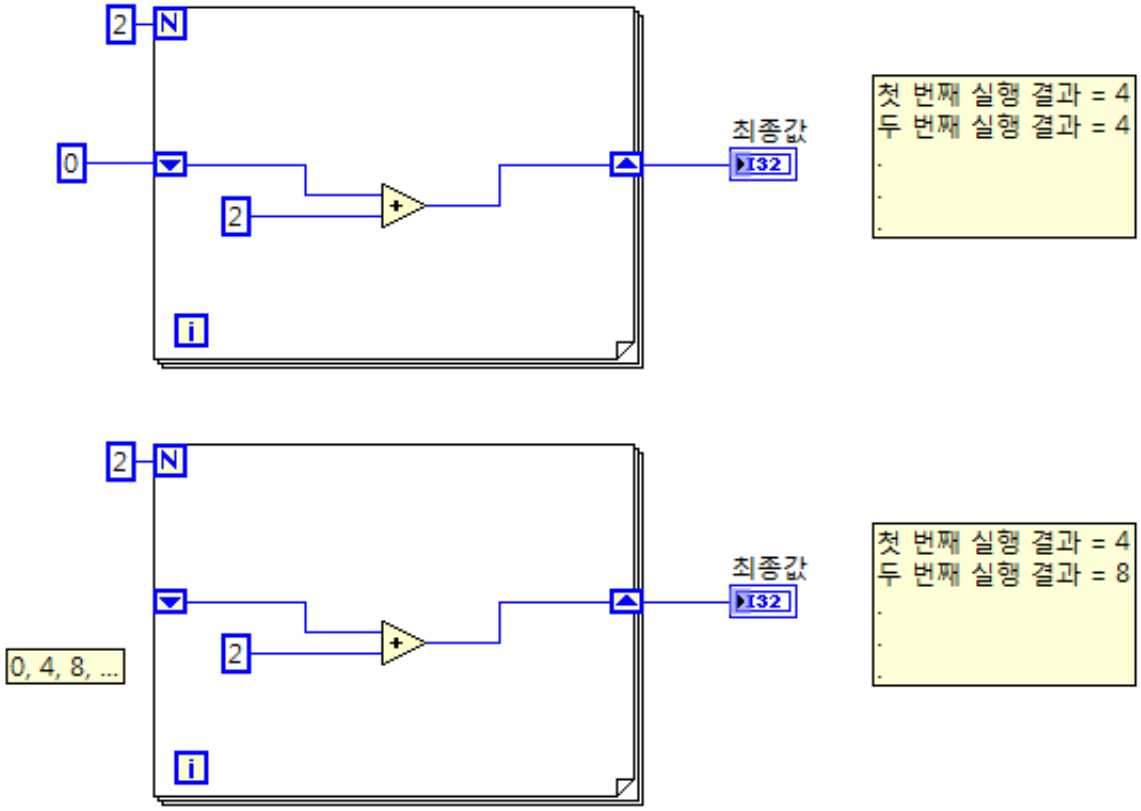
- 다층 시프트 레지스터
  - 과거에 연산한 여러 개의 결과값을 임시 저장





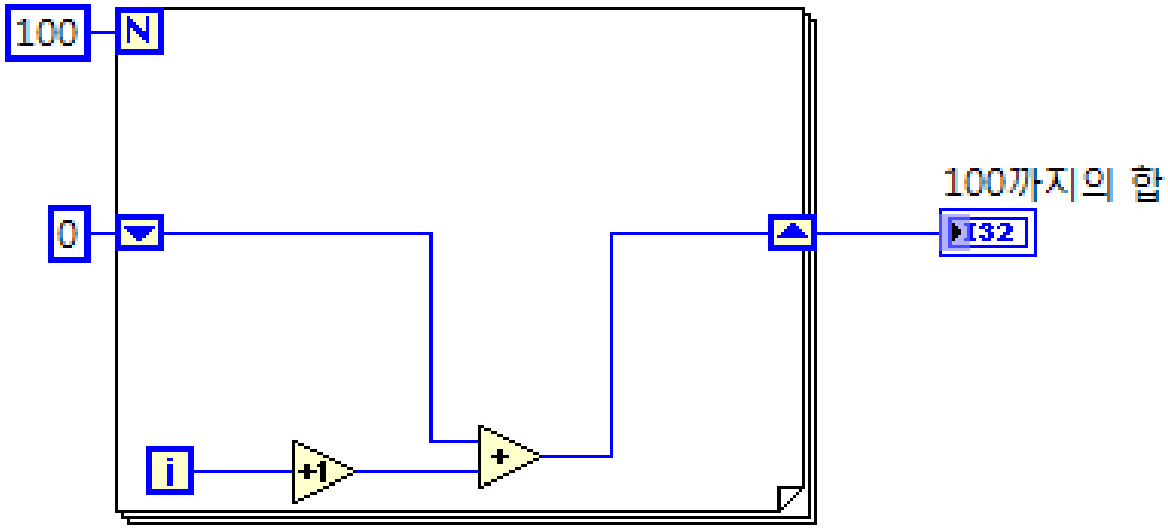
구조 함수 팔레트

- 시프트 레지스터의 초기화



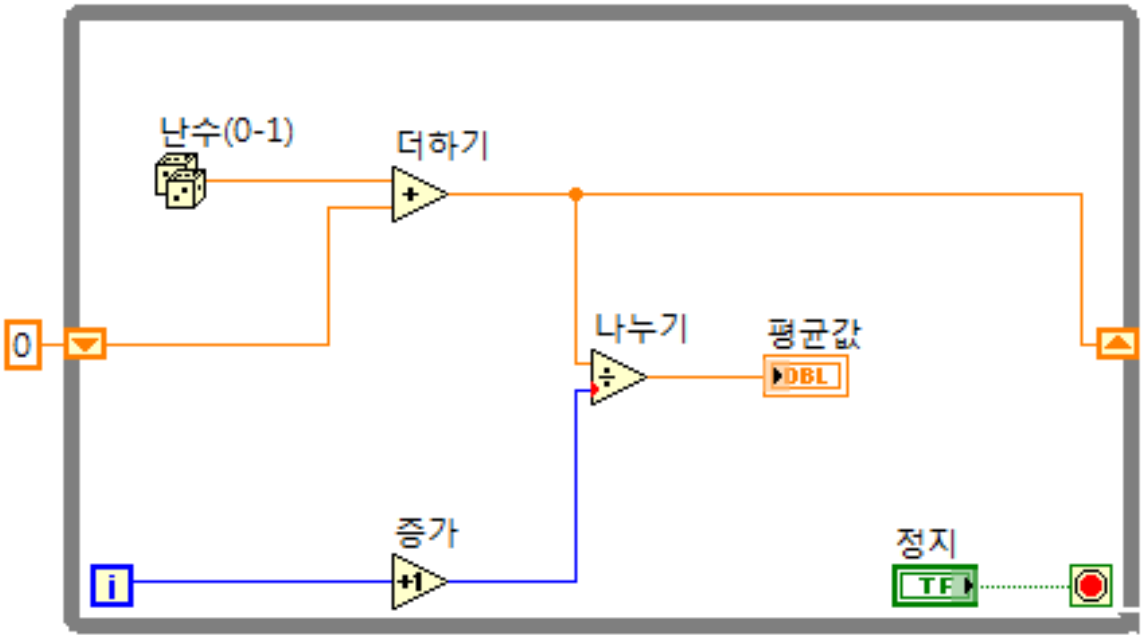
연습문제 9

- 1부터 100까지 더하기



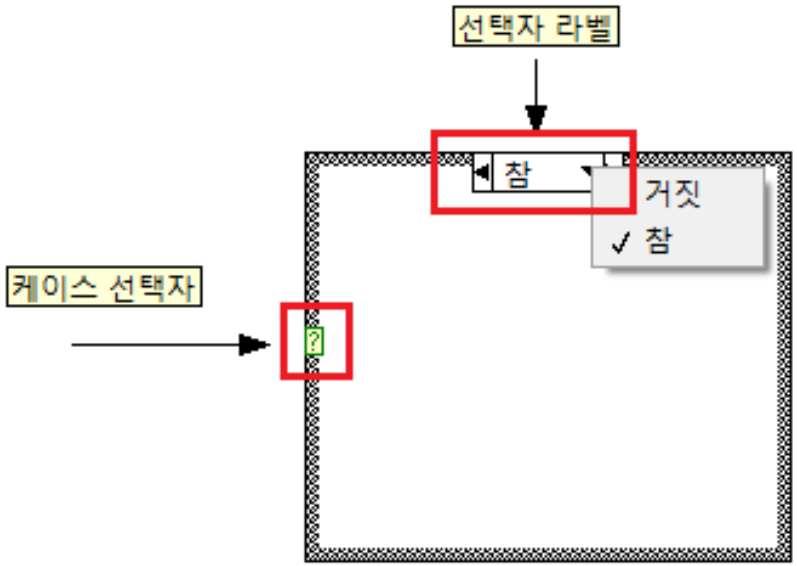
연습문제 10

- 사용자가 정지할 때 까지 발생한 난수의 평균값 구하기



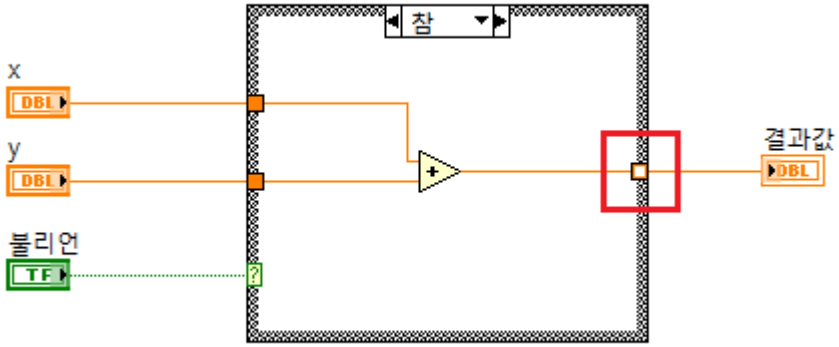
구조 함수 팔레트

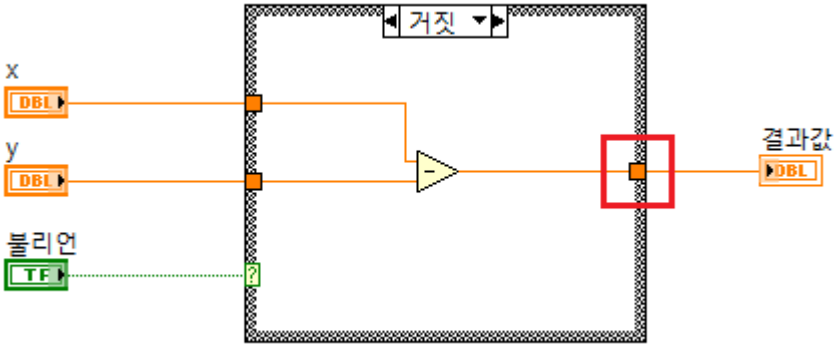
- 케이스 구조
  - IF~ELSE 구문에 해당
  - 케이스 선택자 터미널 / 선택자 라벨
  - 불리언/숫자형/문자열/열거형 선택자 가능



구조 함수 팔레트

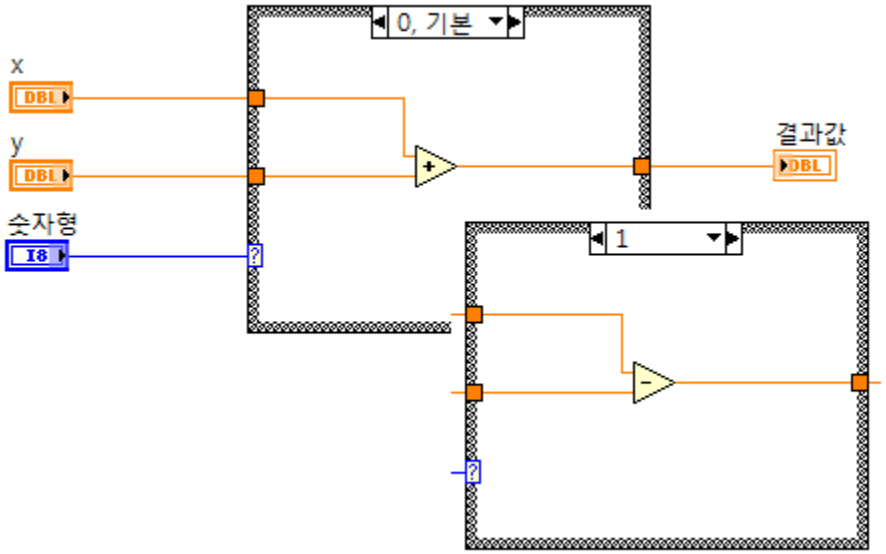
- 케이스 구조
  - 불리언 선택자 입력
  - 참, 거짓 케이스





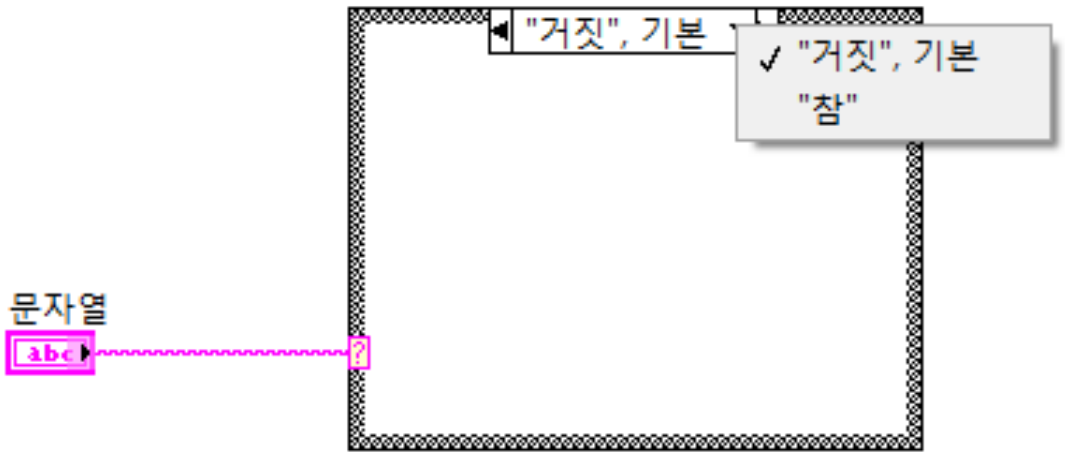
구조 함수 팔레트

- 케이스 구조
  - 숫자형 선택자 입력
  - 0, 1, 2, … 의 정수형 케이스



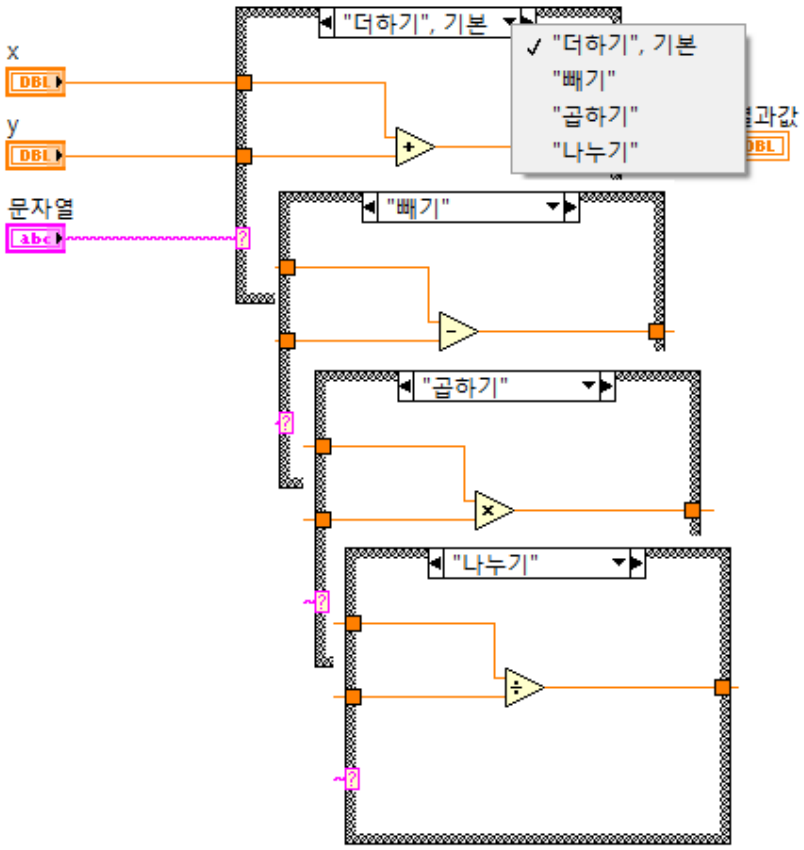
구조 함수 팔레트

- 케이스 구조
  - 문자열 선택자 입력
  - 사용자가 지정한 문자열 케이스
  - 큰 따옴표로(“”) 문자열 표시



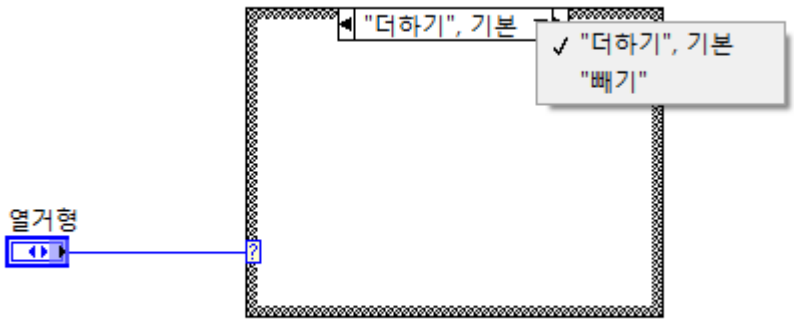
연습문제 11

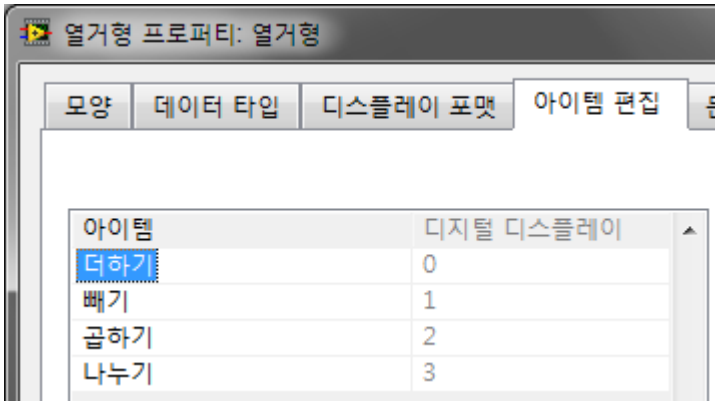
- 문자열 선택자를 이용한 사칙연산 케이스



구조 함수 팔레트

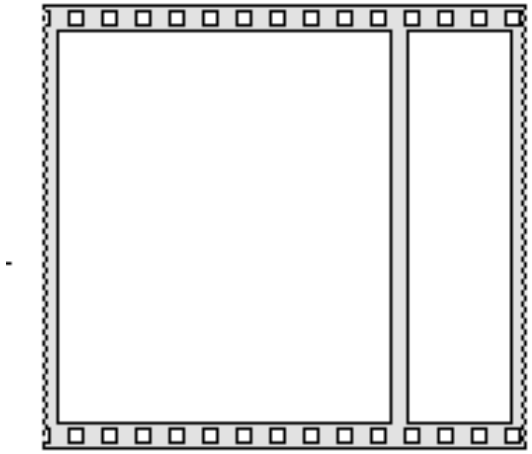
- 케이스 구조
  - 열거형 선택자 입력
  - 열거형 아이템과 동일한 케이스

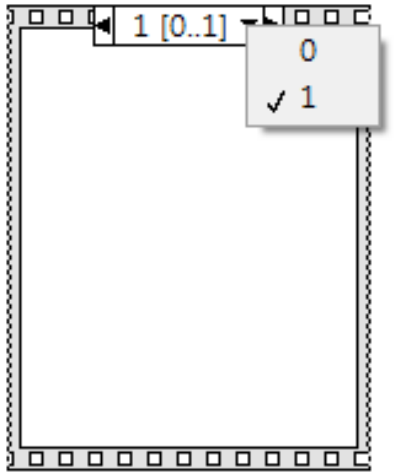




구조 함수 팔레트

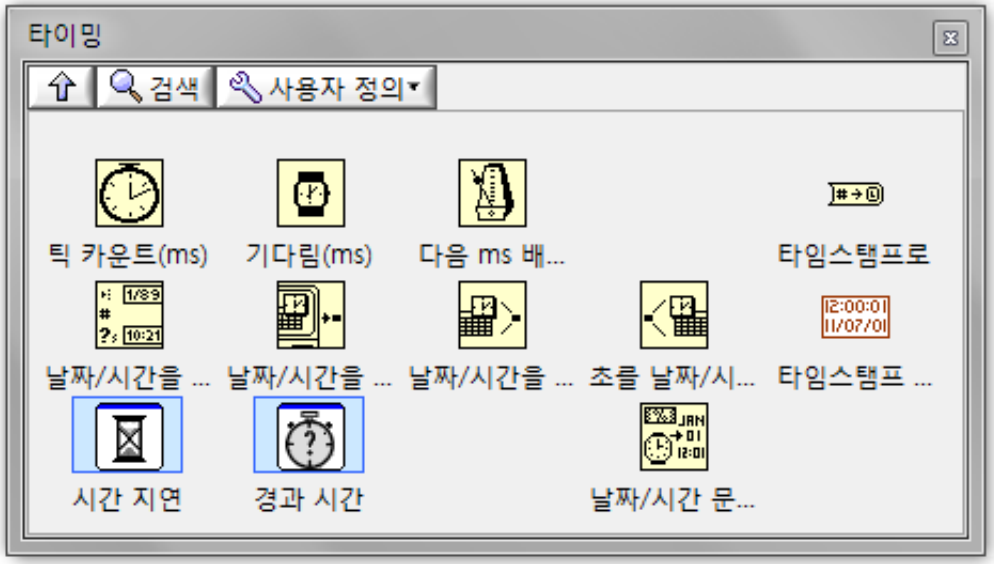
- 시퀀스 구조
  - 실행 순서 조절 기능
  - 플랫 시퀀스 구조 / 다층 시퀀스 구조
  - 코드 부피가 커지고 분석이 복잡해 지는 단점





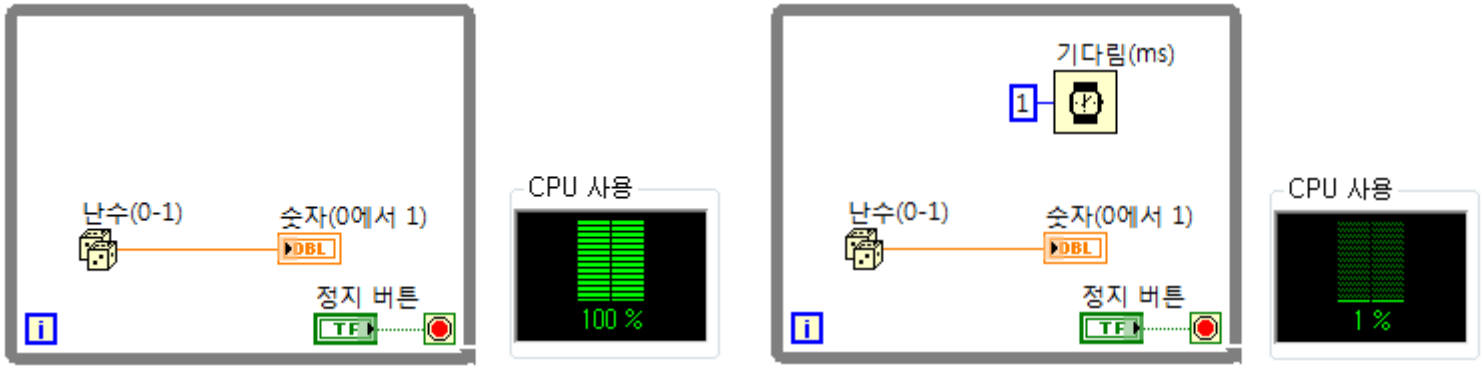
타이밍 함수 팔레트

- 현재의 시간 정보 및 기다림 함수 포함
  - 기다림(ms) 함수 / 다음 ms 배수까지 기다림 함수
  - 틱 카운트(ms) 함수



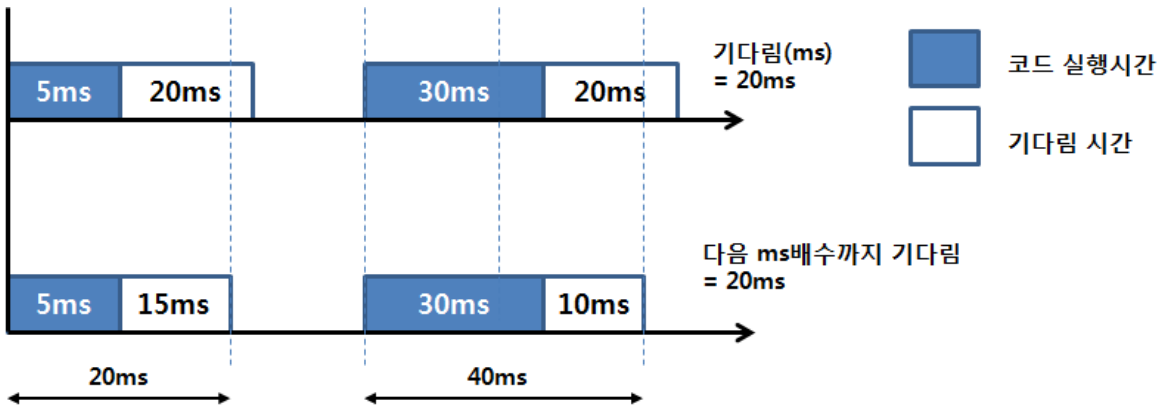
타이밍 함수 팔레트

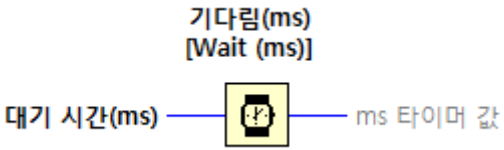
- 기다림(ms) vs. 다음 ms 배수까지 기다림 함수
  - 연산을 정해진 시간만큼 중지
  - CPU 점유율을 낮춤

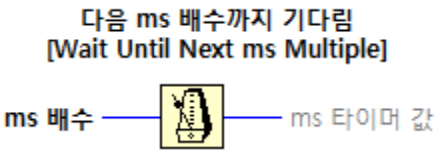


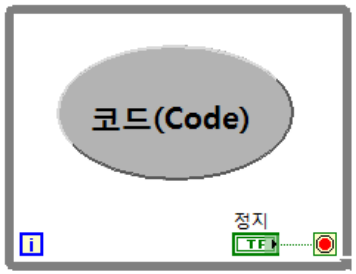
타이밍 함수 팔레트

- 기다림(ms) vs. 다음 ms 배수까지 기다림 함수



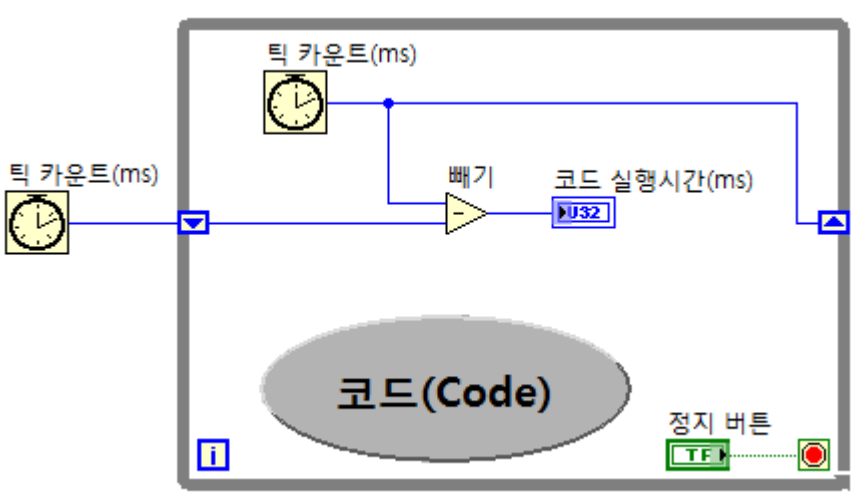


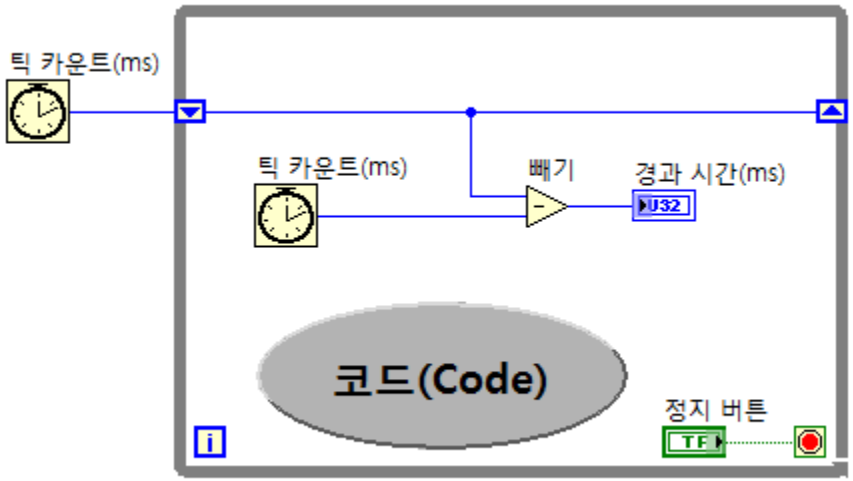




타이밍 함수 팔레트

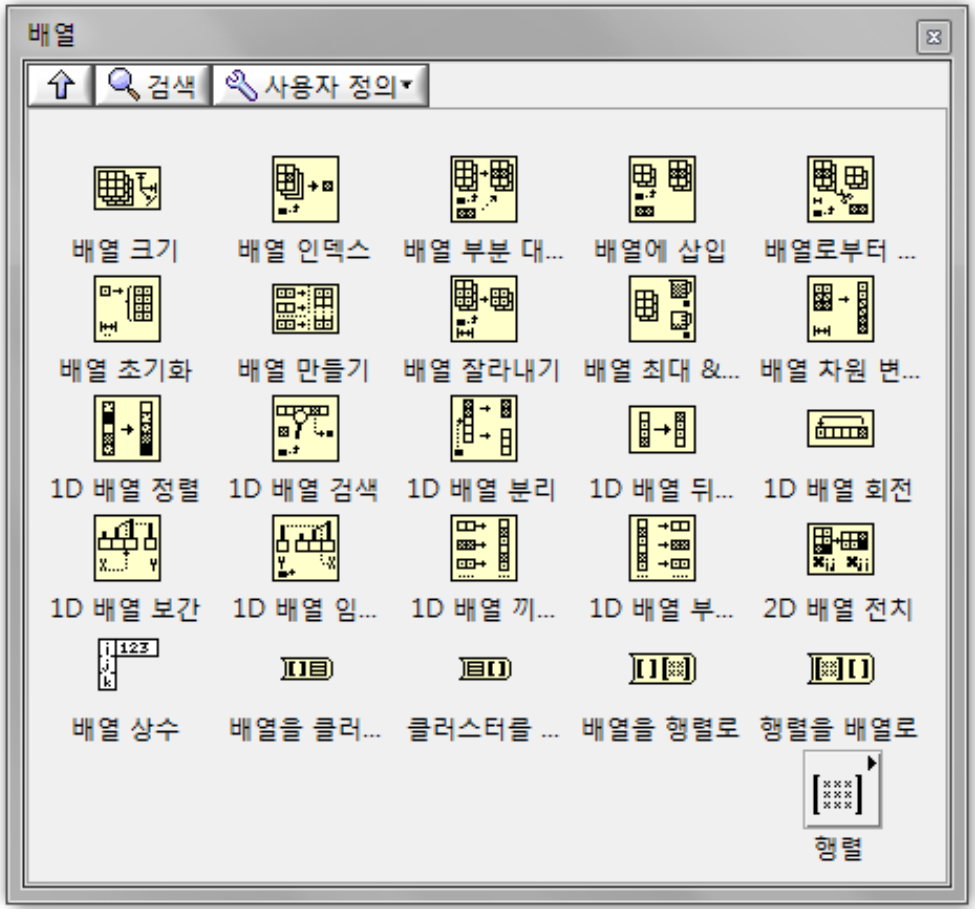
- 틱 카운트(ms) 함수
  - 윈도우가 시작된 시점에서 경과 시간 반환
  - 코드의 실행 시간 / 실행 경과 시간 측정





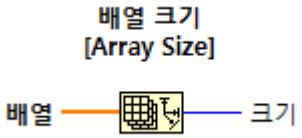
배열 함수 팔레트

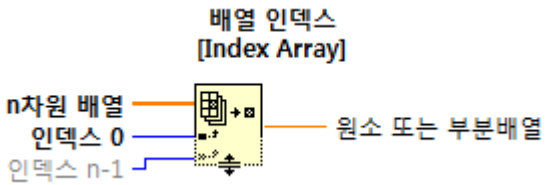
- 배열의 연산에 관련된 함수 포함

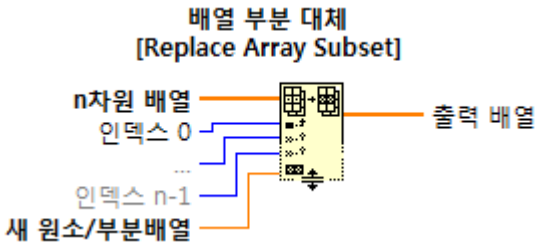


배열 함수 팔레트

- 배열 크기 함수
- 배열 인덱스 함수
- 배열 부분 대체 함수

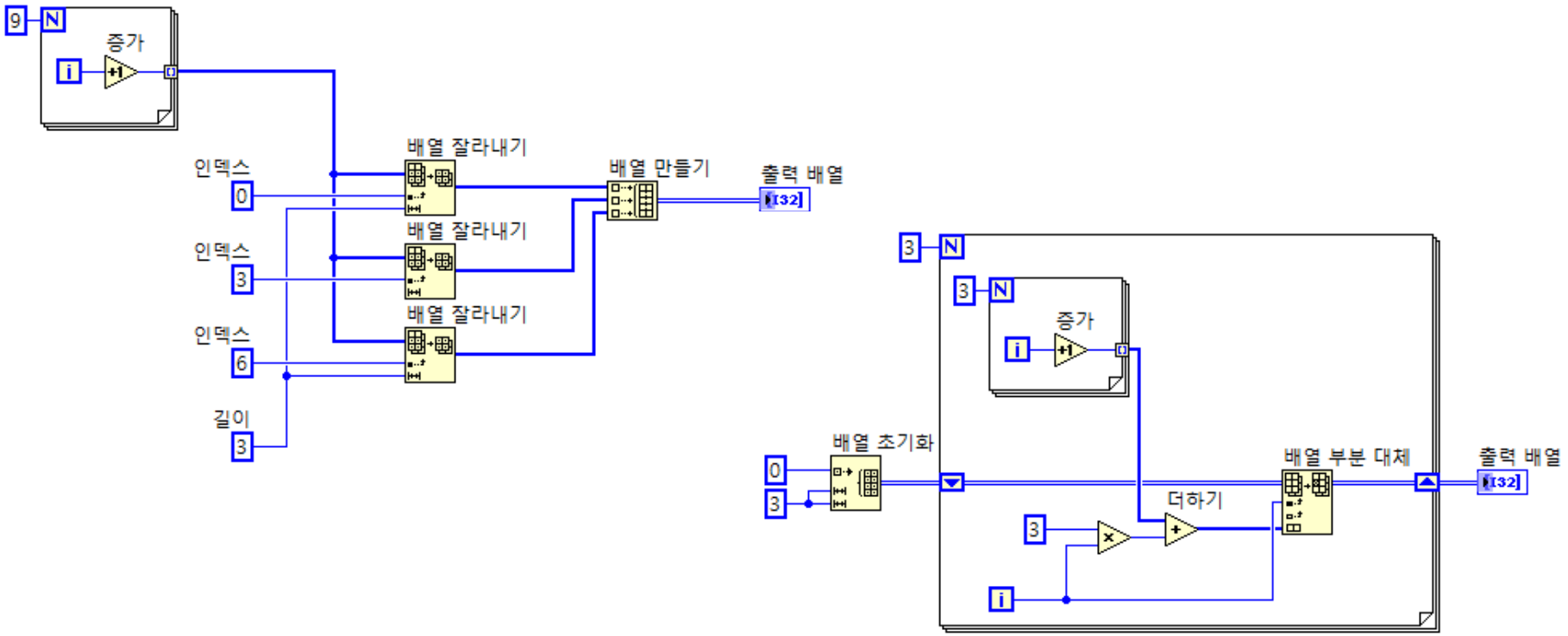






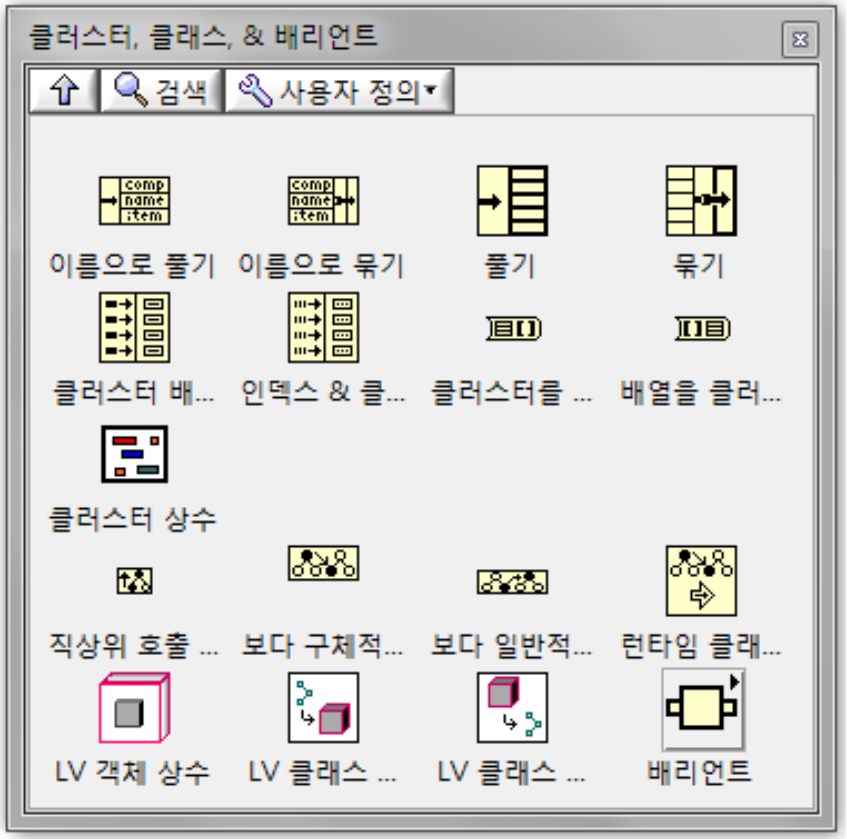
연습 문제 12

- 2차원 배열 생성하기



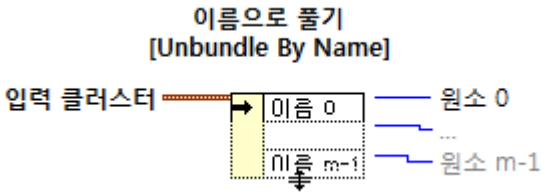
클러스터 함수 팔레트

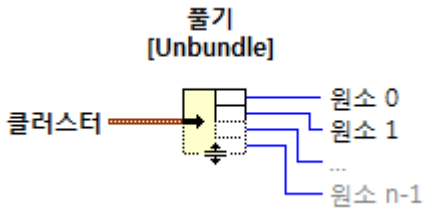
- 클러스터의 원소를 읽거나 변경 함수 모음

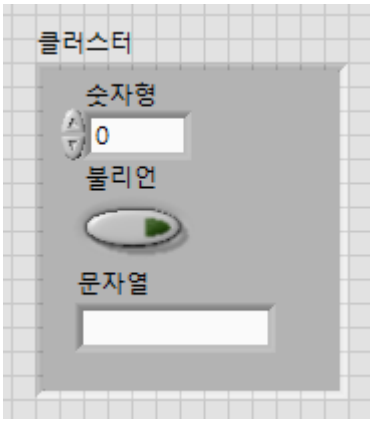


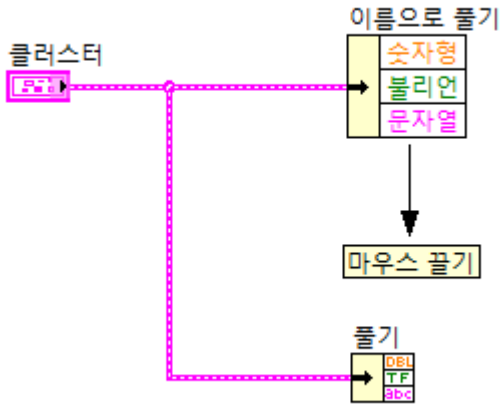
클러스터 함수 팔레트

- 이름으로 풀기 vs. 풀기 함수
  - 클러스터 내의 원소 값 읽기



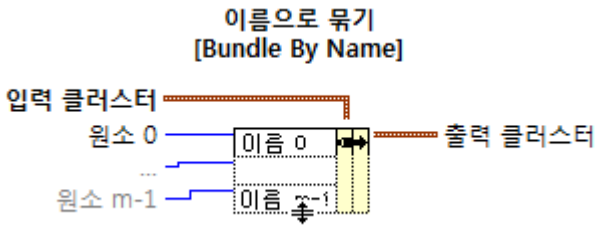


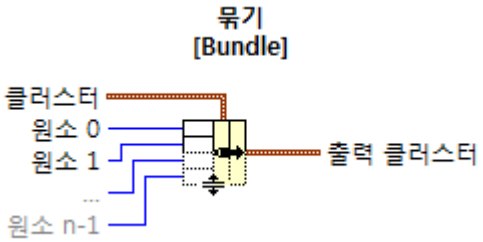


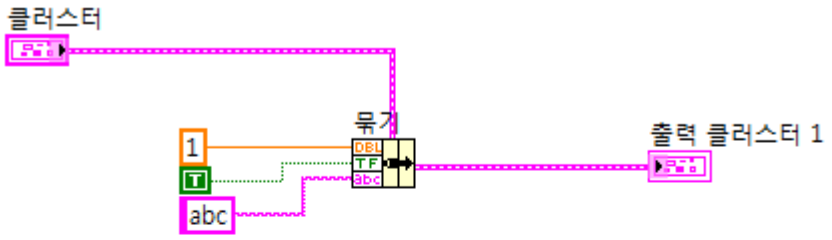


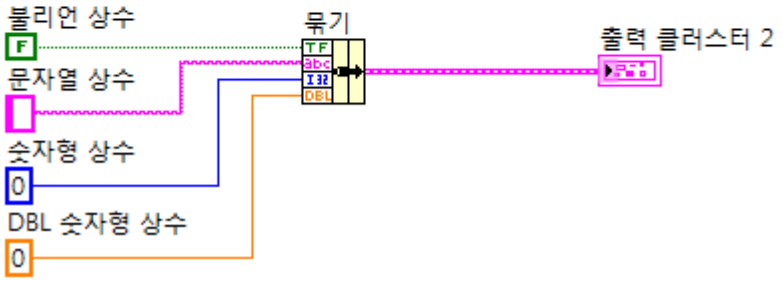
클러스터 함수 팔레트

- 이름으로 묶기 vs. 묶기
  - 클러스터 내의 원소 값 변경
  - 프로그램적으로 클러스터 생성(묶기)



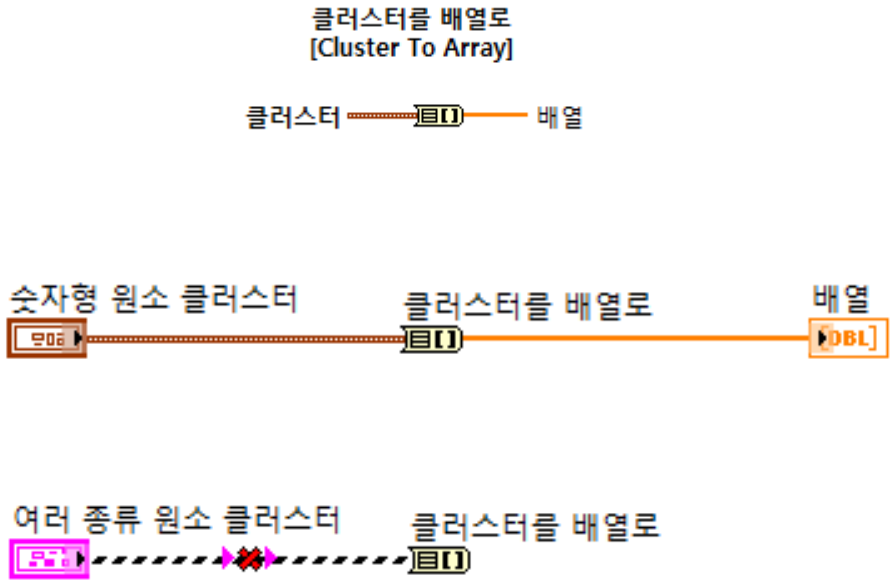






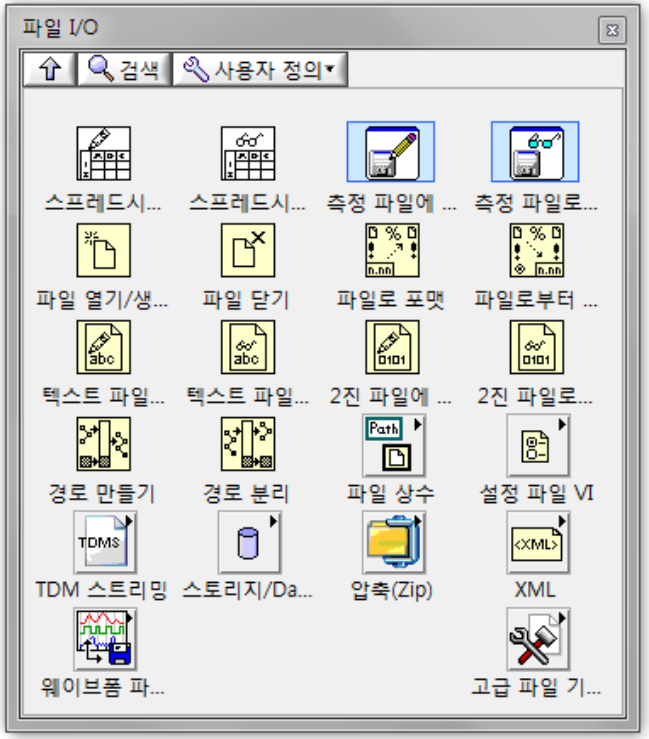
클러스터 함수 팔레트

- 클러스터를 배열로 함수
  - 같은 종류의 원소로 이루어진 클러스터만 변환 가능



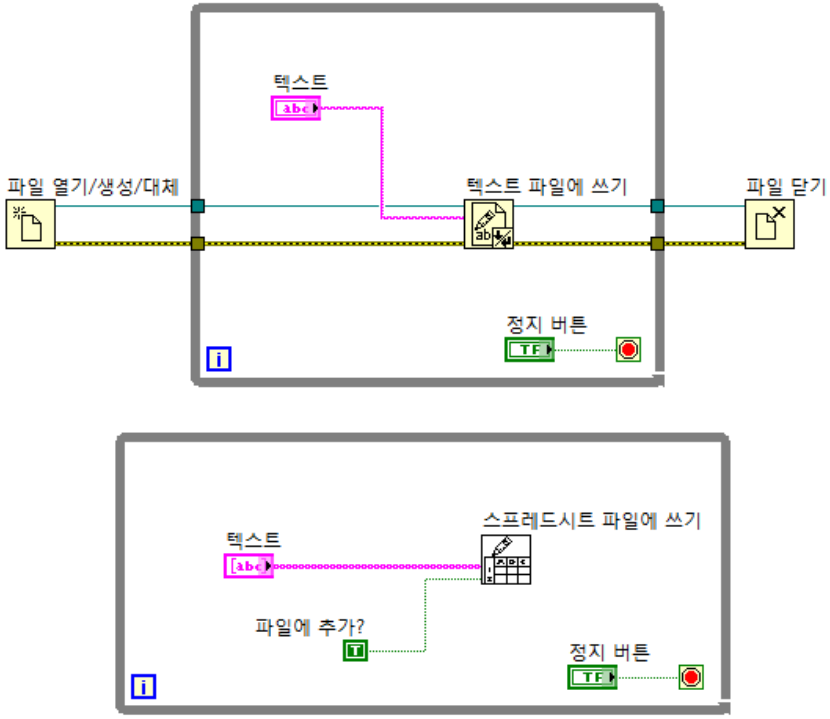
파일 I/O 함수 팔레트

- 아스키(ASCII) 파일 쓰기/읽기
- 바이너리 파일 쓰기/읽기



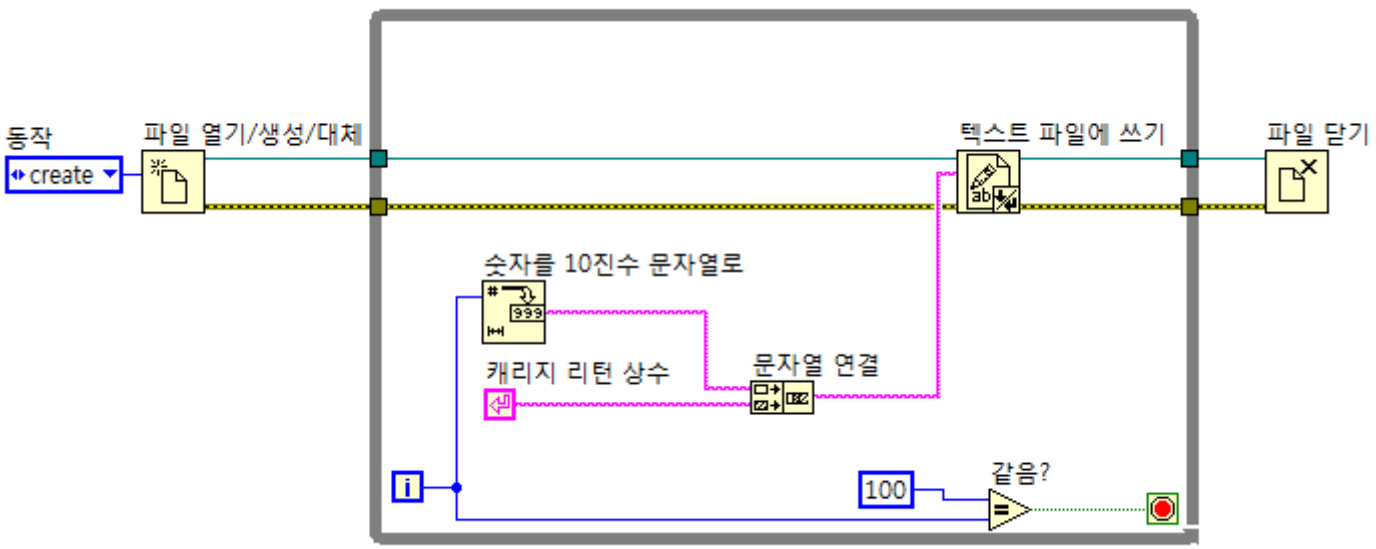
파일 I/O 함수 팔레트

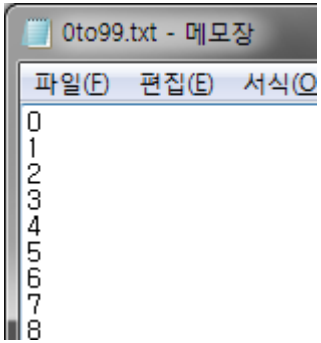
- 아스키(ASCII) 형태 쓰기/읽기 함수
  - 상위 레벨 파일 I/O vs 하위 레벨 파일 I/O
  - 열기 / 읽기(쓰기) / 닫기 기능이 하나의 함수에 모두 포함



연습 문제 13

- 0부터 99까지의 숫자를 텍스트 파일에 쓰기

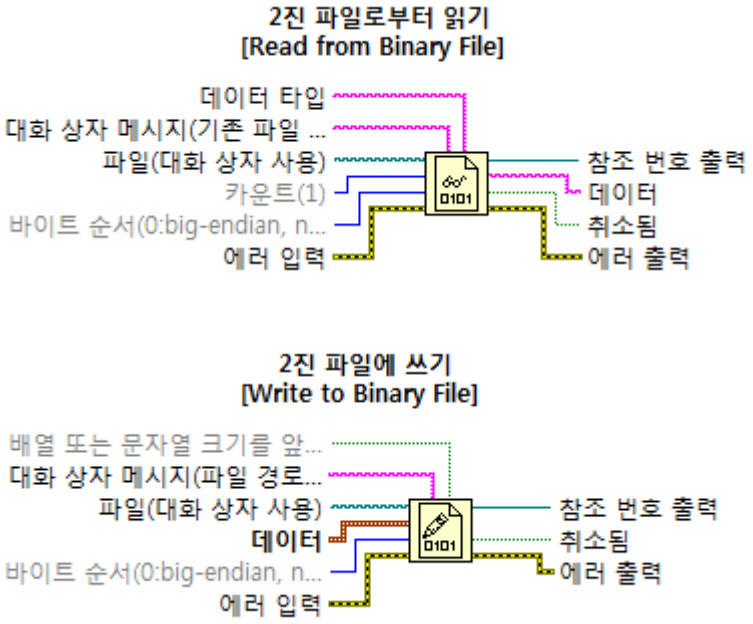




파일 I/O 함수 팔레트

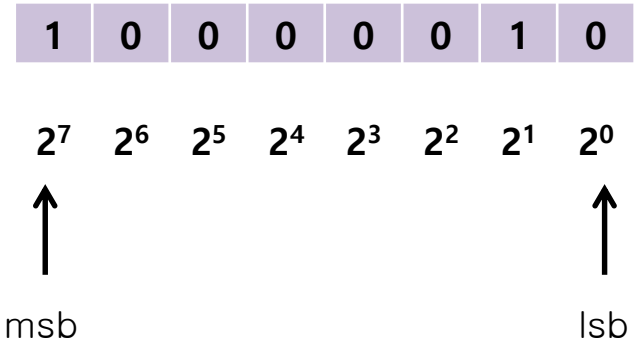
- 바이너리 형태 쓰기/읽기 함수
  - 아스키 형태와 동일한 열기 / 닫기 함수 사용
  - 성능이 중요한 경우 주로 사용

	텍스트 파일	바이너리 파일
속도	늦음	빠름
용량	큼	작음
보안	없음	높음
사용	쉬움	어려움

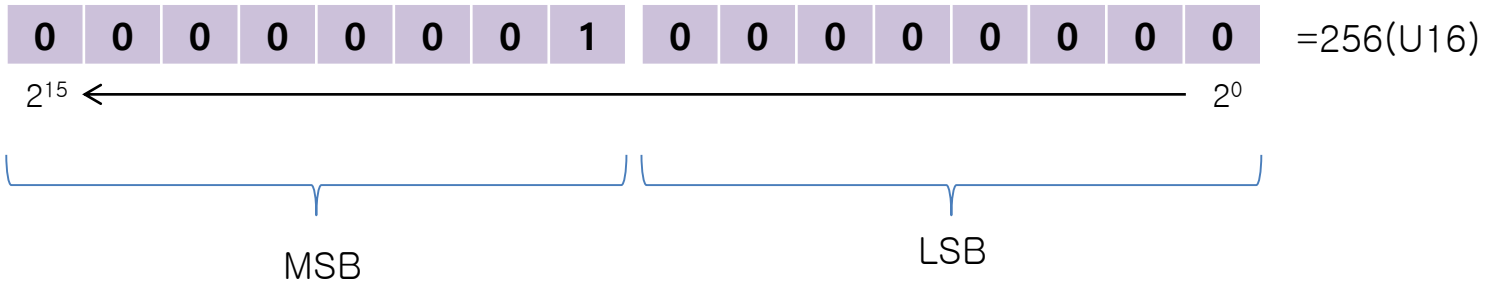


파일 I/O 함수 팔레트

- 최상위 비트(msb) vs. 최하위 비트(lsb)

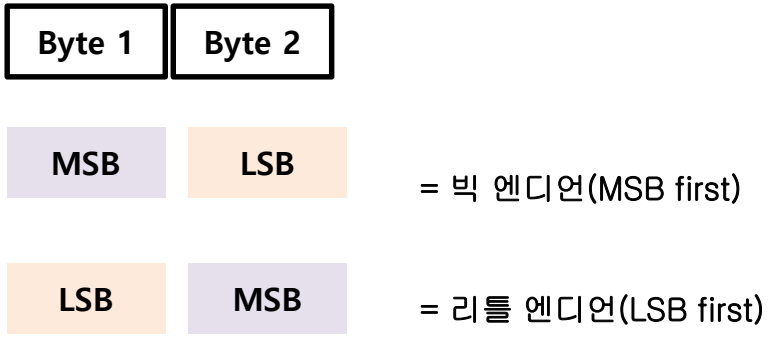


- 최상위 바이트(MSB) vs. 최하위 바이트(LSB)



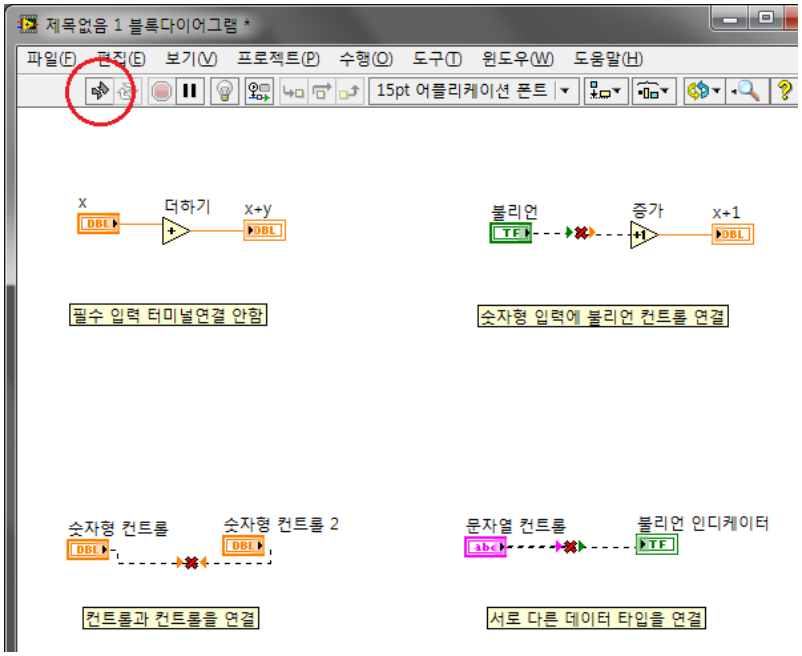
파일 I/O 함수 팔레트

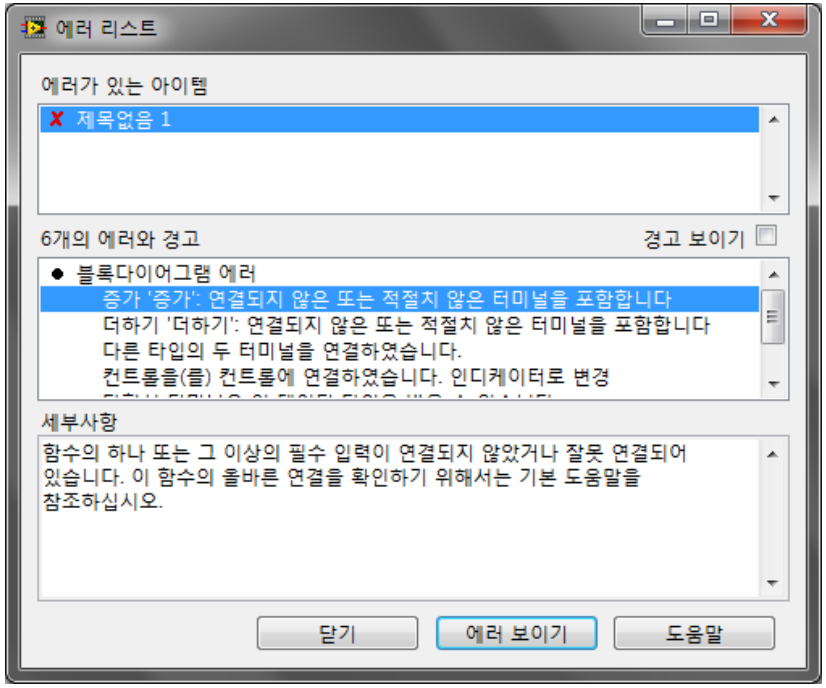
- 빅 엔디언 vs. 리틀 엔디언
  - 운영 체제에 따라서 바이트 배열 방식이 다를 수 있음



랩뷰의 디버깅 도구

- 컴파일
  - 문법적 오류의 수정
  - 깨진 와이어 / 잘못된 함수의 사용 등

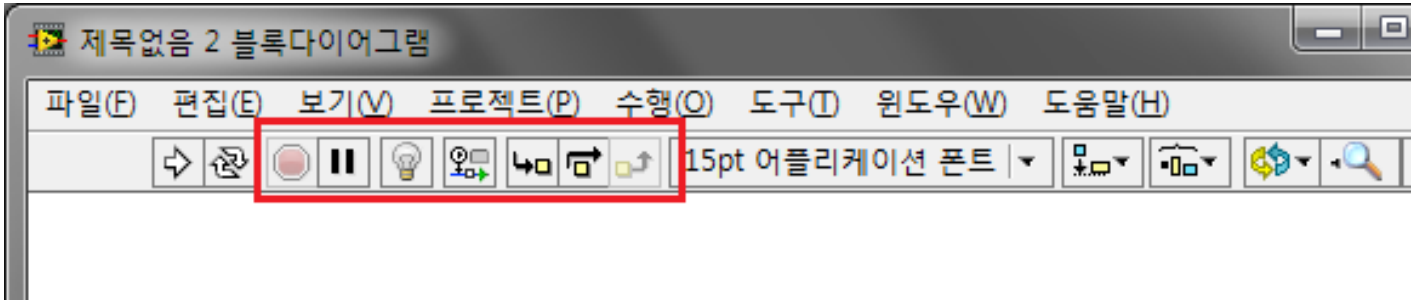




랩뷰의 디버깅 도구

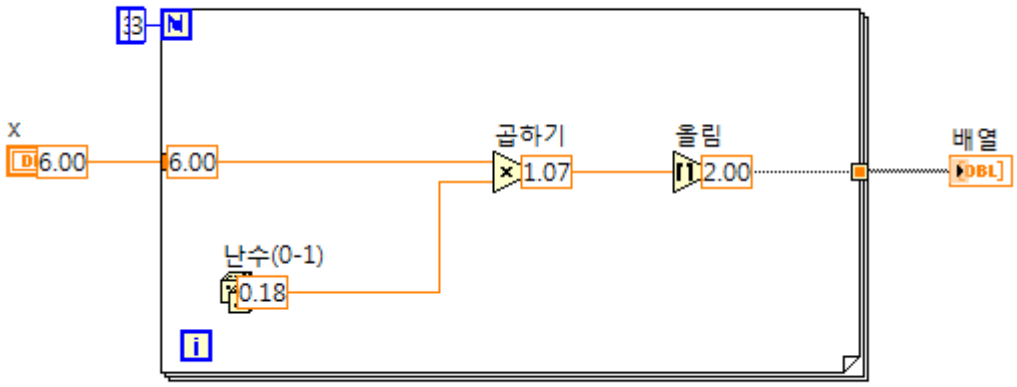
- 실행 강제 종료
  - 실행 중인 VI를 강제로 종료

- 일시 정지
  - 실행 중인 VI의 연산을 일시적으로 정지
  - 다시 버튼을 누르면 실행 재개



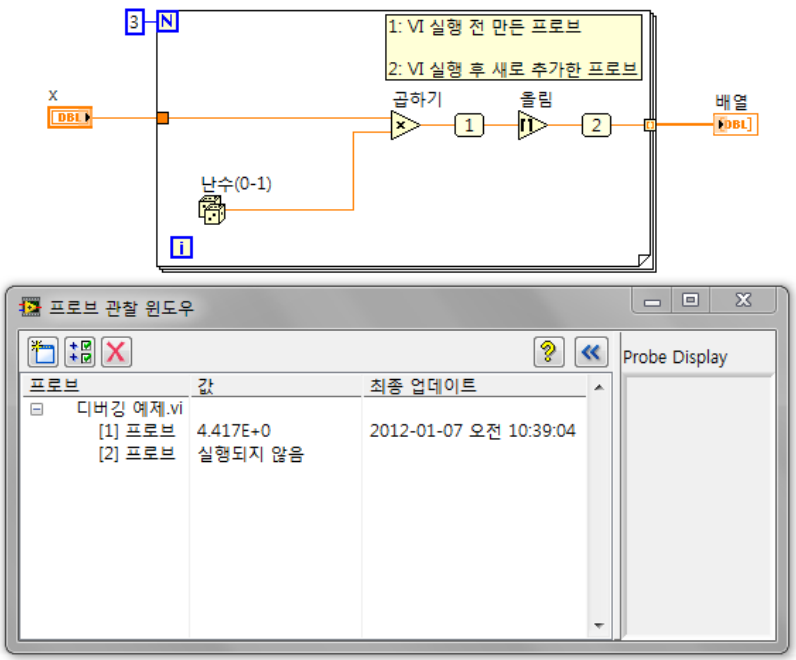
랩뷰의 디버깅 도구

- 실행 하이라이트
  - 와이어를 통한 값의 전달 표시
  - VI의 실행 속도 느려짐
  - 실시간 연산이 필요한 경우 사용 불가



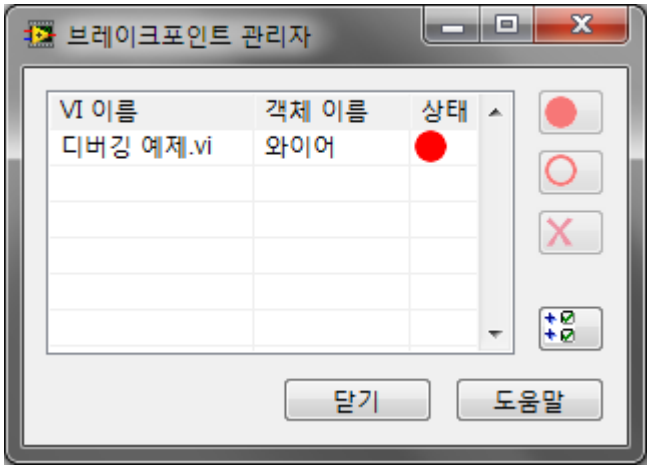
랩뷰의 디버깅 도구

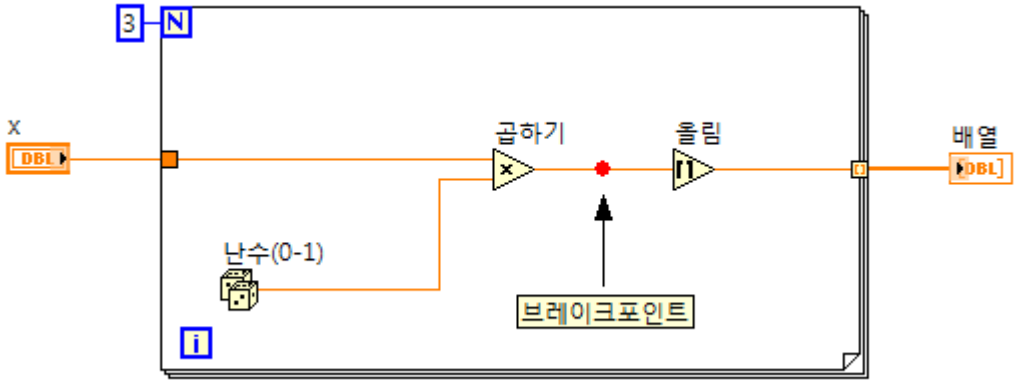
- 프로브
  - 와이어를 통해 전달되는 값을 실시간 감시
  - 와이어 값 유지 옵션



랩뷰의 디버깅 도구

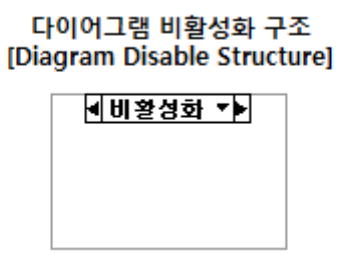
- 단계별 실행
  - 버튼을 눌러 함수 하나하나의 연산을 수동으로 진행
- 브레이크 포인트
  - 일정 지점에서 실행을 일시 정지하고 사용자의 입력을 기다림

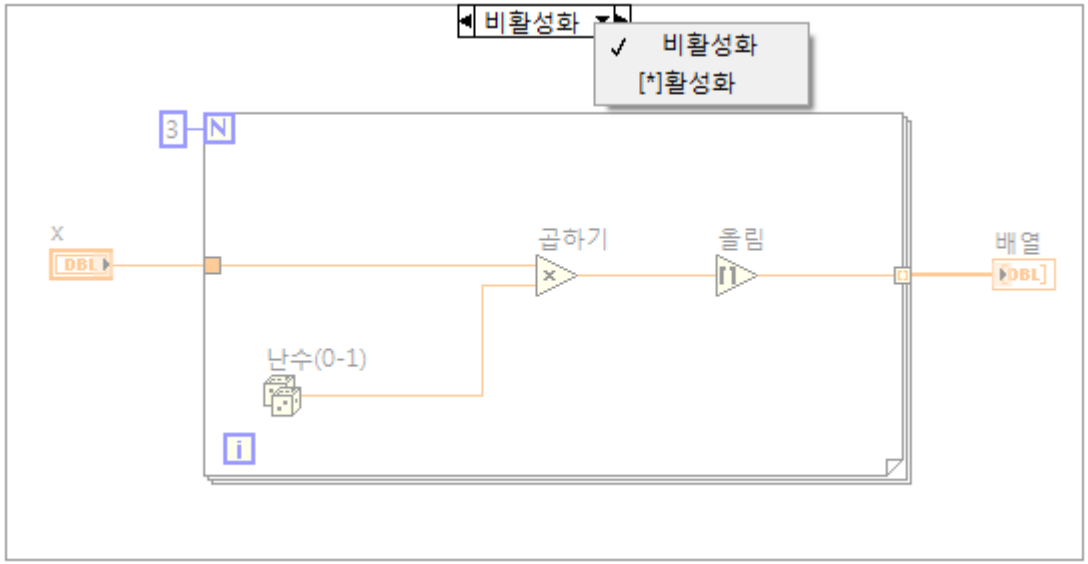




랩뷰의 디버깅 도구

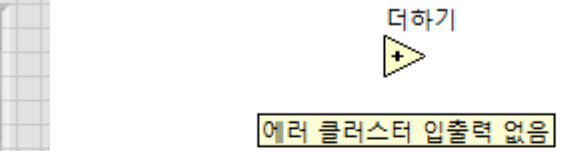
- 다이어그램 비활성화 구조
  - 코드의 일정 부분을 실행하지 않도록 설정
  - 활성화된 부분이 실행

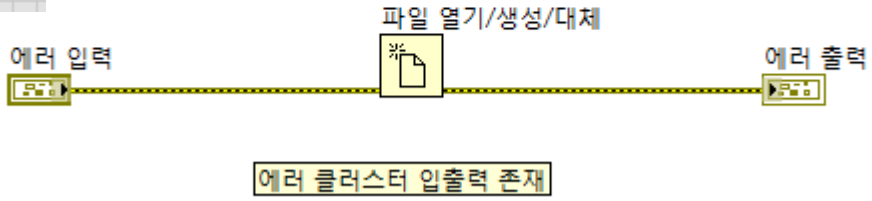




에러 핸들링

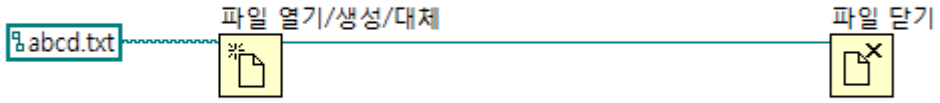
- 에러 핸들링
  - 프로그램 실행 중 에러 발생시 처리 방식
  - 에러 클러스터를 통해 에러 정보 전달

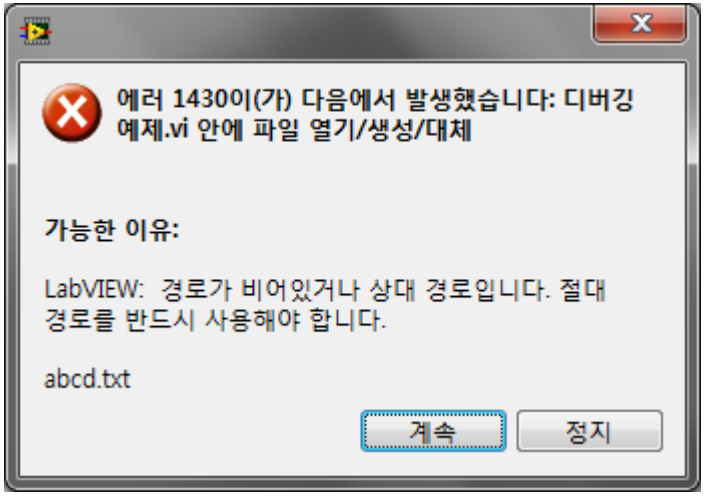




에러 핸들링

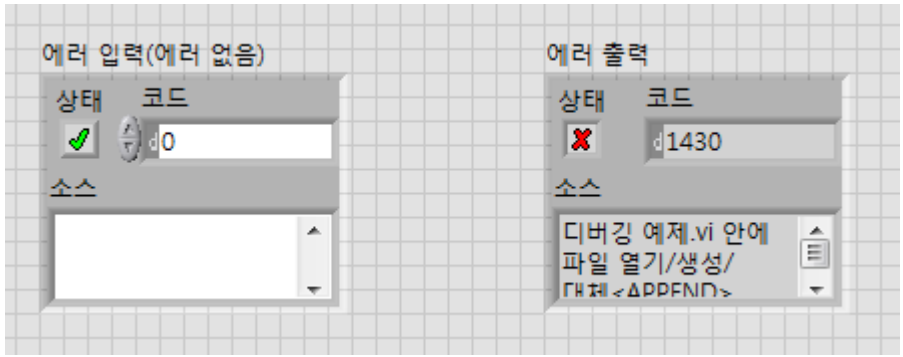
- 자동 에러 핸들링
  - 에러 발생시 랩뷰가 자동적으로 에러 팝업
  - 에러 클러스터 연결하지 않으면 기본 설정으로 자동 에러 핸들링 처리





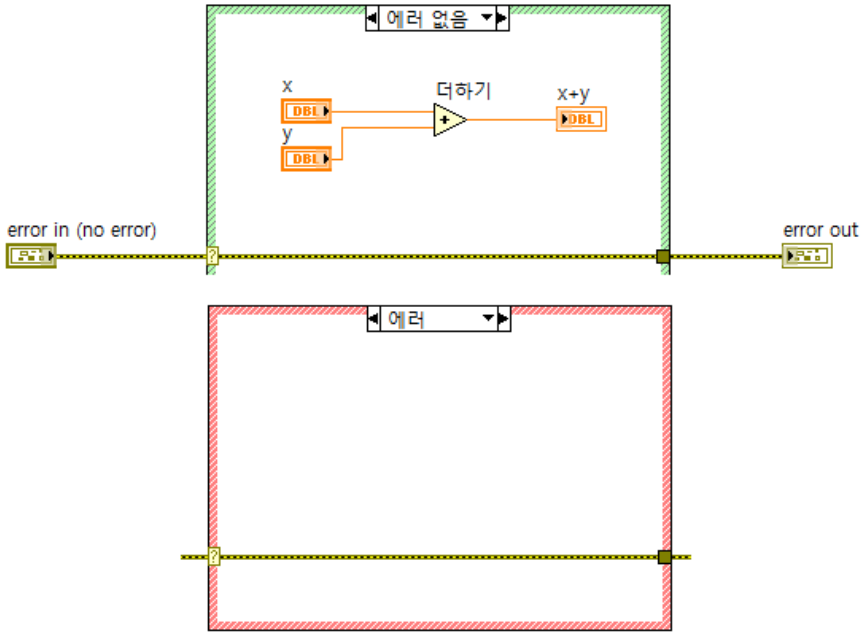
에러 핸들링

- 수동 에러 핸들링
  - 에러 발생 시 사용자가 직접 에러의 처리 방법을 결정할 수 있음
  - 에러 클러스터 연결 시 수동 에러 핸들링으로 전환



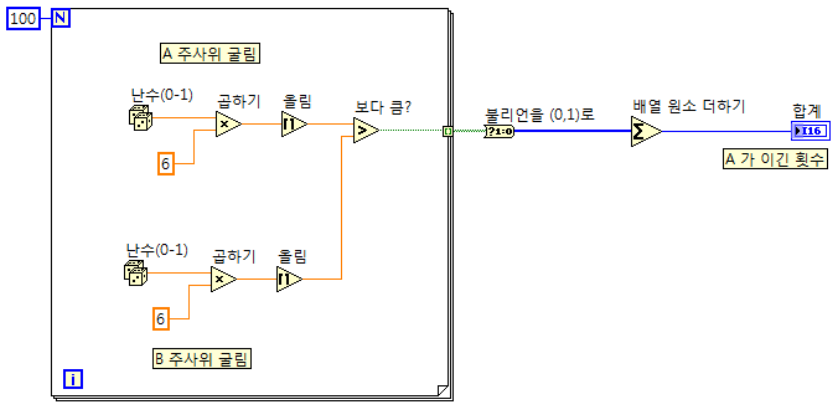
에러 핸들링

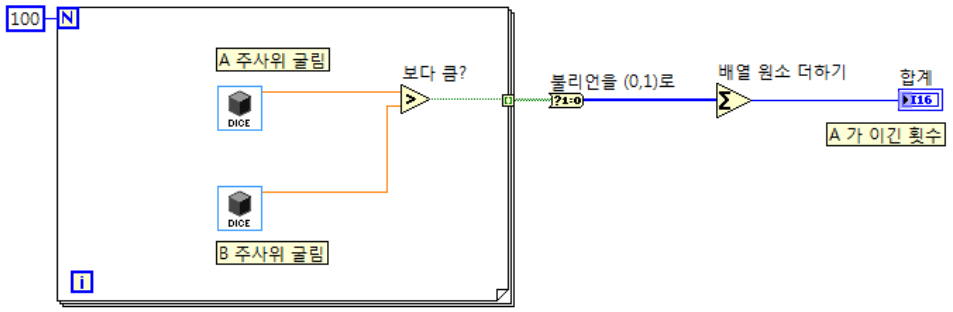
- 수동 에러 핸들링
  - 에러 핸들링 입출력이 있는 함수의 구조
  - 에러 클러스터를 케이스 선택자로 연결



모듈화(Modularity)

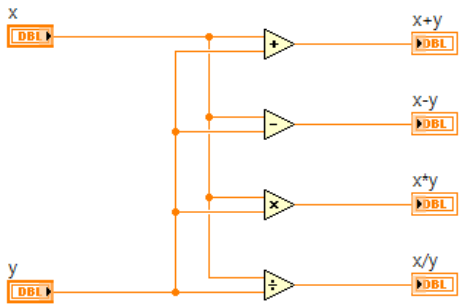
- 모듈(Module)
  - 여러 기능을 구성하기 위해 하나로 묶어 놓은 집합체
- 모듈화의 장점
  - 직관적이고 분석이 쉬운 코드 구성

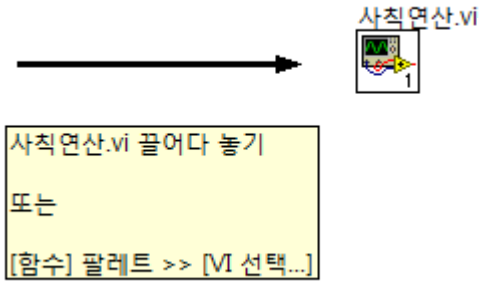


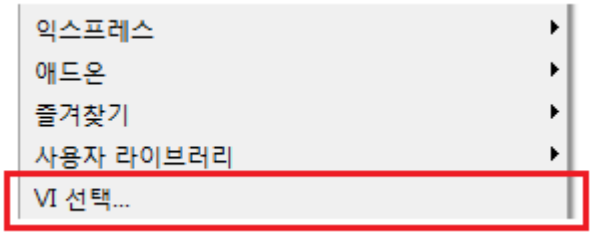


SubVI 만들기

- subVI 란?
  - VI 안에서 사용되는 VI를 지칭
- subVI 불러오기
  - VI 선택 옵션 또는 끌어 놓기

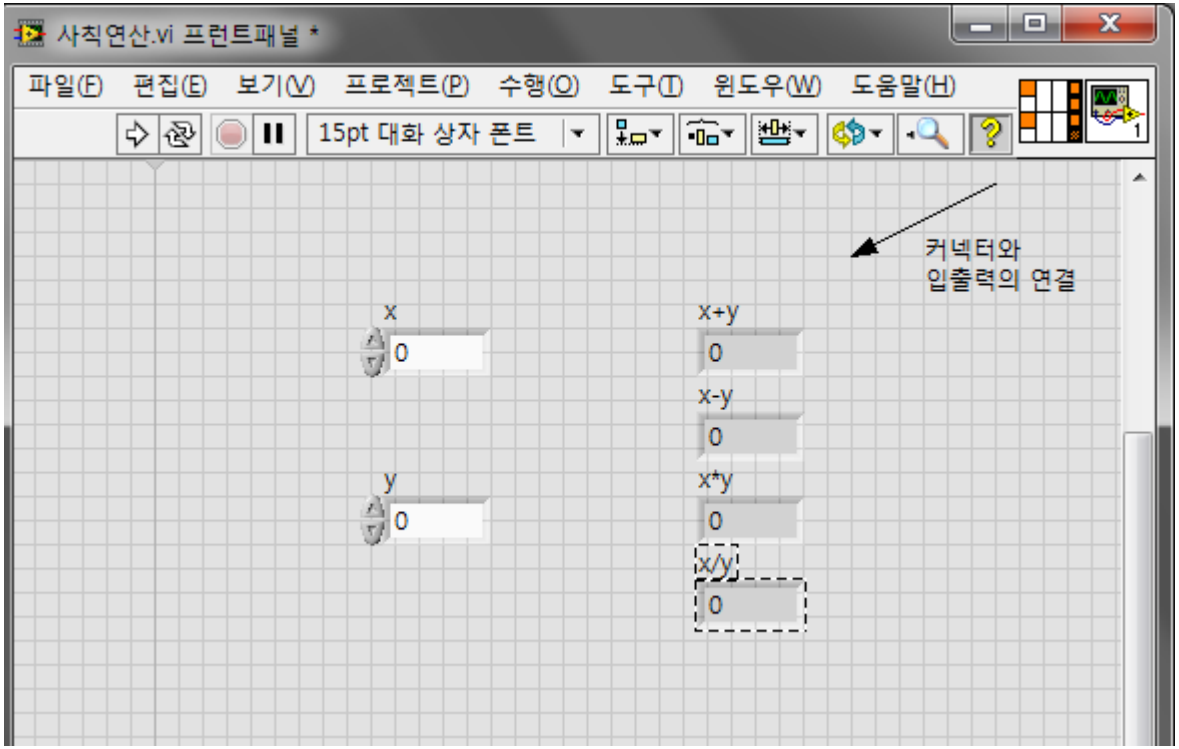


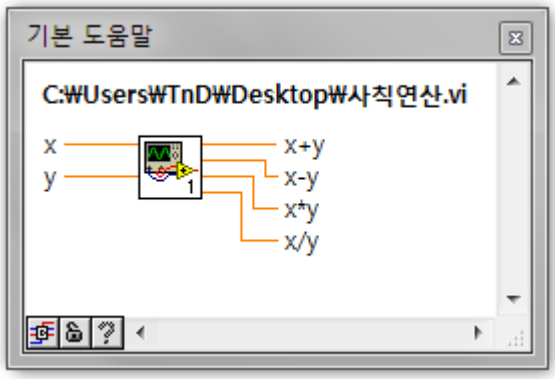




SubVI 만들기

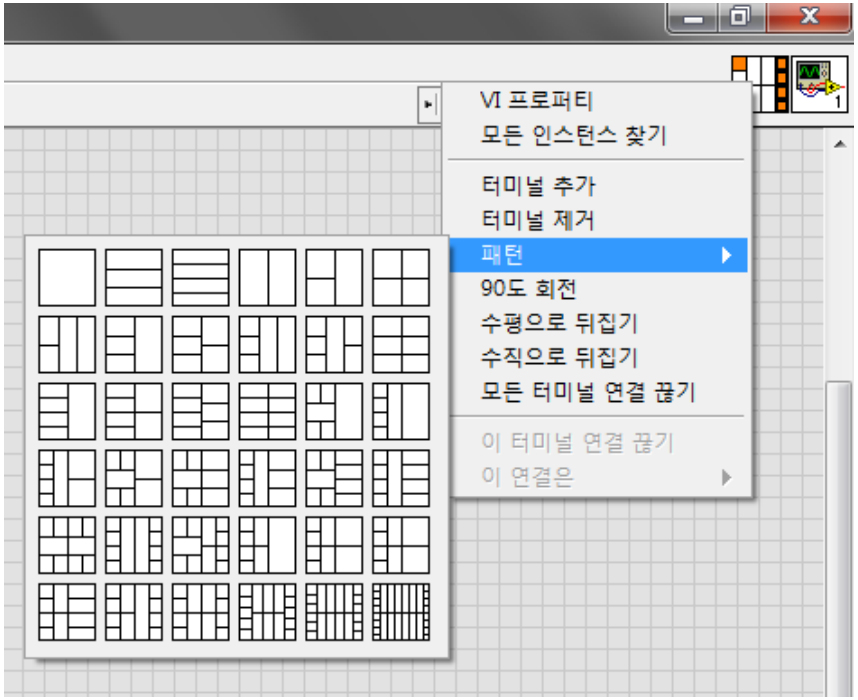
- 커넥터 팬
  - subVI의 입출력 터미널 지정

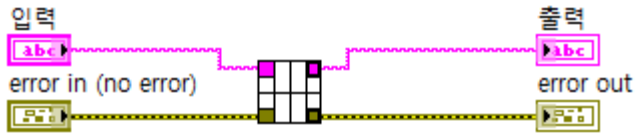




SubVI 만들기

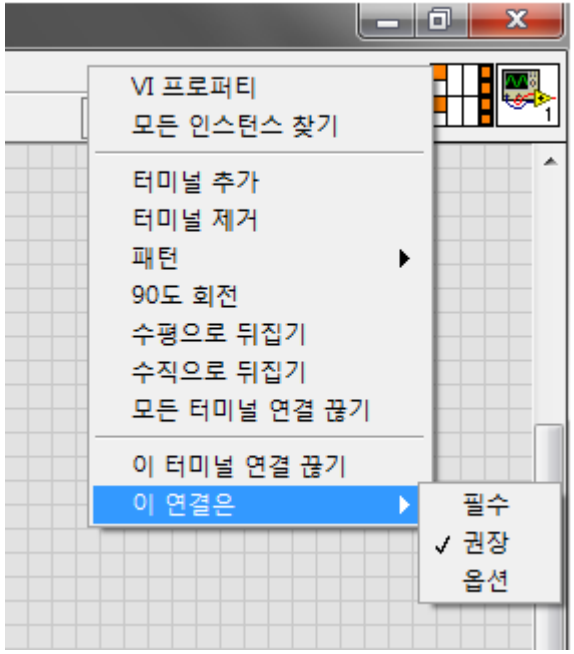
- 커넥터 패턴 선택
  - 입출력 터미널의 개수가 많은 경우
  - 다른 함수와의 연결 위해 기본 패턴 사용 권장

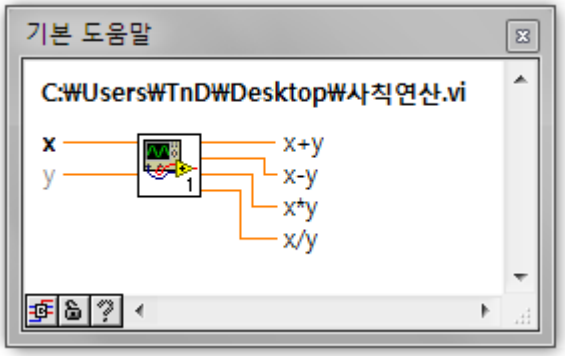




SubVI 만들기

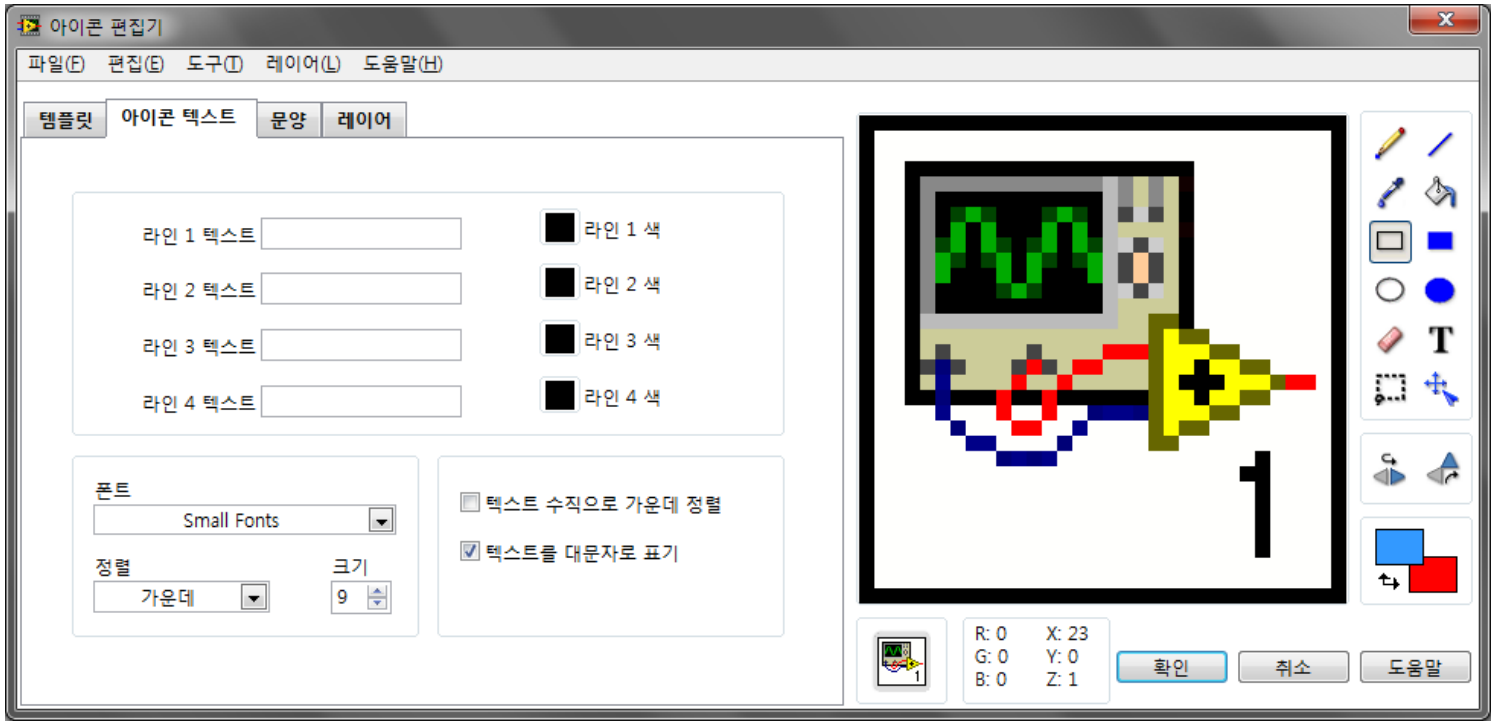
- 입력 터미널의 연결 방식 변경
  – 필수 / 권장 / 옵션 입력 지정





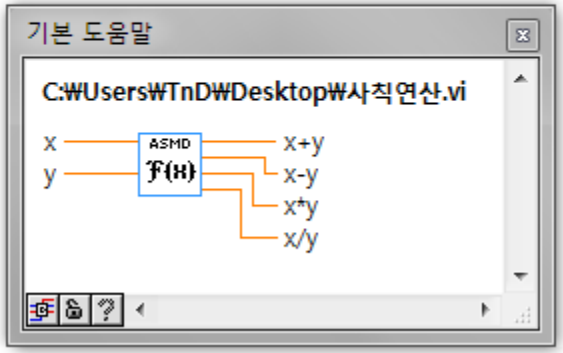
SubVI 만들기

- 아이콘 편집
  - 기본 아이콘 사용 시 혼동 가능성
  - 직관적인 기능 표시 가능



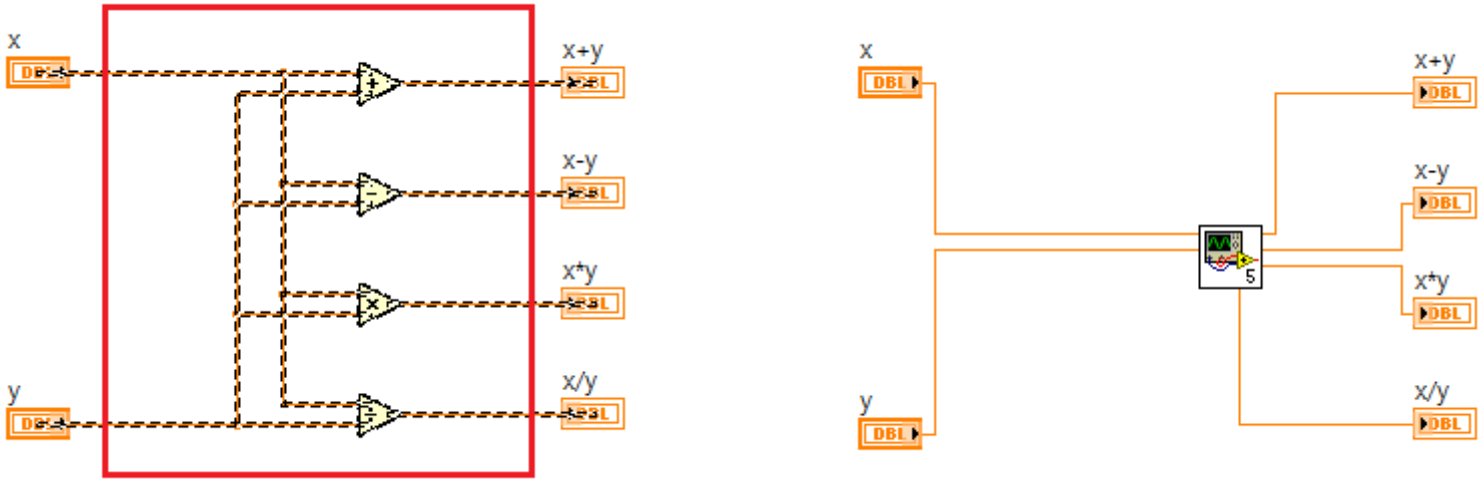
연습문제 14

- 사칙 연산 SubVI 만들기



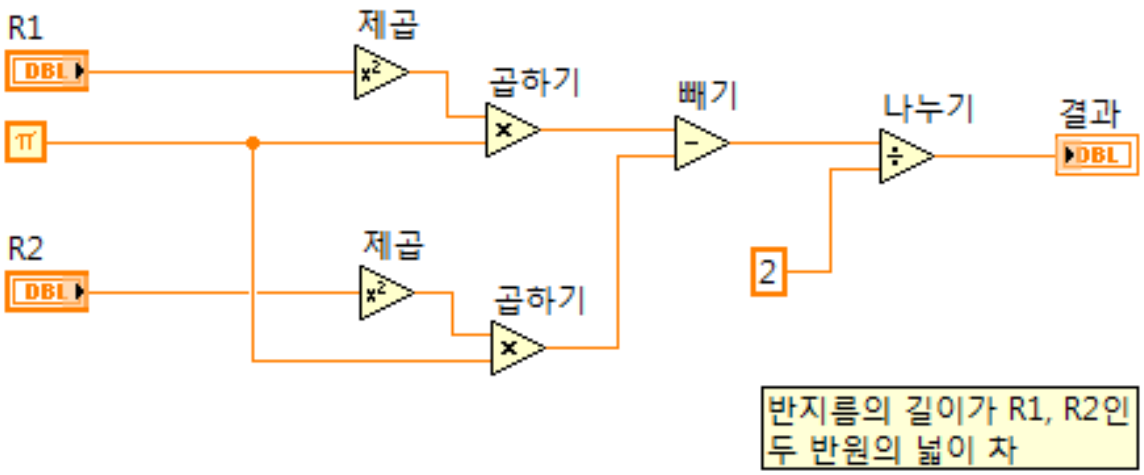
SubVI 만들기

- subVI 생성 옵션
  - 구역 선택 후 [편집] > [subVI 생성] 선택



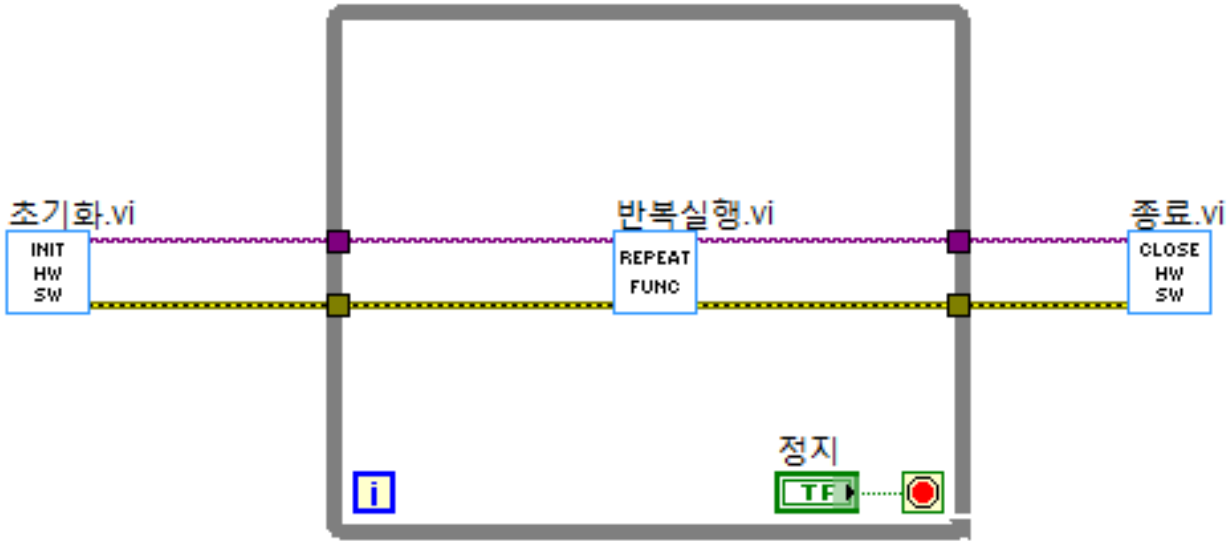
단순 디자인 패턴

- 한 번의 단순 실행
- 수학적 계산이나 시뮬레이션에 적합



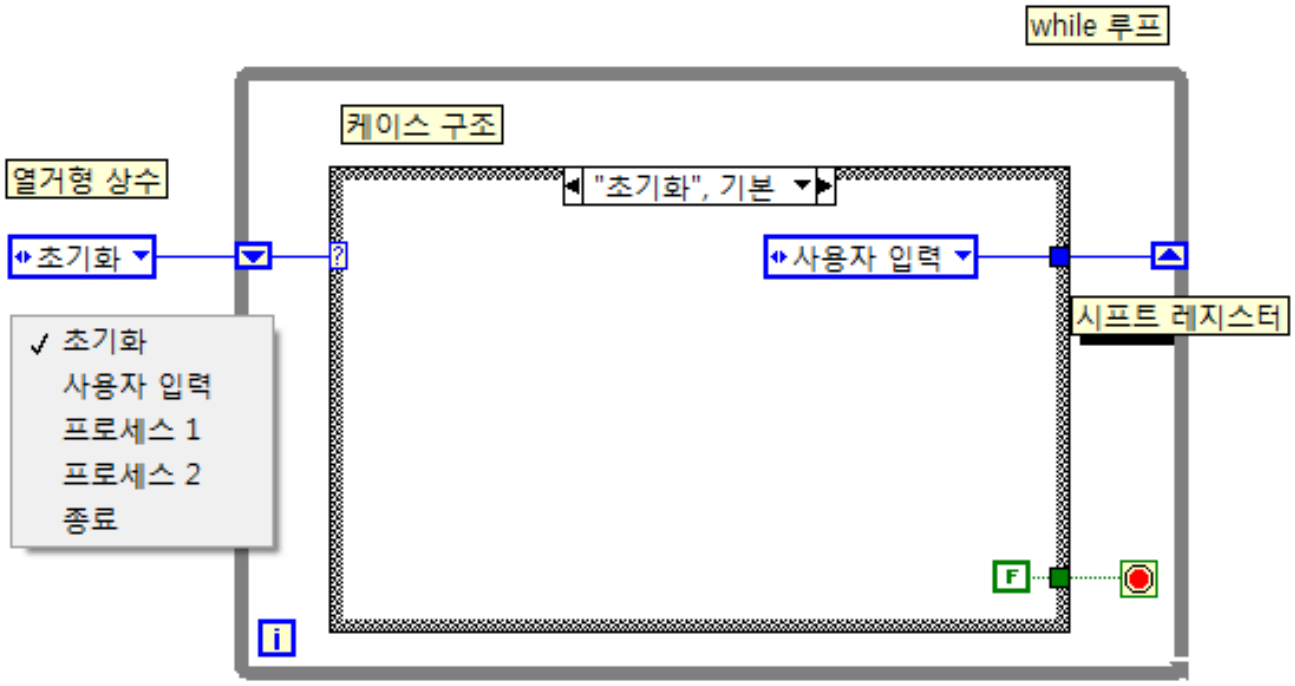
일반 디자인 패턴

- While 루프를 이용한 반복 실행
- 대부분의 프로그램 형태



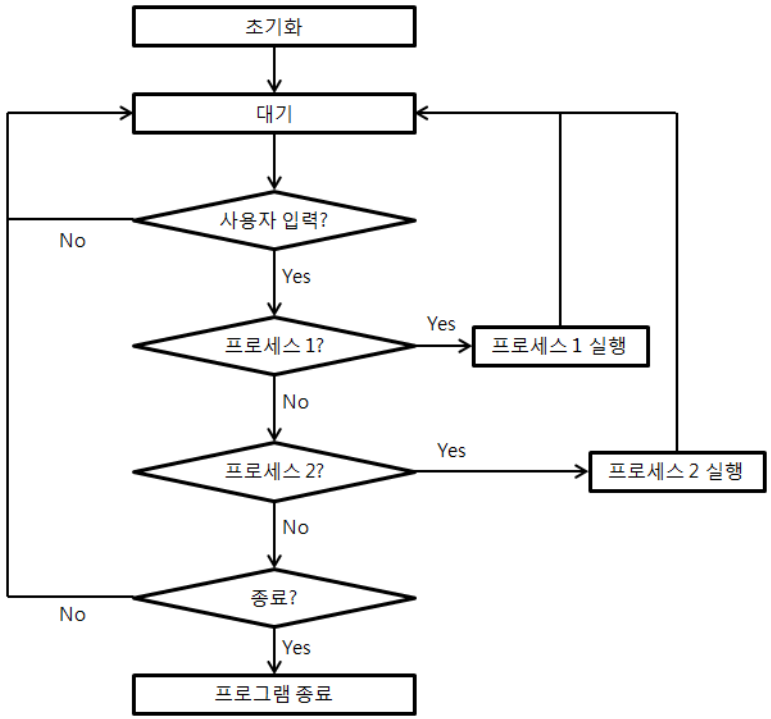
상태 머신 디자인 패턴

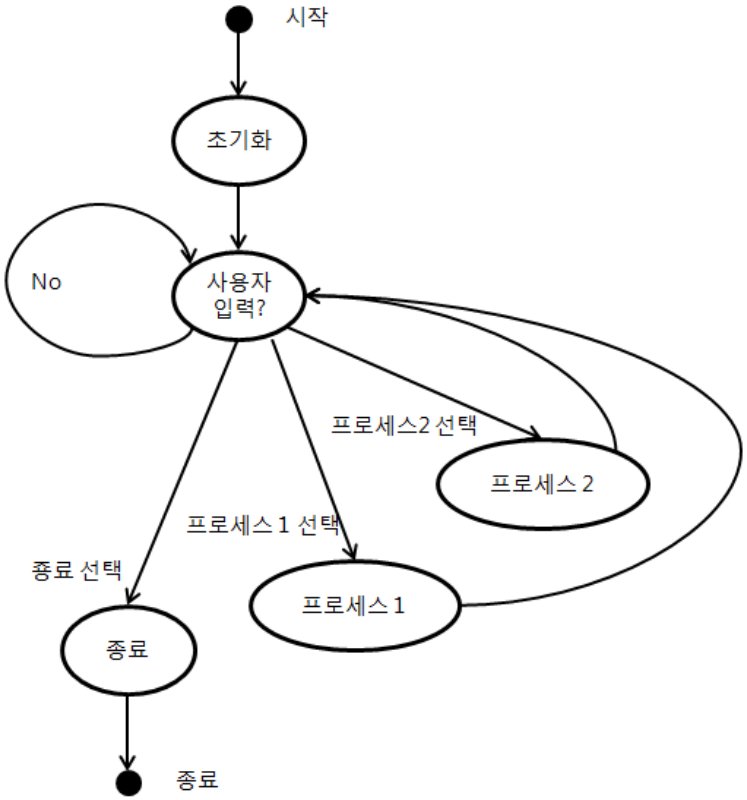
- 구현하는 기능을 하나의 상태로 구성
- 열거형 상수/While 루프 함수/케이스 구조



상태 머신 디자인 패턴

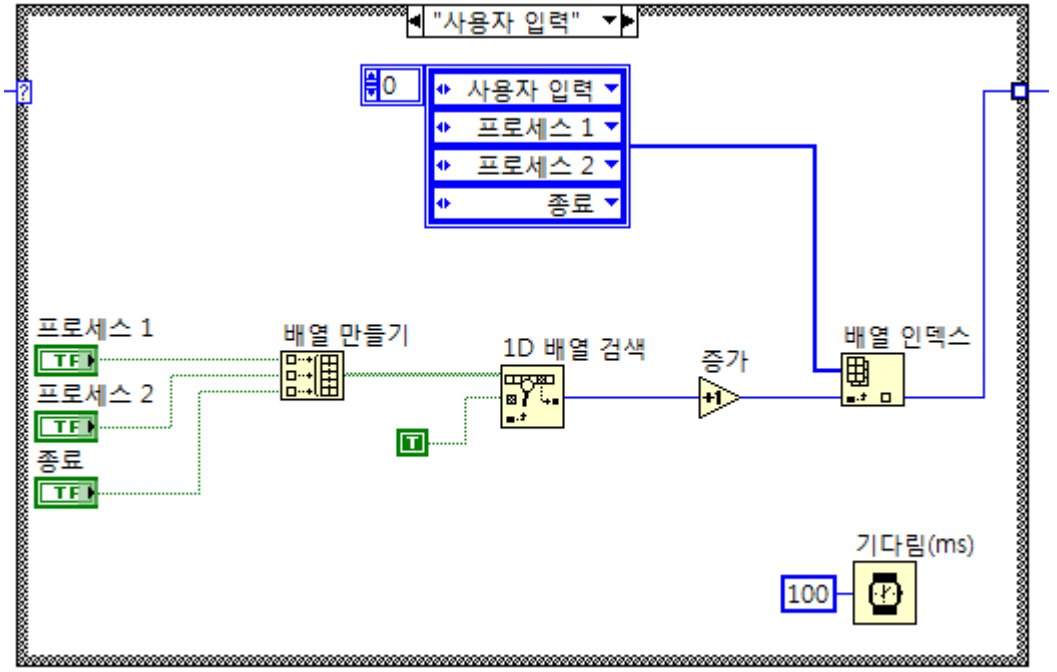
- 상태 전이 다이어그램
  - 상태 머신을 구성하기 위한 설계도





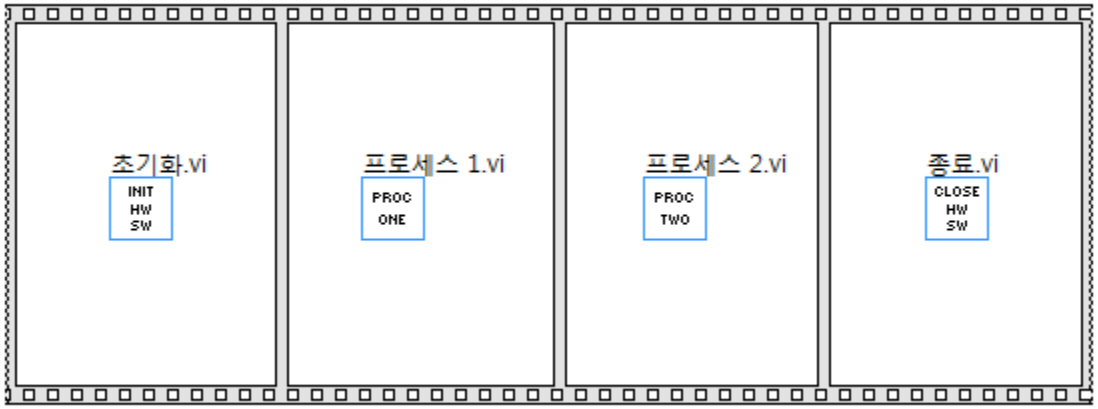
연습 문제 15

- 상태 머신 프로그램 만들기



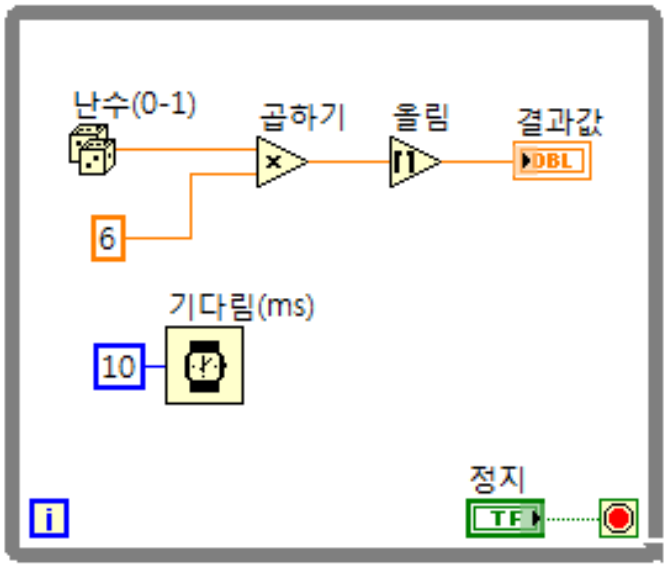
상태 머신 디자인 패턴

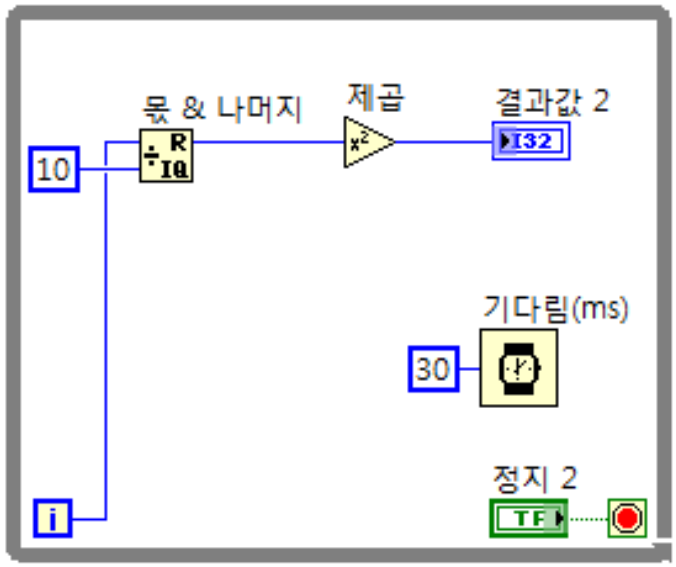
- 상태 머신 디자인 패턴 vs. 시퀀스 구조
  - 하나의 기능을 반복해서 실행 가능
  - 실행 순서를 임의로 바꿀 수 있음
  - 기능의 실행 중 중간에 정지 가능



병렬 디자인 패턴

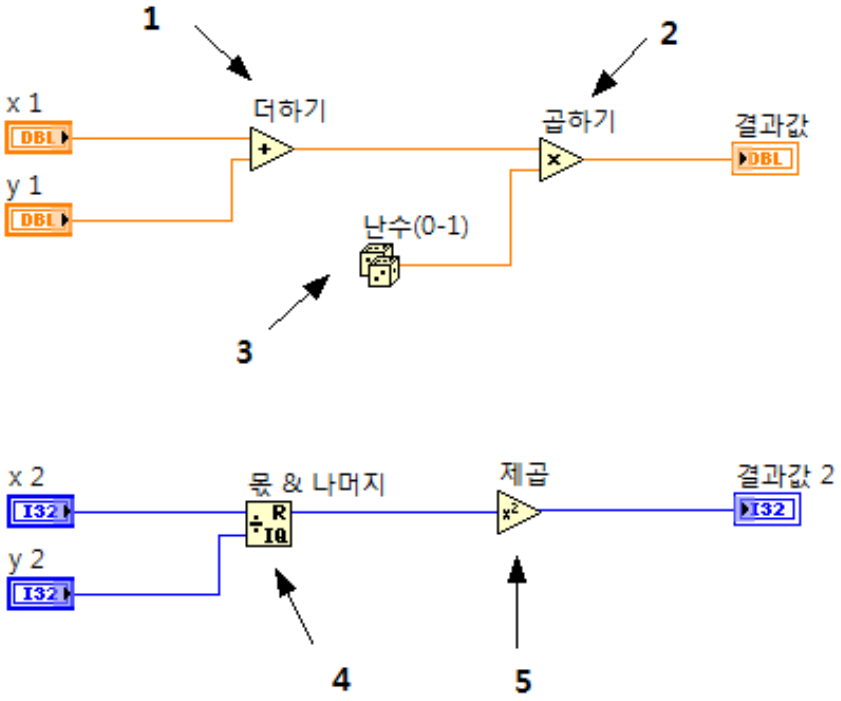
- 여러 개의 While 루프를 사용해 여러 가지 작업을 동시에 수행
  – 멀티 스레드 / 멀티 태스킹





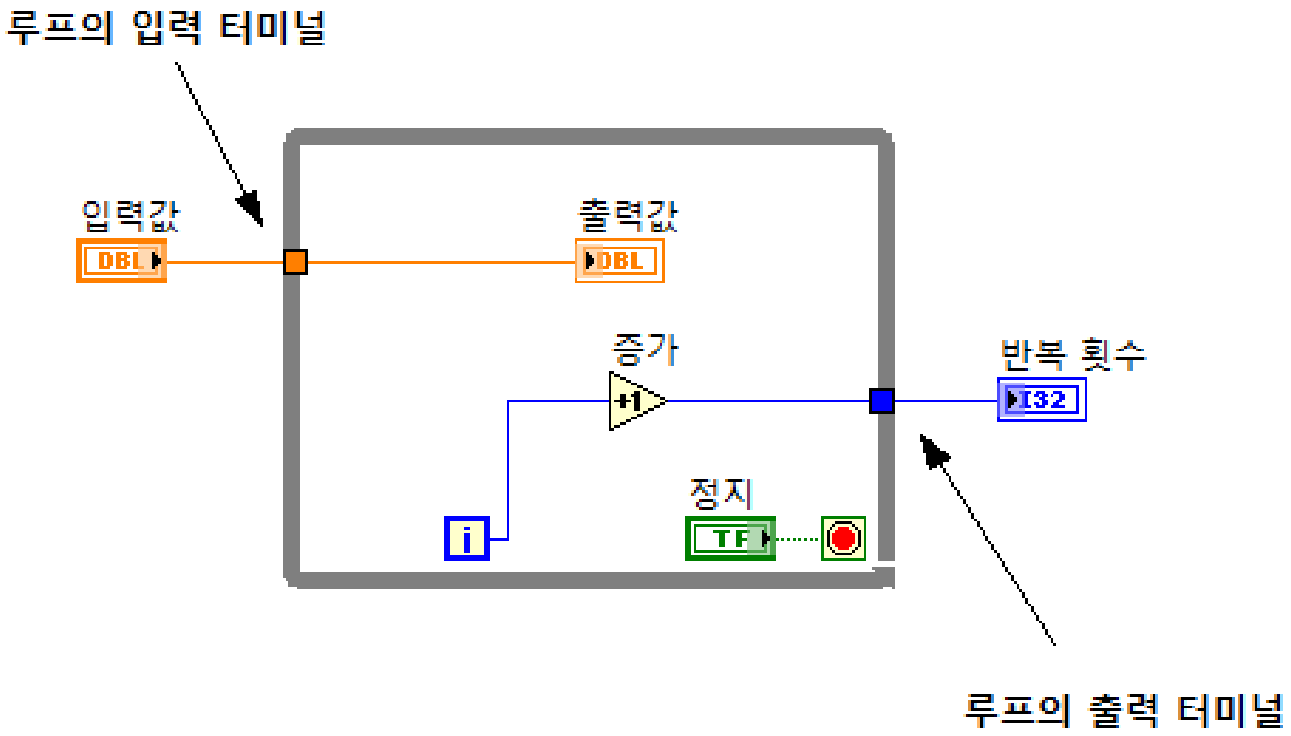
병렬 디자인 패턴

- 랩뷰 함수의 실행 순서
  - 아래의 함수들 중 가장 먼저 실행되는 함수는?



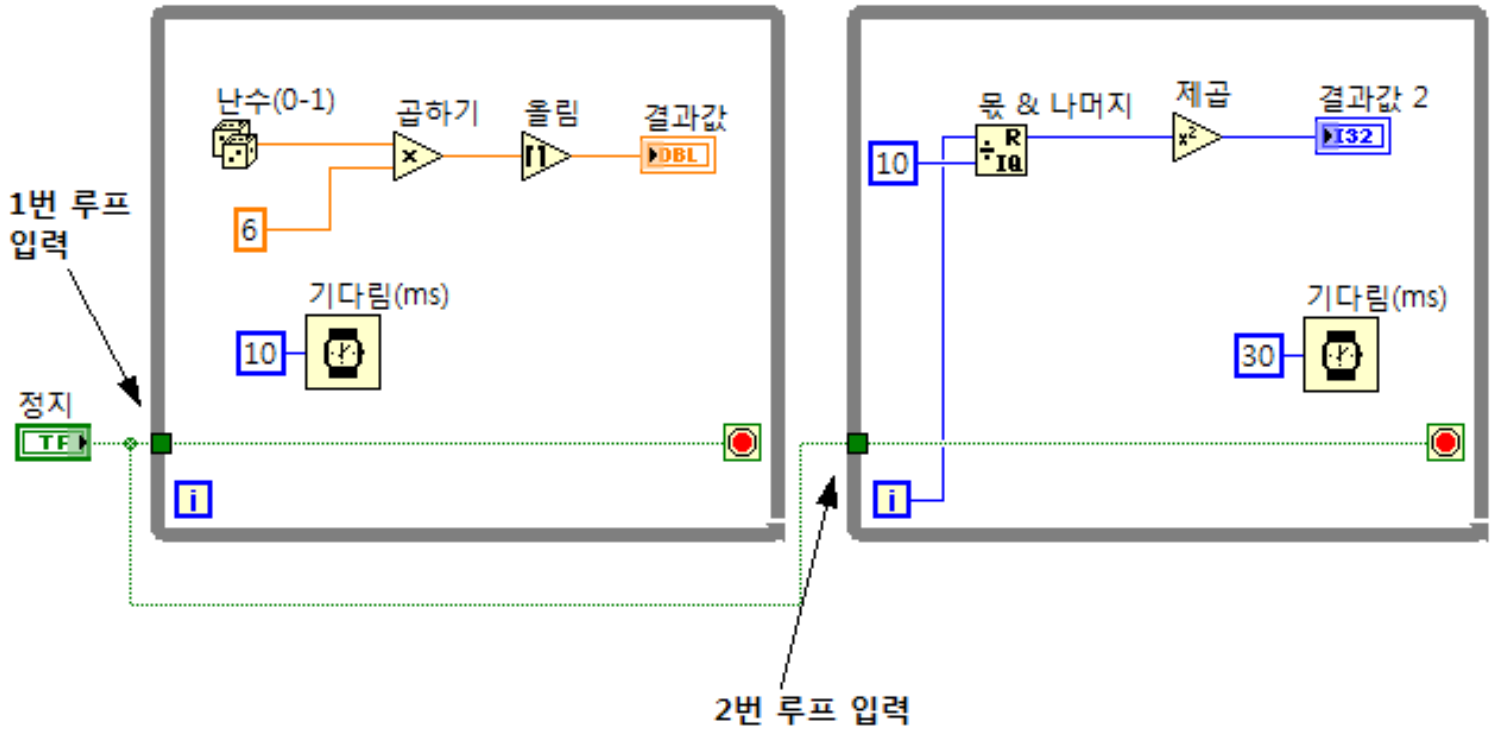
병렬 디자인 패턴

- 루프의 입력과 출력 터널



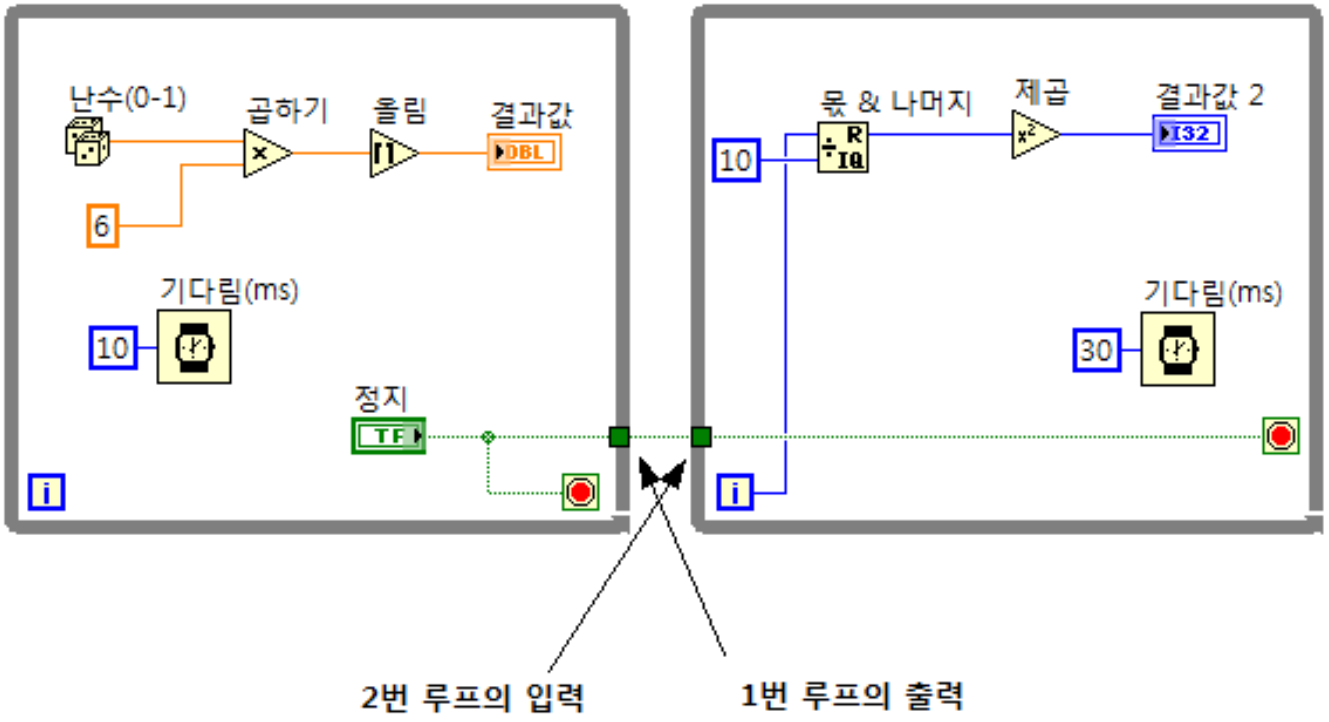
병렬 디자인 패턴

- 실행 중인 루프 간의 데이터 전달
  - 병렬로 실행되는 두 개의 루프를 동시에 정지?



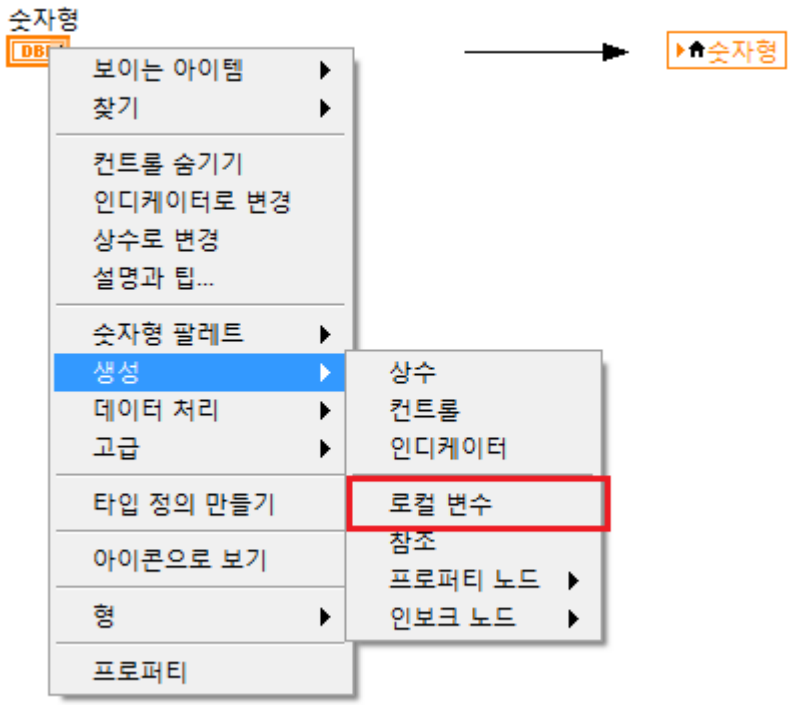
병렬 디자인 패턴

- 실행 중인 루프 간의 데이터 전달
  - 병렬로 실행되는 두 개의 루프를 동시에 정지?



변수

- 로컬 변수
  - 하나의 VI 내에서 데이터 전달

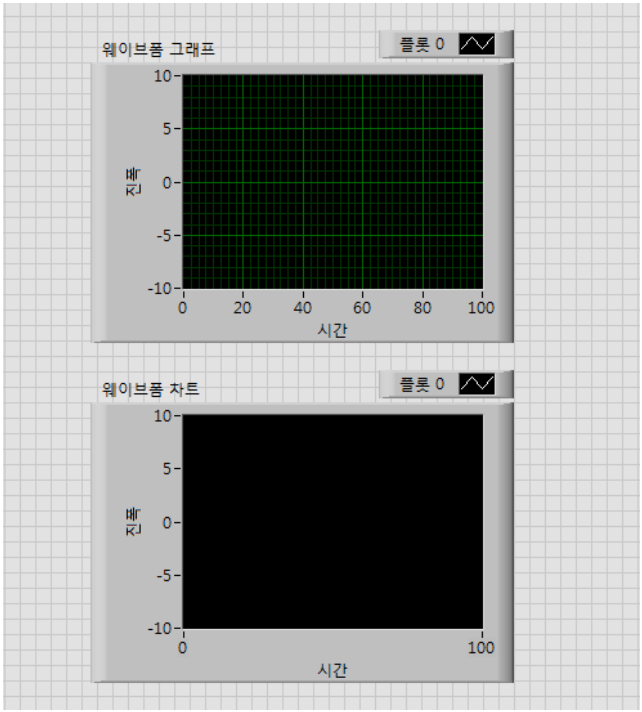


연습 문제 16

- 로컬 변수 사용하여 동시에 루프 종료하기

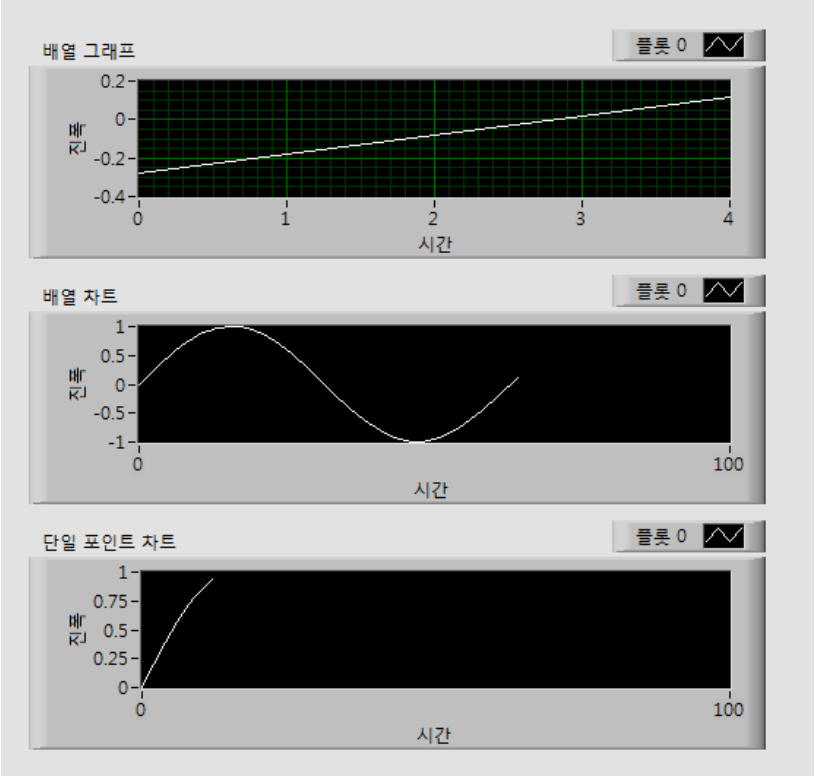
웨이브폼 그래프와 웨이브폼 차트

- 웨이브폼 그래프 vs. 웨이브폼 차트
  - 입력 데이터 형식 다름
  - 과거의 데이터(히스토리) 저장 공간 유무



연습 문제 17

- 웨이브폼 그래프와 웨이브폼 차트의 차이점 알아보기

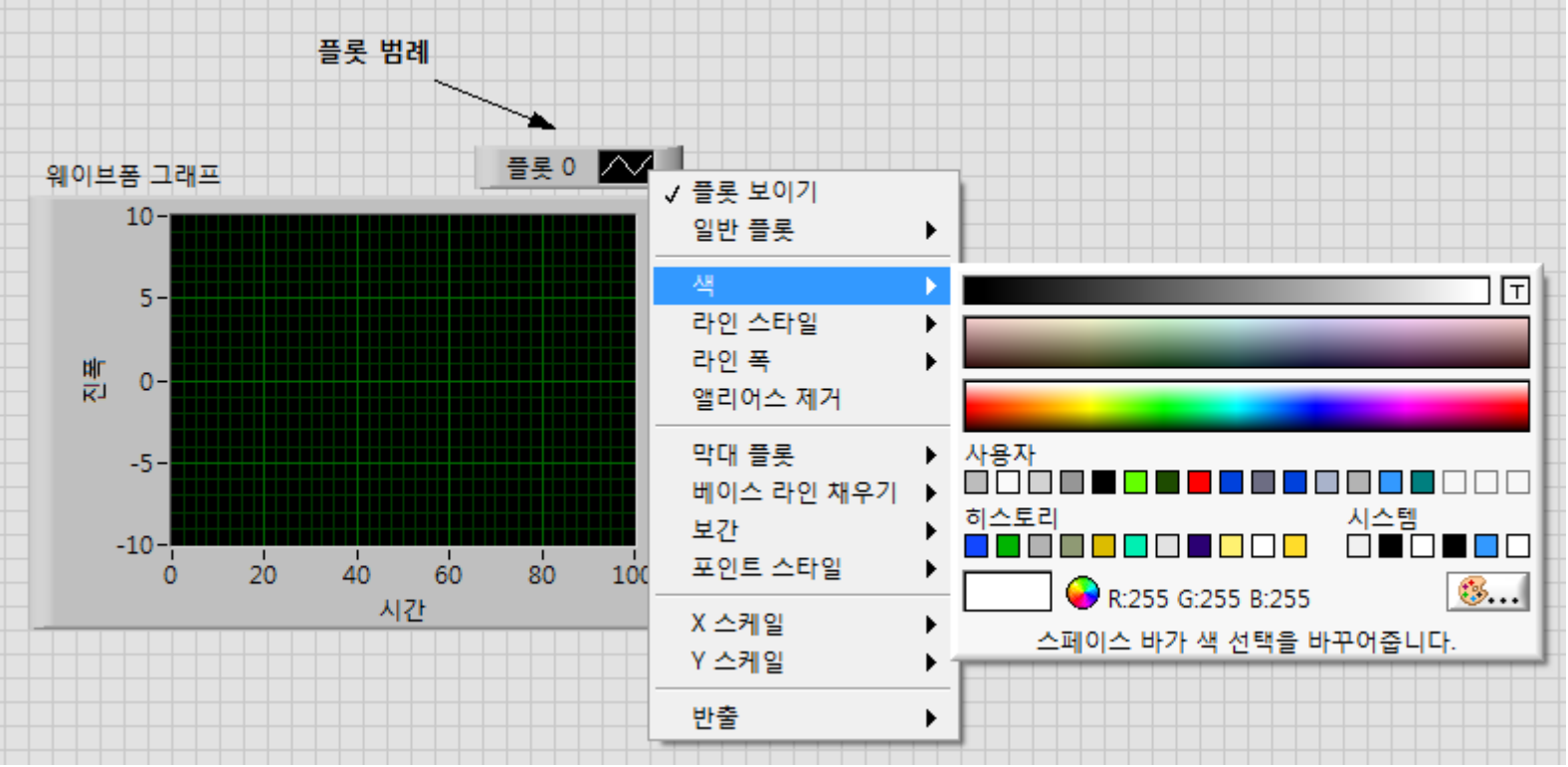


웨이브폼 그래프와 웨이브폼 차트

- 스케일의 설정
  - 수동 설정
  - 오토 스케일 설정

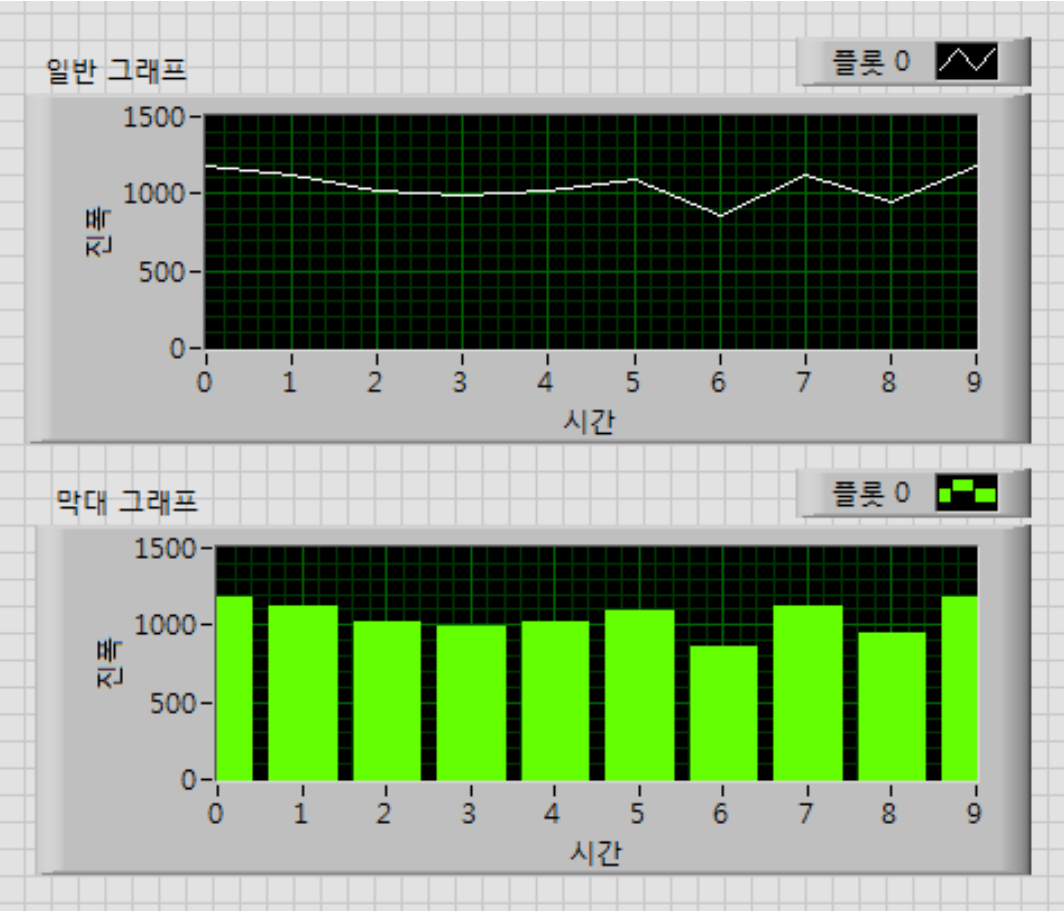
웨이브폼 그래프와 웨이브폼 차트

- 데이터 플롯의 스타일 변경



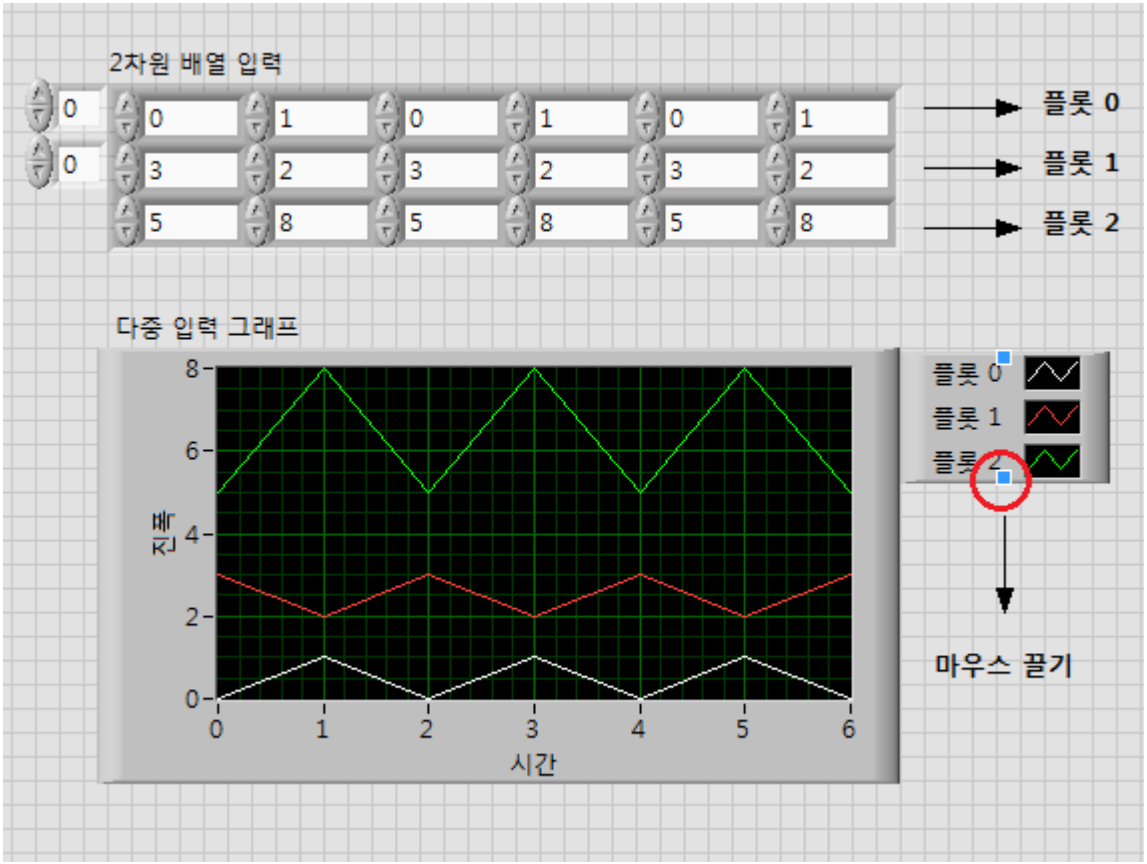
연습 문제 18

- 그래프 스타일 변경하기



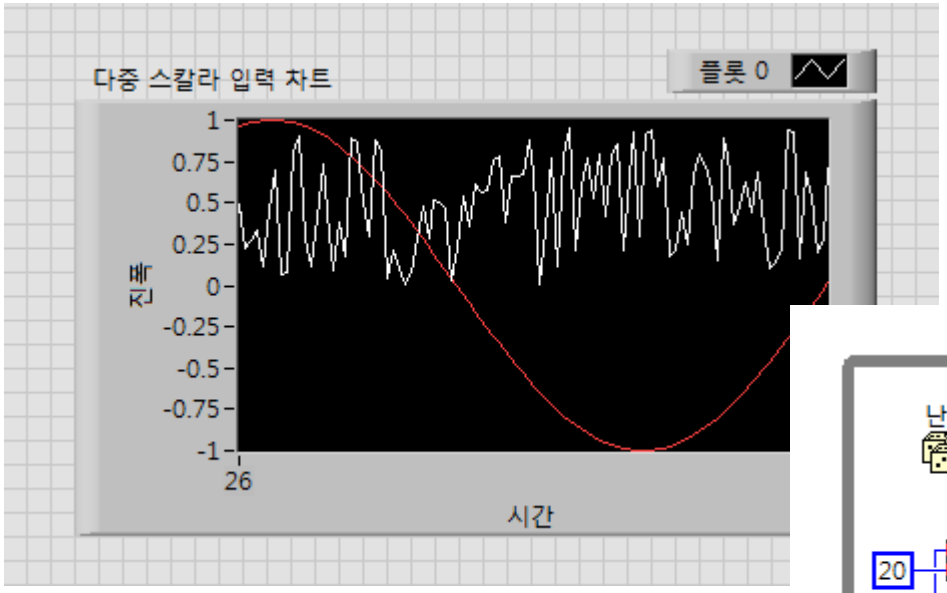
웨이브폼 그래프와 웨이브폼 차트

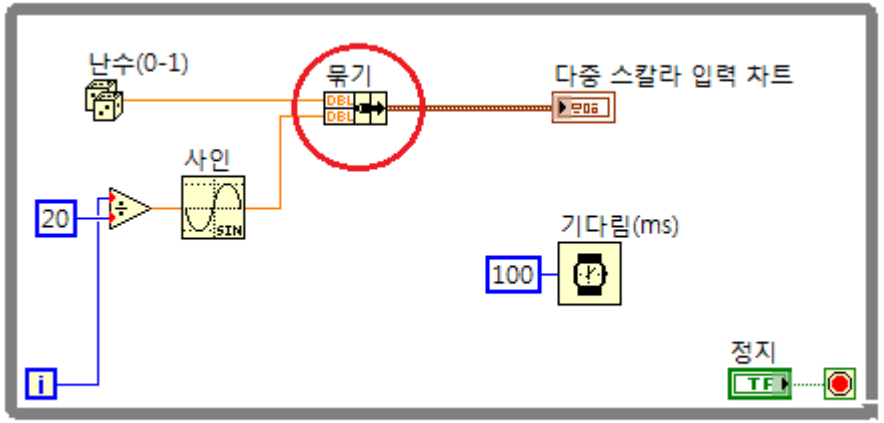
- 다중 데이터 플롯 (웨이브폼 그래프)



웨이브폼 그래프와 웨이브폼 차트

- 다중 데이터 플롯 (웨이브폼 차트)





Questions?