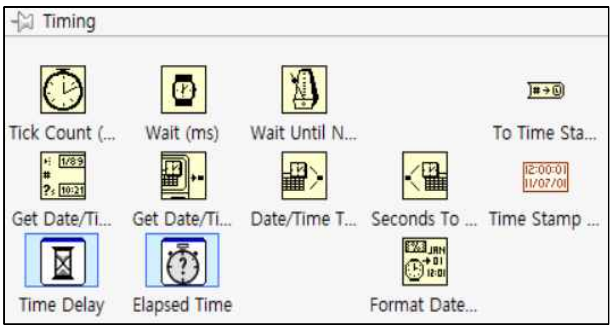
2023.02.27. p. 139 ~ p.160

반복 구조(while/for)의 출력 모드

- 마지막 값만 출력하든지 또는 실행 중 발생하는 모든 값을 출력하든지 두 가지 중 선택하도록 되어 있습니다.
- 구조를 통해서 나가라 때 또는 들어올 때 생기는 접합점들을 터널이라고 하는데, 이 터널에서 마우스 오른쪽 버튼을 클릭하면 "바로가기 메뉴 - 터널 모드 - 마지막 값" 또는 "인덱스 하기"를 볼 수 있습니다.
- '마지막 값' : 인덱싱 비활성화, '인덱스 하기' : 인덱싱 활성화

타이밍 : 반복 구조(While/For)를 사용할 때 시간 지연 및 CPU 점유율을 낮추기 위한 **타이밍 노드**와 함께 사용하는 것이 좋습니다. 특히 while 루트는 언제 종료할지 모르는 프로그램이기 때문에 반드시 타이밍 노드와 함께 사용해야 합니다.

- 기다림(wait)과 다음 ms 배수까지 "기다림"노드를 비교하여 보겠습니다.

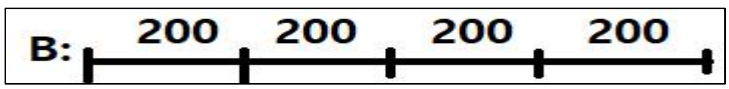


wait(ms) : 작동시간 + 기다림 (ms) 시간 **wait until next ms multiple** : 다음 ms 배수까지 기다림

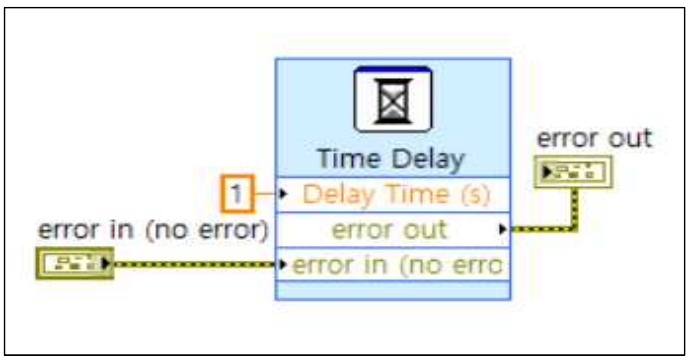
- 만약 A와 B에서 코드를 실행하는 데 있어, 170 "msec"가 걸린다고 가정을 한다면 A의 반복 주기는 270 "msec"가 되는, 반면 B는 반복 주기가 얼마일까요? 힌트는 노드의 이름에 있습니다. 즉 다음 ms 배수까지 기다림 노드는 주어진 시간보다 코드 실행이 오래 걸린다면 주어진 시간의 다음 배수로 주기가 맞춰져서 200 msec가 됩니다.

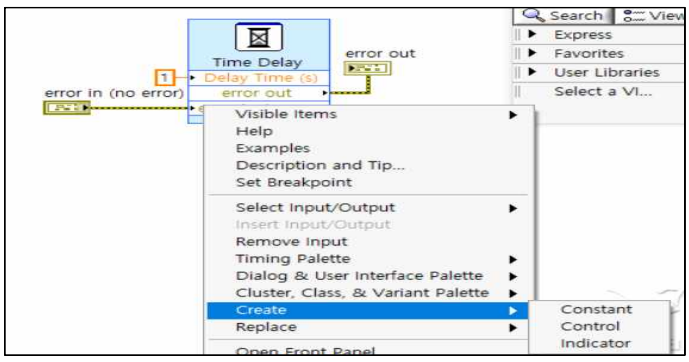
  A : 100 = 기다림(ms)

  B : 100 = 다음 ms 배수까지 기다림

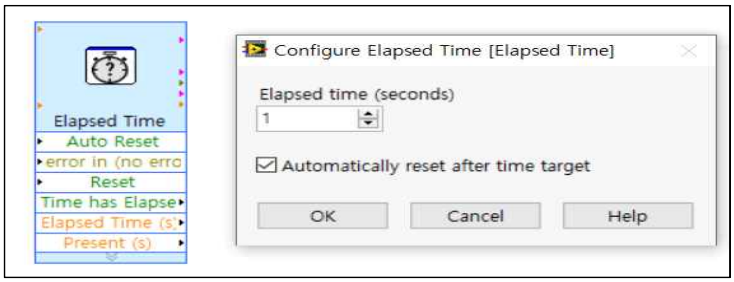


- B : **실행 시간이 170 "msec"이면 이것이 "기다림" 시간과 "다음 ms 배수까지 기다림"보다 시간이 길기 때문에, B : 의문이 생길 수 있는데, 이때는 100*@시간만큼 다음 ms 배수까지 기다림이 진행되는데, 즉 1^{st},2^{nd} 사이의 시간 차이는 200 "ms"** 입니다.
- A는 170 ("진행" 시간) + 100 ("기다림" 시간)이 1 cycle에 270 "msec"가 걸린다는 것을 확인할 수 있다.





시간 지연 노드 : "기다림" 노드와 기능이 똑같습니다. 단 **에러 클러스터 입출력**이 추가되어 있습니다.



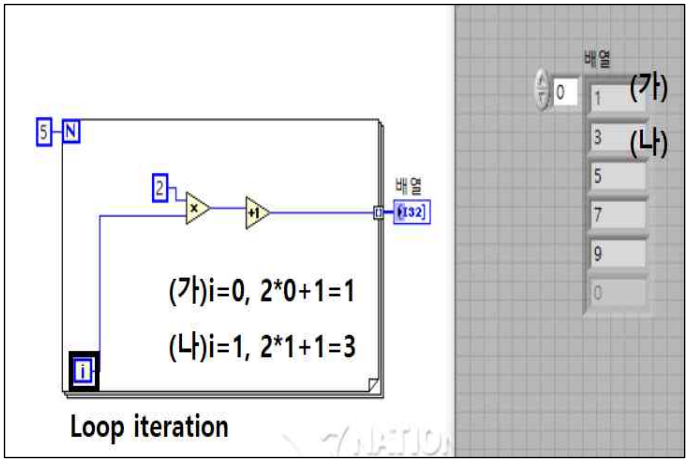
틱 카운트 노드

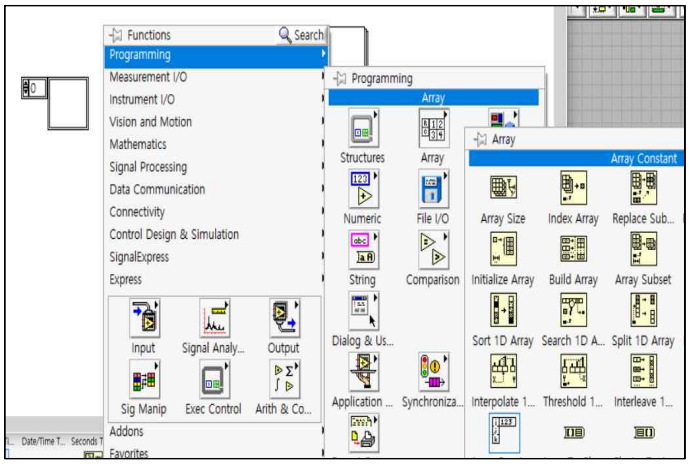
- 경과시간 설정 : 경과시간 노드는 설정한 시간이 경과 되었는지를 체크 해 줍니다. 1초를 설정하였고 1초가 경과 하게 되면 '경과 완료?'에 '참'값을 출력합니다.

틱 카운트 노드 : "초시계"와 같은 기능을 합니다.

배열과 For 루프

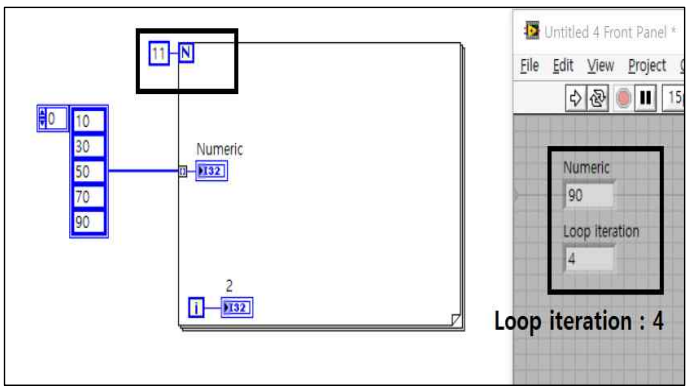
- ex) "1, 3, 5, 7, 9"의 원소를 가진 1차원 배열을 For 루프를 이용하여 만들어 보았습니다.

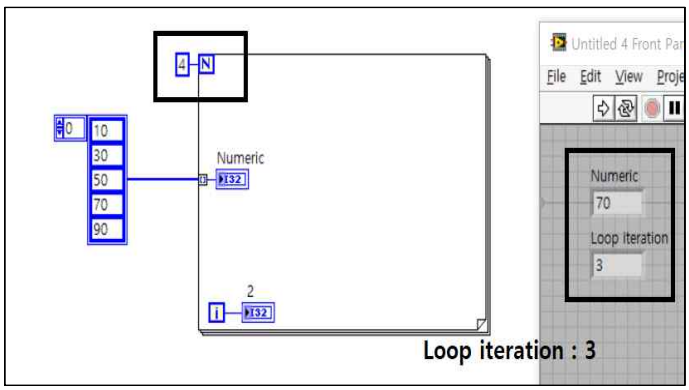




Array Constant

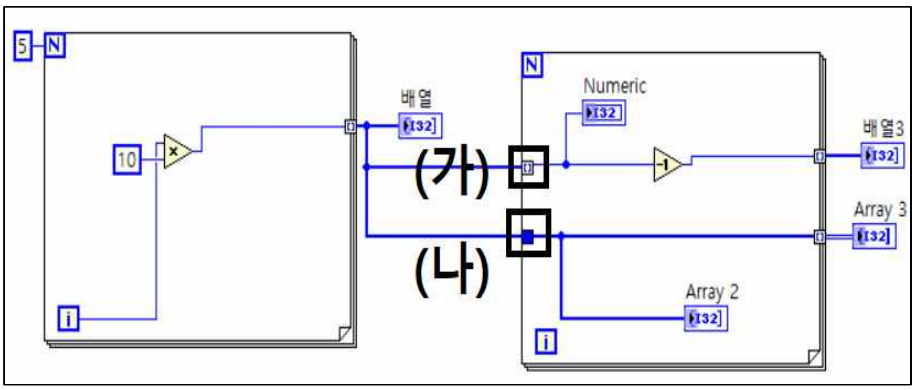
- N에 연결된 값과 배열의 원소 개수 중 **작은 쪽**에 영향을 받습니다.





즉, 왼쪽은 5번, 오른쪽은 3번 박복을 합니다.

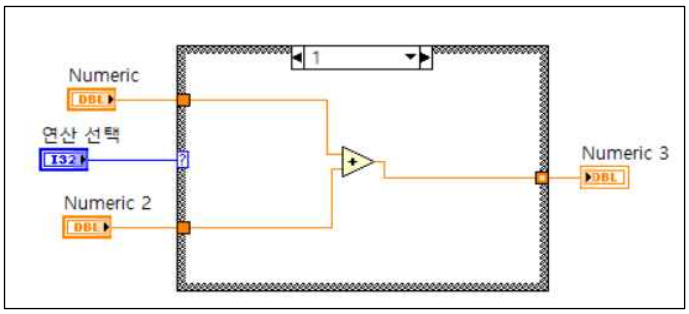
- 왼쪽 : 5번 반복, 오른쪽 : 4번 반복 -> 오른쪽 70 출력

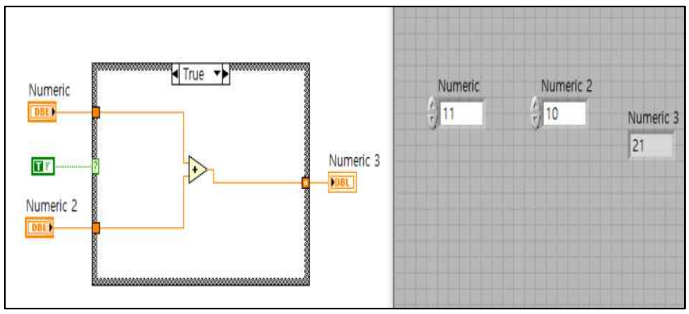


- (가) : 인덱싱 활성화, (나) : 인덱싱 비활성화 -> 배열 3 : For 루프는 5번을 반복하고 각 반복마다 'numeric'에서는 '배열'의 원소, 즉 "0, 10, 20, 30, 40"이 차례로 출력. 'Array2'은 매 반복 마다 [0, 10, 20, 30, 40]이 출력되는 것을 확인할 수 있습니다.

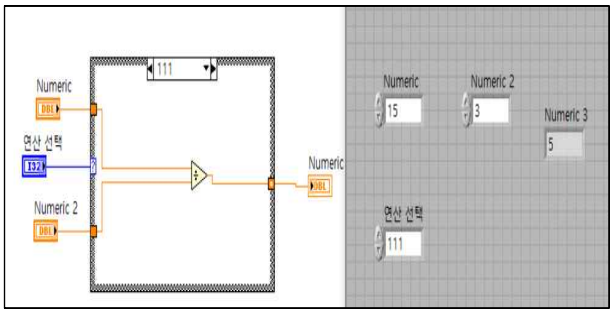
케이스 구조 : 입력하는 값에 따라 실행하는 코드의 내용이 달라질 경우, 케이스 구조를 사용합니다. **'선택자 터미널'**, **'케이스 선택자 라벨'**로 구성

- 선택자 터미널에 어떤 값이 들어오느냐에 따라 실행되는 케이스가 달라집니다.
- 케이스 선택자 라벨을 통해, 케이스를 구분할 수 있습니다.





불리언 데이터 타입을 선택자 터미널에 연결을 하면 '참'케이스와 '거짓'케이스가 자동적으로 생성이 됩니다.
- '1' : a+b=c, '111' : a/b=c, '0' : a*b=c



- 앞에서 정의한 숫자 이외의 값을 입력했을 때는 '기본'이라고 표기 된 케이스를 실행합니다. ex) 연산 선택에 '777'을 입력했을 때는 '기본'이라고 표기된 케이스를 실행합니다. 여기서 '기본'이라는 값을 지우게 되면, 실행 버튼이 깨지게 됩니다. => 선택자 터미널에 연결되어 있는 데이터 타입이 정의한 케이스 이외의 값을 가질 수 있는 숫자형 또는 문자열 등은 반드시 기본 케이스가 정의되어 있어야 합니다. 이것이 케이스 구조를 사용할 때 첫 번째 주의점입니다.
- 케이스 선택자에 불리언 데이터 타입이 연결되어 있는 경우, 기본 케이스가 없습니다. -> 그 이유는 불리언이 가질 수 있는 값은 '참'과 '거짓' 이외의 값을 가질 수 없고, '참' 케이스와 '거짓' 케이스를 모두 만들었기 때문입니다.

- 선택자 터미널에 열거형을 연결하였습니다. 열거형에 정의된 모든 아이템에 해당하는 케이스를 자동으로 생성하려면 케이스 구조의 **"바로가기 메뉴 - 모든 값에 대한 케이스 추가"**를 선택하면 됩니다.
- 선택자 터미널에 에러클러스트를 연결하였습니다. 자동적으로 '에러 없음' 케이스와 '에러'케이스가 생깁니다.

- 케이스 구조에서 출력을 케이스 구조 밖으로 내보낼 때 생기는 출력 터널에 생성한 모든 케이스에 대해 출력값이 전부 연결이 되어 있어야 합니다. 하나라도 연결이 안 되면 출력 터널이 비어 있게 되면서 '터널 지정 잃음'이라는 에러가 발생하게 됩니다.