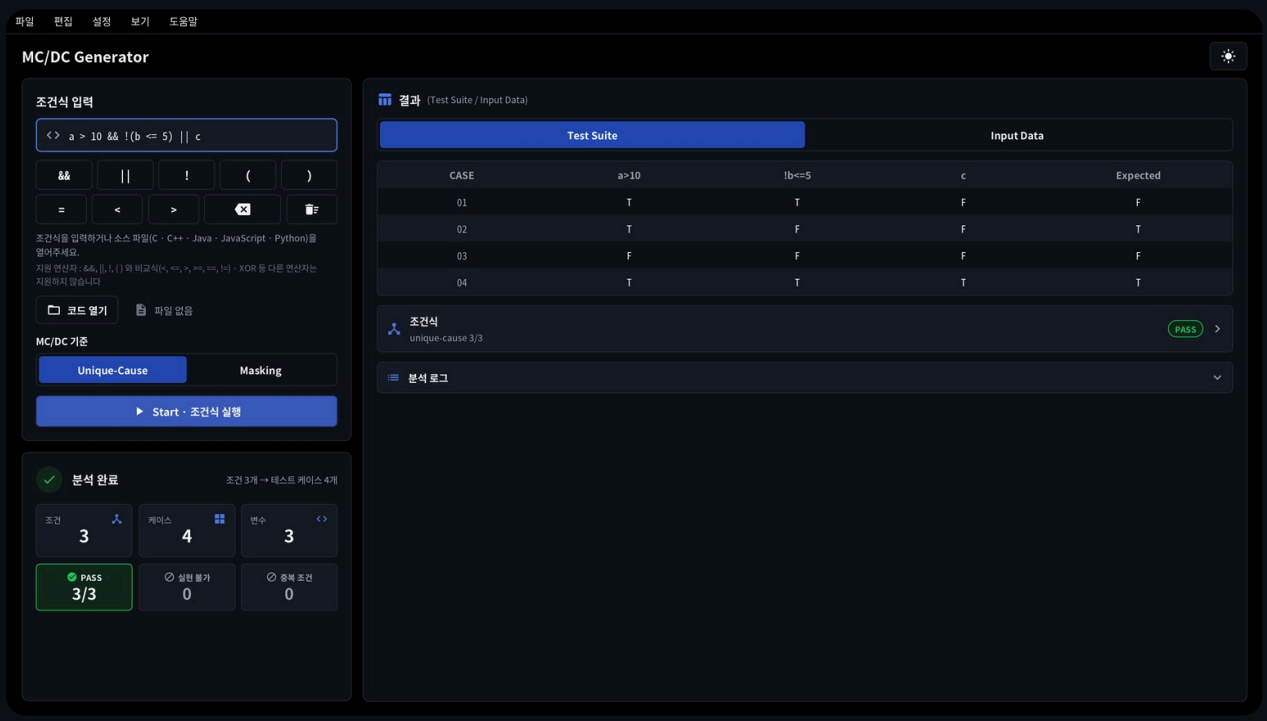
COVERAGELAB

MC/DC Generator

MC/DC 테스트 스위트 생성기

Windows · coveragelab.kr



MC/DC 테스트 스위트 생성기

사용 설명서 · Windows

1. 프로그램 소개
2. 빠른 시작
3. 소스 파일
4. 결과 읽기
5. 통합분석
6. 저장
7. 설정과 단축키
8. 참고와 제한
9. 문의

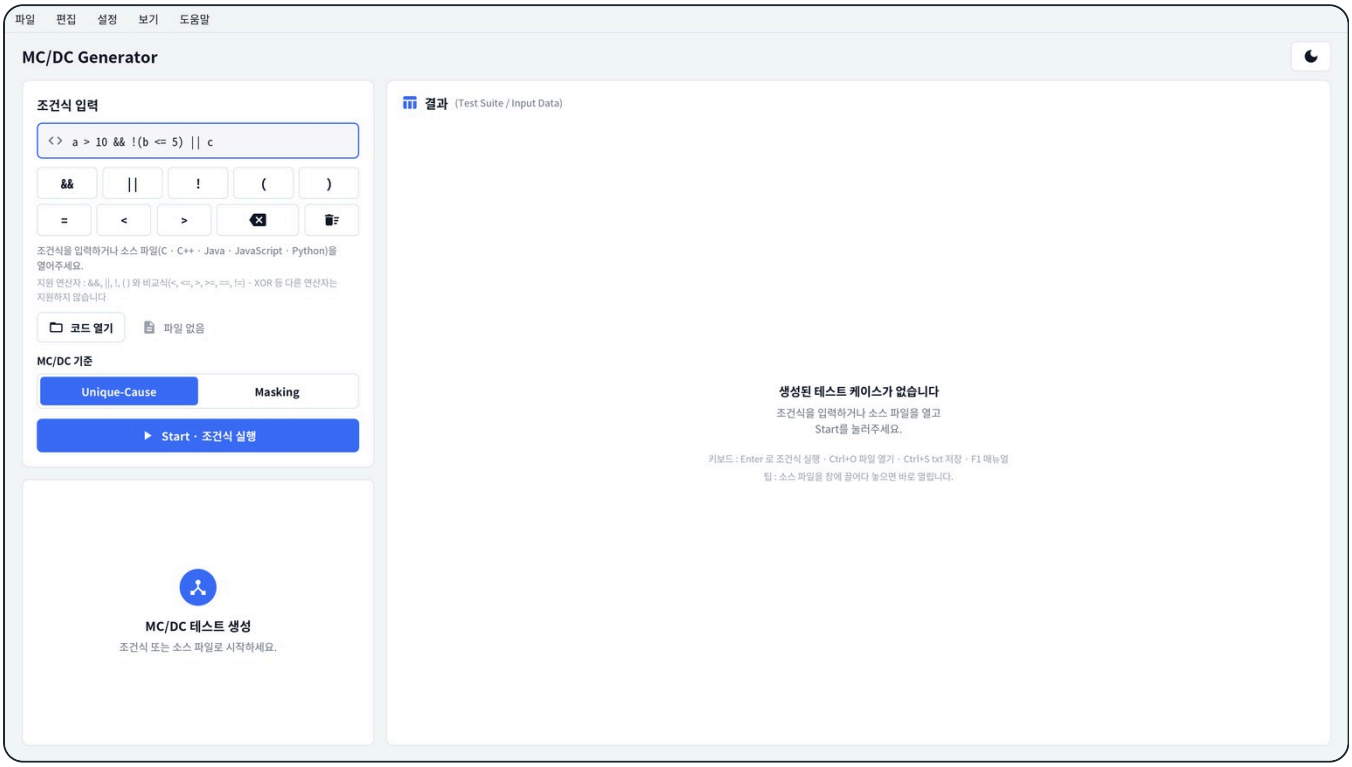
1. 프로그램 소개

MC/DC(Modified Condition / Decision Coverage)는 조건문의 모든 조건에 대해, 그 조건 하나만 바뀌어도 조건문의 결과가 바뀐다는 것을 보여 주는 테스트 케이스 쌍을 요구합니다. 이 프로그램은 직접 입력한 조건식이나 소스 파일에서 찾은 조건문으로부터 **Robin's Rule** 알고리즘으로 테스트 스위트를 만듭니다.

- 서로 다른 조건이 N 개인 **단일 불리언 식**(SBE - 모든 조건이 한 번씩만 나오는 식)에서는 **unique-cause** MC/DC를 만족하는 테스트 케이스를 정확히 **N+1** 개 만듭니다.
- 생성 기준은 **Unique-Cause**(Robin's Rule, N+1)와 **Masking** 두 가지이며, 입력 카드의 토글로 고릅니다 (2절).
- 모든 결과를 만든 뒤 다시 검증합니다(**MC/DC Check**) : 조건마다 독립 영향 쌍을 증명하고, 분석 완료 화면에 **PASS** · **FAIL** · 분석 불가로 표시합니다.
- 진리값 표(**Test Suite**)뿐 아니라 `x > 10` 같은 비교식에 대해 실제 입력값(**Input Data**)도 제안합니다.

2. 빠른 시작

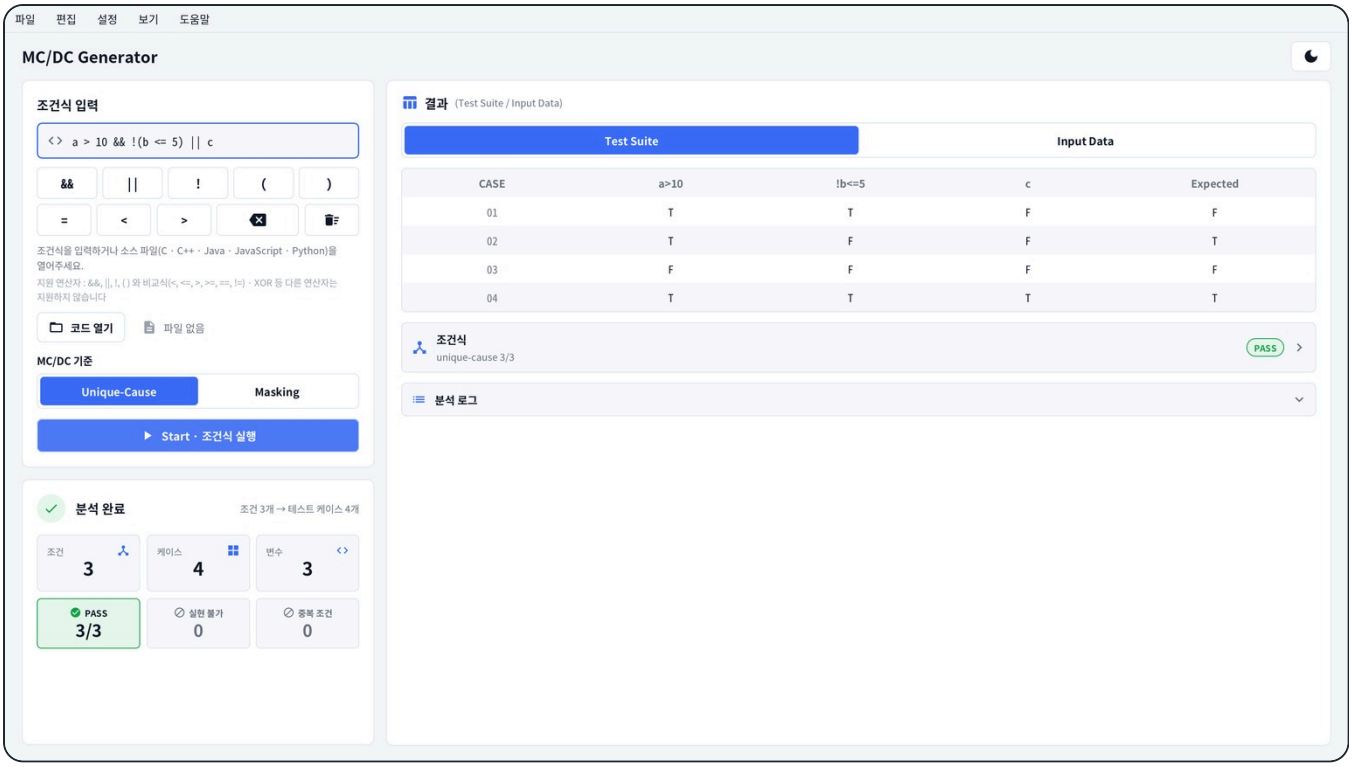
1. **조건식 입력** 칸에 조건식을 입력합니다. 예 : `!(a && b) || c`.
   (`if`와 문장은 빼고 식만 입력합니다.)
2. `Enter`를 누르거나 **Start**를 클릭합니다.
3. 분석 완료 화면에 개수와 MC/DC Check가, 오른쪽 표에 테스트 케이스가 나타납니다.
4. 소스 파일이면 **코드 열기**로 열거나 창에 끌어다 놓은 뒤 **Start**를 누릅니다. 조건문이 하나면 바로 분석하고, 여러 개면 조건문 목록에서 원하는 `Line`을 **더블클릭**하거나 **개별분석** / **통합분석**을 누릅니다 (3절).



조건식을 입력하세요

연산자 버튼(`&&` `||` `!` `(` `)` `=` `<` `>`)은 조건식에 대입하고, ⌫ 는 한 글자를 지우며 휴지통은 조건식 전체를 비웁니다 (`Ctrl+L`은 입력과 결과를 모두 지웁니다).

지원 : `&&`, `||`, `!`, 괄호, 비교식 `< <= > >= == !=`. **미지원** : XOR 등 그 밖의 연산자. 논리 연산자가 아닌 모든 것 - 불리언 변수, 함수 호출, 멤버 접근, 배열 원소, 산술식, 비트 연산식 - 은 하나의 *조건*으로 봅니다. 글자가 같은 두 조건은 같은 조건으로 취급합니다(**중복 조건**).



조건 3개 → 테스트 케이스 4개, 검증 PASS

- 조건식 입력 칸에 대입한 값을 그대로 조건으로 쓰는 식(`a = 3` , `a = b`)을 넣으면 비교가 아니라서 오류로 알려 줍니다. 같은지 비교는 `a == 3` 처럼 `==` 로 씁니다. 대입한 값을 비교하는 식(`(c = getchar()) != EOF`)은 하나의 조건으로 그대로 다룹니다. 소스 파일의 `if (a = 3)` 은 3절처럼 분석하고 경고합니다.
- 비교를 이어 쓴 식(`a < b < c`)도 오류로 알려 줍니다. C 에서는 앞 비교의 결과(0 / 1)를 다음 값과 비교하기 때문입니다. 범위는 `a < b && b < c` 처럼 두 비교를 `&&` 로 잇습니다. 소스 파일에서는 입력값을 `?` 로 두고 경고합니다.
- 상수 조건(`1` , `0` , `true` , `3 > 2`)은 결과가 늘 같아 반대 결과의 케이스를 실현 불가로 표시하고 경고합니다. 증감이 들어간 조건(`n-- > 0` , `i++ < 10`)은 바꾸기 전 값으로 입력값을 만들고 참고로 알려 줍니다. 음수 상수(`a > -5`)는 그대로 읽습니다(-4 / -5).

MC/DC 기준 - Unique-Cause / Masking

Start 버튼 위의 **MC/DC 기준** 토글로 생성 기준을 고릅니다 (선택은 설정에 기억됩니다). 소스 파일도 **Start**를 누르기 전에 여기서 고르면 됩니다. 결과와 저장 파일에는 어느 기준으로 만들었는지 항상 표시됩니다 - 분석 로그와 txt 머리의 `※ 생성 기준 : …` 줄, csv 첫 줄의 `# Criterion : …` 주석.

- **Unique-Cause** (기본값) - 검사하는 조건 하나만 바꾸고 나머지 조건은 모두 같은 두 케이스로 독립 영향을 보입니다 (Robin's Rule, 서로 다른 조건 N개에 케이스 N+1개).
- **Masking** - 다른 조건이 달라도 그 차이가 연산자에 가려져(masked) 결과에 닿지 않으면 같은 쌍으로 인정합니다. `(a && b) || (a && c)` 처럼 같은 조건이 두 번 이상 나오는 식(결합 조건)이나,

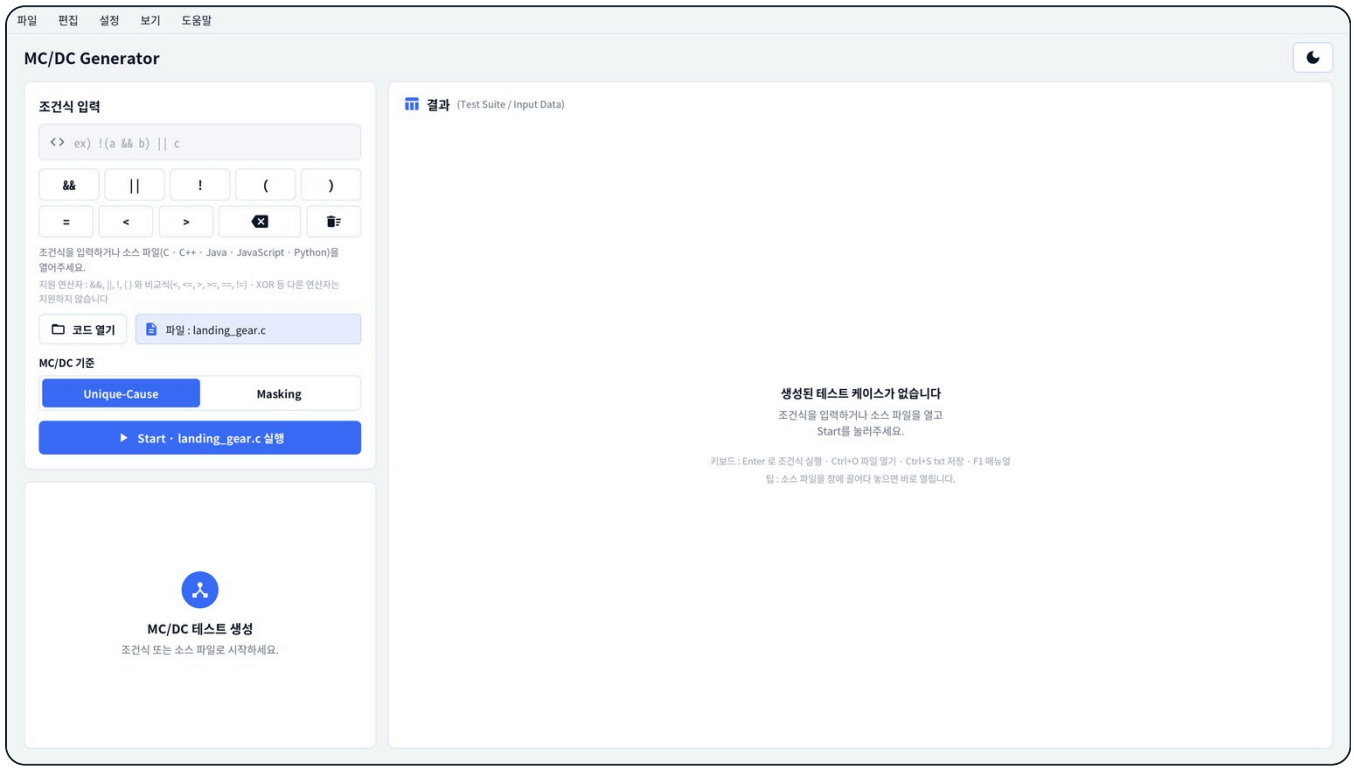
같은 변수의 조건이 충돌해 Unique-Cause 로는 실현 불가 케이스가 생기는 식에서도 실행 가능한 케이스만으로 모든 조건을 보입니다. 조건이 한 번씩만 나오는 식에서는 두 기준의 결과가 같습니다.

Masking으로 만든 결과는 검증 줄이 `MC/DC Check : PASS (masking n/n · unique-cause k)`로, 결합 조건은 "참고 : 결합 조건" 으로 표시되고, 분석 로그와 저장 파일 머리에 `※ 생성 기준 : Masking MC/DC` 줄이 붙습니다. 자세한 설명은 4절의 "MC/DC 기준" 항목에 있습니다.

---

3. 소스 파일

파일 ▸ 코드 열기…(Ctrl+O)나 **코드 열기** 버튼을 누르거나, 파일을 창 위에 끌어다 놓으세요. C · C++ · Java · JavaScript · Python 파일을 확장자로 알아보며, 주석 · 문자열 · 전처리기 줄은 무시합니다. 파일은 창 어디에나 - 조건식 입력란이나 분석 로그 위라도 - 끌어다 놓을 수 있습니다.

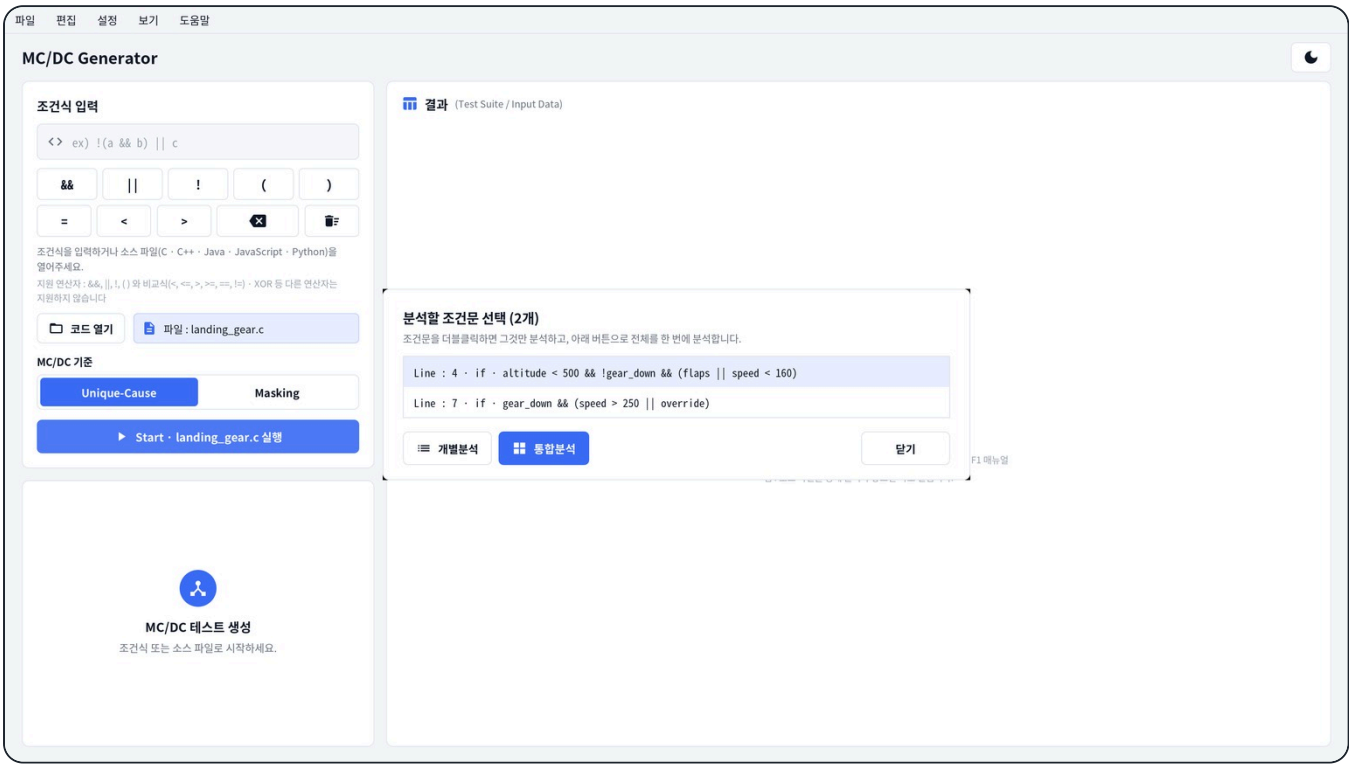


C · C++ · Java · JavaScript · Python 소스도 그대로 열기

- 찾는 조건문 : `if` , `else if` , `while` , `for` (그리고 `do … while`의 `while`). Python : `if` / `elif` / `while`과 `and` , `or` , `not` . C++ 대체 토큰과 JavaScript의 `===` / `!==`은 자동으로 변환합니다.
- 파일을 연 뒤 **Start**를 누르면 선택한 MC/DC 기준으로 분석합니다. 조건문이 하나뿐이면 바로 분석하고, 여러 개면 조건문 목록이 나타납니다 :

동작	결과

조건문을 **더블클릭** (또는 선택 후 Enter)	그 조건문만 분석합니다. 한 번 클릭은 선택만 합니다.
개별분석	모든 조건문을 각각 분석합니다 - 조건문마다 Test Suite와 Input Data 표를 생성합니다.
통합분석	모든 조건문을 분석한 뒤 한 표로 합칩니다 (5절).



파일 안의 결정문을 모두 찾아 줍니다

코드를 불러온 뒤 목록에서 원하는 `Line`을 **더블클릭**하면 그 조건문이 바로 분석됩니다. 한 번 클릭은 선택만 하며, 선택한 뒤 Enter를 눌러도 같습니다.

긴 조건식은 목록에서 줄여서 보여 줍니다. 마우스를 올리면 전체 글자가 보입니다.

대입 · 상수 · 위쪽 코드의 고정 값

소스 파일은 코드에 적힌 그대로 분석합니다. 비교가 아닌 조건은 실제로 참/거짓을 정하는 값으로 입력값을 만들고, 분석 완료 화면의 **확인 필요** 상자와 조건문 행, 분석 로그에 알립니다.

조건	다루는 방식
상수 조건 `while (1)`	결과가 늘 같으므로 반대 결과의 케이스를 실현 불가로 표시하고 경고합니다.

`do { … } while (0)` `if (true)`	

상수 대입
`if (a = 3)`
`if (a = 0)`
	비교가 아니라 대입이라 결과가 늘 T(3) / F(0) 입니다. 반대 케이스는 실현 불가로 표시하고 경고합니다. 안쪽 조건문은 a = 3 인 것으로 경로 값을 만들고, 나올 수 없는 쪽(else) 안의 조건문은 도달 불가가 됩니다.

값 대입 `if (a = b)`	실제 조건은 b(0이 아니면 참)이므로 입력값을 b에 주고, 안쪽 조건문은 a에 b의 값이 들어간 것으로 만듭니다 (`if (a = b) { if (a > 5) … }` 의 안쪽 케이스는 a = b = 6). `==` 를 잘못 쓴 것일 수 있어 경고합니다. 일부러 쓴 대입임을 괄호로 보인 식(`while ((p = p->next))` , `if (!(fp = fopen(…)))`)과 Python의 `:=` 는 참고로만 알립니다.
대입한 값을 비교 `(c = getchar()) != EOF`	정상 관용구입니다. 입력값은 `getchar()` 의 값으로 주고, 안쪽 조건문의 c도 그 값으로 만듭니다 (참고).

증감
`n-- > 0`
`i++ < 10`
	바꾸기 전 값으로 입력값을 만듭니다 (참고).

비교를 이어 쓴 식 `a < b < c`	앞 비교의 결과(0 / 1)를 다음 값과 비교하는 식이라 입력값을 `?` 로 두고 경고합니다. Python의 `a < b < c` 는 두 조건으로 바꿉니다.

위쪽 코드의 고정 값 : 같은 함수에서 조건문보다 앞에 상수가 들어간 변수가 있으면(`b = 0;` 뒤의 `if (a = b)`), 그 조건문이 늘 같은 결과로 보이거나 안쪽 조건문에 도달할 수 없어 보인다고 경고하고, 조건문 행과 분석 로그에 그 경로를 적습니다.

`※ 고정 값 경로 : Line 3 b = 0 → a=b = F → 조건문 결과 항상 F`
`※ 고정 값 경로 : Line 3 b = 0 → Line 4 if = F → 이 조건문 도달 불가`

케이스(표 · 입력값)는 그대로 만들며, 결과는 직접 확인하세요. C · C++ · Java · JavaScript · Python 파일에서 보며, 확실하지 않으면 경고하지 않습니다 : 앞쪽의 if · 반복 · 블록 안에서 바뀌는 변수, 주소를 넘긴 변수(`&x`), 참조, volatile, 안쪽 함수 · 람다가 바꾸는 변수, 함수 호출 뒤의 함수 밖 변수, `#if` 구간 안의 문장, `switch` · `goto` 가 있는 코드, Python의 `try` · `match` 안의 조건문과 `exec` · `eval` 을 쓰는 함수.

4. 결과 읽기

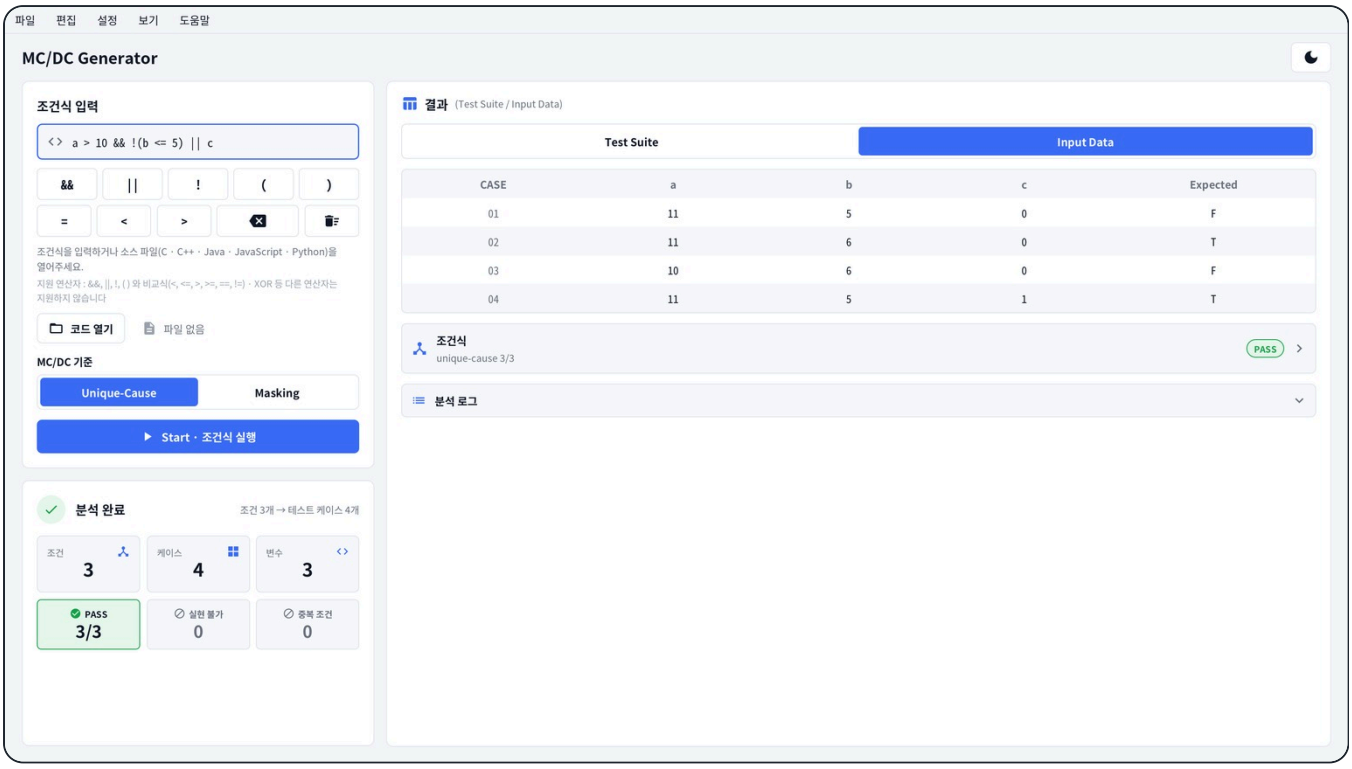
분석 완료 화면

- **조건 / 케이스 / 변수** - 조건문 하나를 분석했을 때, **조건문 / 생성 케이스** - 개별분석, **통합 케이스 / 개별 케이스** - 통합분석.
- **PASS** - 모든 조건의 독립 영향이 증명됨 (unique-cause. masking 으로만 보일 수 있었던 조건이 있으면 조건문 행에 `masking · unique-cause k/N`로 표시); **FAIL** - 표시된 조건의 독립 영향을 증명하지 못함; **분석 불가** - 분석하거나 검증하지 못한 조건문.
- **실현 불가** - 같은 변수에 대한 조건들이 서로 모순이라(예 : `x > 10 && x < 5`) 도달할 수 없는 케이스가 있음. **중복 조건** - 같은 조건이 두 번 이상 나와 N+1 개의 unique-cause 케이스가 보장되지 않음. 자세한 내용은 **확인 필요** 아래에 나옵니다.

결과 표

- **Test Suite** - 조건마다 한 열씩 `T` / `F`, 그리고 그 케이스에서의 조건문 결과 **Expected**.
- **Input Data** - 변수마다 한 열씩, 필요한 진리값을 만드는 입력값의 예(비교식인 경우). `?`는 값을 구하지 못한 경우 - 지원하지 않는 비교식, 비교를 이어 쓴 식(`a < b < c`), 상수를 읽을 수 없는 비교식일 때이며 경고가 함께 나옵니다. 숫자끼리의 비교(`3 > 2`)는 상수 조건으로 다룹니다 (3절).
- 케이스 번호는 `01 … N+1`. 실현 불가 케이스는 빨간 글자와 `※`로, `-`는 그 케이스가 쓰지 않는 변수 (아무 값)를 뜻합니다.
- **중첩 조건문** - 조건문이 다른 조건문 안에 있으면(`if` 안의 `if` , `elif` , 반복문 몸체) 입력값은 바깥 조건문의 도달 조건도 함께 만족합니다. 조건문 행에 그 경로가 표시되고(`도달 조건 : Line 3 if = T · Line 5 if = F`), 경로에만 필요한 변수는 `†`가 붙은 옅은 열로 나옵니다. 경로의 값은 바로 바깥 조건문의 테스트 케이스 가운데 요구 결과를 내는 것을 그대로 쓰므로, 통합분석에서는 안쪽 케이스가 그 바깥 케이스 행에 합쳐집니다. 경로와 함께 만족시킬 수 없는 케이스는 **도달 불가**로 표시됩니다 - 이 프로그램에서는 실행할 수 없는 케이스이므로 MC/DC Check 에도 반영됩니다.
- **MC/DC 기준** - 입력 카드의 토글로 Unique-Cause와 Masking을 고릅니다. Unique-Cause는 검사하는 조건 하나만 바꾸고 나머지 조건은 모두 같은 두 케이스로 독립 영향을 보입니다 (Robin's Rule, 서로 다른 조건 N개에 케이스 N+1개). Masking은 다른 조건이 달라도 그 차이가 연산자에 가려져 결과에 닿지 않으면 같은 쌍으로 인정하므로, `(a && b) || (a && c)`처럼 같은 조건이 두 번 이상 나오는 식이나 같은 변수의 조건이 충돌해 Unique-Cause 로는 실현 불가 케이스가 생기는 식에서도 실행 가능한 케이스만으로 모든 조건을 보입니다. 조건이 한 번씩만 나오는 식에서는 두 기준의 결과가 같습니다. 검증 줄은 `MC/DC Check : PASS (masking n/n · unique-cause k)`로 표시되고, 어떤 케이스로도 결과를 바꿀 수 없는 조건(`a && (a || b)`의 b)은

FAIL로 표시합니다. 화면 표와 분석 로그·저장 파일의 열은 입력한 조건식의 순서이고, 정렬된 식은 "정렬 :" 줄에만 보입니다.



케이스마다 바로 쓸 수 있는 입력값

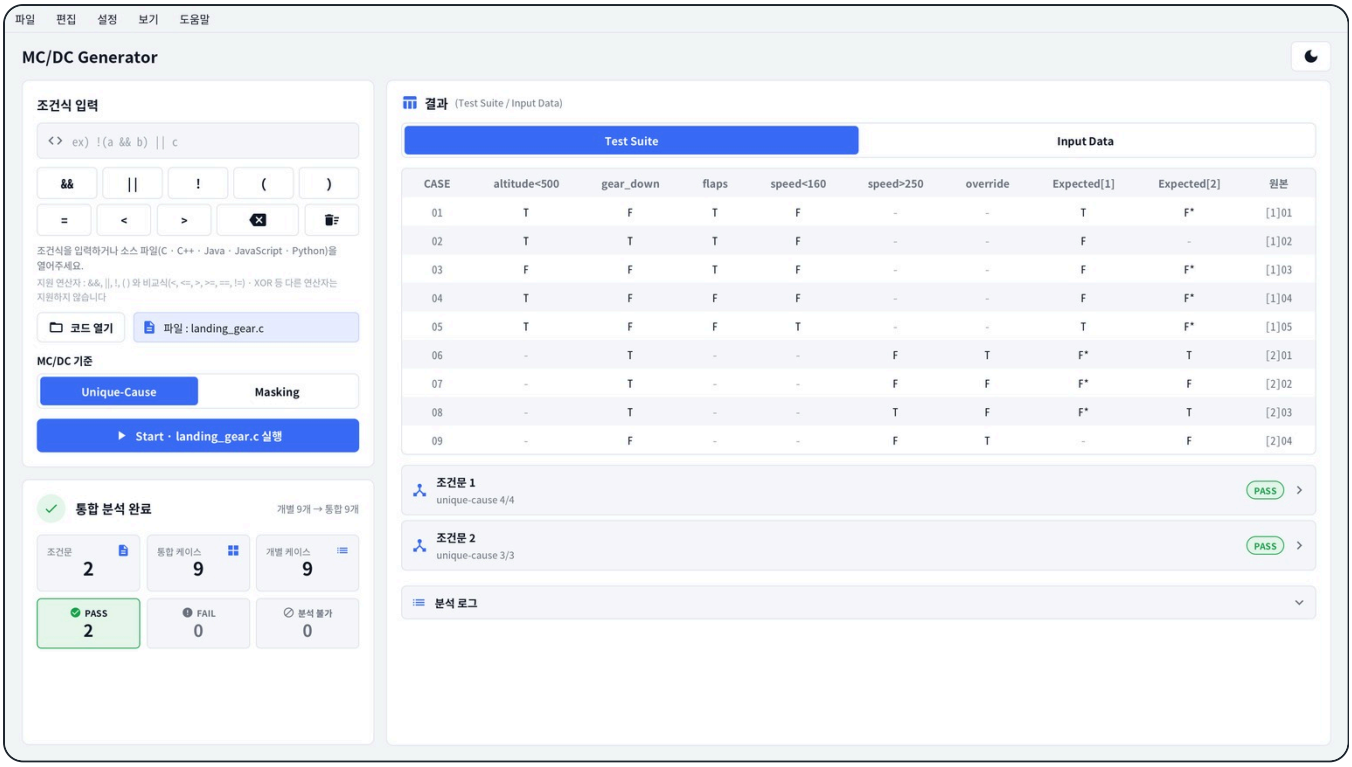
조건문 행과 분석 로그

표 아래에는 조건문마다 소스 줄 번호, 정리된 식, 검증 결과(`unique-cause k/N`, `masking` …)가 한 행씩 나옵니다. **분석 로그**를 펼치면 전체 텍스트 리포트가 보이며, **Save txt**로 저장되는 내용과 같습니다. **편집 ▸ 분석 로그 복사**(Ctrl+Shift+C)로 클립보드에 복사할 수 있습니다.

---

5. 통합분석

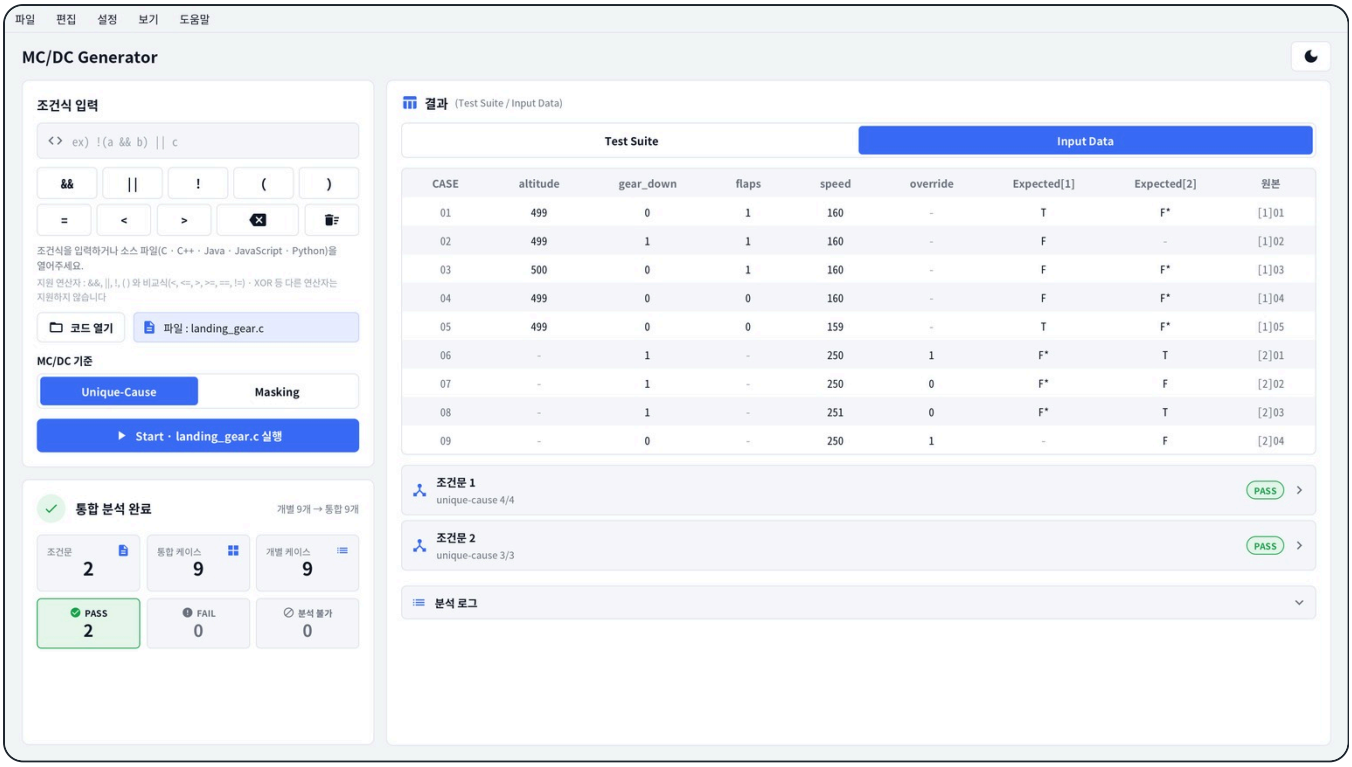
각 조건문의 N+1 케이스를 변수 합집합 기준으로 한 표에 넣고, 공유하는 변수의 값이 충돌하지 않는 케이스끼리 하나로 합칩니다. **원본** 열은 합쳐진 케이스가 어느 조건문에서 왔는지 보여 줍니다. **Expected[n]**은 n 번 조건문의 결과이며, `*`가 붙은 값은 그 조건문에서 나온 케이스는 아니지만 그 행의 값으로 결과가 정해져 계산한 기대 결과입니다(바깥 조건문은 경로 값만으로도 정해집니다). 합친 뒤의 케이스 수에 N+1은 보장되지 않지만, 각 조건문의 케이스가 모두 들어 있으므로 조건문별 MC/DC는 유지됩니다. 서로 배타적인 가지(`if`와 그 `else`)의 케이스는 합쳐지지 않으며, 그 행에서 실행되지 않는 조건문의 Expected는 `-` 입니다. 표의 `-`는 그 행이 다루지 않는 조건·변수(아무 값이나 가능)입니다. 저장 파일에서는 **설정 ▸ 저장할 때 빈 칸(-)**으로 이 칸을 자동으로 채울 수 있습니다 (6절).



통합 분석: 결정문 2개 → 테스트 케이스 9개

6. 저장

메뉴	내용
파일 ▸ Save txt… (Ctrl+S)	텍스트 리포트(분석 로그). 표의 열이 맞도록 Consolas · D2Coding · Courier New 같은 고정폭 글꼴로 여세요.
파일 ▸ Save csv… (Ctrl+Shift+S)	입력 데이터 CSV : `TC` , 변수별 열, `Expected` , `Note` (`infeasible`). BOM이 붙은 UTF-8 이라 Excel 에서 바로 열립니다.



.txt · .csv로 저장해 테스트 도구에 바로

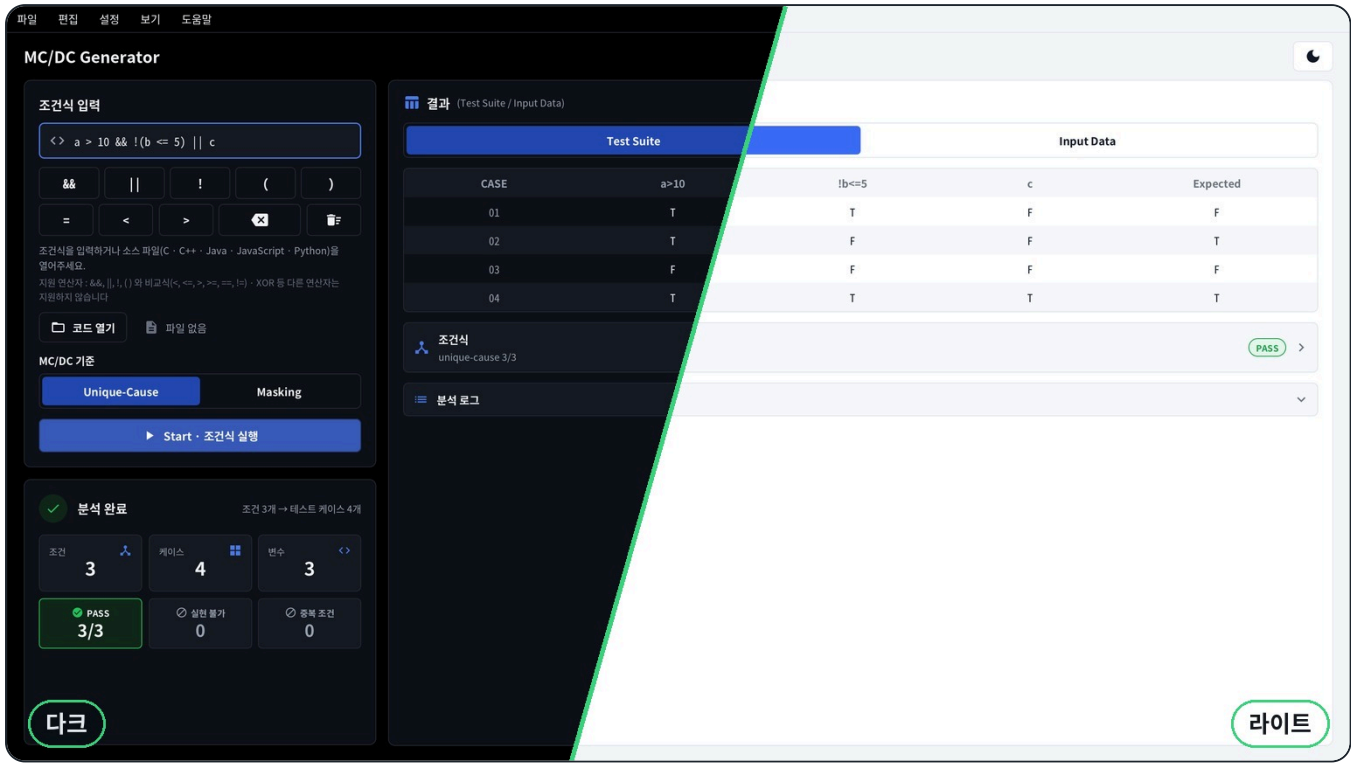
설정 ▸ 저장할 때 빈 칸(-) - 통합분석 결과의 `-`(그 행이 다루지 않는 조건·변수)와 경로 변수 열의 `-`를 저장 파일에서 어떻게 할지 고릅니다 : **그대로 둠 (-)**(기본값) · **자동으로 채움 (조건 T)** · **자동으로 채움 (조건 F)**. 자동으로 채우면 빈 조건 칸은 고른 값(T 또는 F)이 되고, Input Data 에는 그 조건이 그 값이 되도록 하는 입력값이 들어갑니다 - 1 / 0 만이 아니라 조건에 맞는 값입니다. 예를 들어 `a > 50` 이면 T 일 때 a = 51, F 일 때 a = 50, 불리언 변수 `b` 면 1 / 0. 이미 값이 있는 변수로 정해지는 조건은 그 값대로 계산합니다 (a = 51이 있으면 `a > 50` 은 T). 조건이 모두 정해진 조건문의 Expected도 `*`를 붙여 계산해 채우며, 도달하지 않는 조건문의 Expected와 해석할 수 없는 조건은 그대로 `-` 입니다. 화면의 표는 바뀌지 않고, txt 파일 머리에 채운 방식을 적은 줄이 붙습니다.

---

7. 설정과 단축키

- **보기 ▸ 테마** - Windows 설정 따름 · 다크 · 라이트. 제목 줄의 해 / 달 버튼으로 바로 바꿀 수도 있습니다.
- **보기 ▸ 언어** - 시스템 언어 따름, 또는 English · 한국어 · 日本語 · 简体中文 · 繁體中文 · Deutsch · Français · Español · Português (Brasil) · Русский · العربية.
- **설정 ▸ 저장할 때 빈 칸(-)** - 저장 파일의 `-` 칸을 그대로 둘지, 자동으로 채울지 (조건을 T 또는 F로, Input Data는 그에 맞는 값으로) (6절).
- **파일 ▸ 끝내기**와 창의 닫기 버튼은 한 번 더 묻습니다. 생성 결과를 txt / csv로 저장하지 않았으면 "생성 결과를 저장하지 않고 종료하시겠습니까?" 로 알려 줍니다.

- **설정 ▸ 조건문 상세** / **설정 ▸ 분석 로그** - 실행 뒤 조건문 행의 상세와 분석 로그를 **항상 펼침**으로 둘지 **항상 접음**으로 둘지 (기본값은 접음). 행 머리글과 로그 머리글을 눌러 그때그때 펼치고 접을 수도 있습니다.
- 창 크기와 위치, 테마, 언어, MC/DC 기준, 빈 칸·펼침 설정은 다음 실행 때 그대로 유지됩니다.



다크 · 라이트 테마

단축키	동작
Enter	조건식 실행 (조건식 입력 칸에서)
Ctrl+O	소스 파일 열기
Ctrl+S / Ctrl+Shift+S	Save txt / Save csv
Ctrl+L	입력과 결과 지우기
Ctrl+Shift+C	분석 로그 복사
Shift+휠	넓은 표를 가로로 스크롤
F1	매뉴얼
Esc	대화상자 닫기

8. 참고와 제한

- N+1 보장은 단일 불리언 식에 대한 것입니다. 중복 조건이나 강하게 결합된 조건이 있어도 스위트는 만들어지며, 독립 영향을 증명하지 못한 조건을 알려 줍니다. Masking 기준을 고르면 같은 조건이 두 번 이상 나오는 식도 실행 가능한 케이스만으로 모든 조건을 보입니다.
- 입력값은 예시입니다. 같은 진리값을 만드는 값이면 무엇이든 유효합니다.
- 같은 함수 호출이 두 번 나오는 식(`getc() == 'a' || getc() == 'b'`)은 호출마다 값이 달라질 수 있지만 같은 값으로 봅니다. 그런 조건은 입력값을 직접 확인하세요.
- 알고리즘과 엔진은 안드로이드 앱과 동일하므로 두 버전의 결과는 같습니다.

9. 문의

CoverageLab · astesia@me.com
홈페이지 : coveragelab.kr