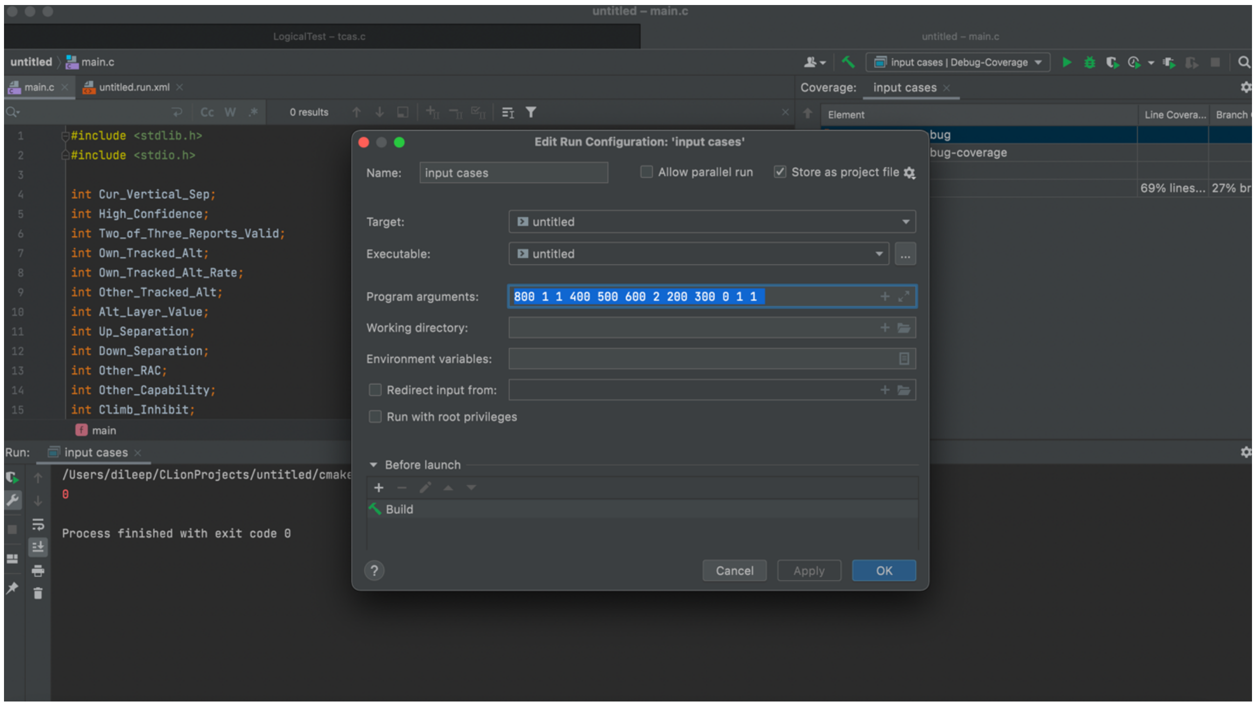
1. Predicate Coverage

Predicate coverage is also called condition coverage. It is
Considered for Boolean expressions. Specifically limited to
Conditional statements. Predicate consists of clauses and
It ensures whether all the Boolean expressions have been
Evaluated to true or false. The condition coverage does not
Necessarily imply branch coverage.

Challenges:

There were certain conditions which were nested i.e.,
Few predicates were dependent on the result of the parent
Branch/ predicate. In order to trace back to these conditions
We need to carefully observe and track back the path. If the
Programs are huge and functionality is bigger than the length
Of the code might be huge and hence tracing and identifying
Those dependent conditions might be time consuming.

Results:

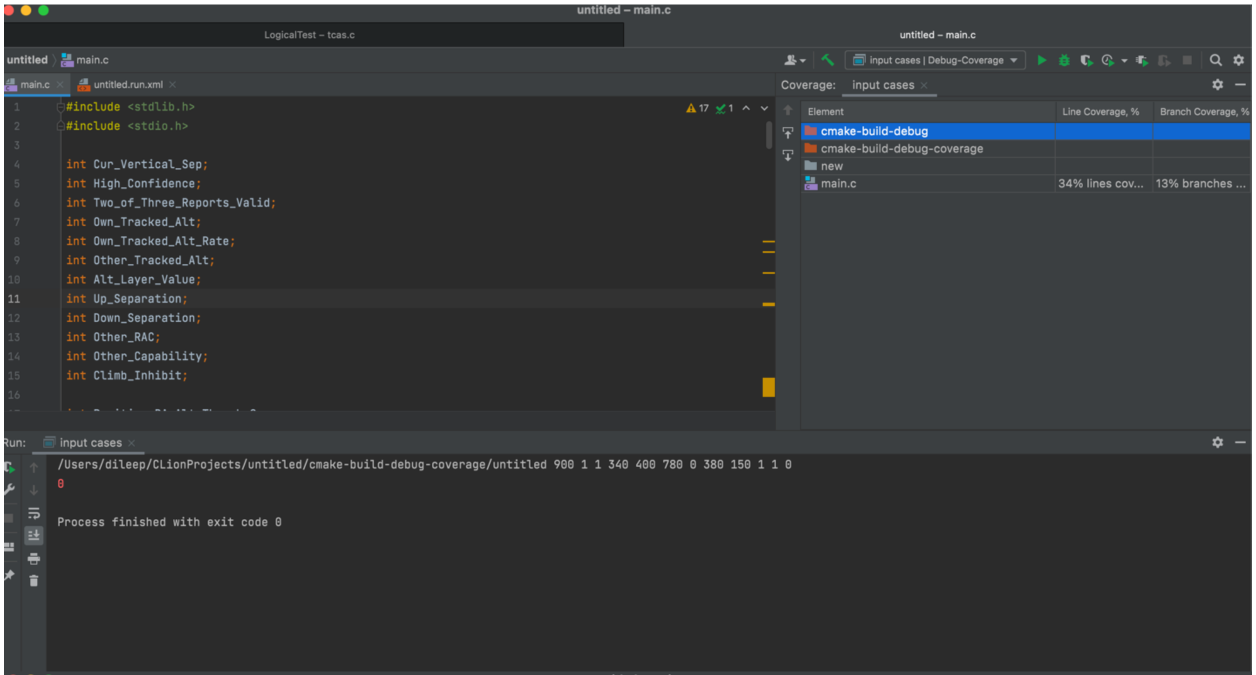


```
#include <stdlib.h>
#include <stdio.h>

int Cur_Vertical_Sep;
int High_Confidence;
int Two_of_Three_Reports_Valid;
int Own_Tracked_Alt;
int Own_Tracked_Alt_Rate;
int Other_Tracked_Alt;
int Alt_Layer_Value;
int Up_Separation;
int Down_Separation;
int Other_RAC;
int Other_Capability;
int Climb_Inhibit;
```

```
/Users/dileep/CLionProjects/untitled/cmake
0

Process finished with exit code 0
```

Edit Run Configuration: 'input cases'

Name: input cases
Allow parallel run: ☐
Store as project file: ✓
Target: untitled
Executable: untitled
Program arguments: 800 1 1 400 500 600 2 200 300 0 1 1
Working directory:
Environment variables:
☐ Redirect input from:
☐ Run with root privileges
Before launch
Build



untitled – main.c

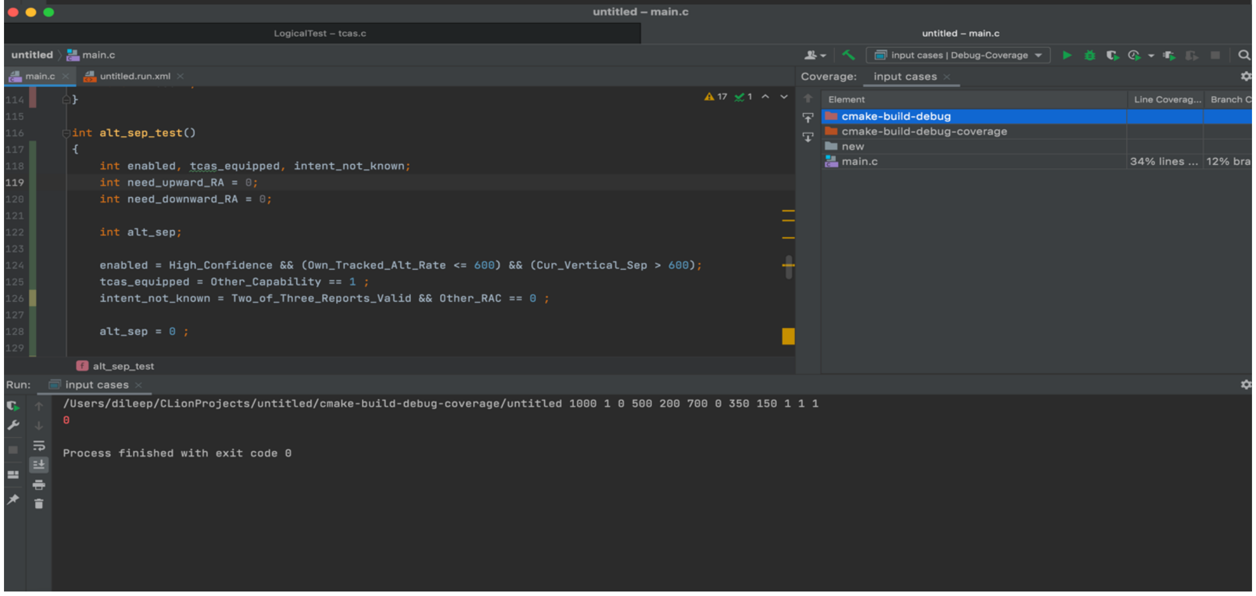
Coverage: input cases

Element	Line Coverage, %	Branch Coverage, %
cmake-build-debug		
cmake-build-debug-coverage		
new		
main.c	34% lines cov...	13% branches ...

```c
#include <stdlib.h>
#include <stdio.h>

int Cur_Vertical_Sep;
int High_Confidence;
int Two_of_Three_Reports_Valid;
int Own_Tracked_Alt;
int Own_Tracked_Alt_Rate;
int Other_Tracked_Alt;
int Alt_Layer_Value;
int Up_Separation;
int Down_Separation;
int Other_RAC;
int Other_Capability;
int Climb_Inhibit;
```

Run: input cases

```
/Users/dileep/CLionProjects/untitled/cmake-build-debug-coverage/untitled 900 1 1 340 400 780 0 380 150 1 1 0
0

Process finished with exit code 0
```



untitled – main.c

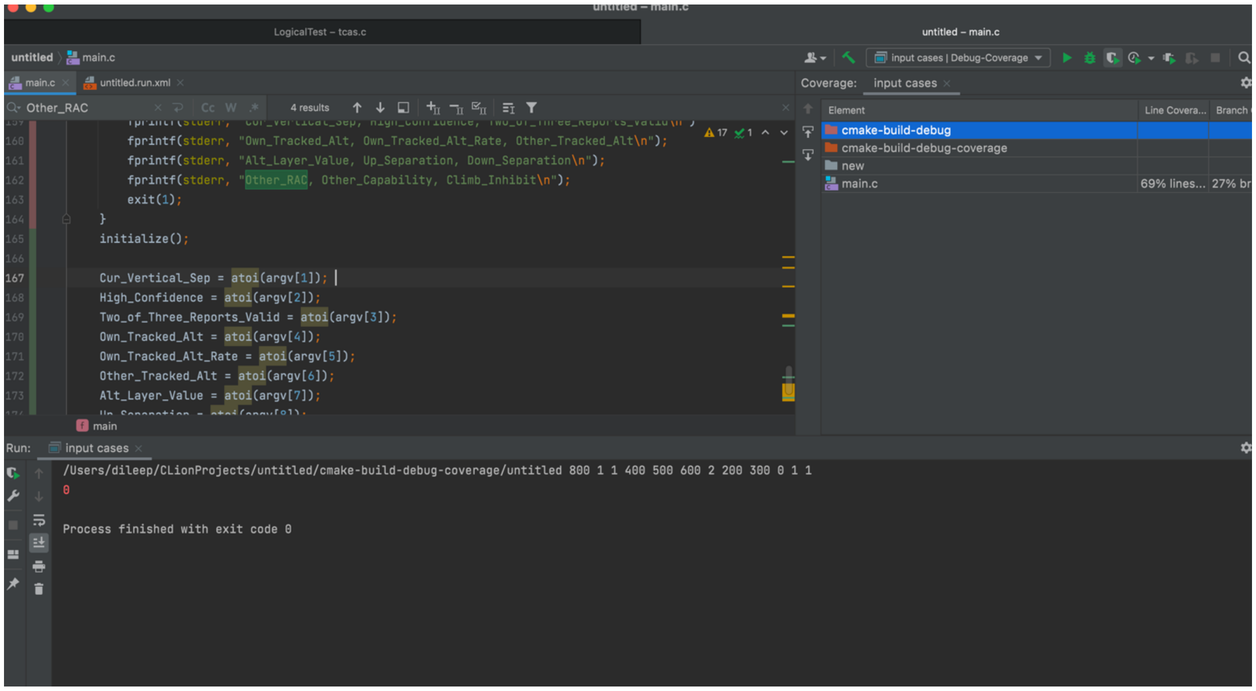
Coverage: input cases

Element	Line Coverag...	Branch C
cmake-build-debug		
cmake-build-debug-coverage		
new		
main.c	34% lines ...	12% bra

```c
}

int alt_sep_test()
{
    int enabled, tcas_equipped, intent_not_known;
    int need_upward_RA = 0;
    int need_downward_RA = 0;

    int alt_sep;

    enabled = High_Confidence && (Own_Tracked_Alt_Rate <= 600) && (Cur_Vertical_Sep > 600);
    tcas_equipped = Other_Capability == 1 ;
    intent_not_known = Two_of_Three_Reports_Valid && Other_RAC == 0 ;

    alt_sep = 0 ;
```

Run: input cases

```
/Users/dileep/CLionProjects/untitled/cmake-build-debug-coverage/untitled 1000 1 0 500 200 700 0 350 150 1 1 1
0

Process finished with exit code 0
```



```c
        fprintf(stderr, "Own_Tracked_Alt, Own_Tracked_Alt_Rate, Other_Tracked_Alt\n");
        fprintf(stderr, "Alt_Layer_Value, Up_Separation, Down_Separation\n");
        fprintf(stderr, "Other_RAC, Other_Capability, Climb_Inhibit\n");
        exit(1);
    }
    initialize();

    Cur_Vertical_Sep = atoi(argv[1]);
    High_Confidence = atoi(argv[2]);
    Two_of_Three_Reports_Valid = atoi(argv[3]);
    Own_Tracked_Alt = atoi(argv[4]);
    Own_Tracked_Alt_Rate = atoi(argv[5]);
    Other_Tracked_Alt = atoi(argv[6]);
    Alt_Layer_Value = atoi(argv[7]);
```

Coverage: input cases

Element	Line Covera...	Branch
cmake-build-debug		
cmake-build-debug-coverage		
new		
main.c	69% lines...	27% br

Run: input cases

```
/Users/dileep/CLionProjects/untitled/cmake-build-debug-coverage/untitled 800 1 1 400 500 600 2 200 300 0 1 1
0

Process finished with exit code 0
```

2. Active Clause Coverage (Modified condition/ Decision Coverage)

This is a form of modified condition/ Decision coverage, where we
Check for every condition in the code has taken all possible output
Values i.e., true or false. Now each condition is tested such that the
Decision outcome is independent. We choose a condition such that
It will affect the outcome; we do not disturb the other conditions and
Alter this chosen condition and check if affects the predicate.
It is obvious that this is a optimized method to find the condition coverage.
We break down and select the cases only which satisfies above condition.
But we cannot guarantee the total 100% code coverage since there might
Be some masked conditions and we might not have 100% confidence
About the code coverage.

Challenges:

There might be some conditional statements which is difficult
To break down into further clauses and identify possible conditions.
Thought this is a better and optimized method compared to condition/predicate
Coverage but, It is not always possible to determine the number of test cases.
The number of test cases required for complex conditions as it requires enough
Test cases to verify every condition can affect the result of its encompassing decision.