# Cases besides functions in C++

In addition to functions changes in C++ source code, we are also able to specify the following cases.

|  |  |  |
| --- | --- | --- |
| Case | Description | example |
| Macro Definition | Macro defined in the source files of the target project, can be divided into Object style and Function style. | #define PI 3.14 //object  #define P(x) x //function |
| Class Definition | Class defined in the source file. | Class A{}; |
| Namespace Definition | Namespace defined in the source file. |  |
| Other Simple Definition | Common definitions such as global variable, class field members etc.  We can now detect whether a simple definition is a field member definition. | int a = 0;  ClassFoo foo; |
| Problem | Code that cannot be recognized, such as macro whose definition cannot be found, and common syntax errors. | TEST(…){} //as we encountered in gtest  int b //missing “;”  int //meaningless |

The current plan to deal with these cases is when we find these cases have been changed, we rerun all test cases.

# Simplifying Java Workflow

Currently, we managed to simplify the Java workflow to only 3 commands, indicating 2 phases of the workflow:

|  |  |  |
| --- | --- | --- |
| Phase | Command | Description |
| Prerequisite | N/A | 1. Setting IUT\_HOME folder -> $HOME/.iut (which contains files that are required to run iut. 2. Modify the “build.gradle” file of the target project |
| Phase1:  Preprocessing | *iut init <projectName>* | Initialize db file and several settings. |
| *iut preproc <projectName>* | instrumentation -> run testcases -> update database |
| Phase2:  Incremental Testing | *iut increm <projectName> <oldVersionPath> <newVersionPath>* | diff to find changed methods -> query for affected testcases -> run affected testcases |