Directed Research Report:

Team Objective-C:

Ankit Maheshwari

Chinmay Sawarkar

Harsh Desai

Sriranjani Babu

Objective-C:

In this Directed research project, we tried to implement and test the Objective-C module of UCC. The module will help UCC project by adding one more language to its arsenal. Objective-C is a reflective, object-oriented programming language. It is used primarily on Apple's Mac OS X and iOS. Objective-C is a thin layer on top of C, and moreover is a *strict superset* of C; it is possible to compile any C program with an Objective-C compiler, and to freely include C code within an Objective-C class.

The project for developing the Objective-C module for UCC started in the previous semester and we were to continue working on the same to finish the project. We started off with integrating the code of Objective-C to UCC which demanded a few changes to the underlying files in UCC. This required for a bit of detailed study of the flow of the UCC code so that we can integrate the Objective-C language to the code. After going through the code, we did the appropriate changes to include Objective-C into UCC.

The second phase of the project started with initial testing of the existing code so that we can find errors or unimplemented modules and to correct them or implement those. The initial testing insinuated that the code needed very little to negligible modification and it seemed to work perfectly. We did rigorous testing of the code so that we could find the shortcomings of the code and improve upon those. The main testing involved in testing each and every functionality of the code based on the UCC demands and UCC functionality needed out of the code, ie, we tested that the results for physical/logical line numbers are coming properly for any type of file supplied. Also, the keywords were getting detected properly for the language.

As suggested by the testing of the existing code base, it was clear that the implementation of the code was complete and just needed to be tested properly. Thus, the third phase involved more and more rigorous testing of the code.

The main modules of code that were needed to be tested were counting and differencing. The counting function tries to identify the length of code in terms of number of physical lines of code and logical lines of code, while the differencing function aims at trying to figure out the differences between two files involved in the differencing function.

We also participated in writing test cases for the Matlab team. The Matlab team was lagging behind and as we had time at spare, we wrote the test cases as demanded from us by the Matlab team. The tests mainly focused on testing the keywords that are present in various of the Matlab modules.

The major categories of tests that we wrote are:

Counting Test cases:

The UCC project is aimed at segregating and identifying the number of logical as compared to number of physical lines in a code base. The counting test cases were mainly aimed at testing various scenarios in which the number of logical lines might or might not vary with the physical lines of code.

Differencing test cases:

These test cases were aimed at testing the differencing module of Objective-C. The major test cases involved in this were testing whether the module is able to correctly identify the code difference in the files. The differencing module compare source files by specifying the directories where the files are located. It generates how the two files differ in code on the basis of “number of new lines”, “number of lines deleted”, “number of lines modified”. This allows us to get an idea as to how the files differ from each other and is highly useful to compare two different versions of the same file.

Stress Testing:

The stress testing focused on testing the code to its limits. The code mainly consisted of performing the count function on a code with 10,000+ lines which are normal in any development environment.

Keywords testing:

This test aimed at testing all the possible keywords that are present in the Objective-C language. This helps us to check if all the keywords are being identified by the code or not.

Note:

The Test files folder is named in accordance with the test case id’s used in the TestCase document. This helps in correlating the test files used for a particular test. For example , The test with Test case id “OBJC\_CommandSwitches\_1” has a folder with that name which contains the test files corresponding to that test.