CSE-4321 Software Testing Final Project - Summary and Discussions

Maximum Effort: Elliot Mai & Benjamin Niccum

December 2, 2024

Summary and Discussions

Summary of Findings

The analysis and testing of the `Printtokens` program identified a total of **13 faults** that were addressed and corrected. These faults primarily stemmed from issues related to token classification, error handling, and program robustness. The corrections were thoroughly tested using unit and program-level tests to ensure functionality and reliability across diverse input scenarios. Below are the key findings:

- **Token Classification Errors**: - Several faults involved incorrect classification of tokens, such as string constants, character constants, and comments. - For example, ID numbers for string and comment tokens were swapped in the `get_token` method, and the `token_type` method incorrectly prioritized special symbols over string constants.
- **Validation and Boundary Issues**: - Methods like `is_char_constant`, `is_num_constant`, and `is_str_constant` lacked proper validation checks, leading to acceptance of malformed tokens or runtime errors due to out-of-bounds access.
- **Error Handling and Usability**: - The `open_character_stream` and `main` methods had insufficient error handling, providing vague feedback or terminating the program unnecessarily with `System.exit`. - A usability improvement was introduced to guide users in providing input when filenames were missing.

- **Code Redundancy and Maintainability**: - Unnecessary variables and duplicate logic were removed in methods like `get_token` and `print_spec_symbol`, simplifying the codebase while retaining functionality.

Discussions on the Testing Process

The testing process involved a comprehensive approach, covering both unit-level and program-level testing. Key observations include:

- **Effectiveness of Unit Tests**: - The unit tests successfully validated individual methods by isolating functionality and identifying specific faults, such as flipped ID numbers in `get_token` and incorrect validation in `is_char_constant`.

- **Program-Level Testing for Integration**: - The program-level tests ensured that all methods worked together seamlessly when processing diverse token streams. This included testing for edge cases like unclosed strings, invalid identifiers, and malformed numeric constants.

- **Use of Edge Coverage**: - Test paths were specifically designed to achieve edge coverage, ensuring that every control flow edge in the program was tested. This approach proved critical in identifying faults related to boundary conditions and EOF handling.

- **Enhancements for Future Testing**: - Automating the test cases could improve testing efficiency and consistency. Additionally, performance testing with large-scale inputs is recommended to evaluate scalability and runtime behavior.

Impacts of the Corrections

The corrections made to the `Printtokens` program resulted in significant improvements:

- **Accuracy**: The corrected logic ensures accurate classification of all token types, meeting the functional requirements of the program.

- **Robustness**: Enhanced validation and error handling make the program more reliable, especially when processing edge cases and malformed inputs.

- **Usability**: Improvements in error messaging and user guidance provide a better experience for both developers and end-users.
- **Maintainability**: Simplified logic and removal of redundant code contribute to a more maintainable codebase, reducing potential future errors.

Challenges and Lessons Learned

The analysis and correction of the `Printtokens` program revealed several challenges:

- **Complex Interactions**: Some faults, such as the flipped IDs in `get_token`, affected multiple methods, requiring careful debugging to locate the fault.
- **Importance of Validation**: The absence of proper boundary checks in methods like `is_num_constant` and `is_str_constant` emphasized the need for rigorous input validation.
- **Value of Comprehensive Testing**: The combination of unit and program-level tests was essential in uncovering both isolated faults and integration issues.

Conclusion

The systematic identification, correction, and testing of faults in the `Printtokens` program demonstrate the importance of rigorous software testing. By addressing 13 distinct faults, the revised version now operates reliably across diverse scenarios, ensuring accurate token classification and robust error handling. The lessons learned underscore the value of thorough validation, comprehensive testing, and maintainable code practices for achieving high-quality software.