Методика тестирования

Дорофеев Андрей

22 апреля 2014 г.

Содержание

1 Введение

1.1 Наименование системы

Полное наименование программного продукта: «Интерпретатор командной оболочки для встраиваемых систем на базе микросхем семейства AVR». Продукт создается в рамках курсовой работы по дисциплине “проектирование трансляторов”.

Производится тестирование версии на ПК. В данном документе не указаны дополнительные действия по тестированию компонентов системы на целевой аппаратной базе.

1.2 Назначение системы

Система предназначена для записи на микросхемы и последующего использования для ускорения разработки и отладки аппаратной части встраиваемой системы, создания единого протокола обмена данными между микроконтроллером и терминальным управляющим устройством.

2 Ссылки на другие документы

3 Подготовка к тестированию

3.1 Проверка запуска

С помощью данного теста устанавливается общая работоспособность системы.

3.1.1 Подготовка аппаратной части

3.1.2 Подготовка программной части

1. Написание главного модуля для тестирования. Модуль должен осуществлять инициализацию компонентов тестируемой системы. После этого необходимо совершить деинициализацию и выход со статусом 0;

2. сборка исполняемого файла.

3.2 Проверка лексического анализатора

С помощью данного теста устанавливается работоспособность лексического анализатора: пропуск валидных токенов, отсеивание с сообщение об ошибке при обнаружении неверного токена.

3.2.1 Подготовка аппаратной части

3.2.2 Подготовка программной части

1. Написание главного модуля, осуществляющего инициализацию тестируемой системы, анализ с помощью системы первого аргумента, представляемого при запуске и деинициализацию системы;

2. сборка исполняемого файла;

3. написание текстового файла, содержащего входные строки;

4. написание скрипта, осуществляющего запуск собранного исполняемого файла с каждой строкой текстового файла с входными строками;

3.3 Проверка синтаксического анализатора

С помощью данного теста устанавливается работоспособность синтаксического анализатора: способность распознавать принадлежность строки специализированному языку.

3.3.1 Подготовка аппаратной части

3.3.2 Подготовка программной части

1. Написание главного модуля, осуществляющего инициализацию тестируемой системы, анализ с помощью системы первого аргумента,

представляемого при запуске и деинициализацию системы;

2. сборка исполняемого файла;
3. написание текстового файла, содержащего входные строки;
4. написание скрипта, осуществляющего запуск собранного исполняемого файла с каждой строкой текстового файла с входными строками;

3.4 Проверка модуля исполнения команд

С помощью данного теста устанавливается работоспособность модуля распознавания команд: корректность ведения таблиц переменных и функций, правильность работы с переменными и пользовательскими функциями.

3.4.1 Подготовка аппаратной части

3.4.2 Подготовка программной части

1. Написание главного модуля, осуществляющего инициализацию тестируемой системы, проведение доступных операций над таблицами переменных и функций напрямую и деинициализацию системы;
2. Написание главного модуля 1, осуществляющего инициализацию тестируемой системы, анализ с помощью системы первого аргумента, представляемого при запуске и деинициализацию системы;
3. Написание главного модуля 2, осуществляющего инициализацию тестируемой системы, добавление собственной функции в таблицу, анализ с помощью системы первого аргумента, представляемого при запуске и деинициализацию системы;
4. сборка исполняемых файлов 1 и 2;
5. написание текстового файла, содержащего входные строки;
6. написание скриптов, осуществляющего запуск собранных исполняемых файлов с каждой строкой текстовых файлов с входными строками:
    - для исп. файла 1, вызов несущ. функции
    - для исп. файла 1, вызов сущ. функции
    - для исп. файла 2, вызов добавленной функции

4 Описание тестов

4.1 Проверка запуска

4.1.1 Инициализация и деинициализация системы

Тестируемые требования	Общая работоспособность.
Необходимые предпосылки	
Входы	
Ожидаемые результаты	Программа заканчивает выполнение без сообщений об ошибках со статусом 0.
Критерии соответствия	Статус при выходе 0.
Процедура тестирования	Запуск исполняемого файла. Проверка статуса завершения.
Ограничения	

4.2 Проверка лексического анализатора

4.2.1 Корректная входная строка

Тестируемые требования	“Система должна предоставлять возможность интерпретировать текстовые команды в ASCII-кодировке”
Необходимые предпосылки	
Входы	текстовый файл, содержащий корректные с лексической точки зрения строки, содержащие все известные токены
Ожидаемые результаты	Программа завершает работу без сообщений об ошибках и со статусом 0
Критерии соответствия	Статус выхода 0
Процедура тестирования	Запуск скрипта
Ограничения	

4.2.2 Некорректная входная строка

Тестируемые требования	“Система должна предоставлять возможность интерпретировать текстовые команды в ASCII-кодировке”
Необходимые предпосылки	
Входы	текстовый файл, содержащий некорректные с лексической точки зрения строки
Ожидаемые результаты	Программа завершает работу с сообщением об ошибке и указанием места во входной строке
Критерии соответствия	Статус выхода 1, сообщение об ошибке
Процедура тестирования	Запуск скрипта
Ограничения	

4.3 Проверка синтаксического анализатора

4.3.1 Корректная входная строка

Тестируемые требования	“Система должна предоставлять возможность интерпретировать текстовые команды в ASCII-кодировке”
Необходимые предпосылки	
Входы	текстовый файл, содержащий строки на специализированном языке. Строки в совокупности должны покрывать все синтаксические конструкции языка.
Ожидаемые результаты	Сообщений об ошибках не выводится. Статусы выхода 0
Критерии соответствия	Статусы выхода 0
Процедура тестирования	Запуск скрипта
Ограничения	

4.3.2 Некорректная входная строка 1

Пропуск разделителя (`;`).

Тестируемые требования	“Система должна предоставлять возможность интерпретировать текстовые команды в ASCII-кодировке”
Необходимые предпосылки	
Входы	текстовый файл, содержащий строки на специализированном языке. Строки содержат единственный тип ошибок — пропущена точка с запятой.
Ожидаемые результаты	Сообщения об ошибках. Статусы выхода -1
Критерии соответствия	Статусы выхода -1 Сообщения о неожиданном токене с указанием корректной позиции пропущенной точки с запятой.
Процедура тестирования	Запуск скрипта
Ограничения	

4.3.3 Некорректная входная строка 2

Неожиданные символы в `control`: `+,-,=,>,<,%,&,!,#`.

Тестируемые требования	“Система должна предоставлять возможность интерпретировать текстовые команды в ASCII-кодировке”
Необходимые предпосылки	
Входы	текстовый файл, содержащий строки на специализированном языке. Строки содержат единственный тип ошибок — перечисленные выше символы находятся в конструкции языка `control`.
Ожидаемые результаты	Сообщения об ошибках. Статусы выхода -1
Критерии соответствия	Статусы выхода -1 Сообщения о неожиданном токене с указанием корректной позиции с неверным токеном.
Процедура тестирования	Запуск скрипта
Ограничения	

4.3.4 Некорректная входная строка 3

Отсутствующий токен `end` в конце `if, while`.

Тестируемые требования	“Система должна предоставлять возможность интерпретировать текстовые команды в ASCII-кодировке”
Необходимые предпосылки	
Входы	текстовый файл, содержащий строки на специализированном языке. Строки содержат единственный тип ошибок — отсутствующий токен `end` в конце `if, while`
Ожидаемые результаты	Сообщения об ошибках. Статусы выхода -1
Критерии соответствия	Статусы выхода -1 Сообщения о неожиданном токене с указанием корректной позиции с неверным токеном.
Процедура тестирования	Запуск скрипта
Ограничения	

4.3.5 Некорректная входная строка 3

Отсутствующий токен `end` в конце `if, while`.

Тестируемые требования	“Система должна предоставлять возможность интерпретировать текстовые команды в ASCII-кодировке”
Необходимые предпосылки	
Входы	текстовый файл, содержащий строки на специализированном языке. Строки содержат единственный тип ошибок — отсутствующий токен `end` в конце `if, while`
Ожидаемые результаты	Сообщения об ошибках. Статусы выхода -1
Критерии соответствия	Статусы выхода -1 Сообщения о неожиданном токене с указанием корректной позиции с неверным токеном.
Процедура тестирования	Запуск скрипта
Ограничения	

4.3.6 Некорректная входная строка 4

Ошибки в рамках `eval`: `=` вместо `==`, незакрытая скобка, пропущенный операнд, пропущенный оператор.

Тестируемые требования	“Система должна предоставлять возможность интерпретировать текстовые команды в ASCII-кодировке”
Необходимые предпосылки	
Входы	текстовый файл, содержащий строки на специализированном языке. Строки содержат перечисленные выше ошибки
Ожидаемые результаты	Сообщения об ошибках. Статусы выхода -1
Критерии соответствия	Статусы выхода -1 Сообщения о неожиданном токене с указанием корректной позиции с неверным токеном.
Процедура тестирования	Запуск скрипта
Ограничения	

4.4 Проверка модуля исполнения команд

4.4.1 Корректность ведения таблиц переменных и функций

Тестируемые требования	“Система должна предоставлять возможность интерпретировать текстовые команды в ASCII-кодировке”
Необходимые предпосылки	
Входы	
Ожидаемые результаты	Статус завершения 0
Критерии соответствия	Статусы выхода 0
Процедура тестирования	Запуск исполняемого файла
Ограничения	

4.4.2 Корректность работы с переменными и функциями 1

Оператор присваивания.

Тестируемые требования	“Система должна предоставлять возможность интерпретировать текстовые команды в ASCII-кодировке”
Необходимые предпосылки	
Входы	текстовый файл с входными стоками, строки содержат синтаксически корректные командные последовательности, разное количество присваиваний: 1-3. Повторное присваивание. Каждый шаг должен сопровождаться командой `echo` для проверки результата.
Ожидаемые результаты	Статус завершения 0, вывод команды соответствует присвоениям
Критерии соответствия	Статусы выхода 0 корректные значения переменных
Процедура тестирования	Запуск исполняемого файла, проверка выведенных значений
Ограничения	

4.4.3 Корректность работы с переменными и функциями 2

Операторы разыменовывания.

Тестируемые требования	“Система должна предоставлять возможность интерпретировать текстовые команды в ASCII-кодировке”
Необходимые предпосылки	
Входы	текстовый файл с входными стоками, строки содержат синтаксически корректные командные последовательности: 1. разыменовывание необъявленной переменной 2. разыменовывание объявленной до этого переменной 3. два разыменовывания подряд 4. разыменовывание в строку в `eval` 5. разыменовывание в число в `eval` Разыменовывание происходит агрументом функции `echo`.
Ожидаемые результаты	Статус завершения 0 1. ошибка о необъявленной переменной с указанием на токен 2. выводится корректное значение 3. выводится корректное значение (два раза) 4. выводится корректное значение 5. выводится корректное значение
Критерии соответствия	Статусы выхода 0 корректные значения переменных
Процедура тестирования	Запуск исполняемого файла, проверка выведенных значений
Ограничения	

4.4.4 Корректность работы с переменными и функциями 3

Вызовы функций.

Тестируемые требования	“Система должна предоставлять возможность интерпретировать текстовые команды в ASCII-кодировке”
Необходимые предпосылки	
Входы	текстовый файл с входными стоками, строки содержат синтаксически корректные командные последовательности: 1. вызов функции, которой нет в таблице 2. вызов функции, которая есть в таблице по умолчанию (`echo`) 3. вызов собственной функции, которая заранее добавлена в таблицу
Ожидаемые результаты	Статус завершения 0 1. сообщение о неизвестной функции с указанием на соответствующий токен 2. корректный вывод аргументов 3. корректное выполнение новой функции
Критерии соответствия	1. сообщение об ошибке, статус -1 2. нет сообщений об ошибках, статус 0, выведенные значения соотв. агрументам 3. нет сообщений об ошибках, статус 0, произведены действия, предусмотренные функцией
Процедура тестирования	1. запуск скрипта 1 для исполняемого файла 1, проверка вывода, проверка статуса 2. запуск скрипта 2 для исполняемого файла 1, проверка вывода, проверка статуса 3. запуск скрипта 1 для исполняемого файла 2, проверка вывода, проверка статуса
Ограничения	

4.4.5 Корректность работы операторов в блоке `eval`

Проверка типов. Корректность выполнения.

Тестируемые требования	“Система должна предоставлять возможность интерпретировать текстовые команды в ASCII-кодировке”
Необходимые предпосылки	

Входы	текстовый файл с входными стоками, строки содержат синтаксически корректные командные последовательности: 1. `echo ( 5 == 'five');` 2. `echo ( 5 >= 'five');` 3. `echo ( 5 <= 'five');` 4. `echo ( 5 > 'five');` 5. `echo ( 5 < 'five');` 6. `echo ( 5 - 'five');` 7. `echo ( 5 + 'five');` 8. `echo ( 5 * 'five');` 9. `echo ( 5 % 'five');` 10. `echo (5 == 5);` 11. `echo (5 == 6);` 12. `echo ('str' == 'str');` 13. `echo ('str' == 'str1');` 14. `echo (5 != 5);` 15. `echo (5 != 6);` 16. `echo ('str' != 'str');` 17. `echo ('str' != 'str1');` 18. `echo (6 + 6);` 19. `echo (6 - 6);`

```
20. echo (6 * 6);
21. echo (5 % 6);
22. echo (6 / 2);
23. echo (6 > 2);
24. echo (6 > 6);
25. echo (6 < 6);
26. echo (6 < 2);
27. echo (6 >= 2);
28. echo (6 >= 6);
29. echo (6 <= 2);
30. echo (6 <= 6);
```

Ожидаемые результаты	Статус завершения 0
	1. `echo ( 5 == 'five');` → ошибка типов 2. `echo ( 5 >= 'five');` → ошибка типов 3. `echo ( 5 <= 'five');` → ошибка типов 4. `echo ( 5 > 'five');` → ошибка типов 5. `echo ( 5 < 'five');` → ошибка типов 6. `echo ( 5 - 'five');` → ошибка типов 7. `echo ( 5 + 'five');` → ошибка типов 8. `echo ( 5 * 'five');` → ошибка типов 9. `echo ( 5 % 'five');` → ошибка типов 10. `echo (5 == 5);` → «t» 11. `echo (5 == 6);` → «t» 12. `echo ('str' == 'str');` → «t» 13. `echo ('str' == 'str1');` → «t» 14. `echo (5 != 5);` → «f» 15. `echo (5 != 6);` → «t» 16. `echo ('str' != 'str');` → «f» 17. `echo ('str' != 'str1');` → «t» 18. `echo (6 + 6);` → «12» 19. `echo (6 - 6);` → «0»

	20. `echo (6 * 6);` → «36» 21. `echo (5 % 6);` → «5» 22. `echo (6 / 2);` → «3» 23. `echo (6 > 2);` → «t» 24. `echo (6 > 6);` → «f» 25. `echo (6 < 6);` → «f» 26. `echo (6 < 2);` → «f» 27. `echo (6 >= 2);` → «t» 28. `echo (6 >= 6);` → «t» 29. `echo (6 <= 2);` → «f» 30. `echo (6 <= 6);` → «t»
Критерии соответствия	• отсутствие ошибок, там, где типы соответствуют • наличие соотв. ошибок, там, где типы не соответствуют • соответствие вывода ожидаемому
Процедура тестирования	1. запуск скрипта 1 для исполняемого файла 1, проверка вывода, проверка статуса 2. запуск скрипта 2 для исполняемого файла 1, проверка вывода, проверка статуса 3. запуск скрипта 1 для исполняемого файла 2, проверка вывода, проверка статуса
Ограничения	