Здравствуйте уважаемые председатель и члены аттестационной комиссии.

Я хочу продемонстрировать вам мою дипломную работу на тему: «Создание библиотеки функций унификации получения входных параметров и систематизации выходных данных в средствах тестирования и диагностики программных средств и оборудования» для Радио-Технического Института им. Минца.

В ходе **комплексного** тестирования программных средств возникает необходимость интерпретации результатов множества тестов. Для решения задачи автоматизации запуска, сбора информации и интерпретации результатов тестирования необходимо привести интерфейсную часть всех тестирующих программ к единообразному виду.

Для данных целей предлагается использовать единую библиотеку с небольшим прикладным программным интерфейсом (API), исключающую возможность административного взаимонепонимания при реализации правил для обработки входных параметров и систематизации выходных данных в средствах тестирования и диагностики.

Данная библиотека в последствии будет интегрирована в систему тестирования, в данный момент разрабатываемую в Радио-Техническом Институте имени Минца, аналоги которой (это OSLTS - система тестирования операционных систем Эльбрус; ТМС — Тестирующая Мотниторинговая Система от компании МЦСТ — Московский Центр Спарк Технологий ) имеют следующие недостатки:

1. Все тесты должно быть представлены в виде единого модуля, а это значит, что при добавлении нового теста приходится перекомпилировать весь модуль.
2. Отсутствие средств унификации вывода в табличной форме
3. Ввод данных осуществляется с помощью глобальных переменных, вследствие чего невозможно поменять входные данные без редактирования исходного кода теста.

При разработке библиотеки использовалась **каскадная модель жизненного цикла** Программного Обеспечения.

Такая модель предполагает, что все этапы, начиная с **Постановки задачи** и заканчивая **Эксплуатацией,** проходят последовательно, причем отсутствует возможность возвращения к предыдущим этапам. Такой подход позволяет ускорить процесс разработки, но при этом необходимо выполнение одного условия. А именно: **Постановка** **задачи** и **Требования** должны быть четко определены.

Первая функция из библиотеки, без вызова которой невозможно дальнейшее использование других функций библиотеки это функция tioInit.

TioInit проводит выделение памяти для нужд библиотеки, а также производит регистрацию всех возможных входных параметров и разбор переданных из командной строки параметров.

Функция принимает 5 аргументов:

version

help

param[]

argc

argv[]

Параметр version содержит версию запускаемого теста

Параметр help содержит краткую информацию о назначении теста

массив структур param содержит все возможные ключи, которые могут передаваться из командной строки с их кратким описанием и именем, с которым будет связано значение ключа.

Также если при запуске теста будет передан параметр —help, то на экран будет выведена информация обо всех зарегистрированных ключах.

Корме того если среди прочих параметров при вызове встретится параметр —version , то на экран будет выведена информация о версии теста.

Следующая группа функций относится к функциям получения значений входных параметров.

Последняя буква в названии говорит о том какой тип будет иметь возвращаемое значение.

Также суффикс Def говорит о том, что если значение не может быть получено, то будет возвращено значение по умолчанию.

При работе этой группы функций могут возникать следующие ошибки:

ПОКАЗЫНЫ НА СЛАЙДЕ.

Функции работы с ошибками

tioDie

tioGetError(void)

Первая из них вызывается для аварийного завершения программы, выводя в поток ошибок сообщение, переданное из второго параметра. Первый параметр задает статус завершения работы приложения.

Константы для status:

TOPASS

TOFAIL

TOTESTNOTSTART

TOINTERNALERROR.

Вторая функция позволяет получить код ошибки последней вызванной функции библиотеки. Если ошибок не было, то функция вернет код TESUC.

Далее функции строчного и табличного вывода

Функции строчного вывода

Суффикс F означает, что вывод форматируемый. Правила форматирования строки, схожи с правилами форматирования большинства стандартных функций.

В зависимости от вызванной функции перед выведением сообщения, сначала будет выведена приставка которая, обозначает к какому типу относится наше сообщение (т.е. обычная информация, отладочная информация, информация об ошибке, информация об предупреждении)

Функции табличного вывода

Первая — инициализирует таблицу,

вторая — заносит данные

третья — формирует и выводит на экран готовую таблицу.

**Демонстрация среды исполнения подпрограмм библиотеки**

Фрагмент вывода автоматического модульного

тестирования функций библиотеки.

А также результат автономного теста, который был написан ранее, и переписан с использованием разработанных функций библиотеки.

**Система контроля версий Git**

CVS, Subversion и Perforce – централизированные

Git является распределенной системой контроля версий, это значит, что каждый раз, когда клиент забирает свежую версию файлов, создаётся полная копия всех данных.

CVS, Subversion, Perforce, Bazaar – патчи

Git считает хранимые данные набором слепков небольшой файловой системы. Каждый раз, когда вы фиксируете текущую версию проекта, Git, по сути, сохраняет слепок того, как выглядят все файлы проекта на текущий момент. Ради эффективности, если файл не менялся, Git не сохраняет файл снова, а делает ссылку на ранее сохранённый файл.

В Git файлы могут находиться в одном из трёх состояний: зафиксированном, изменённом и подготовленном. «Зафиксированный» значит, что файл уже сохранён в вашей локальной базе. К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы. Подготовленные файлы — это изменённые файлы, отмеченные для включения в следующий коммит.

В технологической части дипломной работы было выполнено профилирование функций библиотеки. На слайде изображена диаграмма вызовов функций для теста, использующего функции библиотеки.

Из диаграммы видно, что приблизительно 25% процессорных операций, потраченных на выполнение теста, было израсходовано на функцию drawLine. Сделав небольшие изменения в коде этой функции, удалось сократить количество процессорных операций с 150 000 до 23 000, тем самым улучшив производительность библиотеки в целом.

Подводя итоги можно сказать, что в рамках дипломной работы:

**Были** спроектированы, реализованы и интегрированы в библиотеку функции работы с ошибками библиотеки, функция инициализации, функции получения входных параметров, функции обработки выходных данных.

**Были** продемонстрированы основные возможности библиотеки.

**Было** проведено профилирование разработанной библиотеки

**Каждый раз при** добавлении новых возможностей, для всех нетривиальных функции библиотеки проводилось автоматическое модульное тестирование по принципу черного ящика

**Все выявленные ошибки,** связанные с логикой работы функций были отлажены.

**В разделе по охране труда** и окружающей среды был проведен анализ освещения (естественного и искусственного) и микроклимата помещения, в котором происходила разработка программного обеспечения. А также проанализированы визуальные параметры устройства отображения информации

**В разделе обоснования экономической эффективности разработки** программного обеспечения был рассчитан показатель годового экономического эффекта и срок окупаемости проекта.