Федеральное агентство связи

Сибирский Государственный Университет

Телекоммуникаций и Информатики

(СибГУТИ)

**Пояснительная записка**

к курсовоиу проекту по дисциплине “Технологии разработки программного обеспечения”

**Выполнили:**

**Хорнский А.Д. и Адушев В.П.**

**Группа: ЗП-91**

**Номер студенческого билета:**

**73190184**

**Адрес электронной почты:**

**alexhornsky@yandex.ru**

**Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Новосибирск, 2020 г.**

Содержание

[ВВЕДЕНИЕ 3](#_Toc55825808)

[ПОСТАНОВКА ЗАДАЧИ 4](#_Toc55825809)

[ЭТАПЫ РАБОТЫ НАД ПРОЕКТОМ 5](#_Toc55825810)

[Окончательная структура проекта после завершения работы: 6](#_Toc55825811)

[makefile: 7](#_Toc55825812)

[.travis.yml 7](#_Toc55825813)

[.gitignore 7](#_Toc55825814)

[./src/main.cpp 8](#_Toc55825815)

[./src/func.h 9](#_Toc55825816)

[./src/func.cpp 10](#_Toc55825817)

[./test/test.cpp 11](#_Toc55825818)

[ЗАКЛЮЧЕНИЕ 12](#_Toc55825819)

# ВВЕДЕНИЕ

Цель работы: освоение курса по предмету «Технологии разработки программного обеспечения». Получение опыта использования системы контроля версий GIT, системы сборки приложений make, применения юниттестирования и использование системы Travis CI для автоматизации тестирования изменений.

Задачи:

* Разработка программы согласно ТЗ
* Ведение репозитория программы на Github
* Создание тестов для проверки работы программы
* Подключение средств для непрерывной интеграции

# ПОСТАНОВКА ЗАДАЧИ

В рамках изучения дисциплины «Технологии разработки программного обеспечения» я взял программу «Калькулятор». Данная программа должна удовлетворять следующим условиям:

Функциональность:

* В программе реализовано меню.
* Запрашивает указать какое действие желает выполнить пользователь.
* Считывает вводимое пользователем значение, первое, затем второе.
* В выбранную пользователем желаемую функцию, передает считанные значения, как фактические аргуметы.
* Результат выодится на экран.

В процессе выполнения на всех этапах разработки необходимо проводить автоматизированное тестирование кода на предмет наличия ошибок компиляции, а также юнит-тестирование различных участков кода на предмет корректного выполнения функций.

На всех этапах разработки необходимо использовать систему контроля версий Git, с подключением удаленного репозитория на сервисе Github. Данная система позволяет следить за изменениями кода, а также предоставляет удобный базовый функционал для командной разработки.

Для автоматизированного тестирования кода необходимо использовать систему Travis CI. Данная система позволяет проводить автоматизированное тестирование новых версий на предмет наличия ошибок сборки и тестов.

# ЭТАПЫ РАБОТЫ НАД ПРОЕКТОМ

Для начала работы над проектом необходимо создать локальный репозиторий Git. После создания репозитория в корне необходимо создать файл .gitignore, для игнорирования файлов системой Git. В своем проекте я поместил туда строку вида:

* \*.exe
* src/\*.exe
* test/\*.exe

для игнорирования исполняемых файлов.

Также нужно создать удаленный репозиторий на сайте Github. Для этого воспользуемся GitBash. После создания репозитория можно сразу же отправить туда наш локальный проект.

**git add .** – эта команда добавит в Git все файлы проекта, за исключением .gitignore.

**git commit –m “commit text”** – данная команда подпишет и зафиксирует изменения в проекте.

**git add remote origin** https://github.com/hornsky/mainTRPO.git - эта команда подключит к локальному репозиторию удаленный.

**git push origin master** – отправит локальные изменения на удаленный репозиторий в ветку master.

В процессе разработки мной были использованы различные инструменты системы git, такие как создание и слияние веток, работа с удаленным репозиторием, откат изменений в ветке.

Для тестирования участков кода мной были написаны юнит-тесты для функций в моей программе. Файл с тестами был выделен в отдельную директорию, в которой также содержится тестовая база данных. Для тестирования был использован фреймворк «catch2».

Для удобной сборки используется утилита make. В корневой директории проекта был создан makefile, в котором предусмотрена сборка проекта. Утилита make при запуске автоматически использует makefile из данной директории.

Для интеграции с системой Travis CI в корне проекта создан файл .travis.yml. Данный файл используется системой Travis CI для автоматического тестирования приложения.

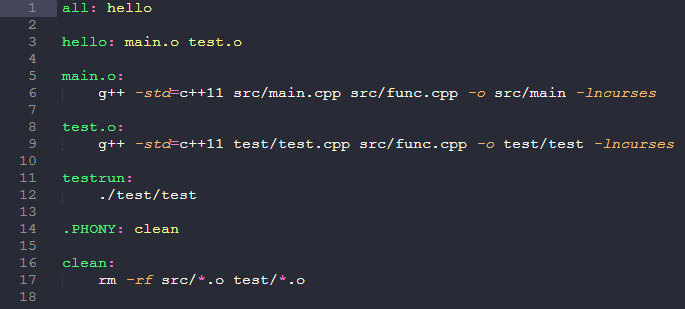
В процессе разработки были созданы: директория ./src содержащая файлы, main.cpp с телом программы, заголовочный файл func.h c прототипами функций и файл func.cpp, содержащий в себе функции для работы программы, текже директория ./test, в данной директории содержится файл test.cpp с тестами функций из func.cpp с тестовыми входными данными.

## Окончательная структура проекта после завершения работы:

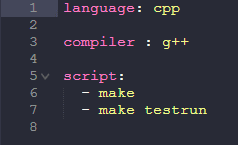
Инструменты Travis CI доступны через сайт. После синхронизации репозитория с GitHub и подключения, Travis будет использовать файл .travis.yml для проверки проекта.

После подключения Travis CI к описанию репозитория можно добавить динамическую иконку прохождения тестов. Код для вставки можно получить на сайте Travis CI и вставить в файл README.md в формате Markdown. После commit в описании появится иконка с текущим статусом прохождения тестов.

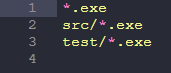
## makefile:



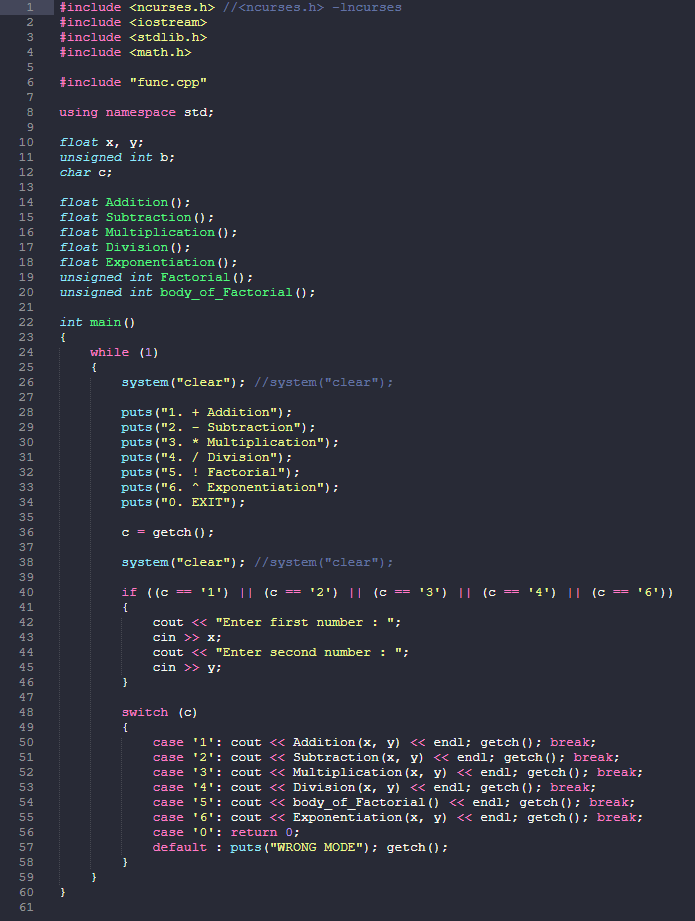
## .travis.yml



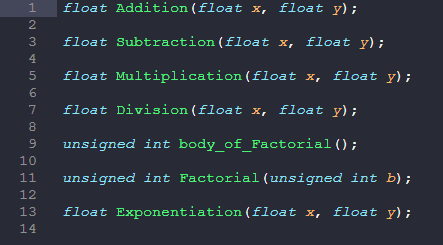
## .gitignore



## ./src/main.cpp



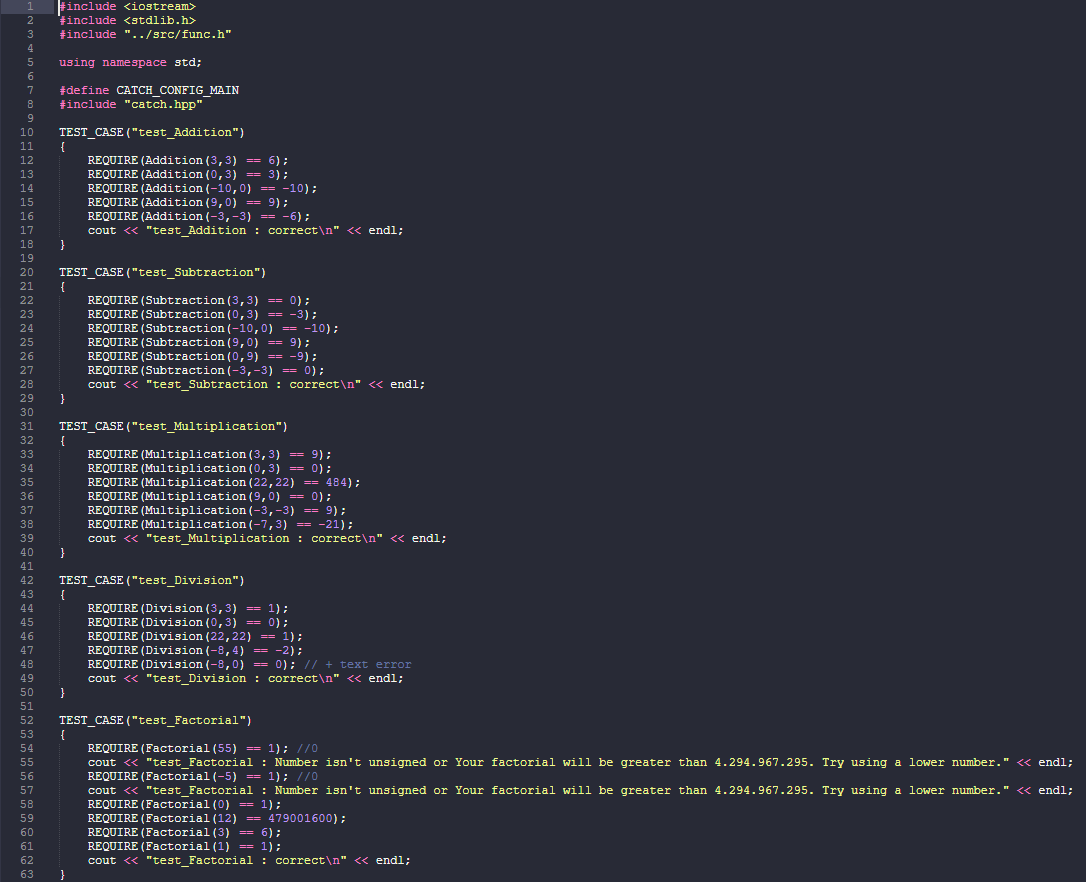
## ./src/func.h

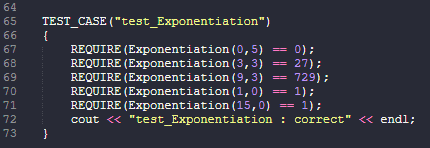


## ./src/func.cpp



## ./test/test.cpp





# ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была разработана программа «Калькулятор». В процессе выполнения работы были применены различные средства и методы разработки, такие как git, Travis CI, make, юниттестирование. В конечном итоге на странице проекта на сайте github в ветку master была загружена рабочая версия программы, готовая к использованию. Данная версия программы удовлетворяет всем требованиям и прошла тестирование. Проект привязан к системе Travis CI для непрерывной интеграции. Соответствующий индикатор расположен в секции README.

https://github.com/hornsky/mainTRPO