Практика

Примером практической работы может служить Репозиторий предмета.

Задачи

Получить ПО для управления роботом в режиме терминала командами, графически псевдографикой.

Группа делится на подгруппы по 2 человека. Каждая подгруппа выполняет работу по одному модулю.
Каждый модуль представляется как статическая(динамическая) библиотека и объединяется в один исполняемый модуль.
Весь проект представлен и разделен при помощи `cmake`, каждая подгруппа сама описывает `cmake` проект.
Также, самостоятельно, пишут скрипты сборки, тестовые данные.
Каждый модуль должен быть протестирован.

Код, в обязательном порядке, должен быть оформлен в соответствии со стилем

Для каждого задания, подгруппа описывает функции модуля, его взаимосвязь с другими модулями, описывается как настраивается модуль, как выстроена взаимосвязь внутренних компонент модуля.

Одна из подгрупп назначется основной и отвечает за приложение

целиком. Делает сводное описание всего приложения, описывается связи всех компонент, рисует графическое представелние взаимодействия. Описывает основной проект `cmake` и координирует работу подгрупп. Осуществляет взаимодействие с преподавателем. Выявляет узкие места и отставания по приложению. Мониторит репозиторий, контролирует `merge`. Также, периодически основная подгруппа предоставляет `merge request` в основной репозиторий с результатами. Основная группа форкается от базового репозитория, остальные подгруппы форкаются от репозитория основной шруппы.

Каждый студент заводит на `github` аккаунт и форкает репозиторий основной группы себе. Далее прописывает `base` удаленный репозиторий группы для актуализации соего репозитория относительно основного и периодически обновляться для того чтобы исключить конфликты.

Каждая группа форкает репозиторий: 2018 и создает свою папку с проектом.

Функции

1. Приложение должно работать с 3 режимах: командном (команды передаются как аргументы командной строки), данный режим включается передачей первого аргумента `console`; меню (работа продолжнается посредством управления меню консоли), данный режим включается передачей первого аргумента `menu`; графическом (работа

продолжается посредством управления псевдографическим меню), данный режим включается передачей первого агумента `gui`.
2. Приложения должно конфигурироваться при помощи настроек хранящихся в файле. Также управлять настройками: сохранять, удалять, изменять. Формат настроек: `java properties`
3. Кодировка всех текстовых файлов с которыми работает приложение `cp866`
4. Команды: смена положения робота (`X1` , `X2` , `X3` , `Y`) с передаче параметров (скорость, ускорение), получение значений датчиков (`N1` , `N2` , `N3` , `N4`), получение настроек (`CE`), установка настроек (`CE`)
5. Получение от робота обратные вызовы (сервер)

Задания

1. Модуль по работе с аргументами командной строки. Обработка аргументов, подготовка команды к исполнению, продолжение работы осуществяется посредством обратного вызова с передачей подготовленной команды.
2. Модуль по работе с меню в режиме терминала. Считывание настроек меню (описание структуры меню хранится в файле, в нем также указывается наименование меню, команда вызова, переходы - для подменю, выходы и отвена)
3. Модуль по работе с графическим меню. Считывание настроек меню (описание структуры меню хранится в файле, в нем также указывается наименование меню, команда

вызова, переходы - для подменю, выходы и отвена)

4. Модуль по работе с настройками в режиме командной строки.
5. Модуль по работе с настройками в режиме консольного меню.
6. Модуль по работе с настройками в режиме графичесчкого меню.
7. Модуль по управлению настройками. Регистрация, сохранения, правила проверки, граничные значения. Хранение в бинарном виде.
8. Модуль по работе с командами приложения. Отвечает за регистрацию команды, удаление, передачи и управелния настройками команды, отслеживание установки обязательных для исполнения параметров.
9. Модуль по работе с сетью. Настройка клиента. Взаимодействие с сервером по протоколу `HTTP`
10. Модуль по работе с сетью. Настройка клиента. Взаимодействие с сервером по бинарному протоколу.
11. Модуль обратного вызова (сервер). Настрока сервера. Взаимодействие с клиентом по протоколу `HTTP` (запросы `PUT`
    или `POST`, тип содержимого `json`, поля: `status` - код статуса выполнения команды (числовой код, 0 - успешное выполнение), `message` - поле сообщения об ошибки или дополнительные сведения (может отсутствовать)).

```
PUT /robo_call HTTP/1.1
Content-Type: application/json
```

```
{"status": 0, "message": "Операция выполнена успешно"}
```

12. Модуль обратного вызова (сервер). Настрока сервера. Взаимодействие с клиентом по бинарному протоколу.

Наименование заданий

1. Разработка программного модуля обработки аргументов команд управления роботом манипулятором
2. Разработка программного модуля терминального меню управления роботом манипулятором
3. Разработка программного модуля псевдографического меню управления роботом манипулятором
4. Разработка программного модуля обработки аргументов команд управления настройками робота манипулятора
5. Разработка программного модуля терминального меню управления настройками робота манипулятора
6. Разработка программного модуля псевдографического меню управления настройками робота манипулятора
7. Разработка программного модуля управления настройками робота манипулятора
8. Разработка программного модуля управления командами робота манипулятора
9. Разработка программного модуля передачи команд роботу манипулятору по бинарному сетевому протоколу
10. Разработка программного модуля передачи команд роботу

манипулятору по HTTP протоколу

11. Разработка программного модуля по приему команд обратной связи от робота манипулятора по HTTP протоколу
12. Разработка программного модуля по приему команд обратной связи от робота манипулятора по бинарному сетевому протоколу