

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Утвержден на заседании кафедры

«Вычислительная техника» _____

" ____ " _____ 20 ____ г.

Заведующий кафедрой

_____ М.А. Митрохин

**ОТЧЕТ ПО УЧЕБНОЙ (ТЕХНОЛОГИЧЕСКОЙ
(ПРОЕКТНО-ТЕХНОЛОГИЧЕСКОЙ)) ПРАКТИКЕ**

(2019/2020 учебный год)

Балаев Кирилл Артемович

Направление подготовки 09.04.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Вычислительные машины, комплексы, системы и сети»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения _____ 1 _____ семестр _____ 2 _____

Период прохождения практики с 06.07.2020 по 18.07.2020

Кафедра «Вычислительная техника»

Заведующий кафедрой д.т.н., профессор, Митрохин М.А.

(должность, ученая степень, ученое звание, Ф.И.О.)

Научный руководитель д.т.н., профессор, Митрохин М.А.

(должность, ученая степень, ученое звание)

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГБОУ ВО "ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ"
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Утвержден на заседании кафедры

«Вычислительная техника» _____

" ____ " _____ 20 ____ г.

Заведующий кафедрой

_____ М.А. Митрохин

**ИНДИВИДУАЛЬНЫЙ ПЛАН
ПРОХОЖДЕНИЯ УЧЕБНОЙ (ТЕХНОЛОГИЧЕСКОЙ
(ПРОЕКТНО-ТЕХНОЛОГИЧЕСКОЙ)) ПРАКТИКИ
(2019/2020 учебный год)**

Балаев Кирилл Артемович

Направление подготовки 09.04.01 Информатика и вычислительная техника

Наименование профиля подготовки Вычислительные машины, комплексы, системы и сети

Форма обучения – очная Срок обучения в соответствии с ФГОС – 2 года

Год обучения _____ 1 _____ семестр _____ 2 _____

Период прохождения практики:

с "6" июля 2020 г. по "18" июля 2020 г.

Кафедра «Вычислительная техника»

Заведующий кафедрой д.т.н., профессор, Митрохин М.А.

(должность, ученая степень, ученое звание, Ф.И.О.)

Научный руководитель д.т.н., профессор, Митрохин М.А.

(должность, ученая степень, ученое звание)

№ п/п	Планируемые формы работы во время практики	Количество часов	Календарные сроки проведения планируемой работы	Подпись руководителя практики
1	Разработка индивидуального плана практики и его согласование с руководителем	2	06.07.2020 г.	
2	Консультирование с руководителем об организации групповой проектной работы в вузе	1	07.07.2020 г.	
3	Самостоятельное изучение рекомендованной нормативно методической базой об организации групповой проектной работы в вузе	12	07.07.2020 - 09.07.2020 г.	
4	Консультирование с руководителем об основной образовательной программе кафедры ВТ	1	07.07.2020 г.	
5	Самостоятельное изучение рекомендованной образовательной программы кафедры ВТ	12	07.07.2020 - 09.07.2020 г.	
6	Консультирование с руководителем плана ведения практики	3	07.07.2020 г.	
7	Изучение учебно-методического материала практики	2	07.07.2020 г.	
8	Разработка указаний для выполнения задания по практике		07.07.2020 - 09.07.2020 г.	
9	Подготовка к проведению дистанционных консультаций по практике	10	07.07.2020 г.	
10	Проведение консультаций (дистанционно) по практике в Discord	8	07.07.2020 - 09.07.2020 г.	
11	Организация групповой работы. Ознакомление с программой практики, порядком прохождения, защиты отчета по практике и требованиями к оформлению отчета	3	07.07.2020 - 09.07.2020 г.	
12	Систематизация учебного материала. Распределение заданий для группы по практике.	8	07.07.2020 - 09.07.2020 г.	
13	Анализ, обсуждение и выполнение бригадами группового задания, разбитого на подзадачи	12	07.07.2020 - 13.07.2020 г.	
14	Консультирование (дистанционное) с промежуточными версиями выполнения индивидуального задания	12	07.07.2020 - 13.07.2020 г.	
15	Проверка (дистанционно) результатов у подгрупп студентов	10	07.07.2020 - 17.07.2020 г.	
16	Анализ результатов ведения групповых проектов занятий с руководителем	2	18.07.2020 г.	
17	Подготовка отчета о практике	2	18.07.2020 г.	
	Общий объем часов	108	06.07.2020 - 18.07.2020 г.	

Магистрант Балаев К.А. _____ "___" _____ 2020 г.

Научный руководитель Митрохин М.А. _____ "___" _____ 2020 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГБОУ ВО "ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ"
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

ОТЧЕТ
О ПРОХОЖДЕНИИ УЧЕБНОЙ (ТЕХНОЛОГИЧЕСКОЙ
(ПРОЕКТНО-ТЕХНОЛОГИЧЕСКОЙ)) ПРАКТИКИ

(2019/2020 учебный год)

Балаев Кирилл Артемович

Направление подготовки 09.04.01 Информатика и вычислительная техника

Наименование профиля подготовки Вычислительные машины, комплексы, системы и сети

Форма обучения – очная Срок обучения в соответствии с ФГОС – 2 года

Год обучения 1 семестр 2

Период прохождения практики:

с "6" июля 2020 г. по "18" июля 2020 г.

Кафедра «Вычислительная техника»

Основные итоги практики:

Балаев К.А. вел у группы бакалавриата практику «Научно-исполнительская практика». На первоначальном этапе были изучены и проанализированы учебно-методические материалы по учебной программе. А так же согласованы с руководителем практики список вариантов заданий на практику. Произвели организацию групповой работы. Ознакомили с программой прохождения, защиты отчета по практике и требованиями к оформлению отчетов. Систематизировали учебно-методический материал для студентов. Выдали групповые задания. Консультировали в течение всей практики по вопросам возникающим в процессе выполнения практики с контролем промежуточных версий. Методологией управления проекта выбран Kanban.

Магистрант Балаев К.А. _____ "___" _____ 2020 г.

Научный руководитель Митрохин М.А. _____ "___" _____ 2020 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГБОУ ВО "ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ"
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

ОТЗЫВ
О ПРОХОЖДЕНИИ УЧЕБНОЙ (ТЕХНОЛОГИЧЕСКОЙ
(ПРОЕКТНО-ТЕХНОЛОГИЧЕСКОЙ)) ПРАКТИКИ

(2019/2020 учебный год)

Балаев Кирилл Артемович

(Ф.И.О. магистра)

Направление подготовки 09.04.01 Информатика и вычислительная техника
Код, название

Наименование профиля подготовки Вычислительные машины, комплексы,
системы и сети

Форма обучения – очная Срок обучения в соответствии с ФГОС – 2 года

Год обучения 1 семестр 2

Период прохождения практики:

с "6" июля 2020 г. по "18" июля 2020 г.

Кафедра «Вычислительная техника»

В процессе прохождения практики Балаев К.А. решал следующие задачи: закрепление и углубление теоретических знаний в сфере информатики и вычислительной техники, а также приобретение практических навыков и компетенций в сфере научной деятельности по профилю подготовки.

За период прохождения практики магистром были проведены подготовительные работы для организации и проведения практики, консультации с подробным анализом и обсуждением порядка проведения и сдачи практики, а также организация групповой работы, выдача групповых заданий на практику и прием отчетов по ним. Во время практики показал себя ответственным, добросовестным руководителем, знающим свой предмет, имеющим представление о современном состоянии науки, владеющим современными общенаучными знаниями по информатике и вычислительной технике, программированию, знаниями о современных технологиях образовательного процесса. Балаев К.А. свободно ориентировался во всех аспектах методики управления групповой работой во время практики.

Практика Балаева К.А. заслуживает оценки «Отлично».
Научный руководитель к.т.н., доцент, Карамышева Н.С. 
07 2020 г.

ВВЕДЕНИЕ

Разработка программного обеспечения – это групповой процесс. Большинство программных продуктов рождается в результате коллективных усилий. Люди являются одной из главных составляющих, необходимых для производства программного продукта. При организации работы программистов важно учитывать не только их технические навыки, но и взаимодействие между ними.

Целью проектно-технологической практики являются закрепление и углубление теоретической подготовки обучающегося, знакомство с практическими аспектами профессиональной деятельности по интеграции программного обеспечения в рамках одного проекта для решения практических задач.

Для достижения цели практики решались следующие **задачи**:

- получение умений по объединению нескольких программных модулей в области решения профессиональных задач с применением программных средств;
- углубление навыков по сбору, обработке, анализу и систематизации информации по теме задания;
- углубление навыков по документированию программных средств, подготовке отчётов;
- получение навыков работы с современными программными и аппаратными средствами информационных технологий.

Теоретической и методологической основой выполненной работы являются существующие материалы, используемые при организации практики «Научно-исполнительской» бакалавриата 2 курса, а также описания базовых алгоритмов обработки данных, находящиеся в публичном доступе и не имеющие ограничений на использование.

При прохождении практики используются подходы группового разделения сфер исследования существующих источников, включая научную,

профессионально-техническую, и учебно-методическую литературу, анализа полученных результатов и консолидации результатов индивидуальных исследований в обобщенный учебно-методический материал.

При выполнении данной работы учитывается индивидуальный опыт, полученный в ходе выполнения группового задания и анализа учебно-методического материала в течение практики.

ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ

Необходимо реализовать и продемонстрировать работу класса из списка, представленного ниже. Конечная программа должна реализовывать непосредственно класс, а также выполнять **считывание входных данных из окна консоли и запись результатов в класс, и отображать все данные в графическом интерфейсе.**

Программа должна обрабатывать данные, вводимые с помощью графического интерфейса.

Конечные программы не должны использовать реализации класса, предоставляемые стандартными или сторонними библиотеками, включая системные.

Отчет должен содержать следующие элементы:

- титульный лист (см. приложение 1);
- индивидуальный план (см. приложение 2);
- лист отчета с краткими итогами (см. приложение 3);
- отзыв руководителя (см. приложение 4);
- содержание
- задачи и цели работы
- ход работы
- краткое описание класса;
- непосредственная реализация класса без использования стандартных или сторонних библиотек, с описанием программы;
- блок-схема программы;
- раздел тестирования готовой программы;
- анализ полученных результатов тестирования.

В качестве результатов выполнения работы предоставляются: отчет с описанием реализованного класса, исполняемая программа, включая ее исходный код.

Каждая группа бакалавров получает один из 20 вариантов заданий и разделяет его выполнение следующим образом:

1. один студент разрабатывает конструктор и деструктор класса и реализует его на любом языке программирования;
2. второй студент разрабатывает методы ввода и вывода, для отображения всех полей класса и их записи в класс, реализованный первым студентом.
3. Третий студент сопоставляет всё вместе и производит тестирование для проверки корректности работы готовой программы.

Для выполнения предложенных заданий учебной практики необходимы навыки, полученные в ходе изучения курса «Программирование», а также базовые знания по алгоритмизации. Для реализации программ допустимо использовать любые доступные языки программирования, за исключением случаев использования готовых или стандартных библиотек.

При проверке качества реализации программы необходимо учитывать корректность реализации основного класса, а также реакцию программы на «граничные» наборы данных, при которых возможно выявление ошибок реализации класса или кода ввода и вывода данных.

Варианты заданий

1. Класс «Автомобиль» (легковой/грузовой, мощность, объем двигателя, цена и т.д.).
2. Класс «Компьютер» (процессор, объем оперативной памяти, цена и т.д.).
3. Класс «Телефон» (производитель, модель, размер экрана и т.д.).
4. Класс «Книга» (тема, автор, издательство, год выпуска и т.д.).
5. Класс «Журнал» (название, тема, тираж и т.д.).
6. Класс «Студент» (ФИО, ВУЗ, факультет, кафедра, форма обучения и т.д.).
7. Класс «Сотрудник» (ФИО, предприятие, должность, зарплата, стаж и т.д.).
8. Класс «Музыкальное произведение» (название, автор, исполнитель, жанр и т.д.)
9. Класс «Фильм» (название, режиссер, студия, жанр и т.д.)
10. Класс «Населенный пункт» (название, количество жителей, географические координаты, виды общественного транспорта и т.д.).
11. Класс «Предприятие» (название, адрес, направление производства, количество сотрудников и т.д.).
12. Класс «Магазин» (название, адрес, торговая площадь, количество сотрудников и т.д.).
13. Класс «Кинотеатр» (название, адрес, количество залов, репертуар и т.д.).
14. Класс «Телевизор» (производитель, размер экрана, поддерживаемые форматы, цена и т.д.).
15. Класс «Компьютерная игра» (название, жанр, производитель, цена и т.д.).
16. Класс «Одежда» (название, тип, материал, размер, цена и т.д.)
17. Класс «Колеса» (типоразмер резины, радиус диска, название резины, название диска и т.д.)
18. Класс «Велосипед» (горный\шоссейный, радиус колес, фирма изготовитель, материал рамы, цена и т.д.)
19. Класс «Отель» (название, количество звезд, этажность, количество номеров и т.д.)
20. Класс «Мебель» (тип, материал, название, фирма изготовитель, габариты и т.д.)

ОРГАНИЗАЦИЯ ДИСТАНЦИОННОГО РАБОЧЕГО ПРОСТРАНСТВА

Для коммуникации со студентами в условиях дистанционного режима обучения был выбран сервис Discord. Это проприетарный бесплатный мессенджер с поддержкой VoIP, видеоконференций, предназначенный для использования различными сообществами геймеров, студентов/школьников и бизнеса. Разработчик — компания Hammer & Chisel (Discord Inc.) из Сан-Франциско. Настольное клиентское приложение реализовано для Windows, macOS и Linux, мобильное приложение — для Android и iOS, также существует веб-клиент. Построен на базе фреймворка Electron. Серверы мессенджера размещены в 11 центрах обработки данных в разных частях мира. По состоянию на 13 мая 2019 года в Discord'е было зарегистрировано более 250 млн чел. при среднем ежемесячном онлайне в 56 млн чел.

Возможности Discord:

- Приложение способно организовывать голосовые конференции с настройкой канала связи и работать по принципу push-to-talk, создавать публичные и приватные чаты для обмена текстовыми сообщениями. Программа имеет браузерную версию, отличием от настольной версии является то, что режим push-to-talk работает только если в браузере активна вкладка с приложением;
- В текстовом чате поддерживается присоединение файлов, картинок, вставка ссылок (для некоторых сайтов работает предпросмотр), форматирование текста и смайлики, размер вложения ограничен объёмом 8 МБ;
- Организация выделенного сервера, где позволяет создавать необходимое количество текстовых каналов и голосовых/видео каналов (позволяет транслировать экран или отдельное приложение на компьютере).

Перед началом практики был подготовлен специальный сервер в Discord с названием «ЛетняяПрактика_2020». Его основные характеристики и настройки:

1. Голосовые каналы:

1.1. Аудитории – голосовые каналы, где могут проводиться проверки работы программы студентов, голос руководителя приглушает голос всех остальных участников беседы, также позволяет каждому участнику использовать транслировать видео с камеры или экрана своего рабочего стола/приложения;

2. Текстовые канал:

2.1. Основной – канал с описанием сервера, условиями прохождения практики и общей информации.

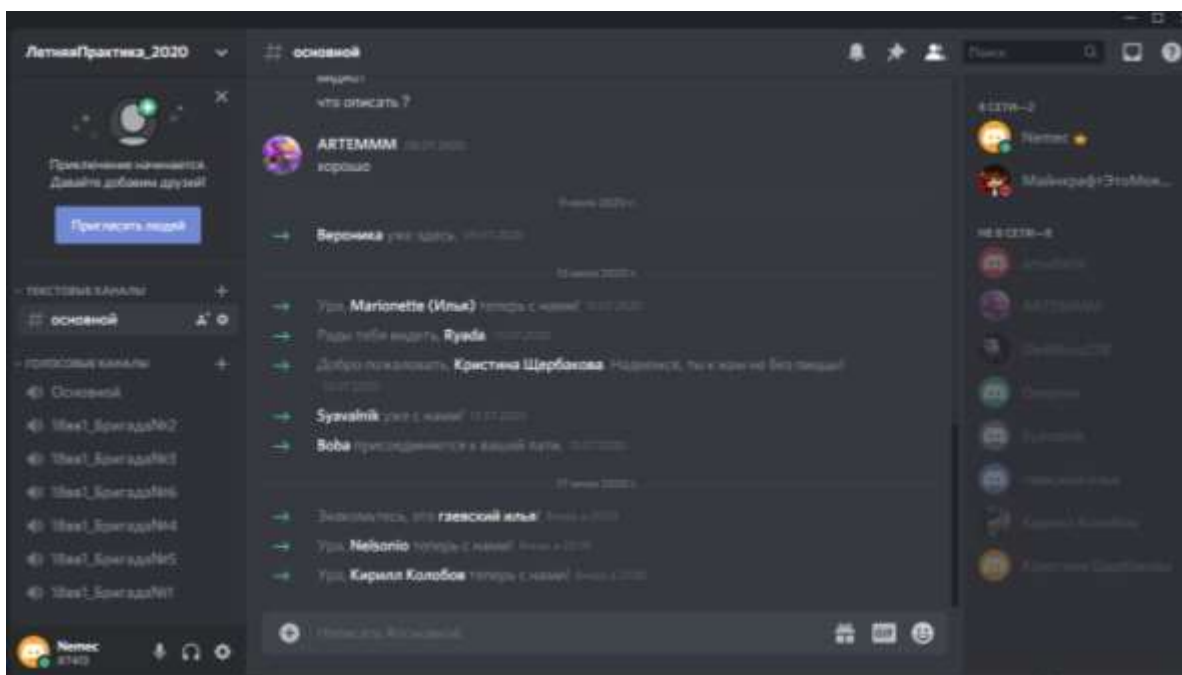


Рисунок 1. Общий вид сервера для практики

Подключиться к серверу можно по постоянной ссылке, которую может создать модератор или владелец сервера. При первом попадании на сервер, пользователь окажется в текстовом канале «основное», где описано всё необходимое для начала работы.

Discord позволяет использовать шаблоны сервера, чтобы моментально создать сервер со всеми сохранёнными в шаблон настройками (каналы, роли, права доступа и так далее). Шаблон данного сервера доступен по ссылке: <https://discord.new/eKY2FRSsAgf6>.

При необходимости срочных извещений всей группе использовалась связь через социальную сеть ВКонтакте. Через старосту группы 18BB1 передавалась срочная информация для всей группы.

Для обмена кодом со студентами в условиях дистанционного режима обучения был выбран сервис GitHub. GitHub— крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки.

Веб-сервис основан на системе контроля версий Git и разработан на Ruby on Rails и Erlang компанией GitHub, Inc (ранее Logical Awesome). Сервис бесплатен для проектов с открытым исходным кодом и (с 2019 года) небольших частных проектов, предоставляя им все возможности (включая SSL), а для крупных корпоративных проектов предлагаются различные платные тарифные планы.

Слоган сервиса — «Social Coding» — на русский можно перевести как «Пишем код вместе». На футболках же печатают совсем другую фразу: «Fork you!» («Ответись!»). С одной стороны, она созвучна с англоязычным ругательством и намекает на неформальную атмосферу. С другой, эти слова напоминают, что создавать новые форки с Git можно легко и безболезненно — традиционно, к созданию веток разработчики проектов с открытым исходным кодом относятся негативно — а также созвучна названию одной из возможностей GitHub — очереди форков.

Каждая из бригад при выполнении задания создавала свой репозиторий куда загружала код своей программы.

Методологией управления проекта выбран kanban. Методология позволяет максимально эффективно использовать имеющиеся в наличии трудовые и материальные ресурсы команды, весь её технический потенциал, когда разработчики разбиваются на группы, каждая из которых отвечает за своё конкретное направление, но может привлекаться в другие направления для решения наиболее важных задач.

Модель Kanban

В основе Kanban лежит непрерывная структура рабочего процесса, дающая командам свободу действий и возможность меняться вместе с изменениями приоритетов. Рабочие задачи представлены карточками на доске Kanban и перемещаются из одной стадии рабочего процесса (столбца) в другую. Обычно рабочий процесс подразделяется на стадии «Предстоит сделать», «В процессе», «На проверке», «Заблокировано» и «Завершено».

Возможность создавать собственные столбцы в зависимости от того, как работает ваша команда, — лучшая особенность Kanban.

Подходы к релизу

В Kanban обновления выпускаются по мере готовности; регулярный график или заранее определенные сроки отсутствуют как таковые.

Теоретически, в рамках Kanban нет нужды устанавливать точный срок для завершения задания. Если задание выполняется раньше (или позже), результат выпускается по мере необходимости — для выпуска не нужно ждать контрольной точки, например собрания по обзору итогов спринта.

Роли в Kanban

Доска Kanban принадлежит всей команде. Некоторые команды привлекают к участию тренера по agile, но Kanban, в отличие от Scrum, не предусматривает

роль «kanban-мастера», который отвечал бы за устранение шероховатостей в процессе работы. Команда несет коллективную ответственность за совместное выполнение заданий на доске и поставку результатов.

Ключевые показатели

Время выполнения и время цикла — важные показатели для команд Kanban. Они отражают, сколько в среднем уходит времени на выполнение работы от начала до конца. Сокращение времени цикла говорит об успехе команды Kanban.

Отношение к изменениям

В Kanban рабочий процесс можно менять в любой момент. В зависимости от приоритетов новые рабочие задачи можно добавлять в бэклог, а существующие карточки — блокировать или вовсе удалять. В случае изменения производительности команды можно соответствующим образом изменить рабочие задачи. Суть Kanban — в гибкости.



Рисунок 2. Процесс работы Kanban

Работа Kanban-команд строится вокруг Kanban-доски, которая используется для визуализации и оптимизации рабочего процесса. Хотя некоторые команды предпочитают реальные доски, их виртуальные аналоги давно стали ключевой функцией любого инструмента agile-разработки ПО: с ними проще отследить процессы, а также организовать совместную работу и доступ из разных мест.

Доски нужны, чтобы визуализировать работу команды, стандартизировать процесс, а также найти и устранить блокиеры и зависимости. И не важно, в какой форме они представлены — физической или цифровой. На стандартной Kanban-доске процесс состоит из трех шагов: «Запланировано», «В работе» и «Сделано». Однако доску можно настроить в соответствии с процессом, принятым в той или иной команде, в зависимости от ее размеров, структуры и целей.

Методика Kanban основана на полной прозрачности работы и обмене информацией по ресурсам в режиме реального времени. Таким образом, Kanban-доска должна стать единственным достоверным источником информации о работе команды.

Подводя итоги, можно сделать вывод, что, методология Kanban побуждает команду к более активной и плодотворной работе в проекте, способствует находить новые точки роста и стремиться к постоянному совершенствованию своих знаний и навыков.

ХОД РАБОТЫ

В понедельник 6 июля 2020 года была проведено извещение студентов о прохождении летней практики, была выдано задание о разделении на подгруппы, который связывались через социальную сеть Вконтакте с руководителем практики, получали ссылка на канал своей подгруппы в дискорде на котором могли получать консультации по практике и задавать свои вопросы, также были представлены учебные материалы для ознакомления.

В среду 8 июля 2020 года после консультации 7 июля с руководителем практики было проведено второе извещение студентов на котором были обговорены условия формирования отчетов, требования к программе, были переданы учебные материалы для изучения системы контроля версий Git, которая могла использоваться как дополнительное задание для студентов претендующих на оценку отлично. Также студентами были созданы свои репозитории GitHub для обмена кодом между членами бригады. Выслана ссылка на репозиторий каждой бригады проверяющему магистру для контроля и оценки работы студентов в течении практики.

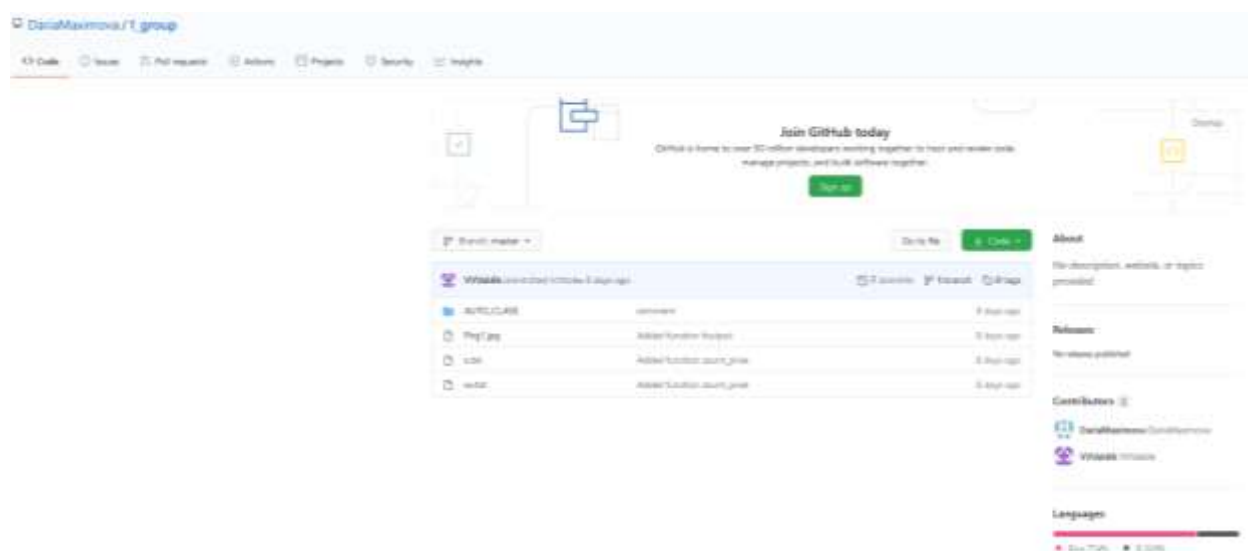


Рисунок 3 – публичный репозиторий для прохождения учебной практики

Система контроля версий Git позволяет организовать удалённую работу над командным проектом. Было решено, что каждая подгруппа студентов заводит свой репозиторий GitHub, в который в течении своей работы отправляет изменения. Пример истории изменений выполненного проекта студентами «Класс Автомобиль» представлена на рисунке 4.

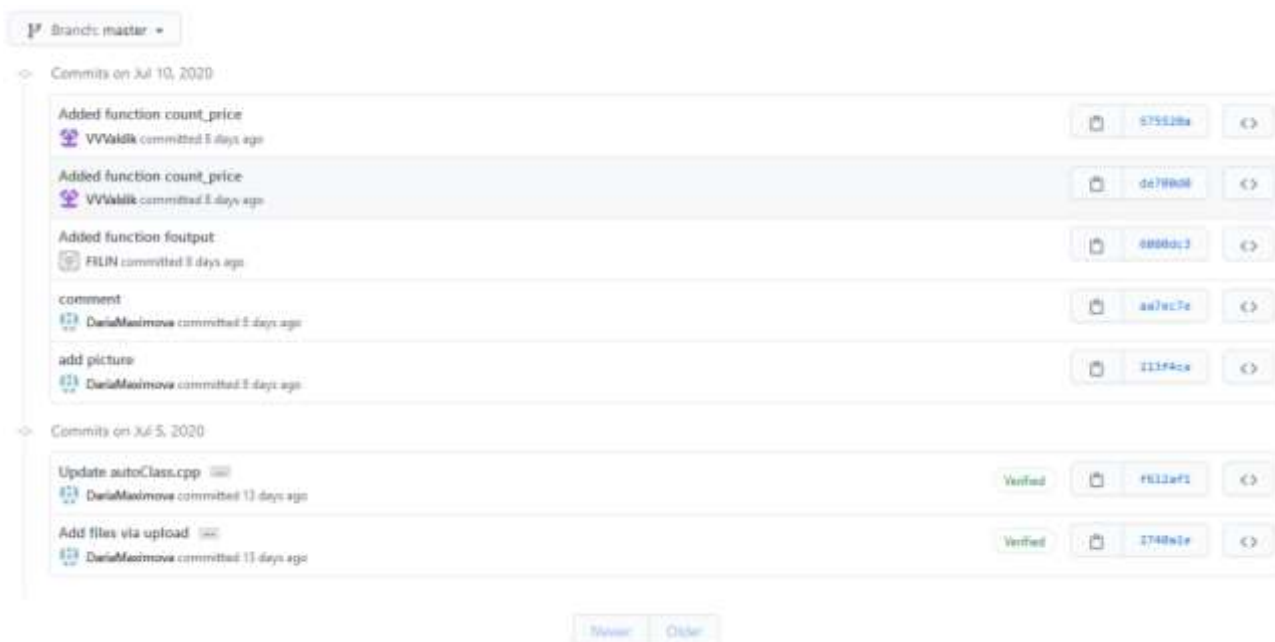


Рисунок 4 – история изменений публичного репозитория одной из бригад

В понедельник 13 июля 2020 года, были получены первые выполненные задания от бригады бакалавров. Была проведена проверка работы программы в голосовом канале сервиса Discord. С помощью демонстрации экрана студенты показали работу программы. Был произведен индивидуальный опрос членов бригады о выполненной работе и выставлены предварительные оценки их деятельности в ходе летней практики.

В течении всей практики студенты могли обращаться за консультацией к проверяющему магистру в социальной сети Вконтакте и сервисе Discord.

АНАЛИЗ ВЫПОЛНЕННОЙ РАБОТЫ

Студенты второго курса бакалавра группы 18ВВ1 разделились на бригады следующим образом:

№ бригады	ФИО	Вариант
1	Коротов В.О	1
	Гросс Д.А.	
	Максимова Д.Р.	
2	Арсентьева У.С.	2
	Ермаков А.В.	
	Горбунова В.В.	
3	Коновалова Д.А.	3
	Репин И.А.	
	Щербакова К.А.	
4	Гаевский И.Р.	4
	Кривоzubов А.К.	
	Махров А.А.	
5	Колобов К.А.	5
6	Пчелинцев А.И.	6
	Федулов В.Д.	
7	Немцев А.А.	16
	Воеков И.С.	
	Канцеров Н.И.	
8	Семенов А.С.	Дистанционно на предприятии
9	Амплеев Д.О.	Дистанционно на предприятии
10	Свищев Д.С.	Дистанционно It-Lab
11	Митрофанов М.П.	Дистанционно It-Lab

Все студенты группы работали бригадами по два и три человека, то есть в бригаде. Каждая бригада провела анализ класса из полученного задания.

Бригада 1.

1. Студент Максимова Д.Р. разработала класс Auto , его конструктор и деструктор функции по вводу и выводу данных класса в консоль на языке C++.

2. Студент Коротов В.О. разработал и проверил функцию для внутренней работы класса на языке C++.

3. Студент Гросс Д.А. произвел тестирование конечного варианта программы на языке программирования C++ и разработал функции для работы с файлами.

Бригада 2.

1. Студент Горбунова В.В. разработала класс его конструктор и деструктор функции по вводу и выводу данных класса в консоль на языке C++.

2. Студент Ермаков А.В. разработал и проверил функцию для ввода и вывода данных класса в консоль на языке C++.

3. Студент Арсентьева У.С. произвела тестирование конечного варианта программы на языке программирования C++.

Бригада 3.

1. Студент Репин И.А. разработал класс его конструктор и деструктор функции по вводу и выводу данных класса в консоль на языке C++.

2. Студент Коновалова Д.А. разработал и проверил функцию для ввода и вывода данных класса в консоль на языке C++.

3. Студент Щербакова К.А. произвела тестирование конечного варианта программы на языке программирования C++.

Бригада 4.

1. Студент Гаевский И.Р. разработал класс его конструктор и деструктор функции по вводу и выводу данных класса в консоль на языке C++.

2. Студент Кривозубов А.К. разработал и проверил функцию для ввода и вывода данных класса в консоль на языке C++.

3. Студент Махров А.А. произвел тестирование конечного варианта программы на языке программирования C++.

Бригада 5.

1. Студент Колобов К.А. разработал класс, его конструктор и деструктор функции по вводу и выводу данных класса в консоль протестировал и отладил программу на языке C++.

Бригада 6.

1. Студент Пчелинцев А.И. протестировал и отладил итоговую программу, был ответственен за репозиторий на GitHub и работу с Git.

2. Студент Федулов В.Д. разработал класс, его конструктор и деструктор функции по вводу и выводу данных класса в консоль протестировал и отладил программу на языке C++.

Бригада 7.

1. Студент Немцев А.А.
2. Студент Воеков И.С.
3. Студент Канцеров Н.И.

Студенты проявили свои знания в предметных областях инженерных направлений, а также знания и навыки, полученные за два года обучения в бакалавриате.

Все студенты выполнили возложенные на них задачи при разделении их внутри на подгруппы (бригады).

Некоторые бригады решили осуществлять работу с репозиторием GitHub с помощью системы контроля версий Git. Им было предоставлено руководство по работе с Git. Результаты работы видны на Рисунках 5 и 6. Возложенную задачу использования системы контроля версий студенты выполнили прекрасно.



Рисунок 5 - Пример работы с Git

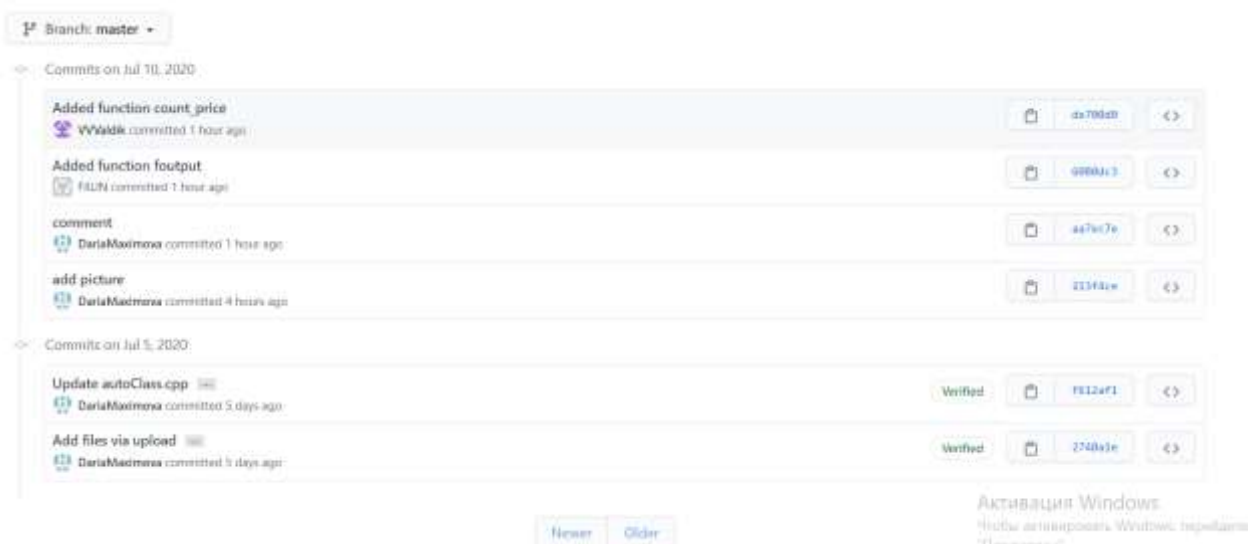


Рисунок 6 - Пример репозитория студентов использовавших Git

В ходе разработки программ у студентов практически не возникало трудностей. В основном проблемы были связаны с необработанными исключениями при разработке программ с графическим интерфейсом. С другой стороны, на данном курсе им ещё не преподавались основы объектно-ориентированного программирования и построения графических интерфейсов.

Основная сложность возникла с оформлением отчёта. Основными типовыми были:

- неправильное заполнение титульных листов;
- неправильное оформление рисунков;
- несоответствие полей страницы заявленным требованиям;
- неверная настройка постановки номера страниц (выставляли на титульных листах);
- отсутствие раздела с подробным описанием работы программы;
- отсутствие раздела тестирования;
- отсутствие рисунка с составными частями изделия (интерфейса);
- небольшие недочеты по оформлению основного текста (отсутствие красных строк, неверное выравнивание текста).

Все проблемы проработаны со студентами индивидуально для каждой работы в дистанционном формате с последующим исправлением и зачетом работы.

ЗАКЛЮЧЕНИЕ

В результате прохождения проектно-технологической практики был получен опыт по выполнению проектной работы и опыт работы в группе на основе учебной практики бакалавриата 2 курса обучения. В их состав входят 20 вариантов заданий на реализацию простых классов, а также разработка графического интерфейса. Студенты второго курса бакалавриата получили опыт работы с системами контроля версий Git, научились работе с сервисом GitHub. Получили навыки удаленной работы в рамках научно-исполнительской практики.

При составлении заданий и групп для их выполнения учитывалось содержание учебного плана бакалавриата 2 курса, а также собственный опыт прохождения учебной практики. При определении заданий учитывалась необходимость постановки групповой работы, чтобы в процессе ее решения студенты выполнили исследование требуемого к реализации класса, разработка и реализация корректного, «дружелюбного» графического интерфейса, и проверка корректности работы реализованного класса с интерфейсом. В процесс выполнения этой объемной работы бакалавры второго курса расширили познания в сфере реализации классов, разработки и построения интерфейса, а также получение навыков тестирования.

Использованные в ходе выполнения проектно-технологической подходы по групповому разделению задач и делегированию ролей потребовали высокой координации деятельности, особенно на этапе консолидации результатов и их анализа.

Все материалы необходимые для выполнения практики студентами, обучающимися по направлению бакалавриата, были направлены в электронной форме, т.к. процесс прохождения практики дистанционный.

Проверка результатов выполнения практики в части работы программы была произведена с помощью записи экранного видео, в котором было показано выполнение всех требований для разработанной программы из

методического указания. В части отчета, были присланы на электронную почту отчеты о групповой работе, после проверки были отправлены либо файлы с замечаниями, либо сообщение о зачтении работы.

СПИСОК ЛИТЕРАТУРЫ

1. ГОСТ 7.32–2001 «Отчет о научно-исследовательской работе. Структура и правила оформления»;
2. ГОСТ 2.105–95 «Общие требования к текстовым документам»;
3. ГОСТ Р 7.0.5–2008 «Библиографическая ссылка. Общие требования и правила составления»;
4. ГОСТ 7.1–2003 «Библиографическая запись. Библиографическое описание. Общие требования и правила составления»;
5. ГОСТ 7.80–2000. «Библиографическая запись. Заголовок. Общие требования и правила составления».
6. Знакомство с Git и GitHub: руководство для начинающих [Электронный ресурс] URL: <https://medium.com/nuances-of-programming/%D0%B7%D0%BD%D0%B0%D0%BA%D0%BE%D0%BC%D1%81%D1%82%D0%B2%D0%BE-%D1%81-git-%D0%B8-github-%D1%80%D1%83%D0%BA%D0%BE%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE-%D0%B4%D0%BB%D1%8F-%D0%BD%D0%B0%D1%87%D0%B8%D0%BD%D0%B0%D1%8E%D1%89%D0%B8%D1%85-54ea2567d76c> (дата обращения: 06.07.2020)
7. Основные принципы ООП [Электронный ресурс] URL: <https://javarush.ru/groups/posts/1966-principih-obhhektno-orientirovannogo-programmirovaniija> (дата обращения: 07.07.2020)
8. Что такое классы в объектно-ориентированном программировании [Электронный ресурс] URL: <https://thecode.media/oop-class/> (дата обращения: 07.07.2020)
9. Основы программирования на языках Си и C++ для начинающих [Электронный ресурс] URL: <http://cppstudio.com/> (дата обращения: 08.07.2020)
10. D.S. Malik C++ Programming: From Problem Analysis to Program Design, Eight Edition 2017.

ПРИЛОЖЕНИЕ А.

КРАТКОЕ РУКОВОДСТВО ПО РАБОТЕ С GIT

Настройка

Настроить свой профиль Git на компьютере (можно глобально, можно будет и после операции clone локально для проекта): поставить user.name ваше <Имя Фамилия> на английском (пример: Ivan Ivanov) и указать свой user.email (если нет или не хотите разглашать, можете поставить фиктивный, по типу ivan.ivanov@notreal.com). Эти данные автоматически указываются в каждом сделанном вами коммите, что позволяет знать, кто сделал этот коммит. Сделать это можно командами:

```
git config --global user.name "Ivan Ivanov"
```

(так как в записи присутствует пробел, её необходимо писать в кавычках)

```
git config --global user.email ivan.ivanov@notreal.com
```

Примечание: чтобы посмотреть текущее значение этих данных, то необходимо набрать просто

```
git config --global user.name
```

и в терминал выведется значение. Также и для email.

Создание проекта

Находясь в папке, куда захотите создать папку с репозиторием, запустите терминал и наберите команду:

```
git Init
```

Проверить состояние репозитория:

```
git status
```

Если есть красные поля (непроиндексированные файлы, или модифицированные и тд), тогда проиндексировать их для подготовки к коммиту:

```
git add .
```

Примечание: `git add` со значением `.` означает проиндексировать все изменения в репозитории на текущий момент.

После индексации изменений, повторить

`git status`

Теперь все поля должны стать зелёными, что означает, что они будут зафиксированы в новом коммите. Выполнить коммит:

`git commit -m "<короткое сообщение коммита, отражающее суть произведенных изменений>"`

`git push`

Примечание: на данном этапе, в терминале у вас могут запросить логин и пароль. Нужно вводить логин и пароль от вашего GitHub аккаунта. Так он авторизируется у вас в системе, и поймёт, что у вас действительно есть права для выполнения данного действия. Если же этих прав у вас нет, выполнение команды будет отклонено (`access denied`).

Просмотреть историю коммитов можно командой:

`git log --oneline`

Примечание: `git log` имеет множество входных параметров и аргументов. В данном примере, параметр `--oneline` позволяет вывести самую короткую информацию о коммите одной строкой. Более подробно о данной команде можно почитать самостоятельно в интернете или книге.

ПРИЛОЖЕНИЕ Б.

ПРИМЕР ОТЧЕТ БРИГАДЫ ПО ПРОЕКТУ

Вариант 1. Разработать консольное приложение, в котором должен быть реализован простой класс «Автомобиль» (легковой/грузовой, мощность, объем двигателя, цена, марка).

Введение

Объектно ориентированное программирование (ООП) – расширение структурного программирования, в котором основными концепциями являются понятия классов и объектов.

Класс — это механизм для создания объектов. В этом смысле класс лежит в основе многих свойств C++. Хотя более детально понятие класса раскрывается в следующих главах, оно столь фундаментально для программирования на C++, что краткий обзор здесь необходим.

Процесс разработки программных продуктов состоит из следующих этапов:

- постановка задачи;
- разработка класса;
- написание и отладка программы;
- тестирование программы.

Для разработки программ существуют различные парадигмы программирования, в данном случае использовался парадигма объектно ориентированного программирования.

1. Постановка задачи

Программа должна быть разработана для работы в операционной системе Microsoft Windows.

Средой разработки была выбрана Microsoft Visual Studio 2017, язык программирования C++.

Необходимо разработать консольное приложение, в котором должен быть реализован простой класс в соответствии с вариантом задания.

Требования к классу:

- Класс должен содержать конструктор без параметров. В конструкторе должны инициализироваться переменные класса значением по умолчанию.

- Класс должен содержать два или более конструкторов с параметрами. В конструкторах должны инициализироваться переменные класса значениями, которые переданы в параметрах конструктора. Если для какой-либо переменной класса в параметрах конструктора не предусмотрен параметр, то переменная должна инициализироваться значением по умолчанию.

- Класс должен содержать деструктор. В деструкторе все переменные класса должны быть затерты (присвоены недействительные значения).

Деструктор является дополнением и продолжением механизма конструктора. Деструктор – это определяемая пользователем функция – член класса. Главное назначение деструктора – освободить ресурсы, выделенные в конструкторе во время жизни объекта.

- Класс должен содержать три или более (в соответствии с вариантом задания) переменных класса. Переменные класса должны быть реализованы в трех секциях: public, protected и private.

- Класс должен содержать одну или более переменную со строковыми данными. Строковые данные должны задаваться только с использованием массива символов (`char str1[100]` или `wchar_t str2[200]`).

- В классе должны быть реализованы два метода: Input и Output. Метод Input должен выполнять ввод данных класса с клавиатуры. Метод Output должен печатать на экране данные класса. Методы должны располагаться в секции public.

Требования к программе:

- Все объекты класса в программе должны создаваться через объявление экземпляра класса на стеке программы.
- В основной функции программы создать объект класса с вызовом конструктора без параметров.
- В основной функции программы создать несколько объектов класса с вызовом конструкторов с параметрами. Должно быть создано столько объектов, сколько реализовано в классе конструкторов с параметрами.
- В основной функции программы объявить массив объектов класса (не более 5).
- В цикле выполнить заполнение массива объектов. Для этого необходимо для каждого объекта в массиве вызвать функцию Input.
- В цикле выполнить печать массива объектов. Для этого необходимо для каждого объекта в массиве вызвать функцию Output.

Распределение работы в команде

Студент №1	Студент №2	Студент №3
Разработка класса Auto	Помощь в разработке класса и создание функции для расчета стоимости автомобиля	Тестирование и работа с файлами

2. Выбор решения

Разработанный класс состоит из следующих функций, которые вызываются в основной функции main:

- 1) Input
- 2) Output

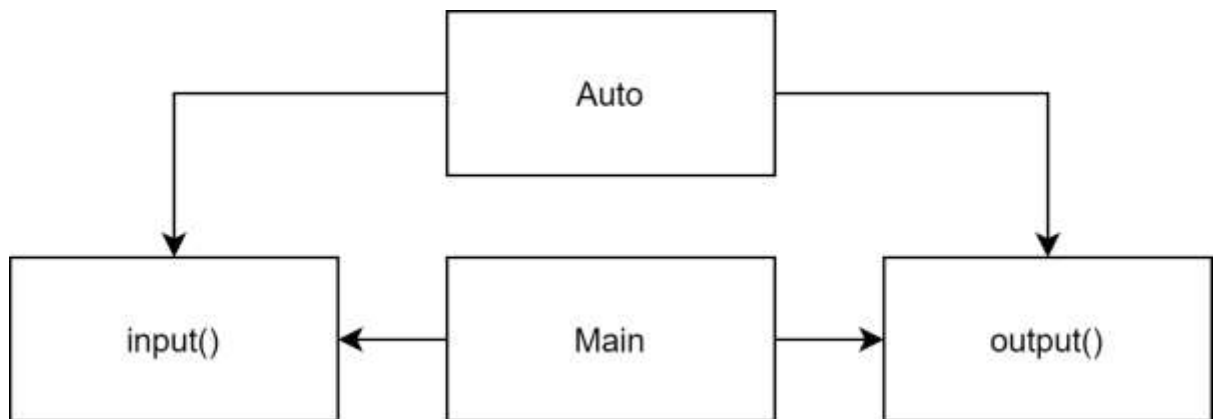


Рисунок 1 - Структура функций программы

- Функция Input используется для ввода данных класса с клавиатуры.
- Функция Output используется для печати на экране данных класса.

Кроме этого, в классе реализованы конструкторы с разным количеством параметров и деструктор.



Рисунок 2 - Схема данных

3. Описание программы

В качестве среды программирования был выбран программный продукт Microsoft Visual studio 2017.

В программе необходимо было реализовать класс Автомобиль. Для этого были написаны заголовочный файл autoClass.h со всеми нужными переменными класса, объявленными функциями и файл с исходным кодом autoClass.cpp, содержащий всю реализацию функций, конструкторы и деструкторы. Файл AUTO_CLASS.cpp содержит в себе основную функцию main().

Задачей студента №1 было разработать класс Автомобиль, его конструктор и деструктор.

В классе использовано три конструктора: первый конструктор – без параметров:

```
Auto::Auto()
{
    init();
    strcpy(type, "-");
    strcpy(brand, "-");
    power = 0;
    volume = 0;
    price = 0;
}
```

Второй конструктор – с двумя параметрами:

```
Auto::Auto(char
typearg[50],
char
brandarg[50]) {
    init();

    strcpy(type, typearg);
    strcpy(brand, brandarg);
    power = 0;
    volume = 0;
```

```

        price = 0;
    }

```

Третий конструктор – с пятью параметрами:

```

Auto::Auto(char
typearg[50],
char
brandarg[50],
float power,
int volume, int
price) {

        init();

        strcpy(type, typearg);
        strcpy(brand, brandarg);
        this->power = power;
        this->volume = volume;
        this->price = price;
    }

```

Так же был реализован один деструктор для уничтожения объекта и очищения памяти.

```

Auto::~Auto()
{

        delete type;
        delete brand;
        power = 0;
        volume = 0;
        price = 0;
    }

```

Функция void Auto::input() необходима при вводе пользователем данных, которые будут использованы для расчета стоимости автомобиля.

```

void
Auto::input()
{

        cout << " Enter type: " << endl;
        cin >> type;
        <...>
        count_price();
    }

```

```
}
```

Функция `void Auto::foutput()` предназначена для вывода данных, которые ввел пользователь, в консоль, а также для записи их в файл `auto.txt`.

```
void
Auto::foutput()
{
    ofstream fout("auto.txt",
ios_base::app);
    cout << endl << " Type is : " << type <<
endl;
    fout << endl << " Type is : " << type <<
endl;
                                <...>
}
```

Функция `void Auto::count_price()` рассчитывает стоимость автомобиля, исходя из данных, которые ввел пользователь. В калькуляторе предложен расчет цены для трех конкретных марок автомобилей (BMW, Lada, Lexus) и других марок. Калькулятор рассчитывает цену по типам: `car` и `truck`, а так же другие типы автомобилей.

```
Void Auto::count_price()
{
    int bmw = 160;
    <...>

    int car = 50000;
    <...>

    if (strcmp(brand, "bmw") == 0) {
        price = bmw * power *
volume;
    }
    <...>

    if (strcmp(type, "car") == 0) {
        price += car;
    }
    <...>
}
```

Функция `void Auto::output()` предназначена для вывода данных объекта класса.

```
void Auto::output (
    cout << endl << " Type is : " << type
    << endl;
    <...>
}
```

Функция `void Auto::init()` осуществляет выделение памяти для переменных `type` и `brand`.

Задачей студента №2 было написать функцию для класса `Auto`, подсчитывающую стоимость автомобиля, исходя из введенных значений(марка авто, его тип, мощность двигателя и т.д.).

Данная функция была объявлена в файле `autoClass.h`, ее реализация находится в файле `autoClass.cpp`.

Стоимость машины будет рассчитываться по следующей формуле:

Стоимость = мощность двигателя * объем двигателя * коэффициент, зависящий от марки авто и + значение, зависящее от типа авто.

Мощность двигателя и объема двигателя заданы как переменные класса и вводятся с клавиатуры в ходе выполнения программы, коэффициенты будем задавать в функции `count_price()`.

```
int bmw = 160;
int lada = 100;
int lexus = 500;
int another = 700;

int car = 50000;
int truck = 100000;
int any = 300000;
```

Далее в коде будем проверять введенный тип авто и его марку с помощью условий. В зависимости от введенной марки мы будем выполнять умножение цены на тот или иной коэффициент и прибавлять значение, в зависимости от типа авто.

```
if (strcmp(brand, "bmw") == 0) {
    price = bmw * power * volume;
}
else if (strcmp(brand, "lada") == 0) {
    price = lada * power * volume;
}
```

```

else if (strcmp(brand, "lexus") == 0) {
    price = lexus * power * volume;
}
else{
    price = another * power * volume;
}

if (strcmp(type, "car") == 0) {
    price += car;
}
else if (strcmp(type, "truck")==0) {
    price += truck;
}
else {
    price += any;
}

```

Функция count_price() будет вызываться сразу после ввода данных авто в input()

```

void Auto::input() {
    cout << " Enter type: " << endl;
    cin >> type;
    cout << " Enter brand: " << endl;
    cin >> brand;
    cout << " Enter power: " << endl;
    cin >> power;
    cout << " Enter volume: " << endl;
    cin >> volume;
    count_price();
}

```

Значение стоимости машины подсчитано и записано в поле price класса Auto.

Задачей Студента № 3 Было разработать foutput для работы с файлами, а точнее функция должна записывать значения объекта класса в файл "auto.txt" и протестировать готовую целую программу. По заданию нужно было создать класс описывающий автомобиль. Для этого мы создали переменные внутри класса type для типа автомобиля, brand для марки автомобиля, power для мощности двигателя, volume для объема двигателя. В классе присутствует различные конструкторы и деструкторы и функции для работы с классом

```

void Auto::foutput() {
    ofstream fout("auto.txt", ios_base::app);
    cout << endl << " Type is : " << type << endl;
    fout << endl << " Type is : " << type << endl;
    cout << " Brand is : " << brand << endl;
    fout << " Brand is : " << brand << endl;
    cout << " Power is : " << power << endl;
}

```

```

fout << " Power is : " << power << endl;
cout << " Volume is : " << volume << endl;
fout << " Volume is : " << volume << endl;

cout << " Price is : " << price << endl;
fout << " Price is : " << price << endl;
}

```

В строчках с `fout` можно увидеть запись значений в файл.

```

Auto* cars = new Auto[3];
for (int i = 0; i < 3; i++) {
    cars[i].input();
    cars[i].foutput();
}

```

В основной функции `main` вызываем метод класса `Auto` `foutput` для записи введённых с клавиатуры значений в файл.

4. Схема программы

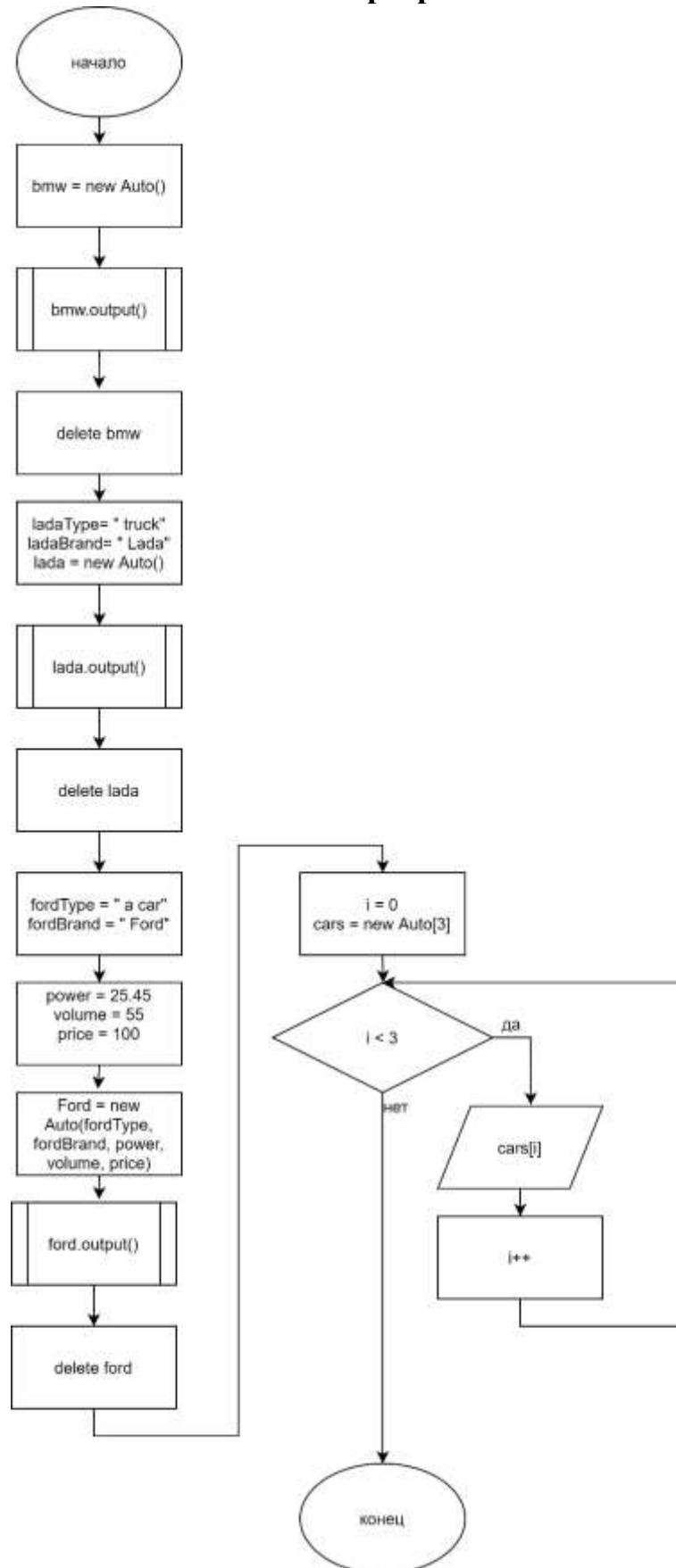


Рисунок 3 - Схема программы

5. Отладка и тестирование

Microsoft Visual Studio 2017 предоставляет все средства, которые необходимы для разработки и отладки программы. Для отладки программы использовались следующие инструменты этого программного продукта: точки останова, анализ содержимого локальных и глобальных переменных, пошаговое выполнение кода.

Тестирование программы производилось во время ее написания, а также после завершения создания программы.

Результаты тестирования приведены в Приложении А на рисунках А.1-А.5.

6. Работа с Git

GitHub — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. Кроме размещения кода, участники могут общаться, комментировать правки друг друга, а также следить за новостями знакомых. С помощью широких возможностей Git программисты могут объединять свои репозитории — GitHub предлагает удобный интерфейс для этого и может отображать вклад каждого участника в виде дерева. Для проектов есть личные страницы, небольшие Вики и система отслеживания ошибок. Прямо на сайте можно просмотреть файлы проектов с подсветкой синтаксиса для большинства языков программирования.

Работа с Git Студента №1

Был создан репозиторий на сервисе GitHub. С помощью Git репозиторий был клонирован на компьютер, благодаря команде «git clone».

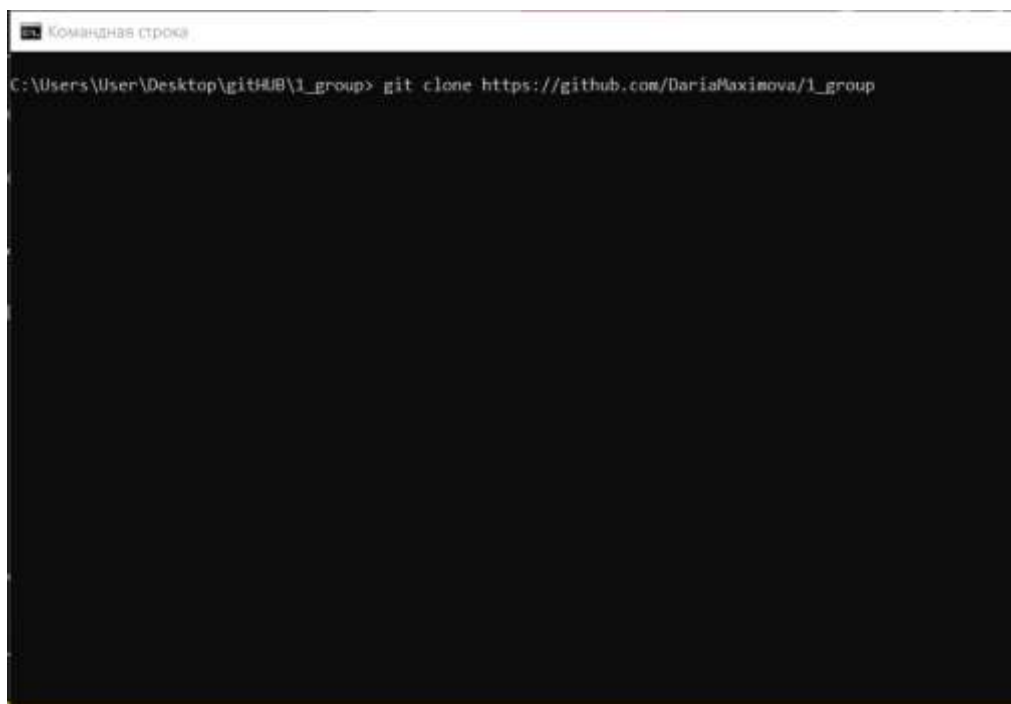


Рисунок 4 – Команда «git clone»

После добавления всех файлов необходимо их добавить на GitHub. Это можно сделать с помощью команды «git add .».



Рисунок 5 – Команда «git add .»

После этого нужно добавить коммиты для всех изменений с помощью команды «git commit».

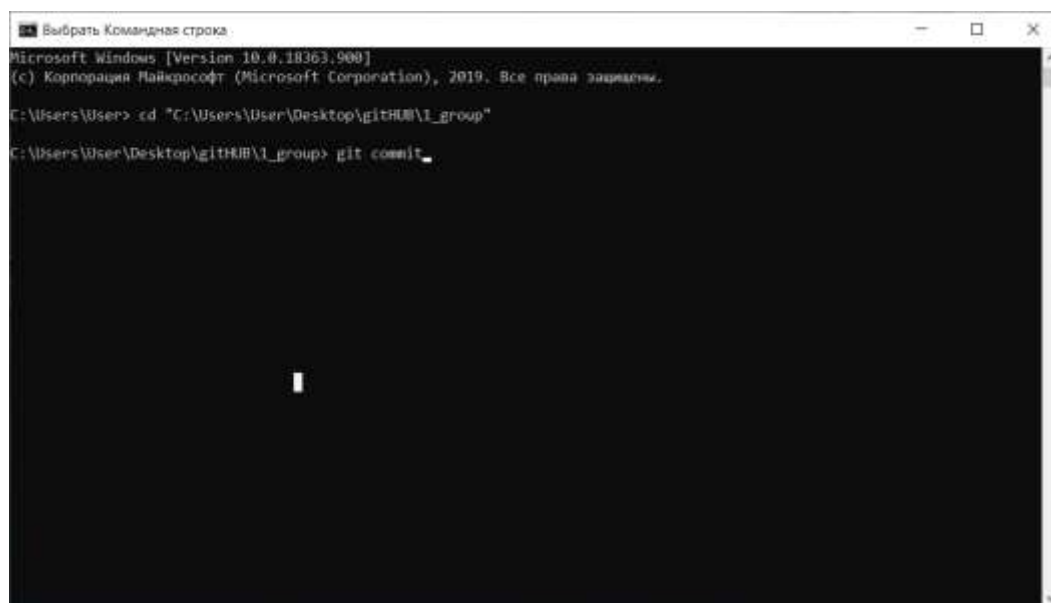


Рисунок 6 – Команда «git commit»

Необходимо отобразить все изменения на GitHub с помощью команды «git push origin master».



Рисунок 7 – Команда «git push origin master»

Для получения изменений с сервера GitHub и сливания изменений в локальную копию проекта, используется команда «git pull».

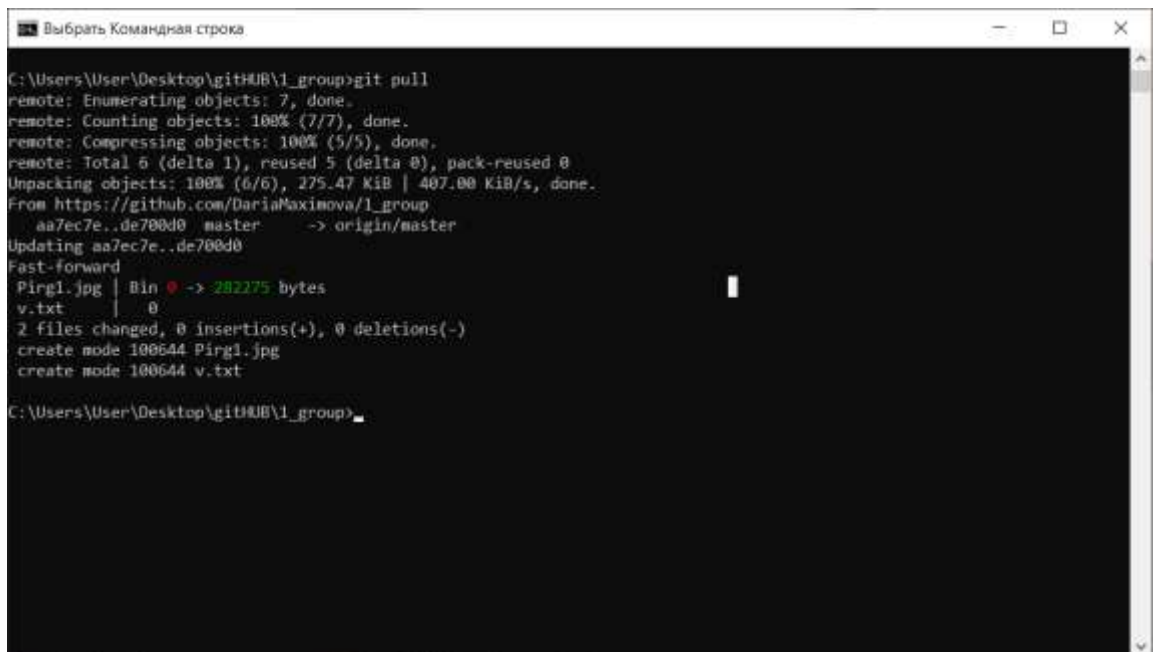


Рисунок 8 – Команда «git pull»

На GitHub отображаются все комментарии, которые были оставлены при работе над проектом. Видно, какой пользователь и когда оставил commit.



Рисунок 9 – Список commit

Работа с Git Студента №2

Для командной работы была использована система контроля версий Git и сервис онлайн хостинга репозитория GitHub.

С помощью команды `git clone` репозиторий был скопирован на компьютер.

```
( [redacted] )>git clone https://github.com/DariaMaximova/1_group.git
Cloning into '1_group'...
remote: Enumerating objects: 85, done.
remote: Counting objects: 100% (85/85), done.
remote: Compressing objects: 100% (62/62), done.
remote: Total 85 (delta 23), reused 56 (delta 14), pack-reused 0
Unpacking objects: 100% (85/85), done.
```

Рисунок 10 – Команда `git clone`

После добавления необходимого функционала к проекту, необходимо содержимое рабочей дериктории в индекс с помощью команды `git add`.

```
git add .
```

Рисунок 11 – Команда `git add`

Далее файлы необходимо проиндексировать и закоммитить с помощью команды `git commit -m "comment"`

```
( [redacted] )>git commit -m "Added function count_price"
```

Рисунок 12 – Команда `git commit`

Теперь необходимо все изменения отправить на сервер GitHub с помощью функции `git push origin master`

```
>git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 250 bytes | 250.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/DariaMaximova/1_group.git
de700d0..575520a master -> master
```

Рисунок 13 – Команда `git push`

Команду `git pull` была использована для получения всех изменений с сервера GitHub и сливания их в локальную папку с проектом.

Работа с Git Студента №3

Мы использовали программу Git и GitHub, чтобы работа в команде была более удобной.

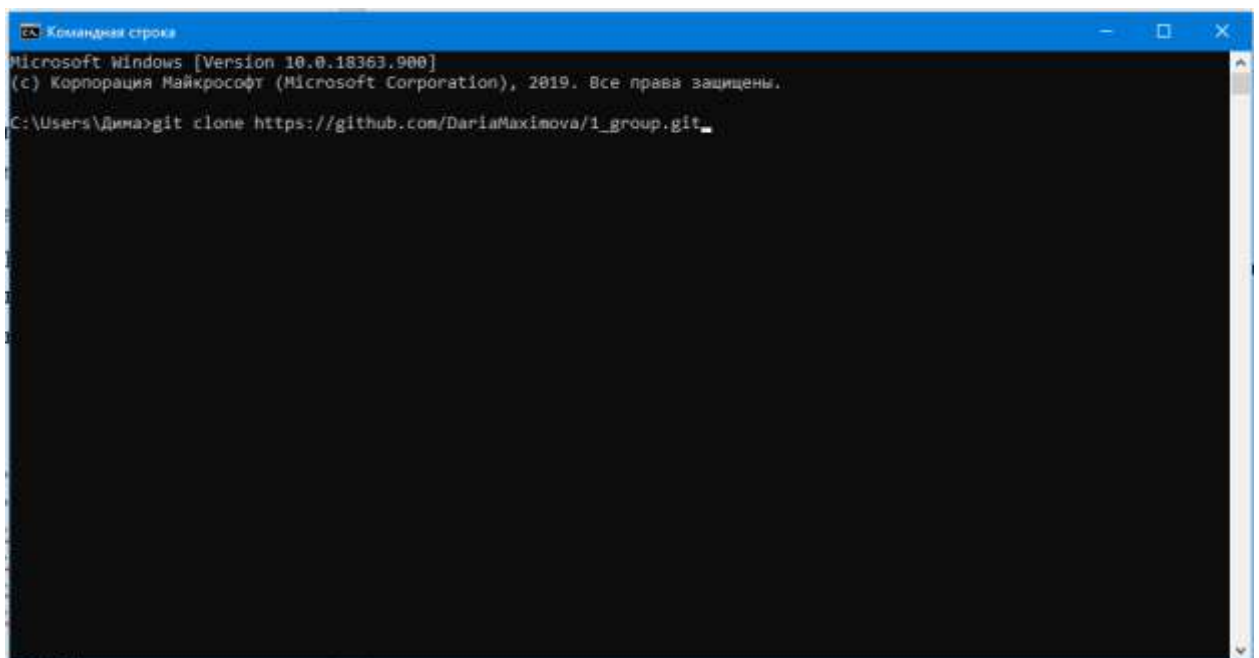
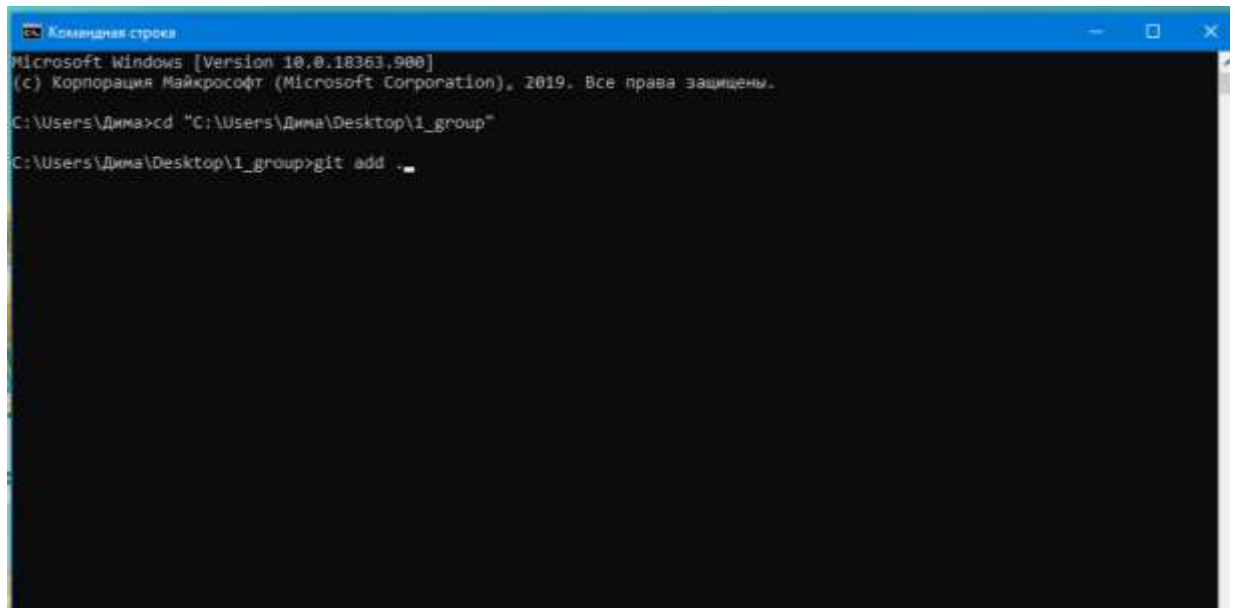


Рисунок 14 - Склонировали удаленный репозиторий себе на компьютер.

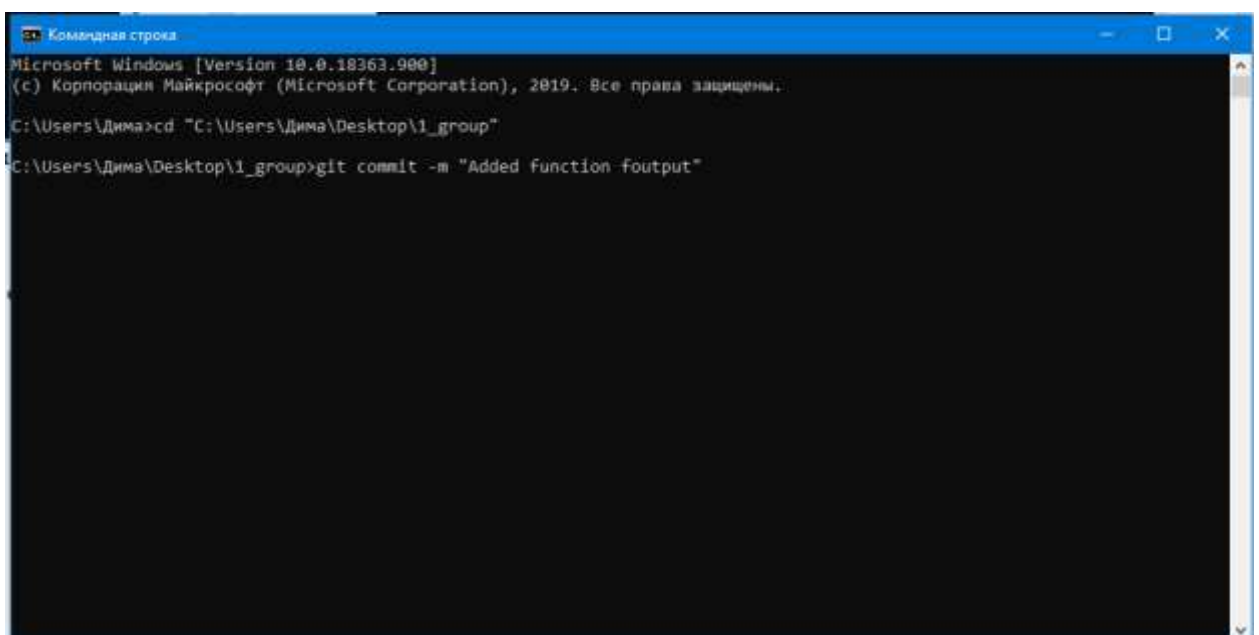
После внесения изменений в проект необходимо все изменения отправить на сервер Github.



```
Командная строка
Microsoft Windows [Version 10.0.18363.900]
(c) Корпорация Майкрософт (Microsoft Corporation), 2019. Все права защищены.

C:\Users\Дима>cd "C:\Users\Дима\Desktop\i_group"
C:\Users\Дима\Desktop\i_group>git add .
```

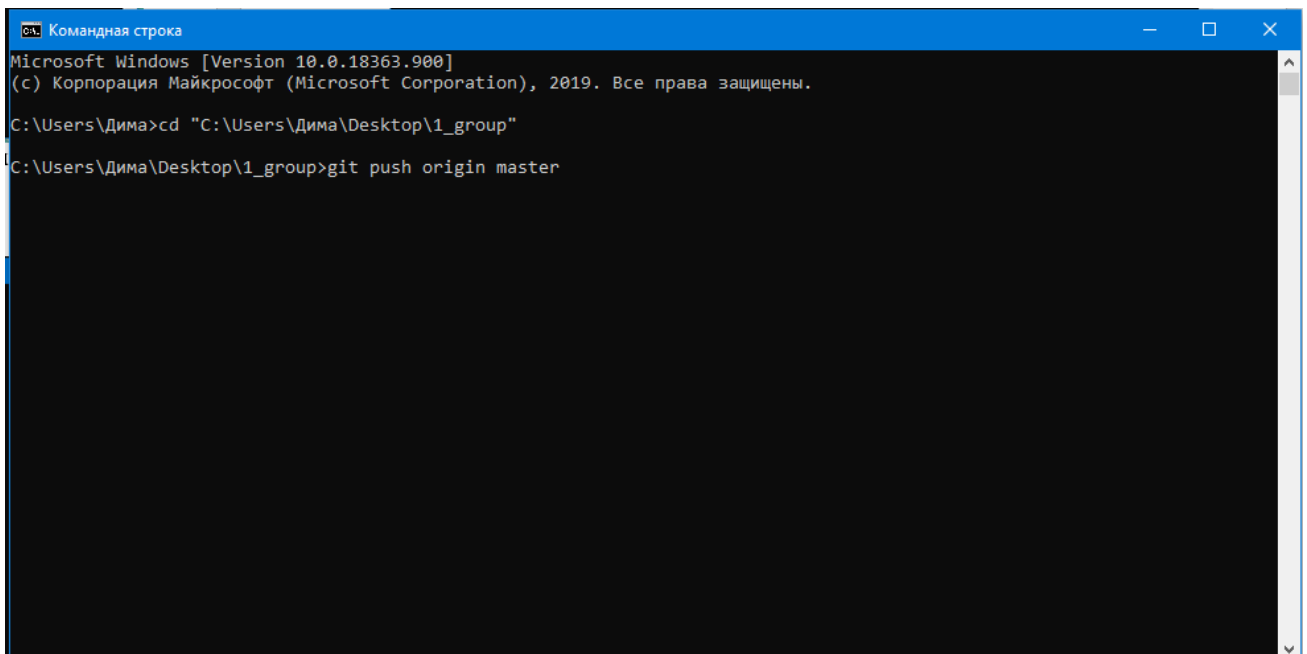
Рисунок 15 - Для того чтобы добавить в индекс все файлы, в которых были изменения, а также новые файлы использовалась команда `git add`.



```
Командная строка
Microsoft Windows [Version 10.0.18363.900]
(c) Корпорация Майкрософт (Microsoft Corporation), 2019. Все права защищены.

C:\Users\Дима>cd "C:\Users\Дима\Desktop\i_group"
C:\Users\Дима\Desktop\i_group>git commit -m "Added function foutput"
```

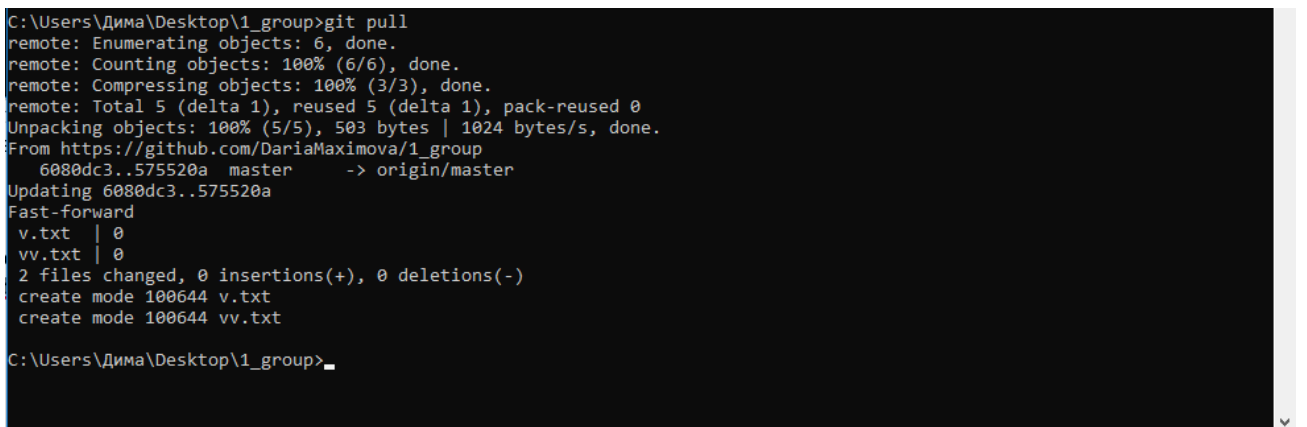
Рисунок 16 - Для того чтобы одновременно проиндексировать и закоммитить файлы использовалась команда `git commit -m "Added function foutput"`.



```
Командная строка
Microsoft Windows [Version 10.0.18363.900]
(c) Корпорация Майкрософт (Microsoft Corporation), 2019. Все права защищены.

C:\Users\Дима>cd "C:\Users\Дима\Desktop\1_group"
C:\Users\Дима\Desktop\1_group>git push origin master
```

Рисунок 17 - Чтобы загрузить данные на наш удаленный репозиторий использовалась команда `git push origin master`.



```
C:\Users\Дима\Desktop\1_group>git pull
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 5 (delta 1), reused 5 (delta 1), pack-reused 0
Unpacking objects: 100% (5/5), 503 bytes | 1024 bytes/s, done.
From https://github.com/DariaMaximova/1_group
   6080dc3..575520a  master    -> origin/master
Updating 6080dc3..575520a
Fast-forward
 v.txt | 0
 vv.txt | 0
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 v.txt
 create mode 100644 vv.txt
C:\Users\Дима\Desktop\1_group>_
```

Рисунок 18 - Для получения изменения сервера от других участников использовалась команда `git pull`.

Заключение

В ходе выполнения выполнения практического задания были получены навыки работы с классами. Получены и усовершенствованы навыки в среде программирования Microsoft Visual Studio 2017. Получен опыт работы с Git и Github.

В процессе выполнения практического задания создан класс, описывающий характеристики автомобиля, а также создана функция для подсчета цены для автомобиля, исходя из введенных значений.

Список литературы

1. Стенли Липпман – Основы программирования на C++.
2. Герберт Шилдт - Самоучитель C++.
3. Википедия. <https://ru.wikipedia.org/wiki/GitHub>

Приложение А

```
Call 1st constructor
Enter type:
truck
Enter brand:
tesla
Enter power:
600
Enter volume:
12
Enter price:
1000000

Type is : truck
Brand is : tesla
Power is : 600
Volume is : 12
Price is : 1000000
```

Рисунок А.1 – Вводим значения с клавиатуры

```
Call 1st constructor

Type is : -
Brand is : -
Power is : 0
Volume is : 0
Price is : 0

Memory has been cleaned. Good bye.
Call 2nd constructor

Type is : truck
Brand is : Lada
Power is : 0
Volume is : 0
Price is : 0

Memory has been cleaned. Good bye.
Call 3d constructor

Type is : a car
Brand is : Ford
Power is : 25.45
Volume is : 55
Price is : 100

Memory has been cleaned. Good bye.
```

Рисунок А.2 – Вызываем конструкторы с разным числом параметров.

Приложения (Демонстрация работы программы):

```
Call 1st constructor
Call 1st constructor
Enter type:
truck
Enter brand:
tesla
Enter power:
120
Enter volume:
4
Enter price:
500000

Type is : truck
Brand is : tesla
Power is : 120
Volume is : 4
Price is : 500000

Enter type:
car
Enter brand:
audi
Enter power:
500
Enter volume:
4
Enter price:
100000

Type is : car
Brand is : audi
Power is : 500
Volume is : 4
Price is : 100000
```

Рисунок А.3 - Тест №1

```
Call 3d constructor

Type is : truck
Brand is : Ford
Power is : 25.45
Volume is : 55
Price is : 100

Memory has been cleaned. Good bye.
```

Рисунок А.4 - Тест №2

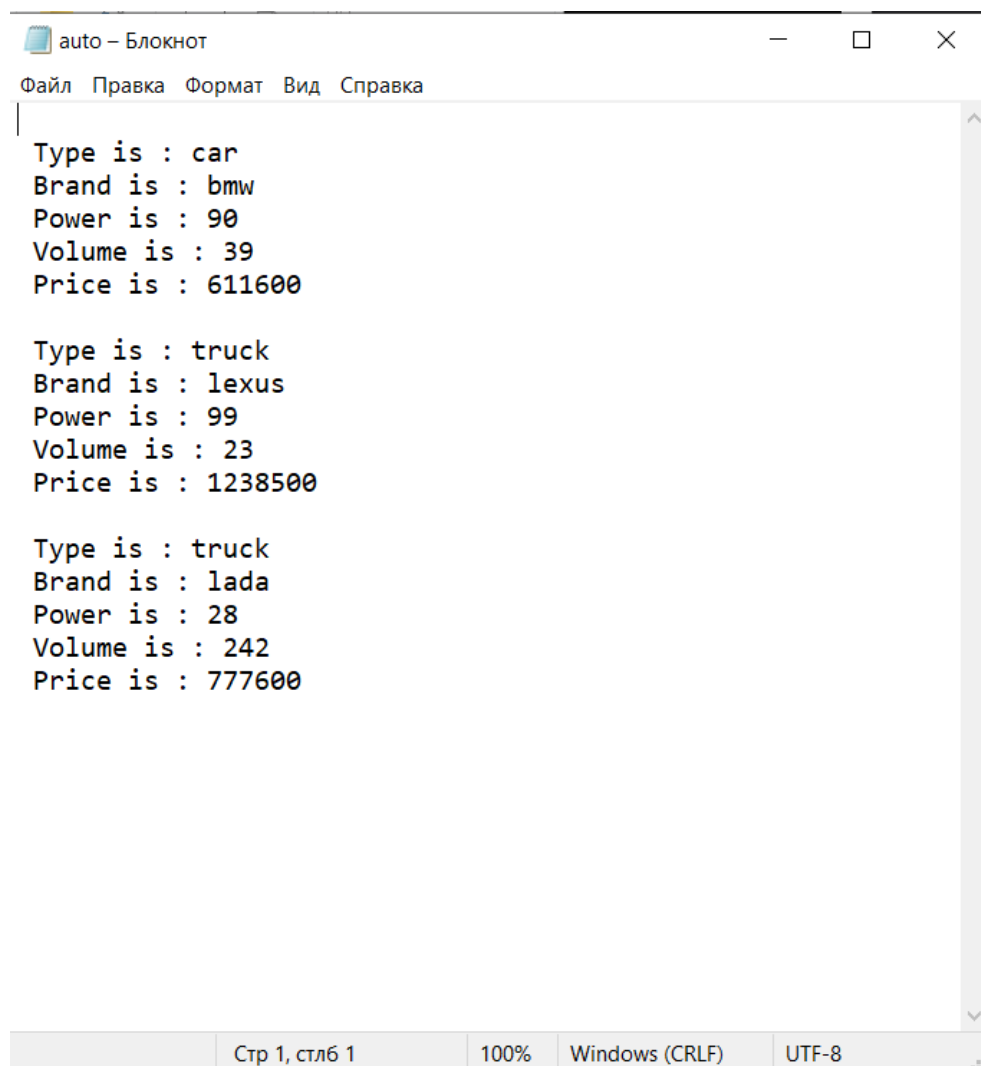


Рисунок А.5 – Файл с записанными данными

Приложение В. Листинг программы

Файл AUTO_CLASS.cpp

```
#define _CRT_SECURE_NO_WARNINGS
#include "pch.h"
#include "autoClass.h"
#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    ofstream fout ("auto.txt", ios_base::trunc);

    Auto* BMW = new Auto ();
    BMW->output();
    delete BMW;

    char ladaType[50]= " truck";
    char ladaBrand[50]= " Lada";
    Auto* Lada = new Auto(ladaType, ladaBrand);
    Lada->output();
    delete Lada;

    char fordType[50] = " a car";
    char fordBrand[50] = " Ford";
    float power = 25.45;

    int volume = 55;
    int price = 100;
    Auto* Ford = new Auto(fordType, fordBrand, power, volume,
price);
    Ford->output();
    delete Ford;

    Auto* cars = new Auto[3];
    for (int i = 0; i < 3; i++) {
        cars[i].input();
        cars[i].foutput();
    }
    fout.close();
}
```

Файл autoClass.cpp

```
#define _CRT_SECURE_NO_WARNINGS
#include "pch.h"
#include "autoClass.h"
#include <iostream>
#include <fstream>
```

```

#include <stdio.h>
#include < string.h >

using namespace std;

Auto::Auto() {

    init();
    cout << " Call 1st constructor\n";
    strcpy(type, "-");
    strcpy(brand, "-");
    power = 0;
    volume = 0;
    price = 0;
}

Auto::Auto(char typearg[50], char brandarg[50]) {
    init();
    cout << " Call 2nd constructor\n";
    strcpy(type, typearg);
    strcpy(brand, brandarg);
    power = 0;
    volume = 0;
    price = 0;
}

Auto::Auto(char typearg[50], char brandarg[50], float power, int
volume, int price) {
    init();
    cout << " Call 3d constructor\n";
    strcpy(type, typearg);
    strcpy(brand, brandarg);
    this->power = power;
    this->volume = volume;
    this->price = price;
}

Auto::~Auto()
{
    cout << " Memory has been cleaned. Good bye." << endl;
    delete type;
    delete brand;
    power = 0;
    volume = 0;
    price = 0;
}

void Auto::input() {
    cout << " Enter type: " << endl;
    cin >> type;
    cout << " Enter brand: " << endl;
    cin >> brand;
    cout << " Enter power: " << endl;

```

```

        cin >> power;
        cout << " Enter volume: " << endl;
        cin >> volume;
        count_price();
    }

    void Auto::foutput() {
        ofstream fout("auto.txt", ios_base::app);
        cout << endl << " Type is : " << type << endl;
        fout << endl << " Type is : " << type << endl;
        cout << " Brand is : " << brand << endl;
        fout << " Brand is : " << brand << endl;
        cout << " Power is : " << power << endl;
        fout << " Power is : " << power << endl;
        cout << " Volume is : " << volume << endl;
        fout << " Volume is : " << volume << endl;

        cout << " Price is : " << price << endl;
        fout << " Price is : " << price << endl;
    }

    void Auto::count_price() {
        int bmw = 160;
        int lada = 100;
        int lexus = 500;
        int another = 700;

        int car = 50000;
        int truck = 100000;
        int any = 300000;

        if (strcmp(brand, "bmw") == 0) {
            price = bmw * power * volume;
        }
        else if (strcmp(brand, "lada") == 0) {
            price = lada * power * volume;
        }
        else if (strcmp(brand, "lexus") == 0) {
            price = lexus * power * volume;
        }
        else{
            price = another * power * volume;
        }

        if (strcmp(type, "car") == 0) {
            price += car;
        }
        else if (strcmp(type, "truck")==0) {
            price += truck;
        }
        else {
            price += any;
        }
    }

```

```

    }
}

void Auto::output() {
    cout << endl << " Type is : " << type << endl;
    cout << " Brand is : " << brand << endl;
    cout << " Power is : " << power << endl;
    cout << " Volume is : " << volume << endl;
    cout << " Price is : " << price << endl << endl;
}

void Auto::init() {
    type = new char[50];
    brand = new char[50];
}

```

Файл autoClass.h

```

#pragma once
#define _CRT_SECURE_NO_WARNINGS
#include "pch.h"
#include <iostream>

class Auto {
public:
    Auto();
    Auto(char typearg[50], char brandarg[50]);
    Auto(char typearg[50], char brandarg[50], float power, int
volume, int price);
    ~Auto();
    void input();
    void foutput();
    void count_price();
    void output();
    char *type;
    char *brand;
protected:
    float power;
    int volume;
private:
    void init();
    int price;
};

```