

# Технологии программирования

## ЛАБОРАТОРНАЯ РАБОТА № 5

Разработка консольного приложения с применением технологий управления проектом и командой, контроля версий и сборки проекта. Для выполнения лабораторной работы сформировать группы из 2-3 человек и сообщить сведения о составе группы преподавателю и лектору.

ЛАБОРАТОРНАЯ РАБОТА № 5.....	1
Методические рекомендации.....	2
Управление проектами.....	2
Порядок действий при создании проекта в Github Projects.....	2
Порядок действий при создании проекта в Trello.....	3
Порядок действий при создании проекта в Targetprocess.....	3
Непрерывная сборка и тестирование приложений.....	4
Подключение Travis-CI на примере простого проекта.....	5
Сборка проекта с Github Action.....	6
Настройка Unit-тестов для проекта на C.....	8
Unit тесты на основе фреймворка GoogleTest.....	8
Unit тесты на основе фреймворка CUnit.....	8
Проверка покрытия кода тестами.....	8
Литература по Travis-CI.....	9
ЗАДАНИЕ 1. МЕНЕДЖМЕНТ ПРОЕКТА В СТИЛЕ KANBAN.....	9
Упражнение 1.1. Изучить возможности управления проектами.....	9
Упражнение 1.2. Создать проект и распределить задачи.....	9
ЗАДАНИЕ 2. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ.....	10
ЗАДАНИЕ 3. ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ БАЗЫ ДАННЫХ И НАПОЛНЕНИЕ ДАННЫМИ.....	10
Упражнение 3.1. Изучить функционал Vertabelo.....	10
Упражнение 3.2. Спроектировать базу данных.....	10
Упражнение 3.3. Сгенерировать sql-скрипт для sqlite.....	11
ЗАДАНИЕ 4. НЕПРЕРЫВНАЯ СБОРКА ПРОЕКТА И ТЕСТИРОВАНИЕ.....	11
ЗАДАНИЕ 5. РАЗРАБОТКА ПРИЛОЖЕНИЯ.....	11
Упражнение 5.1. Подключить репозиторий группы.....	11
Упражнение 5.2. Документирование проекта.....	11
Упражнение 5.3. Разработать приложение.....	12
ВАРИАНТЫ:.....	12
Вариант 1. «Автопарк».....	12
Вариант 2. «Рыболовная флотилия».....	13
Вариант 3. «Воздушный извозчик».....	14
Вариант 4. «Ипподром».....	15
Вариант 5. «Музыкальный салон».....	16
Вариант 6. «Туристическое бюро».....	17
Вариант 7. «Пушной аукцион».....	18
Вариант 8. «Цветочная оранжерея».....	19
Вариант 9. «Парфюмерный базар».....	20
Вариант 10. «Автомастерские».....	21

# Методические рекомендации

## Управление проектами

Для управления проектами применяются различные системы и решения, например Github Projects, Trello, Jira, Targetprocess и другие.

Руководства и рекомендации по системам управления проектами:

- Github Project<sup>beta</sup>  
<https://docs.github.com/en/issues/trying-out-the-new-projects-experience/quickstart>
- Github Project  
<https://help.github.com/en/github/managing-your-work-on-github/about-project-boards>
- Trello  
<https://trello.com/ru/guide>
- Targetprocess  
Познакомиться с видео <https://youtu.be/EaqCxDmEeSw> и руководством <https://www.targetprocess.com/guide/how-to-start/basic-scrum/>.

## Порядок действий при создании проекта в Github Projects

1. Ознакомьтесь с документацией по управления проектами в GitHub (для Beta версии <https://docs.github.com/en/issues/trying-out-the-new-projects-experience/quickstart> и для текущей версии — [Managing project boards](#)).
2. Тимлид команды в репозитории создаёт Проект, например для текущей версии Github Project (см. рис. ниже):

maryiad / project

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security Insights Settings

### Create a new project

Coordinate, track, and update your work in one place, so projects stay transparent and on schedule.

**Project board name**

test proejct Указать имя проекта

**Description (optional)**

Выбрать шаблон проекта

**Project template**

Save yourself time with a pre-configured project board template.

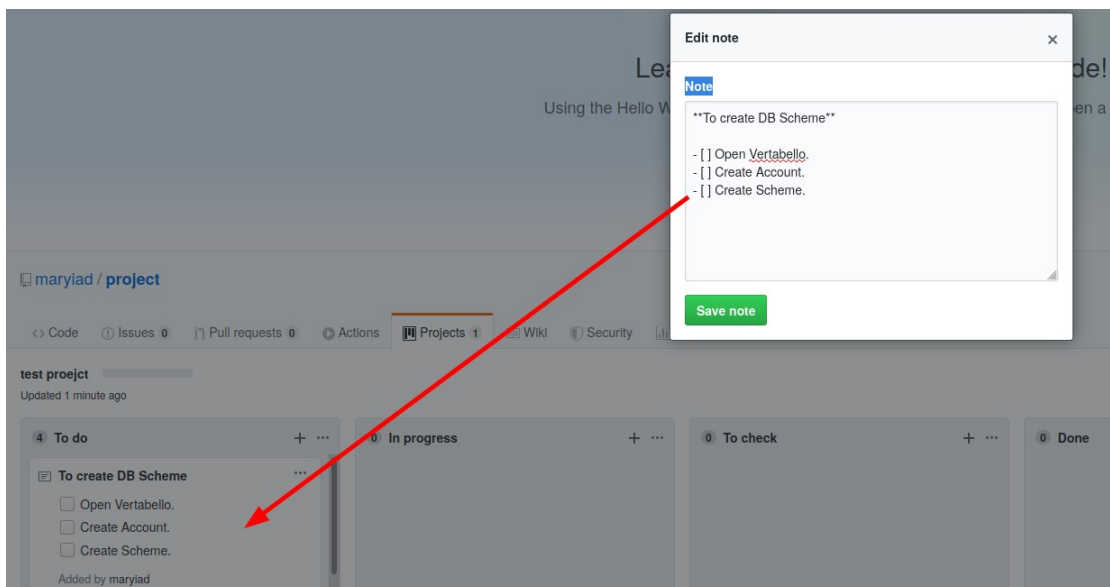
Template: Basic kanban

Create project

3. К сгенерированным спискам To do, In progress, Done добавляет список To

check.

4. Для добавления в список задачи нажмите символ + в заголовке списка. Например, описание задачи с чеклистом:



5. Как только задача готова, перемещается в список To check и передается на тестирование участнику команды, который выполняет тестирование.
6. После прохождения тестов задача перемещается в список Done.

## Порядок действий при создании проекта в Trello

1. Ознакомьтесь с документацией <https://trello.com/guide/trello-101>.
2. Тимлид команды создаёт учётную запись в Trello.
3. Создаёт команду и отправляет приглашения другим участникам команды.
4. Создаёт доску.
5. Создаёт списки для задач, например Новая, В работе, Тестирование, Готово.
6. Добавляет задачи в список Новая и, по возможности, распределяет задачи между участниками команды.
7. Как только задача готова, перемещается в список Тестирование и передаётся на тестирование участнику команды, который выполняет тестирование.
8. После прохождения тестов задача перемещается в список Готово.

## Порядок действий при создании проекта в Targetprocess

- 1) Зарегистрироваться, создать проект на <http://www.targetprocess.com/pricing/>, создав учётную запись для лидера проекта

Targetprocess 3 предлагает пользователям ряд новых функций, которые включают возможность использования 4х различных представлений одних и тех же данных:

- Доска;
- Список;
- Быстрый просмотр;
- Таймлайн.

В дополнение, каждый пользователь может создать множество личных

представлений данных («Views»), которые максимально удобно и быстро помогают в достижении поставленной цели, в зависимости от роли и требований самого пользователя. Владелец продукта или менеджер проекта быстро получит необходимый срез данных для планирования бэклогов или отслеживания статуса проделанной работы в проекте, менеджер по работе с клиентами информацию о статусе выполнения требований конкретного клиента, а руководство отдела/компании таймлайн для планирования ресурсов и проектов на ближайшее полугодие.

История targetprocess - <https://medium.com/targetprocess-software/targetprocess-25d3bec2fa5e>

2) После входа выбрать управление проектом в стиле Scrum.

3) Познакомиться с управлением проектом. Добавить других членов команды, используя меню **Add** и выбрав **User**.

4) Подготовить список высокоуровневых требований (Feature) и user stories (примеры [http://en.wikipedia.org/wiki/User\\_story](http://en.wikipedia.org/wiki/User_story)) к консольному приложению на основе данных из заданий 2, 3 и 4 текущей лабораторной работы.

Features и user stories должны быть действительными, которые будут реализованы.

На основе User Stories создать задачи (tasks) и распределить их между участниками команды.

## Непрерывная сборка и тестирование приложений

Travis CI — распределённый веб-сервис для сборки и тестирования программного обеспечения, использующий GitHub в качестве хостинга исходного кода. Программная составляющая сервиса также располагается на GitHub. Представлена в двух редакциях — коммерческая (<https://travis-ci.com>), которая может применяться для частных репозиторий с ограничением 30 дней на использование, и публичная (<https://travis-ci.org>), которая может применяться в публичных репозиториях без ограничений по времени.

Веб-сервис поддерживает сборку проектов на множестве языков, включая C, C++, D, JavaScript, Java, PHP, Python, Ruby, Swift и другие. Разные проекты с открытым исходным кодом используют Travis CI для непрерывной интеграции кода.

GitHub Actions, релиз которого состоялся 13 ноября 2019 года, позволяет легко автоматизировать все рабочие процессы в области сборки и тестирования программного обеспечения.

GitHub Actions обеспечивает ряд преимуществ:

1. Предоставляет возможность реагировать на события внутри GitHub, избавляя тем самым от необходимости привлечения внешнего инструмента и пересылки данных через другой сервис.
2. Предоставляет более дешевый тариф, чем отдельный сервис для CI/CD.

3. Позволяет повторно задействовать общие рабочие процессы.

## Подключение Travis-CI на примере простого проекта

1. Создайте репозиторий на github.
2. Клонировать репозиторий локально и не забудьте о первом коммите для создания ветки master.
3. Создайте в репозитории проекта простое приложение, например такой main.c:

```
/* main.c */
#include <stdio.h>

void main (void)
{
    printf ("Hello World!");
}
```

4. Создайте в репозитории проекта Makefile:

```
# Makefile for Hello World project

hello: main.c
    gcc -o hello main.c -I.
clean:
    rm -f hello
```

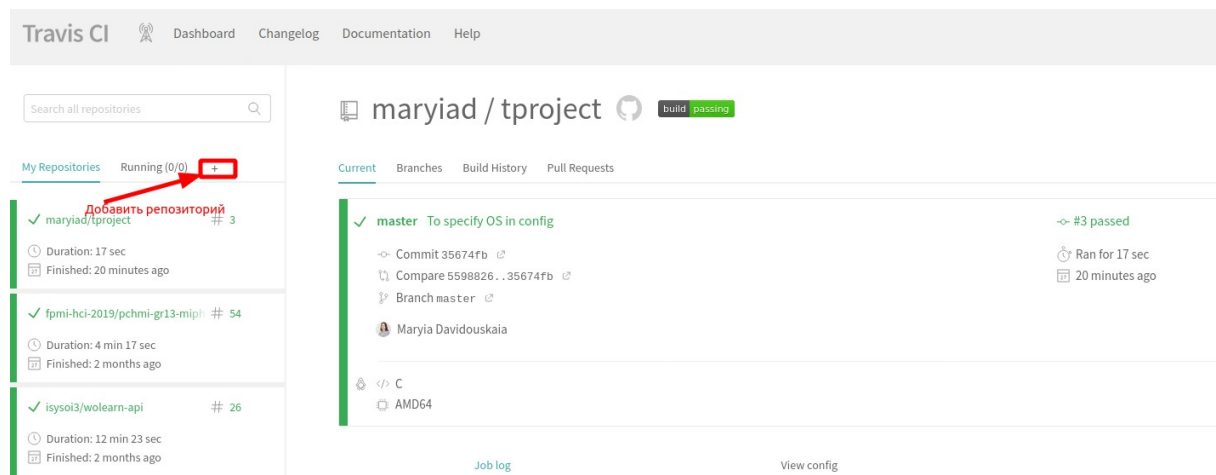
5. Создайте в репозитории проекта .travis.yml. Обратите внимание, что файл .travis.yml является скрытым и его имя начинается с точки:

```
language: c
os: linux

script: make
```

6. Подключите Travis-CI:

- a) Для приватного репозитория используем платную версию Travis CI. Для этого перейдите по адресу <https://travis-ci.com>. Для публичных репозиторий рекомендуется <https://travis-ci.org>.
- b) На главной странице нажмите кнопку Sign Up with GitHub.
- c) Подтвердите авторизацию с учетной записью на github. Пжл., учитывайте, что платная версия для теста доступна в течение 30 дней.
- d) В результате откроется окно как в примере (см. ниже) и нажмите кнопку +, чтобы добавить репозиторий, который будет отслеживаться Travis CI.



е) В окне добавления проекта найдите свой проект и нажмите на кнопку переключателя для подключения проекта к Travis.

ф) Теперь можем перейти в проект, в котором будут отражены результаты сборки.

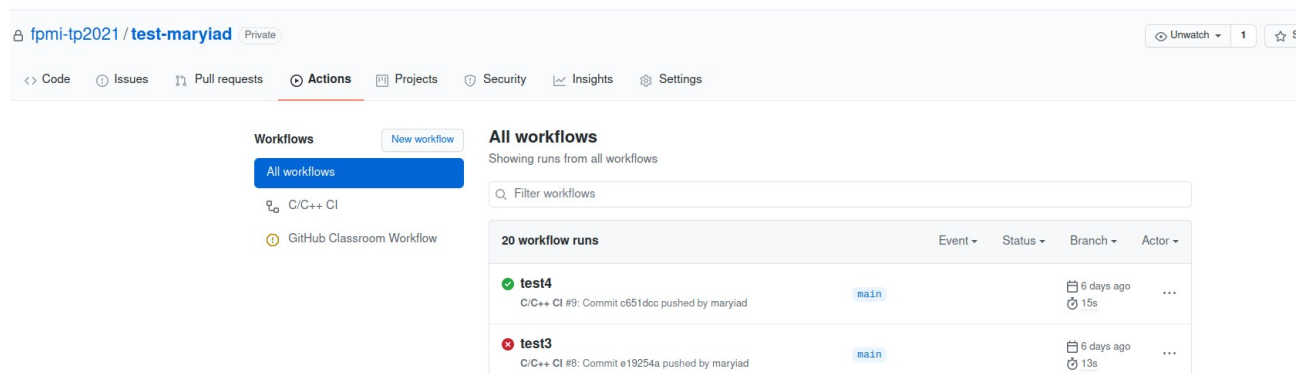
Подробнее о настройке Travis CI на примере проекта на Rust можно почитать в статьях:

- «[Как настроить сборку и тестирование для Open Source проекта на Rust под Linux с помощью Travis](#)»;
- «[Непрерывная интеграция \(CI\) для GitHub проектов на C/C++ с Cmake-сборкой](#)»;
- «[Building a C Project](#)».

## Сборка проекта с Github Action

Просмотрите руководство по Github Actions к лабораторной работе 3 — [Как пользоваться Github Actions](#).

В интерфейсе репозитория на github на вкладке Actions создаются и настраиваются все процессы и этапы сборки (см. рис. ниже).

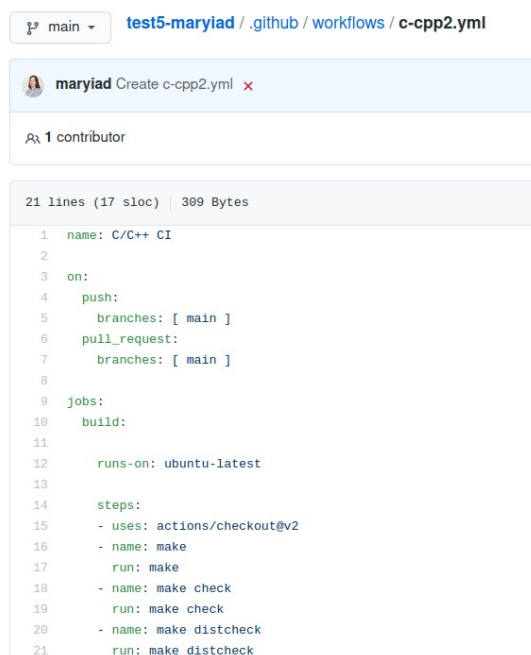


На приведенном рисунке демонстрируются две конфигурации сборки C/C++ + CI (активная) и Github Classroom Workflow (на паузе). В правой части отображаются результаты процедуры сборки.

Для того, чтобы выполнялась сборка, в репозиторий добавлен скрытый

каталог `.github`, содержащий каталог `workflows` с файлами конфигураций сборки.

Пример файла, описывающего процедуру сборки с применением `github actions`:



The screenshot shows a GitHub Actions workflow file named `c-cpp2.yml` in the `test5-maryiad` repository. The workflow is triggered on push to the `main` branch or a pull request to `main`. It defines a single job named `build` that runs on `ubuntu-latest`. The job contains three steps: `checkout` (using `actions/checkout@v2`), `make` (using `make`), and `make check` (using `make check`). The `make` step is configured with `run: make`. The `make check` step is configured with `run: make check`. The `make distcheck` step is configured with `run: make distcheck`.

```
1 name: C/C++ CI
2
3 on:
4   push:
5     branches: [ main ]
6   pull_request:
7     branches: [ main ]
8
9 jobs:
10  build:
11
12    runs-on: ubuntu-latest
13
14    steps:
15      - uses: actions/checkout@v2
16      - name: make
17        run: make
18      - name: make check
19        run: make check
20      - name: make distcheck
21        run: make distcheck
```

Для того, чтобы сборка выполнялась корректно, Makefile проекта должен содержать следующие цели:

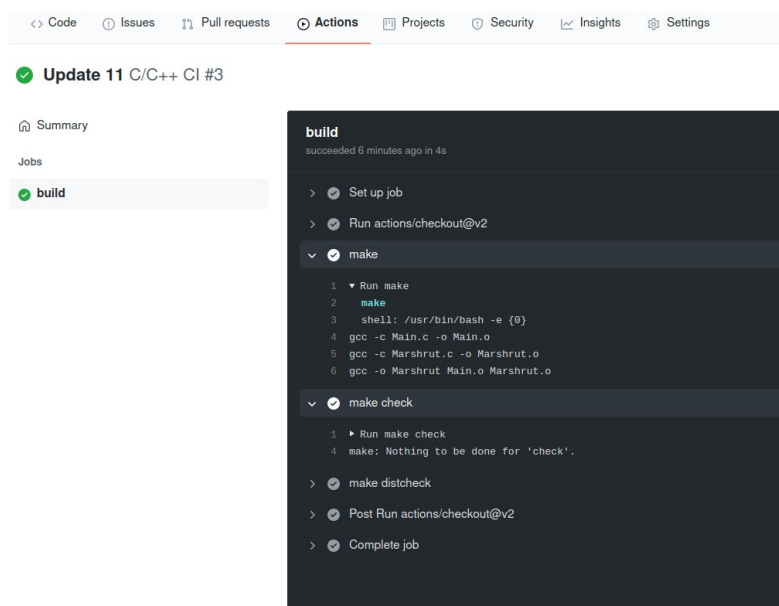
check:

```
shell: /usr/bin/bash -e {0}
```

distcheck:

```
shell: /usr/bin/bash -e {0}
```

Результат сборки представлен на рисунке ниже:



## Настройка Unit-тестов для проекта на C

### *Unit тесты на основе фреймворка GoogleTest*

Фреймворк GoogleTest (gtest) поддерживает компиляторы gcc, clang, MSVC 2015+ и системы сборки Bazel и CMake. При использовании данного фреймворка разрешается использовать утилиту cmake вместо make.

Познакомьтесь с примером настройки Unit тестов для проекта на примере фреймворка для тестирования Google Test — [Example on how to integrate gtest test into github actions](#) (для Github Actions) и <https://github.com/deftio/travis-ci-cpp-example> (для Travis CI).

Познакомьтесь с материалами для фреймворка Google Test (<https://github.com/google/googletest>) и изучите пример [GoogleTest Primer](#).

Изучите статьи как создать тесты для проекта на C, используя фреймворк для тестирования Google Test:

- [Unit testing C code with gtest](#);
- [Using Google Test to Unit Test C Code](#);
- [Example on how to integrate gtest test into github actions](#);
- [A quick introduction to the Google C++ Testing Framework](#).

### *Unit тесты на основе фреймворка CUnit*

Фреймворк CUnit позволяет создать и поддерживать тесты для проектов на языке программирования C.

Изучите документацию и примеры:

- [Документация по CUnit](#);
- <https://github.com/excalibur-sa/CUnit-examples>;
- <https://github.com/jacklicn/CUnit>;
- <https://github.com/zetalog/cunit>;
- пример настройки тестов с использованием CUnit и Github Actions в репозитории <https://github.com/alrevuelta/cONNXr>:
  - рабочий процесс workflow для сборки на ОС macOS — <https://github.com/alrevuelta/cONNXr/blob/master/.github/workflows/macOS.yml>;
  - рабочий процесс workflow для сборки на ОС Ubuntu — <https://github.com/alrevuelta/cONNXr/blob/master/.github/workflows/ubuntu.yml>.

### *Проверка покрытия кода тестами*

Для проверки покрытия кода тестами используем утилиту gcov, которая



входит в состав пакета компилятора gcc и сервис Codecov <https://about.codecov.io/language/c/>.

При использовании CUnit изучите примеры настройки Unit тестов и использования утилиты gcov и Codecov-сервиса:

- <https://gcc.gnu.org/onlinedocs/gcc/Gcov.html>
- <https://about.codecov.io/tool/cunit/>
- <https://about.codecov.io/blog/how-to-set-up-codecov-with-c-and-github-actions/>

При использовании GoogleTest (gtest) изучите примеры настройки Unit тестов и использования утилиты gcov:

- <https://medium.com/@naveen.maltesh/generating-code-coverage-report-using-gnu-gcov-lcov-ee54a4de3f11>

## Литература по Travis-CI

Основные материалы

1. «[Как настроить сборку и тестирование для Open Source проекта на Rust под Linux с помощью Travis](#)»;
2. «[Непрерывная интеграция \(CI\) для GitHub проектов на C/C++ с Cmake-сборкой](#)»;
3. «[Building a C Project](#)»

Дополнительные материалы

4. <https://habr.com/ru/post/352282/>
5. <https://habr.com/ru/post/338126/>
6. <https://maxkrasnov.ru/note/travis-ci-settings>
7. <http://automation-remarks.com/travis-ci-na-sluzhbie-u-avtomatizatsii/>
8. <http://nano.sapegin.ru/all/travis-ci>

## ЗАДАНИЕ 1. МЕНЕДЖМЕНТ ПРОЕКТА В СТИЛЕ KANBAN

### Упражнение 1.1. Изучить возможности управления проектами

1. Выбрать систему управления проектами из перечня: *Github Project*, или *Trello* или *Targetprocess*.
2. Познакомиться с учебными материалами для выбранной системы управления проектами.

### Упражнение 1.2. Создать проект и распределить задачи

1. Создать проект типа Kanban в выбранной системе управления проектами согласно:

- Порядок действий при создании проекта в Github Projects
  - Порядок действий при создании проекта в Trello
  - Порядок действий при создании проекта в Targetprocess
2. Добавить в проект участников.
  3. Добавить списки и задачи.
  4. Распределить задачи.
  5. Использовать проект для управления командой.

## ЗАДАНИЕ 2. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

1. Познакомиться с документом [UML and C.pdf](#).
2. Реализовать диаграмму вариантов использования для иллюстрации функциональных требований согласно варианту команды.
3. Используя диаграмму классов, создать диаграмму файлов консольного приложения, иллюстрирующую файлы, типы, атрибуты, функции и отношения.

Для разработки использовать десктопные приложения:

- VisualParadigm Community Edition (<http://visual-paradigm.com/>)
- StarUML 2 (<http://staruml.io/>)

или онлайн-сервисы:

- Diagrams.net (<https://app.diagrams.net>)
- Creately (<http://creately.com/>)
- GenMyModel (<http://www.genmymodel.com/>)

## ЗАДАНИЕ 3. ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ БАЗЫ ДАННЫХ И НАПОЛНЕНИЕ ДАННЫМИ

Для решения необходимо использовать <http://www.vertabelo.com> для проектирования архитектуры базы данных.

### Упражнение 3.1. Изучить функционал Vertabelo

Ознакомьтесь с видео <https://youtu.be/hU-A08K08-Y>.

### Упражнение 3.2. Спроектировать базу данных

Спроектируйте базу данных для хранения данных о пользователях и других сущностях согласно варианту задания.

База данных должна содержать таблицу пользователей и таблицы под каждый тип сущности приложения.

Например, для банковского приложения создаются таблицы: BANK\_ACCOUNTS, BANK\_CLIENTS, BANK\_USERS, BANK\_CONFIG. В банковском приложении сведения о счетах всех пользователей хранятся в таблице BANK\_ACCOUNTS. Данные клиентов хранятся в таблице BANK\_CLIENTS. Данные о пользователях, имеющих доступ в приложение, — в таблице BANK\_USERS. Основная конфигурационная информация банка должна храниться в таблице BANK\_CONFIG. К конфигурационной информации относится:

- размер процентов для того или иного типа счет;
- период начислений процентов;

- сведения о штрафе в случае овердрафт;
- максимальное количество транзакций для CheckingAccount;
- сроки для OverdraftAccount.

### Упражнение 3.3. Сгенерировать sql-скрипт для sqlite

Сгенерируйте файл SQL-скрипта для БД SQLite. Импортируйте файл в Valentina Studio или SQLBrowser и заполните таблицы БД данными.

## ЗАДАНИЕ 4. НЕПРЕРЫВНАЯ СБОРКА ПРОЕКТА И ТЕСТИРОВАНИЕ

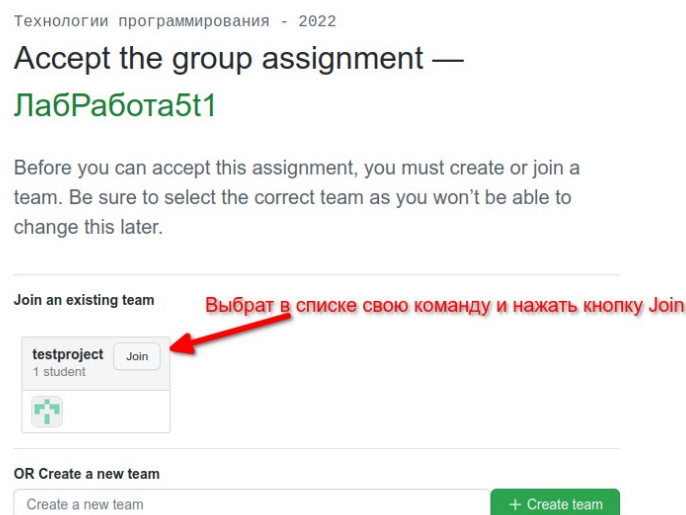
1. Изучите основные материалы по использованию Github Actions (или travis-ci) для сборки и тестирования консольного проекта.
2. Продемонстрируйте результаты сборки проекта, выполнения тестов (не менее 3 тестов на каждый файл проекта) и отчет по покрытию кода тестами.

## ЗАДАНИЕ 5. РАЗРАБОТКА ПРИЛОЖЕНИЯ

### Упражнение 5.1. Подключить репозиторий группы

Подключить репозиторий на github согласно ссылке для Вашей группы. Репозиторий создаёт тимлид.

Тимлид сообщает участникам название команды. Остальные участники команды переходят по ссылке для репозитория и выбирают свою команду (см. рисунок ниже).



В процессе выполнения разработки проиллюстрировать работу с несколькими ветками в репозитории проекта, добавление и приём коммитов, слияние и перемещение веток и т. д.

## Упражнение 5.2. Документирование проекта

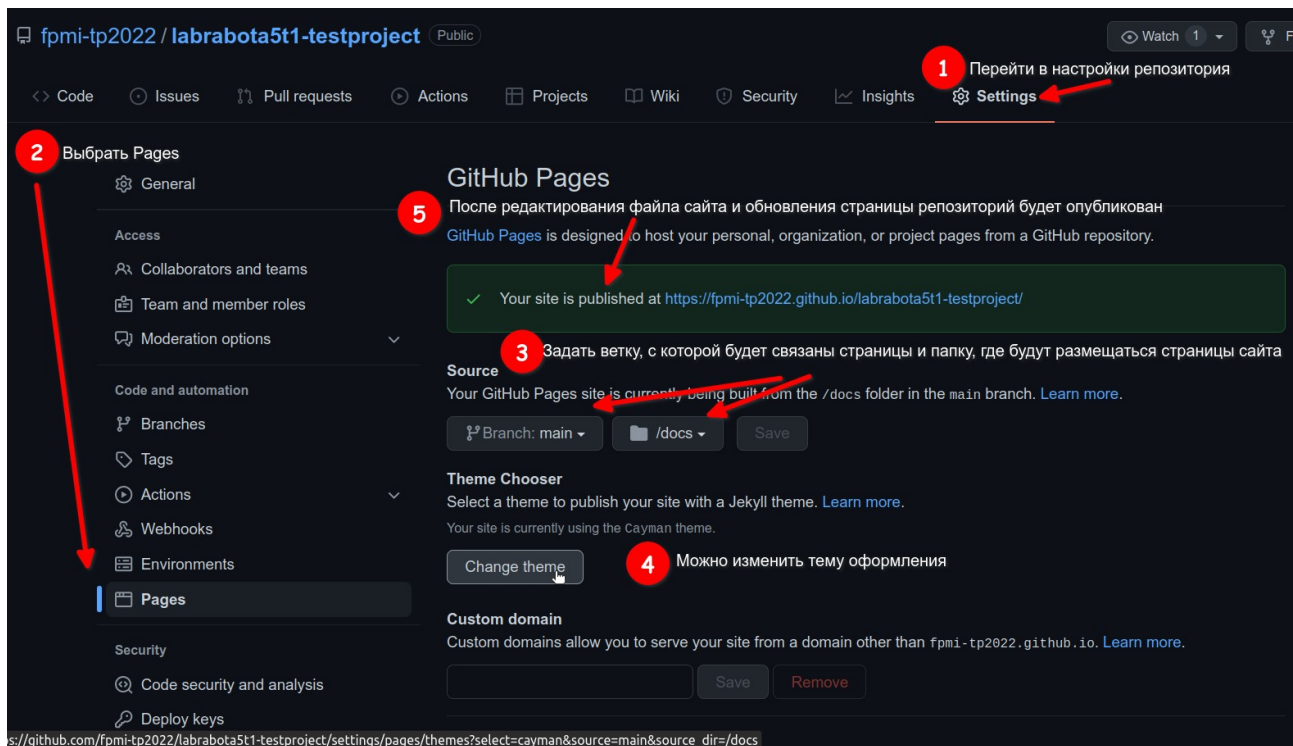
### Описание проекта в wiki

Изучить документацию <https://guides.github.com/features/wikis/> и документировать проект в Readme и wiki репозитория согласно следующим требованиям:

1. Файл Readme и wiki оформить с помощью синтаксиса Markdown.
2. Структура файла Readme должна быть следующей:
  - **Project Name:** в данном блоке указать название проекта.
  - **Description:** Краткое описание проекта и его функциональности в 3-5 предложениях.
  - **Installation:** Последовательность шагов, как установить приложение локально.
  - **Usage:** Рекомендации как использовать приложение после установки. Может содержать скриншоты.
  - **Contributing:** Сведения об авторах проекта и какие задачи реализовывали.
3. Структура страниц wiki должна быть следующей:
  - **Главная страница:** содержит краткое описание задачи и ссылки на другие материалы и разделы.
  - **Функциональные требования:** описание функциональных требований, диаграммы Use case, текстовые сценарии.
  - **Диаграмма файлов приложения:** диаграмма файлов и описание.
  - **Дополнительная спецификация:** ограничения, требования к безопасности, надежности и другое.
  - **Схема базы данных:** страница содержит схему базы данных в виде изображения и ссылку на sql-файл.
  - **Презентация проекта:** ссылка на презентацию проекта, в которой должно быть отражено распределение задач в команде, требования к приложению, схема базы данных, как была организована работа с репозиторием и проектом и др.

### Описание проекта команды в Github Pages

1. Изучить курс <https://lab.github.com/githubtraining/github-pages>
2. В командном репозитории для проекта тимлид создаёт сайт для github Pages (см. рис. ниже).



Пример сайта на основе Github Pages доступен по ссылке — <https://fpmi-tp2022.github.io/labrabota5t1-testproject/>

3. Добавить структуру страниц, аналогичную wiki и еще одну отдельную страницу, на которой разместить ссылки на учебные репозитории по итогам изучения курса <https://lab.github.com/githubtraining/github-pages> для каждого участника команды.

## Упражнение 5.3. Разработать приложение

Разработать консольное приложение на языке Си, которое позволяет пройти аутентификацию и авторизацию и выполнять операции согласно варианту задания.

Проект должен состоять из нескольких .с файлов. Структура проекта должна соответствовать модели КИС, содержать папки bin, build, includes, src.

Документацию проекта, включая постановку задачи и другие материалы представить в вики проекта.

Проиллюстрировать различные виды запросов к базе данных, обеспечить хранение данных в различных форматах, включая изображения.

## ВАРИАНТЫ:

### Вариант 1. «Автопарк».

Автопарк осуществляет обслуживание заказов на перевозку грузов, используя для этой цели свой парк автомашин и своих водителей. Водитель, выполнивший заказ, получает 20 % от стоимости перевозки.

#### Управление автопарком владеет информацией:

об автомашинах: номер машины, марка, пробег на момент приобретения, грузоподъемность;

о водителях: табельный номер, фамилия водителя, категория, стаж, адрес, год рождения;

о выполненных заказах: дата, фамилия водителя, номер машины, километраж, масса груза, стоимость перевозки.

Водитель автопарка может узнать информацию: о своей машине, только свои

данные - смотрите ниже по пункту, помеченному \* (звездочкой).

**Необходимо выполнить:**

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:
  - по указанному водителю – перечень выполненных заказов за указанный период (\*);
  - по указанной машине – общий пробег и общую массу перевезенных грузов (\*);
  - по каждому водителю – общее количество поездок, общую массу перевезенных грузов, сумму заработанных денег (\*);
  - по водителю, выполнившему наименьшее количество поездок, – все сведения и количество полученных денег;
  - по автомашине с наибольшим общим пробегом – все сведения(\*);
3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать функцию, который при добавлении информации в таблицу заказов, проверяет, не превышает ли масса груза грузоподъемности машины, и если это так, то запрещает помещать информацию в таблицу(\*).
5. Создать функцию, которая за указанный период определяет количество денег, начисленных каждому водителю за перевозки. В качестве параметра передать начальную дату периода и конечную дату периода. Результаты занести в специальную таблицу (\*).
6. Создать функцию, которая за указанный период определяет количество денег, начисленных указанному водителю за перевозки. В качестве параметра передать начальную дату периода, конечную дату периода и фамилию водителя(\*).

## **Вариант 2. «Рыболовная флотилия».**

Флотилия рыболовных траулеров осуществляет поиск косяков рыбы и его отлов. С этой целью каждый траулер отправляется в рейс на некоторое количество суток и посещает ряд мелководных участков моря, называемых банками. Каждая банка имеет свое название. Выловленная рыба классифицируется своим названием, количеством и качеством.

**Управление флотилией владеет информацией:**

*о траулерах:* название траулера, водоизмещение, дата постройки;

*о членах команды* фамилия, должность, дата приема на работу, год рождения;

*о результатах рейсов:* название траулера, дата выхода в море, дата возвращения, название посещенной банки, название, качество и количество выловленной рыбы.

**Член команды может узнать информацию:** только свои данные - смотрите ниже по пункту, помеченному \* (звездочкой).

**Необходимо выполнить:**

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:
  - по указанному траулеру – перечень выполненных рейсов за указанный период с выдачей сведений об общем количестве выловленной рыбы;

- по указанной банке – перечень и количество выловленной рыбы по видам рыбы;
- по банке, на которой было выловлено максимальное количество рыбы низкого качества, – сведения о датах рейсов и траулерах, ее посещавших;
- по траулеру, выловившему наибольшее количество рыбы, – сведения о капитане и посещенных траулером банках с указанием дат выхода и возвращения;
- по членам команд – перечень людей, которые на указанную дату должны быть отправлены на пенсию.

3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.

4. Создать функцию, который при добавлении информации в таблицу результатов рейса выполняет обновление информации в таблице статистики, где для каждого траулера хранится информация об общем количестве выловленной рыбы.

5. Создать функцию, которая за указанный период осуществляет начисление премий членам экипажа за внеплановый отлов рыбы. В качестве параметра передать начальную дату периода, конечную дату периода, плановое задание, среднюю стоимость килограмма рыбы. Результаты занести в специальную таблицу(\*).

6. Создать функцию, которая за указанный период осуществляет начисление премии указанному члену экипажа(\*).

### **Вариант 3. «Воздушный извозчик».**

Отряд грузовых вертолетов осуществляет доставку грузов и людей в высокогорном районе. Каждый вертолет обслуживается экипажем из трех пилотов, постоянно закрепленных за ним. Летчики получают по 5 % от стоимости обычного рейса и 10 % от стоимости спецрейса.

#### **Командир авиаотряда владеет информацией:**

*о вертолетах:* номер вертолета, марка, дата изготовления, грузоподъемность, дата последнего капитального ремонта, летный ресурс времени до следующего капитального ремонта;

*о членах экипажа:* табельный номер, фамилия, должность, стаж, адрес, год рождения, номер вертолета;

*о выполненных рейсах:* дата, номер вертолета, код рейса, масса груза, количество перевезенных людей, длительность полета, стоимость рейса.

**Член экипажа может узнать информацию:** только свои данные или данные о своём вертолете - смотрите ниже по пункту, помеченному \* (звездочкой).

#### **Необходимо выполнить:**

1. Создать таблицы БД с учетом ограничений целостности данных.

2. Используя оператор Select, выдать следующую информацию:

- по каждому вертолету – общее количество часов, которые они налетали после капитального ремонта, и ресурс летного времени (\*);

- по каждому вертолету – перечень выполненных рейсов с указанием общей массы перевезенных грузов и количества человек за указанный период(\*);
- по всем вертолетам, выполнявшим спецрейсы, – общее количество рейсов, общая масса перевезенных грузов, общая сумма заработанных денег;
- по всем вертолетам, выполнявшим обычные рейсы, – общее количество рейсов, общая масса перевезенных грузов, общая сумма заработанных денег;
- по вертолету, выполнившему максимальное количество рейсов, – все сведения об его экипаже и количестве заработанных денег;
- по экипажу, заработавшему максимальное количество денег, – все сведения о выполненных им рейсах.
- по экипажу(или члену экипажа), – все сведения о выполненных им рейсах (\*).

3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.

4. Создать функцию, который при внесении информации в таблицу рейсов, проверяет, не будет ли превышен ресурс летного времени для вертолета, и если это так, то запрещает вносить информацию в таблицу.

5. Создать функцию, которая за указанный период определяет количество денег, начисленных экипажам авиаотряда за перевозки. В качестве параметра передать начальную дату периода и конечную дату периода. Результаты занести в специальную таблицу.

6. Создать функцию, которая за указанный период определяет количество денег, начисленных указанному летчику(\*).

7. Создать функцию, которая за указанный период определяет количество денег, начисленных указанному летчику за указанный рейс(сы) или спецрейс(сы) (\*).

## **Вариант 4. «Ипподром».**

Ипподром регулярно проводит состязания в беге лошадей. Лошади на соревнования заявляются владельцами, в состязании лошадь участвует вместе с жокеем, который в различных состязаниях может работать с различными лошадьми. Жокеи входят в штат ипподрома.

### **Управление ипподромом владеет информацией:**

*о лошадях:* кличка лошади, возраст, стаж участия в соревнованиях, владелец, заплаченная цена;

*о жокеях:* фамилия жокея, стаж, год рождения, адрес;

*о проведенных забегах:* дата, номер забега, кличка лошади, фамилия жокея, занятое место.

**Жокей может узнать информацию:** только свои данные, проведенные забеги и данные о своей лошади - смотрите ниже по пункту, помеченному \* (звездочкой).

Владелец лошади может узнать информацию: только данные своих лошадей и проведенные ими забеги - смотрите ниже по пункту, помеченному \*\* (двумя звездочками).



**Необходимо выполнить:**

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:
  - по лошади, побеждавшей максимальное количество раз, – сведения о ней, датах соревнований и о жокеях (\*\*);
  - по жокею, участвующему наибольшее количество раз в забегах, – сведения об этом жокее и об общем количестве забегов, в которых он участвовал ;
  - по указанному жокею – сведения о датах забегов, участвовавших в них лошадях и занятых местах(\*);
  - по указанному владельцу – список его лошадей с указанием дат забегов и занятых мест(\*\*);
  - по всем забегам – все сведения о проводимых забегах и участвовавших в них лошадях за указанный период.
3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать функцию, который запрещает помещать информацию в таблицу забегов при отсутствии соответствия кличек лошадей и фамилий жокеев с аналогичными данными в таблицах лошадей и жокеев.
5. Создать функцию, которая осуществляет распределение призового фонда для тройки призеров последнего забега в соотношении 50 %, 30 %, 20 %. В качестве параметра передать величину призового фонда. Результаты занести в специальную таблицу.
6. Создать функцию, которая за указанный период выводит по указанному жокею сведения о забегах, где он участвовал(\*).

**Вариант 5. «Музыкальный салон».**

Постоянно работающий музыкальный салон продает компакт-диски с записями определенных исполнителей, поступающие от различных компаний-производителей.

**Управление салона владеет информацией:**

*о компакт-дисках:* код компакта, дата изготовления, компания-производитель, цена одного компакта;

*об исполнителях музыкальных произведений:* название музыкального произведения, автор, исполнитель, код компакта;

*о поступлении и продаже компакт-дисков:* дата операции, код операции (поступление или продажа), код компакта, количество экземпляров.

**Покупатель может узнать информацию:** только данные - смотрите ниже по пункту, помеченному \* (звездочкой).

**Необходимо выполнить:**

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:
  - по всем компактам – сведения о количестве проданных и оставшихся компактв одного вида по убыванию разницы;
  - по указанному компактв – сведения о количестве и стоимости компактв, проданных за указанный период;
  - по компактв, купленному максимальное количество раз, – выдать все сведения о нем и музыкальных произведениях(\*);
  - по наиболее популярному исполнителю – сведения о количестве проданных компактв с его произведениями(\*);
  - по каждому автору – сведения о количестве проданных компактв с его записями и сумме полученных денег.
3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать функцию, который запрещает помещать информацию о продаже компактв в таблицу, если суммарное количество проданных компактв превысит суммарное количество поступивших.
5. Создать функцию, которая за указанный период определяет количество поступивших и проданных компактв по каждому виду. В качестве параметра передать начальную дату периода и конечную дату периода. Результаты занести в специальную таблицу.
6. Создать функцию, которая по заданному коду компактв выводит информацию о результатах его продажи за указанный период(\*).

## **Вариант 6. «Туристическое бюро».**

Бюро путешествий осуществляет обслуживание экскурсионных маршрутов, используя для этой цели свои автобусы и экипажи, состоящие из трех человек, которые строго закреплены за своим автобусом. Экипаж, выполнивший заказ на обслуживание, получает 20 %, от стоимости обслуживания. Стоимость билета договорная и зависит от длины туристической трассы и числа участников экскурсии.

Бюро владеет информацией:

*об экскурсионных маршрутах:* название маршрута, начальный пункт, конечный пункт, протяженность пути;

*об автобусах:* номер автобуса, название, общая величина пробега;

*о членах экипажа:* фамилия, табельный номер, стаж, категория, адрес, год рождения, номер автобуса;

*о выполненных рейсах:* номер автобуса, дата отбытия, дата прибытия, название маршрута, количество перевезенных пассажиров, стоимость билета.

Член экипажа может узнать информацию: только свои данные - смотрите ниже по пункту, помеченному \* (звездочкой).

**Необходимо выполнить:**

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:
  - по указанному автобусу – перечень выполненных рейсов за указанный период;
  - по указанному автобусу – общее количество поездок, количество перевезенных пассажиров и полученных денег(\*);
  - по каждому экипажу – количество начисленных денег за указанный период (\*);
  - по наиболее дорогому маршруту – сведения об автобусах, экипажах и стоимости билетов;
  - по автобусу с наибольшим суммарным пробегом – сведения о количестве перевезенных пассажиров и величине пробега.
3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать функцию, который при добавлении информации в таблицу экскурсий, проверяет соответствие вносимой информации той информации, которая хранится в основных таблицах.
5. Создать функцию, которая за указанный период определяет количество денег, начисленных каждому экипажу за обслуживание экскурсий. В качестве параметра передать процент отчисления, начальную дату периода и конечную дату периода. Результаты занести в специальную таблицу.
6. Создать функцию, которая на указанную дату выводит информацию о количестве денег, начисленных указанному экипажу (\*).

**Вариант 7. «Пушной аукцион».**

Управление звероводческими совхозами регулярно проводит пушные аукционы, куда звероводческие совхозы представляют свою продукцию. Пушнина выставляется по определенной цене, но в результате аукциона она может быть продана по более высокой или низкой цене.

Управление звероводческими совхозами владеет информацией:

*о зверофермах:* номер зверофермы, адрес, фамилия директора, телефон;

*о выставленной на аукцион пушнине:* номер зверофермы, название меха, сорт, количество единиц, заявленная цена;

*о результатах аукциона:* номер зверофермы, название меха, сорт, количество проданных единиц, продажная цена, категория покупателя (меховая фабрика, ателье, частное лицо).

Звероферма владеет информацией: данные о своих шкурках - смотрите ниже по пункту, помеченному \* (звездочкой).

**Необходимо выполнить:**

1. Создать таблицы БД с учетом ограничений целостности данных.

2. Используя оператор Select, выдать следующую информацию:

- по звероферме, чей мех получил самую высокую цену, – все сведения о ней;
- по каждой категории покупателей – общее количество купленных единиц, общая сумма уплаченных денег;
- по каждой звероферме – прибыль от продажи пушнины (\*);
- по всем зверофермам, которые продали шкурки по цене выше средней аукционной цены, – все сведения о них;
- по звероферме, получившей максимальную прибыль, – все сведения о ней, количестве проданной пушнины и величине прибыли.

3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.

4. Создать функцию, который запрещает внесение сведений в таблицу результатов аукциона, если вносимые сведения не соответствуют реальности.

5. Создать функцию, которая определяет список звероферм с указанием сведений о них, которые получили прибыль от аукциона меньше, чем планировалось. В качестве параметра передать плановый процент прибыли .

6. Создать функцию, которая по указанной звероферме выдает величину прибыли, полученную на аукционе.

## **Вариант 8. «Цветочная оранжерея».**

Цветочная оранжерея выращивает различные виды цветов и продает на заказ составленные из них композиции. Каждая композиция имеет свое название и может состоять как из цветов одного вида, так и из цветов разного вида. Заказ обычно выполняется в течение нескольких дней. При выполнении заказа в течение суток дополнительно взимается плата в размере 25 %. При выполнении заказа в течение двух суток дополнительно взимается плата в размере 15 %.

Дирекция оранжереи владеет информацией:

*о цветах:* название цветка, сорт, стоимость одного цветка;

*о композициях:* название композиции, название входящего в композицию цветка, сорт, количество единиц;

*о выполнении заказов:* дата принятия заказа, дата выполнения заказа, название композиции, количество единиц, покупатель.

Покупатель владеет информацией: данные о своих заказах - смотрите ниже по пункту, помеченному \* (звездочкой).

### **Необходимо выполнить:**

1. Создать таблицы БД с учетом ограничений целостности данных.

2. Используя оператор Select, выдать следующую информацию:

- по всем заказам – сумма полученных денег за указанный период (\*);
- по композиции, пользующейся максимальным спросом, – все сведения о ней;

- по всем заказам – сведения о количестве выполненных заказов по срочности;
- по всем заказам – сведения о количестве использованных цветов по видам и сортам за указанный период;
- по всем заказам – сведения о количестве проданных композиций и сумме полученных денег по видам композиций.

3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.

4. Создать функцию, который запрещает увеличивать цену на цветы, если стоимость композиции увеличивается более чем на 10 %.

5. Создать функцию, которая осуществляет внесение информации в таблицу заказов, затем рассчитывает стоимость заказа и вносит сведения о нем в специальную таблицу. В качестве параметра передать все сведения о заказе.

6. Создать функцию, которая на указанную дату выводит всю информацию о полученных заказах.

## **Вариант 9. «Парфюмерный базар».**

Постоянно работающий парфюмерный базар характеризуется участием в ней оптовых фирм-поставщиков и оптовых фирм-покупателей, при этом все сделки заключаются через специальных маклеров, имеющих доступ ко всем товарам.

Управление базаром владеет информацией:

*о маклерах:* фамилия маклера, адрес, год рождения;

*о товаре:* название товара, вид, цена единицы товара, оптовая фирма-поставщик, срок годности, количество поставленных единиц;

*о заключенных сделках:* дата сделки, название товара, вид, количество проданных единиц, фамилия маклера, оптовая фирма- покупатель.

Маклер владеет информацией: данные о своих сделках - смотрите ниже по пункту, помеченному \* (звездочкой).

### **Необходимо выполнить:**

1. Создать таблицы БД с учетом ограничений целостности данных.

2. Используя оператор Select, выдать следующую информацию:

- по каждому названию товара – сведения о проданном количестве и общей стоимости за указанный период;
- по каждому названию товара – перечень фирм-покупателей с указанием сведений о количестве единиц и стоимости купленного ими товара по каждой фирме-покупателю ;
- по виду товара, пользующемуся наибольшим спросом, – сведения о количестве и стоимости проданного товара по каждой фирме-покупателю (\*);
- по маклеру, совершившему максимальное количество сделок, – сведения о нем и фирмах-поставщиках;

- по каждой фирме-поставщику – список маклеров с указанием сведений о количестве и стоимости проданного ими товара по каждому маклеру.

3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.

4. Создать функцию, которая по совершении очередной сделки обновляет по каждому маклеру сведения о количестве единиц проданного товара и сумме сделки в специальной таблице статистики(\*).

5. Создать функцию, которая на указанную дату выполняет обновление данных таблицы, содержащей сведения о товаре, вычитая из количества поставленных единиц количество проданных единиц до указанной даты, и удаляет соответствующие строки из таблицы заключенных сделок. В качестве параметра передать дату периода.

6. Создать функцию, которая на указанную дату выводит всю информацию о совершенных сделках(\*).

## **Вариант 10. «Автомастерские».**

Городская служба хозяйствования имеет в своем распоряжении несколько автомастерских, каждая из которых проводит обслуживание автомобилей определенных марок. При этом выполняются следующие виды работ:

- замена отдельных элементов кузова, подбор краски и покраска;
- замена ремней, регулировка клапанов, замена маслосъемных колпачков;
- замена ведущих и ведомых шестерен;
- замена масла, замена фильтров.

Каждая автомастерская имеет свой штат работников.

Городская служба владеет информацией:

*о мастерских:* номер автомастерской, адрес, перечень марок ремонтируемых машин, список мастеров;

*об отремонтированных машинах:* госномер, марка, год выпуска, фамилия владельца, номер техпаспорта, адрес владельца;

*о выполненных работах:* номер мастерской, дата поступления, дата завершения ремонта, госномер, вид ремонта, стоимость ремонта, мастер.

Мастер владеет информацией: данные о своих работах - смотрите ниже по пункту, помеченному \* (звездочкой).

### **Необходимо выполнить:**

1. Создать таблицы БД с учетом ограничений целостности данных.

2. Используя оператор Select, выдать следующую информацию:

- по указанной автомастерской – перечень выполненных работ каждым мастером за указанный период;
- по указанному мастеру – сведения о выполненных ремонтах и отремонтированных автомашинах (\*);

- по каждой марке отремонтированных машин – сведения о номерах автомастерских, датах и видах ремонта, фамилиях мастеров;
- по каждой автомастерской – сведения об общем количестве проведенных ремонтов и общей выручке;
- по автомастерской с наибольшим количеством ремонтов – все сведения о проведенных ремонтах и отремонтированных машинах.

3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.

4. Создать функцию, который при добавлении информации в таблицу выполненных работ обновляет хранящуюся в специальной таблице по всем мастерам информацию о количестве выполненных работ по видам работ.

5. Создать функцию, которая за указанный период определяет для указанной автомастерской количество и перечень выполненных работ по видам. В качестве параметра передать номер автомастерской, начальную дату периода и конечную дату периода. Результаты занести в специальную таблицу.

6. Создать функцию, которая на указанную дату выводит информацию о количестве выполненных работ для указанной мастерской(\*).