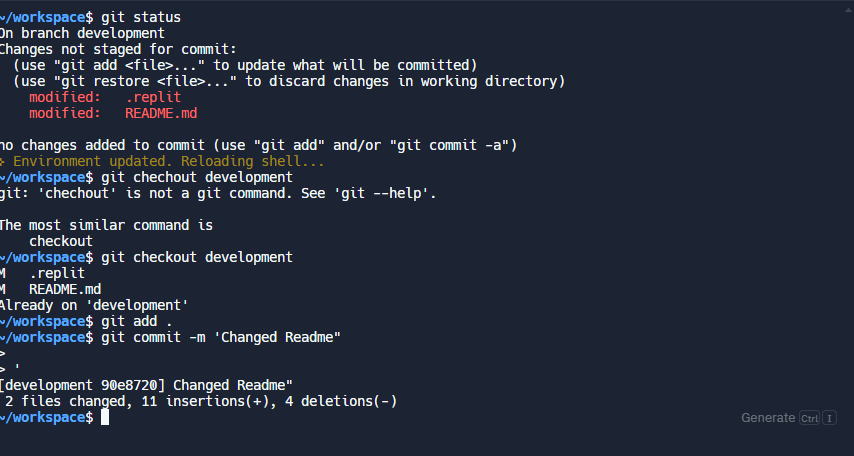
# Отчёт по лабораторной работе 2

Группа 11б

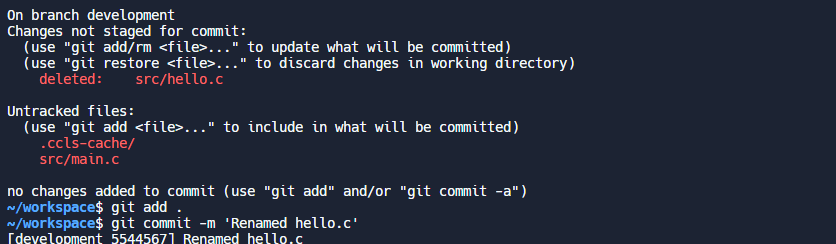
Степанов Антон

# Задание 1

5.



6.



7.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

8-9.

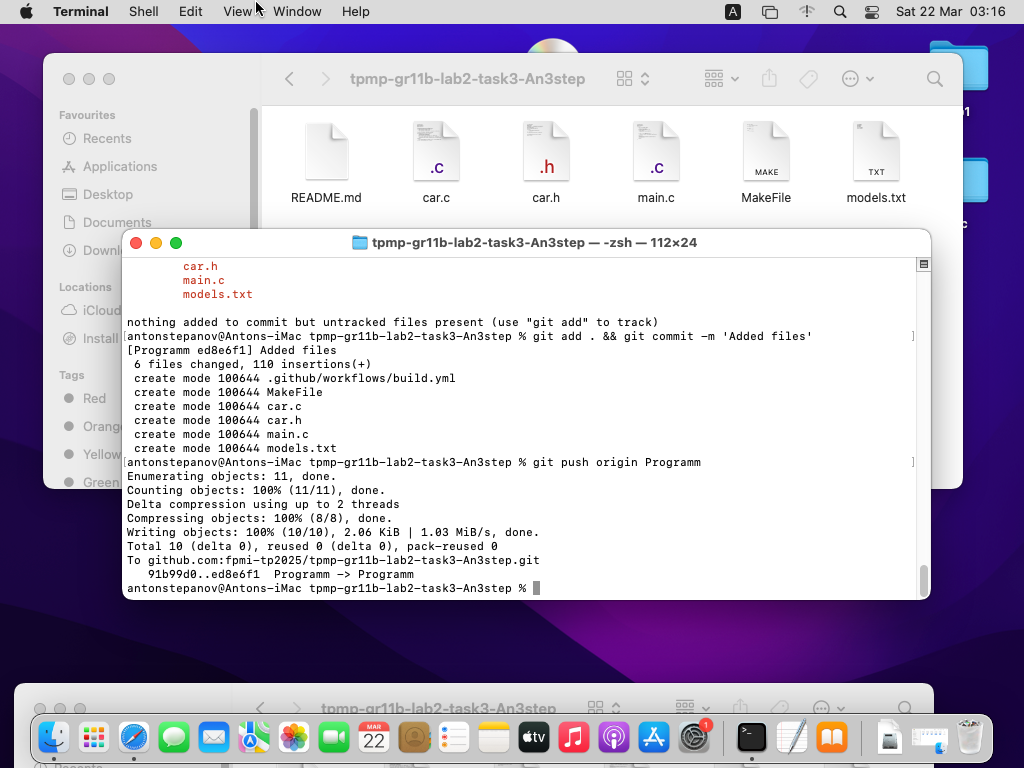
Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

# Задание 2



# Задание 3



# Задание 4



**1. Приведите примеры облачных IDE.**

* **Repl.it (или Replit)**: Поддерживает различные языки, возможность совместной работы.
* **GitHub Codespaces**: Полностью интегрированная среда разработки, которая работает в браузере.
* **CodeSandbox**: Подходит для разработки веб-приложений с поддержкой фреймворков, таких как React.
* **Glitch**: Инструмент для быстрого прототипирования и создания веб-приложений.
* **AWS Cloud9**: Облачная IDE от Amazon с поддержкой выполнения кода на сервере.

**2. Опишите процесс подключения репозитория в repl.it.**

1. Войдите в Replit.
2. Создайте новый repl или откройте существующий.
3. Нажмите на вкладку "Version Control" или "Import from GitHub".
4. Введите URL-адрес репозитория GitHub или выберите из списка доступных репозиториев.
5. Нажмите "Import" для загрузки содержимого репозитория в ваш Replit проект.

**3. Что такое «сборка»?**

Сборка — это процесс компиляции исходного кода в исполняемую программу. Он включает в себя преобразование кода в объектные файлы и связывание их в единый исполняемый файл с помощью компилятора и компоновщика.

**4. Какие утилиты могут использоваться для сборки?**

* **Make**: Утилита для управления процессом сборки, использующая Makefile.
* **CMake**: Кроссплатформенная система для автоматизации сборки проектов.
* **Ninja**: Быстрая система сборки, разработанная для проектирования.
* **Gradle**: Система автоматизации сборки, часто используемая для Java-проектов.
* **Ant**: Утилита для автоматизации сборки, в основном для Java.

**5. Для чего служит Makefile?**

Makefile служит для автоматизации процесса сборки проекта. Он содержит правила, которые определяют, как компилировать и связывать исходные файлы, а также как очищать проект или выполнять другие задачи.

**6. Что такое цель в Makefile? Приведите пример.**

Цель — это имя задачи, которую следует выполнить, например, создание исполняемого файла. Пример:

makefile

all: program

**7. Что такое связка в Makefile? Приведите пример.**

Связка — это процесс объединения объектных файлов и библиотек для создания исполняемого файла. Пример связки:

program: file1.o file2.o

gcc -o program file1.o file2.o

**8. Что такое зависимость в Makefile? Приведите пример.**

Зависимость — это файл, необходимый для создания целевой цели. Например, если исполняемый файл зависит от объектных файлов:

program: file1.o file2.o

Здесь file1.o и file2.o являются зависимостями для цели program.

**9. Что такое правило в Makefile? Приведите пример.**

Правило определяет, как создать цель из ее зависимостей. Пример:

file1.o: file1.c

gcc -c file1.c

**10. Что такое макроопределение в Makefile? Приведите пример.**

Макроопределение — это переменная, содержащая значения, которые могут быть использованы в правилах. Пример:

CC = gcc

CFLAGS = -Wall -g

program: file1.o

$(CC) $(CFLAGS) -o program file1.o

**11. Какая связка используется для очистки проекта?**

Обычно для очистки проекта используется цель clean, которая удаляет объектные файлы и исполняемый файл:

clean:

rm -f \*.o program

**12. В чем заключается модель КИС для проектов на языке C?**

Модель КИС (Конструкция, Инициализация, Сборка) для проектов на C подразумевает создание структуры проекта, где код организован в функции и модули, каждая функция отвечает за свою часть, и проходят этапы инициализации и сборки.

**13. Что такое стандарты кодирования и почему рекомендуется им следовать?**

Стандарты кодирования — это правила и рекомендации по написанию кода. Рекомендуется следовать им для обеспечения читаемости, поддерживаемости и совместимости кода среди команды разработчиков.

**14. Какой должна быть структура проекта согласно модели КИС и правилам сборки?**

Структура проекта в соответствии с моделью КИС (Конструкция, Инициализация, Сборка) и правилами сборки может выглядеть следующим образом:

my\_project/

│

├── src/ # Исходные файлы

│ ├── main.c # Основной файл программы

│ ├── zoo.c # Реализация функций зоопарка

│ └── file\_operations.c # Реализация функций работы с файлами

│

├── include/ # Заголовочные файлы

│ ├── zoo.h # Объявления структур и функций зоопарка

│ └── file\_operations.h # Объявления функций работы с файлами

│

├── tests/ # Тестовые файлы

│ └── test\_zoo.c # Тесты для функций зоопарка

│

├── Makefile # Makefile для сборки проекта

│

└── README.md # Документация проекта

**15. Что такое GitHub Actions и для каких задач применяется данный функционал?**

GitHub Actions — это функциональность GitHub, позволяющая автоматизировать процессы разработки, такие как тестирование, сборка и развертывание кода. Она позволяет создавать встроенные рабочие процессы (workflow), которые могут автоматически выполняться при определенных событиях, таких как коммиты, запросы на слияние или версии релиза.

**16. Что такое workflow в GitHub Actions? Приведите пример.**

Workflow в GitHub Actions представляет собой набор заданий (jobs) и шагов (steps), которые выполняются в определенной последовательности. Он определяется в файле YAML и может быть настроен для запуска по событиям.

Пример workflow для автоматического тестирования при каждом пуше в ветку main:

name: CI

on:

push:

branches:

- main

jobs:

build:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Set up Python

uses: actions/setup-python@v2

with:

python-version: '3.8'

- name: Install dependencies

run: |

python -m pip install --upgrade pip

pip install -r requirements.txt

- name: Run tests

run: |

pytest

**17. Что такое event в GitHub Actions? Приведите пример.**

Event в GitHub Actions — это триггер, который запускает выполнение workflow. Событие может быть связано с различными действиями в репозитории, такими как пуш, создание запроса на слияние, создание релиза и др.

on:

push:

branches:

- main

Здесь workflow будет запускаться каждый раз, когда кем-то будет выполнен пуш в ветку main.

**18. Что такое job в GitHub Actions? Приведите пример.**

Job — это группа шагов (steps), которые выполняются на одной и той же виртуальной машине. Каждый job может выполняться параллельно или последовательно в зависимости от настроек.

Copy

jobs:

build:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Build application

run: |

make

**19. Что такое step в GitHub Actions? Приведите пример.**

Step — это отдельный шаг внутри job, который выполняет определенное действие. Это может быть выполнение команды, установка зависимости или использование другого действия (action).

Пример:

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Build application

run: |

make

**20. Что такое action в GitHub Actions? Приведите пример.**

Action — это отдельный модуль кода, который выполняет одну конкретную задачу и может быть повторно использован в различных workflow. Вы можете использовать существующие действия, доступные в GitHub Marketplace, или создавать свои собственные.

Пример использования действия для проверки кода:

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Run a linter

uses: github/super-linter@v3

В этом примере github/super-linter — это действие, которое будет проверять код на наличие ошибок и несоответствий стандартам кодирования.