МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики» (МТУСИ)**

Кафедра «Системное программирование»

**ОТЧЕТ**

по лабораторной работе № 1 дисциплина **«Введение в информационные технологии»**

Выполнил: Студент гр.БВТ2207

Кузьмин И.А.

“ ” 2023г.

Проверил:

“ ” 2023г.

Москва, 2023

# Введение

# Цель работы:

Научиться пользоваться терминалом Git, заливать с помощью команд.

# Задание:

На вход программе подаются 3 коэффициента квадратного уравнения. Программа должна находить корни квадратного уравнения.

Залить данную задачу на локальный Git и GitHub.

# Реализация

В процессе выполнения лабораторной работы использованы следующие информационные источники [1-2].

**Практическая реализация**

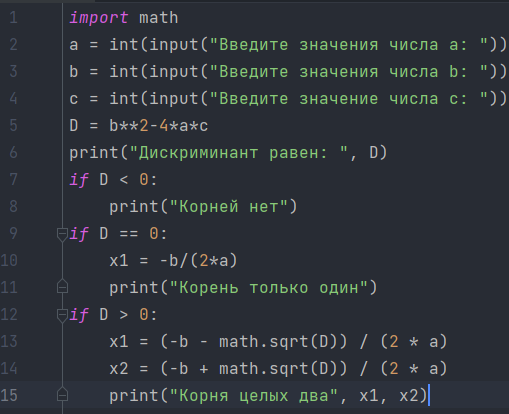


Рис. 1 – код программы

Данный код выполняет вычисление дискриминанта и нахождение корней квадратного уравнения вида `ax^2 + bx + c = 0`.

В начале кода импортируется модуль `math`, который содержит функции для математических операций, в данном случае используется функция `sqrt` (квадратный корень).

Затем пользователю предлагается ввести значения трех переменных `a`, `b` и `c` с помощью функции `input`. Каждое введенное значение преобразуется в целое число с помощью функции `int` и сохраняется в соответствующую переменную.

Далее вычисляется дискриминант `D` по формуле `b^2 - 4ac` и результат выводится на экран с помощью функции `print`.

Затем код проверяет несколько условий с помощью конструкции `if`:

- Если дискриминант меньше нуля (`D < 0`), то выводится сообщение "Корней нет".

- Если дискриминант равен нулю (`D == 0`), то вычисляется значение одного корня по формуле `-b/(2a)` и результат выводится на экран.

- Если дискриминант больше нуля (`D > 0`), то вычисляются значения двух корней по формулам `(-b - sqrt(D))/(2a)` и `(-b + sqrt(D))/(2a)`, где `sqrt` - функция квадратного корня из модуля `math`. Затем значения корней выводятся на экран.

Таким образом, код позволяет определить, сколько корней имеет квадратное уравнение и вычислить их значения, если они существуют.

**Создание репозитория:**

**Git init - выполняет следующие действия:**

1. Создает скрытую папку .git в текущем каталоге. В этой папке будут храниться все данные репозитория, включая история коммитов, ветки, конфигурационные файлы и другая информация.
2. Инициализирует пустой репозиторий Git в указанном каталоге или в текущем каталоге. Это означает, что Git будет начинать отслеживать изменения файлов в этом каталоге.
3. Создает файл .gitignore в корне репозитория. В этом файле можно указать шаблоны файлов или папок, которые Git должен игнорировать при отслеживании изменений.

После выполнения команды git init, вы сможете использовать другие команды Git, такие как git add, git commit, git branch и другие, для управления версиями вашего кода и сотрудничества с другими разработчиками.

**Git status:**

**выполняет следующие действия:**

1. Выводит имя текущей ветки, на которой вы находитесь. Обычно имя текущей ветки отображается в строке "On branch <branch\_name>".
2. Показывает список измененных файлов, которые уже добавлены в индекс, но еще не закоммичены. Они отображаются в разделе "Changes to be committed".
3. Показывает список измененных файлов, которые не были добавлены в индекс. Они отображаются в разделе "Changes not staged for commit".
4. Показывает список непроиндексированных (неслеживаемых) файлов в рабочем каталоге. Они отображаются в разделе "Untracked files".

Git add:

используется для добавления файла **main.py** в индекс репозитория Git. При выполнении этой команды Git начинает отслеживать изменения в указанном файле и готовит его для последующего коммита.

Git commit:

Фиксирует изменения

Git push:

1. Git проверяет текущую ветку, на которой вы находитесь, и связанный с ней удаленный репозиторий. Ветка и удаленный репозиторий могут быть предварительно настроены с помощью команды **git remote** или при клонировании репозитория.
2. Git отправляет изменения, которые находятся в ветке, на удаленный репозиторий. Это включает в себя новые коммиты, обновления существующих коммитов и другие изменения в истории.
3. Если ваше локальное состояние отстает от удаленного репозитория (например, кто-то другой внес изменения в удаленный репозиторий после вашего последнего **git pull**), Git может отклонить вашу операцию **git push**. В этом случае вам нужно будет сначала выполнить **git pull**, чтобы получить последние изменения из удаленного репозитория, а затем снова выполнить **git push**.
4. Git возвращает информацию о процессе пуша, включая добавленные коммиты и изменения в удаленном репозитории.

# Заключение

# Список использованных источников