### ****Шпаргалка по GitHub****

### ****Настройка Git:****

1. git config --global user.name "John Doe"  
   Устанавливает ваше имя для всех Git-репозиториев на компьютере. Это имя будет использоваться в коммитах.
2. git config --global user.email "johndoe@example.com"  
   Устанавливает вашу электронную почту для всех Git-репозиториев. Это нужно для идентификации автора коммитов.
3. git config --global core.editor nano  
   Устанавливает текстовый редактор nano для Git (например, для написания сообщений коммитов).  
   Для Windows можно использовать notepad:  
   git config --global core.editor notepad  
   Это нужно, если вы не хотите использовать vim (который может быть сложным для новичков).

### ****Основные команды:****

1. git add <файл>  
   Добавляет файл в "индекс" (staging area), чтобы Git начал отслеживать изменения в этом файле.
2. git commit  
   Фиксирует изменения, которые вы добавили с помощью git add, и сохраняет их в истории коммитов.  
   Если просто написать git commit, откроется текстовый редактор для написания сообщения коммита.
3. git commit -m "My commit message"  
   Фиксирует изменения с коротким сообщением, описывающим, что было сделано.
4. git log  
   Показывает историю коммитов в текущей ветке.
5. git log -n 5  
   Показывает последние 5 коммитов в истории.
6. git log --oneline  
   Показывает историю коммитов в сокращенном виде (по одной строке на коммит).
7. git log --oneline --graph  
   Показывает историю коммитов в сокращенном виде с графическим отображением веток
8. git log --oneline --graph --all  
   Показывает историю всех коммитов во всех ветках с графическим отображением.

### ****Работа с ветками:****

1. git switch <ветка>  
   Переключает вас на указанную ветку.
2. git switch -c <новая-ветка>  
   Создает новую ветку и сразу переключается на нее.
3. git branch  
   Показывает список всех локальных веток. Текущая ветка будет выделена символом \*.
4. git branch <имя-ветки>  
   Создает новую ветку, но не переключается на нее.
5. git branch -v  
   Показывает список веток с информацией о последнем коммите в каждой.
6. git branch -d <ветка>  
   Удаляет указанную ветку (если она уже слита с другой веткой).
7. git checkout <ветка>  
   Переключает вас на указанную ветку.
8. git checkout -b <новая-ветка>  
   Создает новую ветку и сразу переключается на нее.

### ****Сравнение изменений:****

1. git diff  
   Показывает разницу между текущими изменениями в рабочей директории и последним коммитом.
2. git diff <веткаA> <веткаB>  
   Показывает разницу между двумя ветками.

### ****Отмена изменений:****

1. git restore --staged <файл>  
   Убирает файл из индекса (staging area), но сохраняет изменения в рабочей директории.

### ****Слияние веток:****

1. git merge <ветка>  
   Сливает указанную ветку с текущей.
2. git mergetool --tool=vimdiff  
   Открывает инструмент для разрешения конфликтов слияния (в данном случае vimdiff).

### Команды и их объяснения Submodule

#### 1. ****Команды для работы с подмодулями:****

1. git submodule add <URL> <путь>
2. **Цель**: Добавить подмодуль (внешний репозиторий) в ваш репозиторий по указанному пути.

**Пример**:

ubmodule add https://github.com/Darkness1853/programming-C.git lab3-git/programming-C

**Объяснение**: Добавляет репозиторий programming-C как подмодуль в директорию lab3-git/programming-C.

2.git submodule init

**Цель**: Инициализировать подмодули, указанные в файле .gitmodules.

1. git submodule status
   * **Цель**: Показать состояние всех подмодулей в репозитории.
2. git submodule update
   * **Цель**: Обновить подмодули, загрузив их содержимое.

git submodule update

* + **Объяснение**: Загружает данные подмодулей, если они еще не были загружены.

1. git rm -r <путь>
   * **Цель**: Удалить подмодуль из индекса Git.

git rm -r programming-C

* + **Объяснение**: Удаляет подмодуль programming-C из индекса Git.

1. rm -rf <директория>
   * **Цель**: Удалить локальную копию подмодуля из директории .git/modules.

rm -rf .git/modules/programming-C

* + **Объяснение**: Удаляет подмодуль programming-C с диска.

1. cat .gitmodules
   * **Цель**: Просмотреть содержимое файла .gitmodules.
2. echo '[submodule "lab3-git/programming-C"]' > .gitmodules
   * **Цель**: Создать файл .gitmodules и добавить в него запись для подмодуля.

echo '[submodule "lab3-git/programming-C"]' > .gitmodules

echo ' path = lab3-git/programming-C' >> .gitmodules

echo ' url = https://github.com/Darkness1853/programming-C.git' >> .gitmodules

* + **Объяснение**: Ручное создание или исправление файла .gitmodules.

1. git add .gitmodules
   * **Цель**: Добавить файл .gitmodules в индекс Git.

git add .gitmodules

* + **Ошибка**: fatal: pathspec '.gitmodules' did not match any files.
  + **Причина**: Файл .gitmodules отсутствовал в репозитории.

#### 2. ****Команды для работы с ветками:****

1. git branch
   * **Цель**: Показать список локальных веток.

git branch

* + **Объяснение**: Выводит список всех локальных веток, выделяя текущую ветку символом \*.

1. git checkout <ветка>
   * **Цель**: Переключиться на указанную ветку.

git checkout master

* + **Объяснение**: Переключает рабочую директорию на ветку master.

1. git switch <ветка>
   * **Цель**: Аналог git checkout, но только для переключения веток.

git switch Mergesort-Impl

* + **Объяснение**: Переключает рабочую директорию на ветку Mergesort-Impl.

#### 4. ****Команды для работы с файловой системой:****

1. mkdir <директория>
   * **Цель**: Создать новую директорию.

mkdir lab3-git

* + **Объяснение**: Создает директорию lab3-git.

1. ls
   * **Цель**: Показать содержимое текущей директории.

ls

* + **Объяснение**: Выводит список файлов и папок в текущей директории.

#### 5. ****Команды для работы с удаленными репозиториями:****

1. git clone <URL>
   * **Цель**: Клонировать удаленный репозиторий на локальную машину.

git clone https://github.com/Darkness1853/programming-C.git

* + **Объяснение**: Клонирует репозиторий programming-C в текущую директорию.

#### 6. ****Команды для фиксации изменений:****

1. git add <файл>
   * **Цель**: Добавить файл или директорию в индекс Git.

git add .gitmodules

* + **Объяснение**: Добавляет файл .gitmodules в индекс для последующего коммита.

1. git commit -m "<сообщение>"
   * **Цель**: Зафиксировать изменения в репозитории с указанным сообщением.

git commit -m "Добавлен подмодуль programming-C"

* + **Объяснение**: Создает коммит с сообщением "Добавлен подмодуль programming-C".