Оглавление

Оглавление

[IDE 1](#_cu1r3qjeiyjq)

[Примеры 1](#_olji95bged95)

[VS Code 1](#_tx8bv0vdnfez)

[Сравнение VS, PyCharm, VS code 2](#_hjt8t88bj9rc)

[Небольшой пример 3](#_hh7466u02817)

[Пример кода: 3](#_1ib3x5uh3n7o)

[Полезные ссылки по теме 4](#_l95v75bno2vh)

[GIT основы 5](#_bh076whyoq3l)

[GIT работа 6](#_gb6jdws4unmb)

[git config 6](#_hmjljqmz9pwg)

[Настройки для работы с git 6](#_k074tnfa8uuc)

[GUI против CLI git 7](#_okcmbywyv0u4)

[CLI: HTTPS 9](#_8dknc1phpkrj)

[CLI: SSH 9](#_1qgizkwwzq6k)

# IDE

**IDE (Integrated Development Environment)** — это программа, которая объединяет инструменты для разработки, отладки и тестирования кода в одном интерфейсе.

## Примеры

Для C++

* Visual Studio (с поддержкой C++)
* CLion
* Code::Blocks
* Eclipse CDT
* …

Для Python

* PyCharm
* Spyder
* IDLE (встроенная в Python)
* …

## VS Code

**Visual Studio Code (VS Code)** — это бесплатный, лёгкий и кросс-платформенный **редактор кода** с поддержкой расширений, который позволяет работать с множеством языков программирования, обеспечивая отладку, управление Git и интеграцию с различными инструментами.

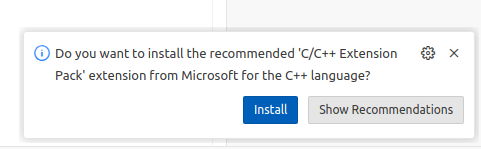
В отличие от полноценной IDE – **VS code** в базовой версии – это просто редактор текстовых документов, который можно **очень гибко** настроить под себя.   
  
Можно представить, что **VS code** – это некоторая основа \ скелет \ платформа, в которую можно устанавливать различные дополнения. Часть IDE специализируются под один язык программирования (или под несколько похожих), поскольку **VS code** можно настраивать под свои задачи, его можно превратить в IDE под те языки программирования, которые нужны именно вам, добавить или убрать плагины.

## Сравнение VS, PyCharm, VS code

| Приложение | Специализация | Производительность | Стоимость | Расширяемость |
| --- | --- | --- | --- | --- |
| PyCharm | Python | Тяжеловесная | Ограниченная **бесплатная**  Полная  **платная** | Ограничена плагинами JetBrains |
| Visual Studio | С++ С# | Тяжеловесная | Ограниченная **бесплатная**  Полная  **платная** | Ограничена расширениями Microsoft |
| Visual Studio Code | Универсальная | Легковесная | **Бесплатная** | Высокая (поддержка пользовательских плагинов) |

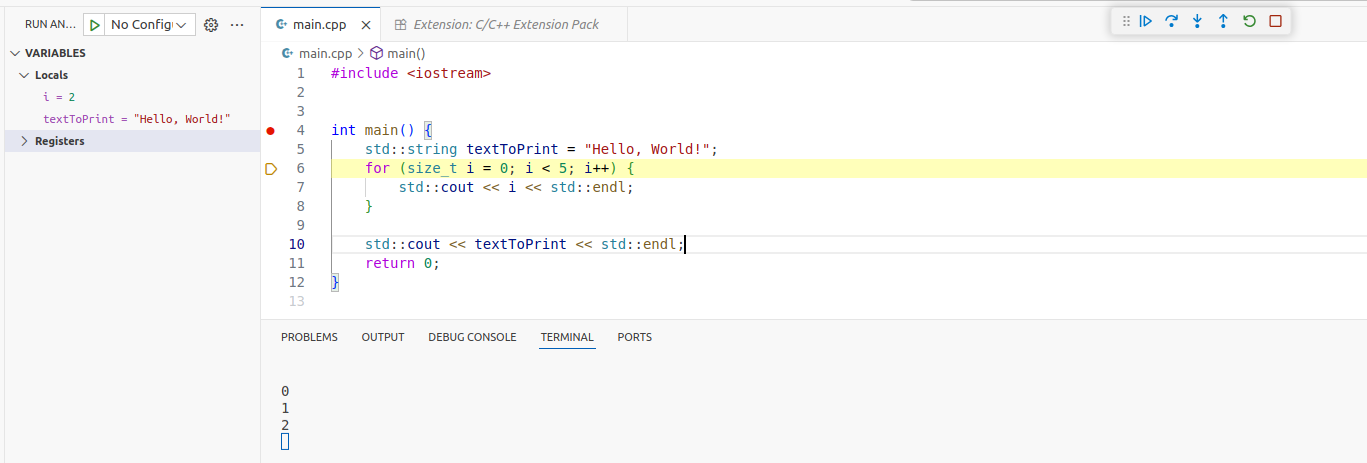
Инструкцию по установке и настройку VS code – в презентации: [1. Среда разработки (IDE)](https://docs.google.com/presentation/d/1pIF6R9z4KzYMeddQUz3gn_J-olG1AFp-TfB0cAUiL5Y/edit?usp=sharing)

При написания файла на C++ vs code может предложить установить вам расширение под C++:



## Небольшой пример

Во время мастер класса была запущена простая программа на C++ и установлена точка останова:



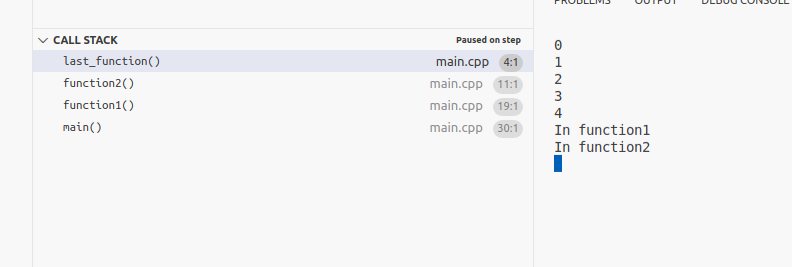
С помощью интерфейса отладки последовательно пройдены несколько шагов программы – при этом рассмотрены основные элементы интерфейса:

Где располагаются значения переменных, стек вызовов, как проводить шаги отладки.

### Пример кода:

| #include <iostream>  int last\_function() {  std::cout << "In last function" << std::endl;  return 0; } int function2() {  std::cout << "In function2" << std::endl;  last\_function();  std::cout << "exit from function2" << std::endl;  return 0; }  int function1() {  std::cout << "In function1" << std::endl;  function2();  std::cout << "Exit from function1" << std::endl;  return 0; }  int main() {  std::string textToPrint = "Hello, World!";  for (size\_t i = 0; i < 5; i++) {  std::cout << i << std::endl;  }    function1();   std::cout << textToPrint << std::endl;  return 0; } |
| --- |

Стек вызовов функций:



Упомянуты прочие расширения.

## Полезные ссылки по теме

* Инструкция по установке и создании первого проекта на C++ в VS Code: https://code.visualstudio.com/docs/cpp/config-mingw
* Документация по работе с VS Code: https://code.visualstudio.com/docs
* Просмотр всех расширений для VS Code: https://marketplace.visualstudio.com/VSCode

# GIT основы

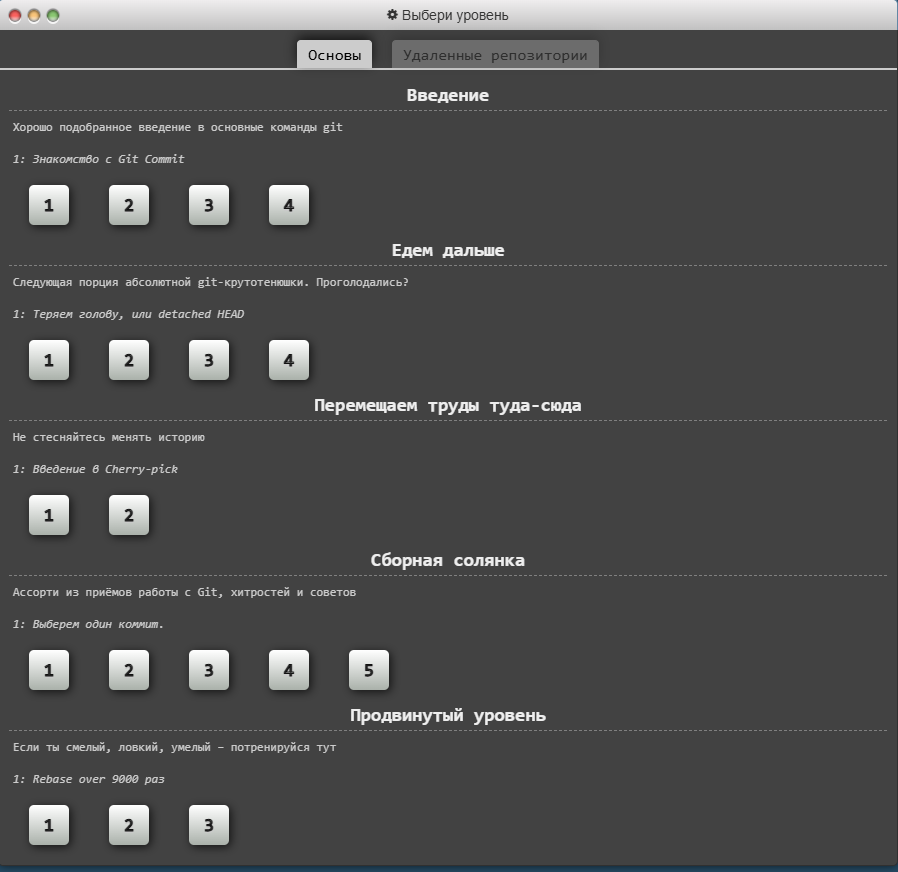
В презентации [2 git 1 Основы](https://docs.google.com/presentation/d/15iY195b4HyXN9wwVrZOj__uIm0YaP9kjP1BY5Tzxbmw/edit?usp=sharing) рассмотрены основные понятия по типу

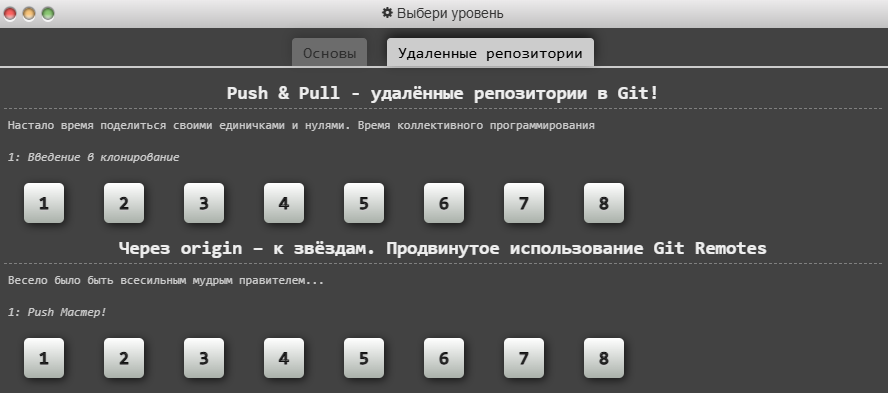
* Репозиторий
* Коммит
* Ветка
* Основные операции

В той же презентации рассмотрена инструкция по регистрации в github.

Для успешного освоения материала будет плюсом, если вы пройдете курс:  
<https://learngitbranching.js.org/?locale=ru_RU>

Это интерактивный тренажер.

Вы можете выбрать задачу из блока основных или из блока “Удаленные репозитории”



# GIT работа

В презентации [3 git 2 работа](https://docs.google.com/presentation/d/1o-DAWPefA8ImygtGFulerrnvIlRAXJdFx3SYA4ly9QI/edit?usp=sharing) рассмотрены:

* Настройка системы для работы с git
* Настройка VS Code для работы с git github
* Как работать с git в vs code:
  + Клонирование, сохранение и отправка изменений
* Как работать с git из командной строки с помощью токенов:
  + Клонирование, сохранение и отправка изменений
* Как работать с git из командной строки с помощью ssh:
  + Клонирование, сохранение и отправка изменений

Рассмотрим рассмотренное в презентации

## git config

git config — это команда для настройки параметров Git на глобальном, локальном или системном уровнях. Она определяет поведение Git, идентификацию пользователя и другие параметры.

Статья о конфигурации git из документации:   
[https://git-scm.com/book/ru/v2/Настройка-Git-Конфигурация-Git](https://git-scm.com/book/ru/v2/%D0%9D%D0%B0%D1%81%D1%82%D1%80%D0%BE%D0%B9%D0%BA%D0%B0-Git-%D0%9A%D0%BE%D0%BD%D1%84%D0%B8%D0%B3%D1%83%D1%80%D0%B0%D1%86%D0%B8%D1%8F-Git)

Рассмотрим полезные несколько параметров для настройки:

* core.editor – то, с помощью какой программы изменения текста будет производиться работа с git (например, в каком редакторе текста вы будете прописывать название и описание коммитов)
* color.ui – настройка цветовой подсветки вывода git команд
  + Больше параметров в [https://git-scm.com/book/ru/v2/Настройка-Git-Конфигурация-Git](https://git-scm.com/book/ru/v2/%D0%9D%D0%B0%D1%81%D1%82%D1%80%D0%BE%D0%B9%D0%BA%D0%B0-Git-%D0%9A%D0%BE%D0%BD%D1%84%D0%B8%D0%B3%D1%83%D1%80%D0%B0%D1%86%D0%B8%D1%8F-Git)
  + Больше параметров в git config --help

## Настройки для работы с git

В презентации описана пошаговая инструкция по настройке системы к работе с git. Однако плюсы и минусы этих подходов описаны кратко и оговорены устно. Рассмотрим плюсы и минусы работы с git с помощью GUI интерфейсов (редакторы кода, IDE и прочее) и с помощью CLI (интерфейс командной строки).

### GUI против CLI git

GUI – Graphical User Interface. Редакторы кода или IDE часто предоставляет интеграцию работы с git. Часть разработчиков пользуются GUI на постоянной основе. GUI удобный и простой в освоении, легко создавать ветки и работать с репозиторием, поскольку под основные действия есть отдельные кнопки и не нужно держать в голове множество команд и флагов к ним.

CLI– Command Line Interface. Интерфейс командной строки, основанный на текстовых командах, предоставляющий возможность работы с git. CLI предоставляет возможность работы с полным спектром команд git, что позволяет сделать работу более гибкой и удобной.

У обоих подходов есть свои плюсы и минусы, рассмотрим их в таблице:

| Характеристика | GUI | CLI |
| --- | --- | --- |
| Порог входа | Низкий – основная функциональность реализуется через отдельные кнопки | Высокий – нужно знать некоторый минимум основных команд (остальные можно либо искать в интернете, либо просматривать с помощью --help) |
| Визуальная наглядность | Наглядный diff, подсветка изменений | Текстовый вывод с автоматической подсветкой (можно отключить)  Менее наглядно |
| Контроль над операциями | Ограниченный GUI приложения | Полный контроль (доступны все команды) |
| Разрешение конфликтов | Легко | Сложнее |
| Автоматизация | Почти отсуствует | Полная |
| Работа на удаленных серверах | Не возможна | Возможна |
| Скорость работы на больших проектах | Низкая | Наивысшая |
| Риски совершения ошибки | Низкие (чаще всего опасные операции подсвечиваются или дополняются уведомлениями) | Высокие – ваши команды применяются сразу |
| Тонкая настройка сценариев работы | Отсутствует | Возможна благодаря системны alias и git alias |
| Доступность на различных системах | Зависит от доступности приложений | Доступна везде, где есть git и консоль |
| Возможность сложных массовых операций | Ограничена интерфейсом | Почти не ограничена  Например добавить в индекс все файлы, кроме файлов, с расширением “.log”, которые начинаются на “sys”: git add --all -- ':!sys\*.log' ':!\*/sys\*.log' |

По таблице можно сделать вывод, что CLI предоставляет максимальный контроль, но может быть труден в освоении, в то время как GUI интуитивно понятен, хоть и ограничен.

Заметка автора:

Первое взаимодействие с git у меня было в CLI, это было сложно, однако на “большой дистанции” это стало скорее плюсом, чем минусом. Сейчас я использую оба подхода:

* Основные действия по типу клонирования, коммитов, создания и работы с ветками, отправки изменений я делаю в терминале
* Разрешение конфликтов, проверка файлов на изменения – в GUI

Стоит также отметить, что для VS Code есть расширения, которые показывают, кто и когда менял конкретную строку кода и прочую информацию об истории изменения файлов – это в определенных условиях удобная функция.

В интерфейсе командной строки (CLI) можно работать с помощью HTTPS токенов и с помощью SSH. Рассмотрим эти подходы.

#### CLI: HTTPS

Устаревший способ работы с git, поскольку он менее безопасный. Принцип работы следующий – на github получается токен с определенными правами, который потом используется каждый раз в качестве пароля при взаимодействии с удаленными репозиториями (клонирование, скачивания новых файлов из удаленного репозитория или отправка изменений в него). Это не очень удобно, поскольку его нужно будет где-то хранить, копировать и вводить при любом действии с удаленным репозиторием. Если хранить его неаккуратно, то он может попасть в руки злоумышленников.   
  
На данным момент не рекомендуется использовать HTTPS токен в качестве аутентификатора. Их уместно использовать для скриптов автоматизации (CICD), если обращаться с ними аккуратно. В этом сценарии использование токенов оправдано, к тому же github позволяет сгенерировать токен с набором специфических прав – например, только просматривать репозитории, или просматривать (или изменять) только один конкретный репозиторий.

#### CLI: SSH

Рекомендуемый способ работы с удаленными репозиториями. Принцип работы следующий – вы создаете на своем устройстве пару ключей (открытый и закрытый). Открытый ключ вы загружаете на github. Открытый ключ можно передавать куда угодно – по нему не получится получить закрытый и действовать от вашего имени. Закрытый ключ остается локально на вашем компьютере и никуда не передается, что повышает безопасность. К тому же есть техническая возможность зашифровать файл с закрытым ключом паролем (даже если злоумышленник получит доступ к компьютеру, он не сможет скомпрометировать ваш ключ).

Вторым плюсом работы с github с помощью ssh ключей заключается в том, что вам не нужно вводить свои данные при работе с удаленным репозиторием – подтверждение производится с помощью пары ключей (публичный у github, закрытый у вас).

Потому рекомендуется использовать ssh ключи. Инструкция по настройке представлена в презентации.