



Объектно-ориентированное программирование

2023



Как со мной связаться?

Старший преподаватель кафедры 806

Дзюба Дмитрий Владимирович

ddzuba@yandex.ru



Подключитесь
online



Что будет в курсе?

Отчетность по курсу рейтинг

5-балльная система	Рейтинговая система	Европейская система
5 - Отлично	90-100	A
4 – Хорошо	82-89	B
	75-81	C
3 - Удовлетворительно	67-74	D
	60-66	E
2 - Неудовлетворительно	Менее 60	F

Балы даются:

1. Сделанная и сданная Лабораторная работа (7 шт) – до 14 баллов.
2. Зачет до 30 баллов



№	Критерий	Процент
1	Лабораторная работа сдана в течении 2х недель после выдачи	3
2	Созданы Unit-тесты покрывающие все возможные варианты работы программы (code-coverage около 100%)	4
3	Хорошее качество кода/алгоритмов	3
4	Студент может объяснить используемые алгоритмы, отвечает на дополнительные вопросы без ошибок	4

Критерий приема ЛР

Где что?

- Материалы лекций
- Лабораторные работы

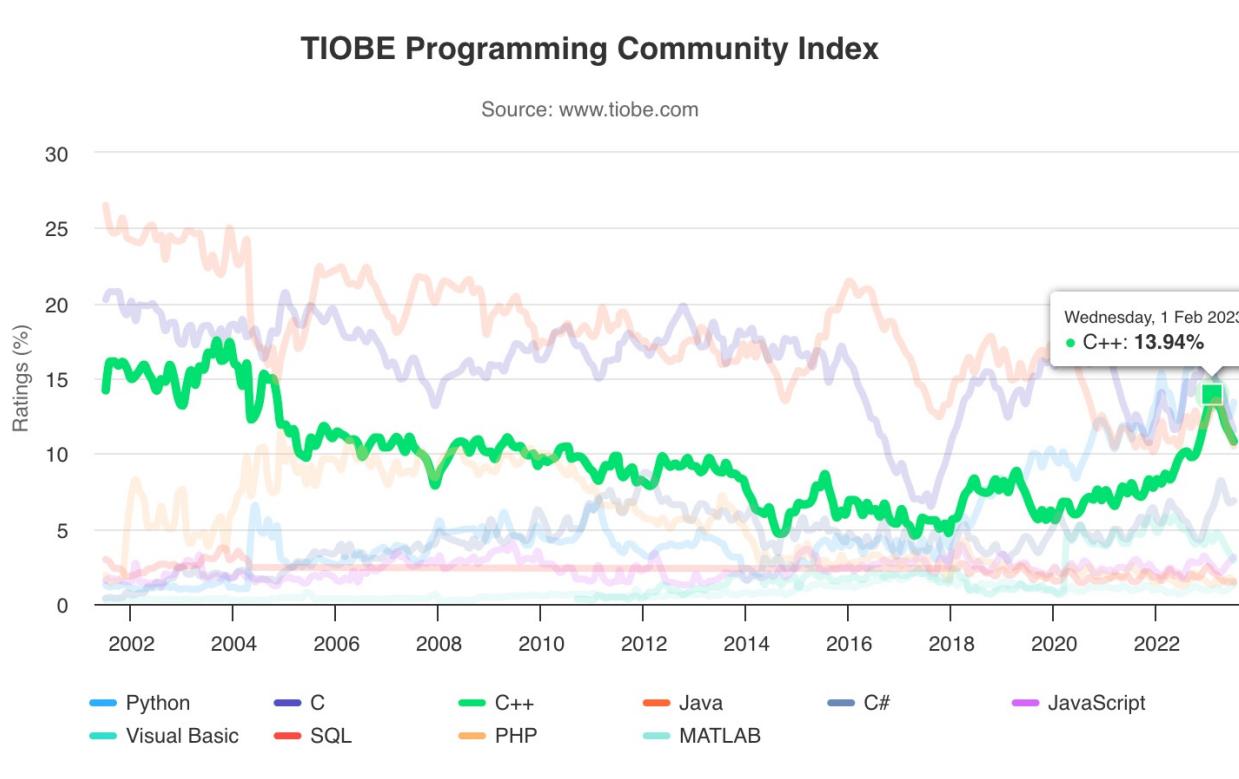


Объектно-ориентированное программирование

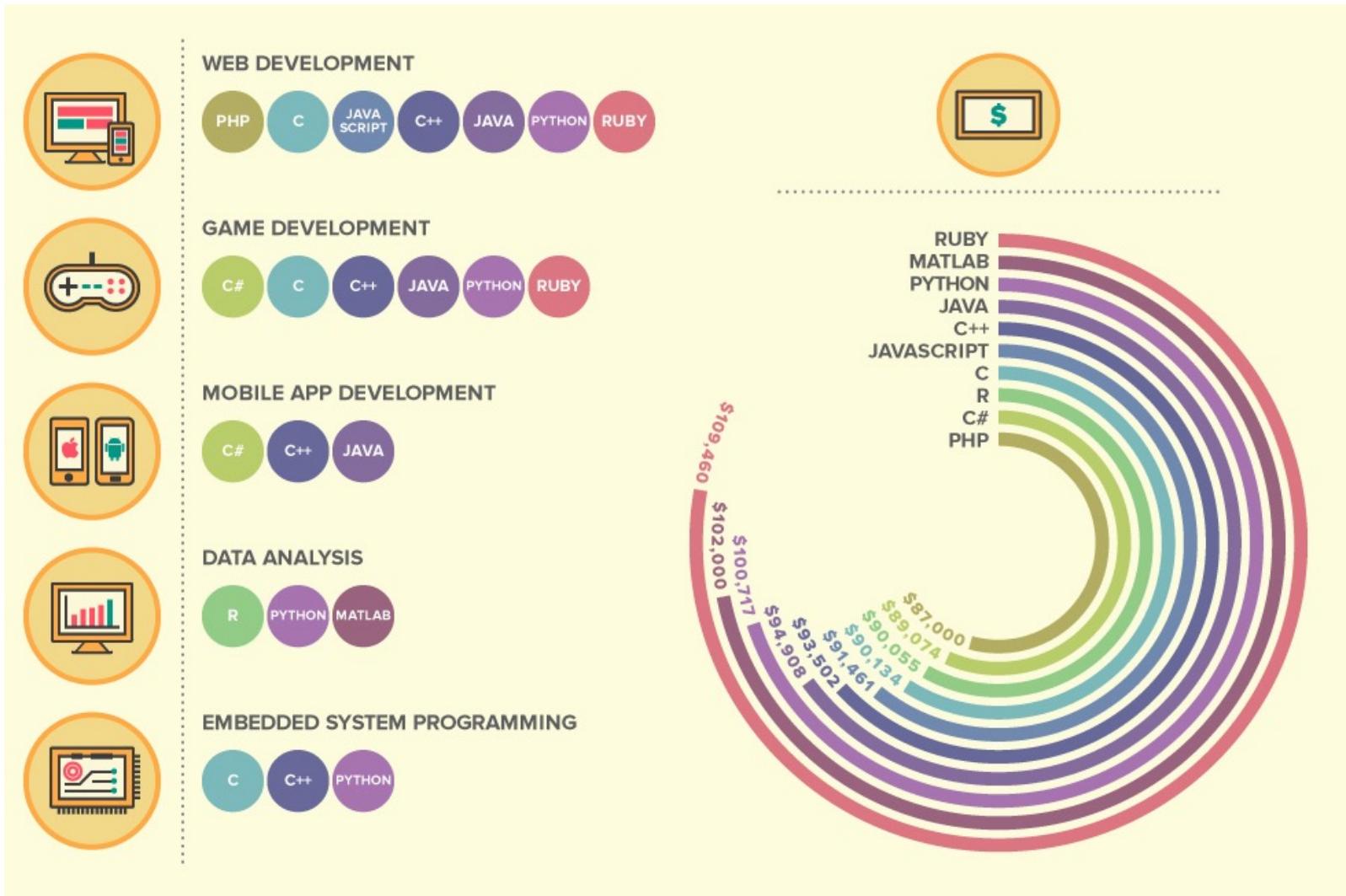
М8О-209Б-22, М8О-203Б-22, М8О-201Б-22,
М8О-208Б-22, М8О-207Б-22, М8О-205Б-22,
М8О-210Б-22, М8О-202Б-22, М8О-212Б-22,
М8О-206Б-22

<https://lms.mai.ru/course/view.php?id=7393>

Почему C++



Компании, в которых
востребовано знание C++
Билайн, МТС, Касперский, VK,
Yandex ...



План занятия

Изучаем необходимые для работы с C++ инструменты

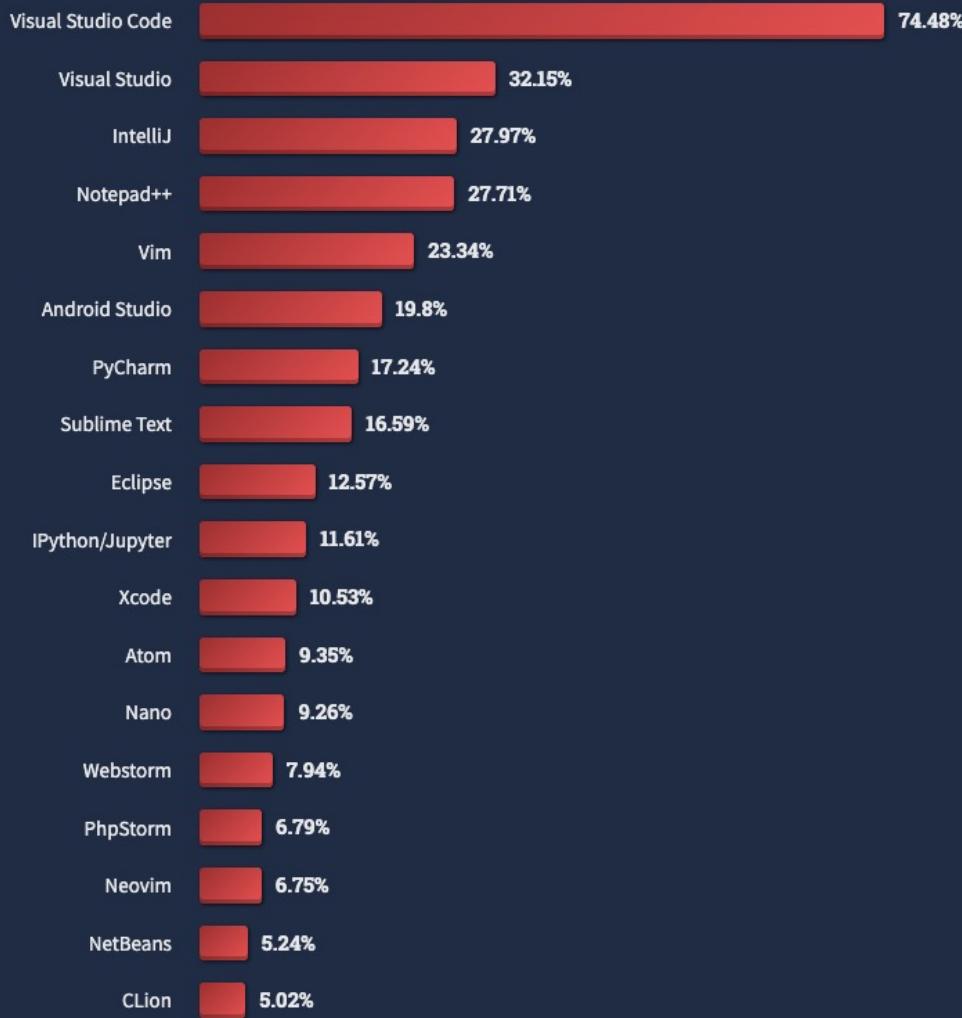
Пишем первую программу на C++



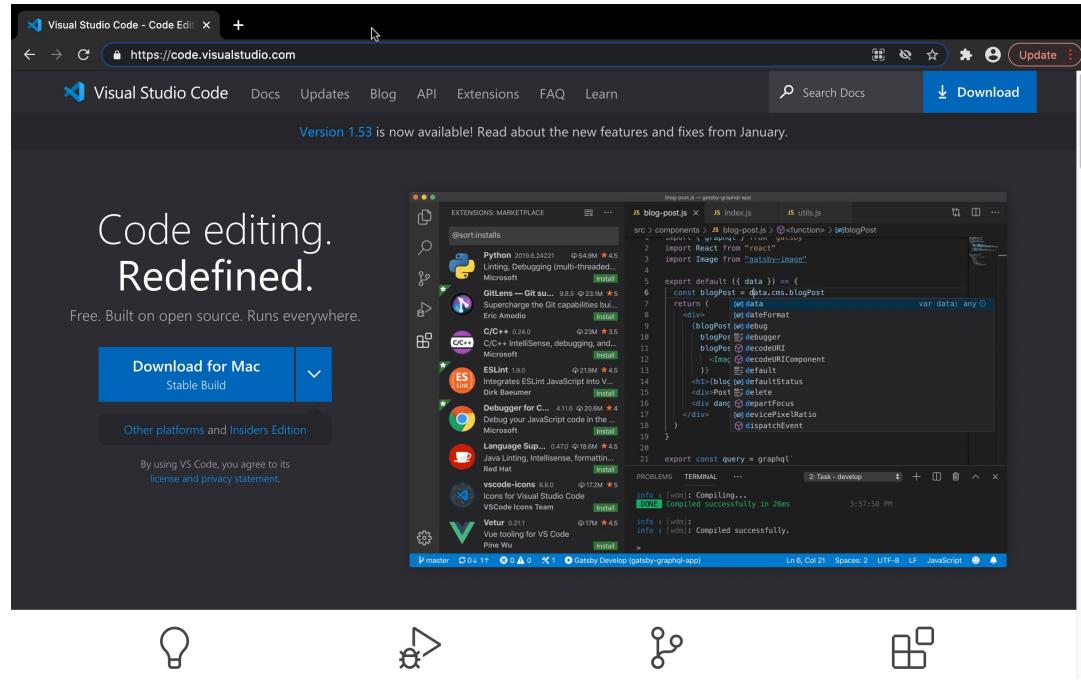
Visual Studio Code

СРЕДА РАЗРАБОТКИ



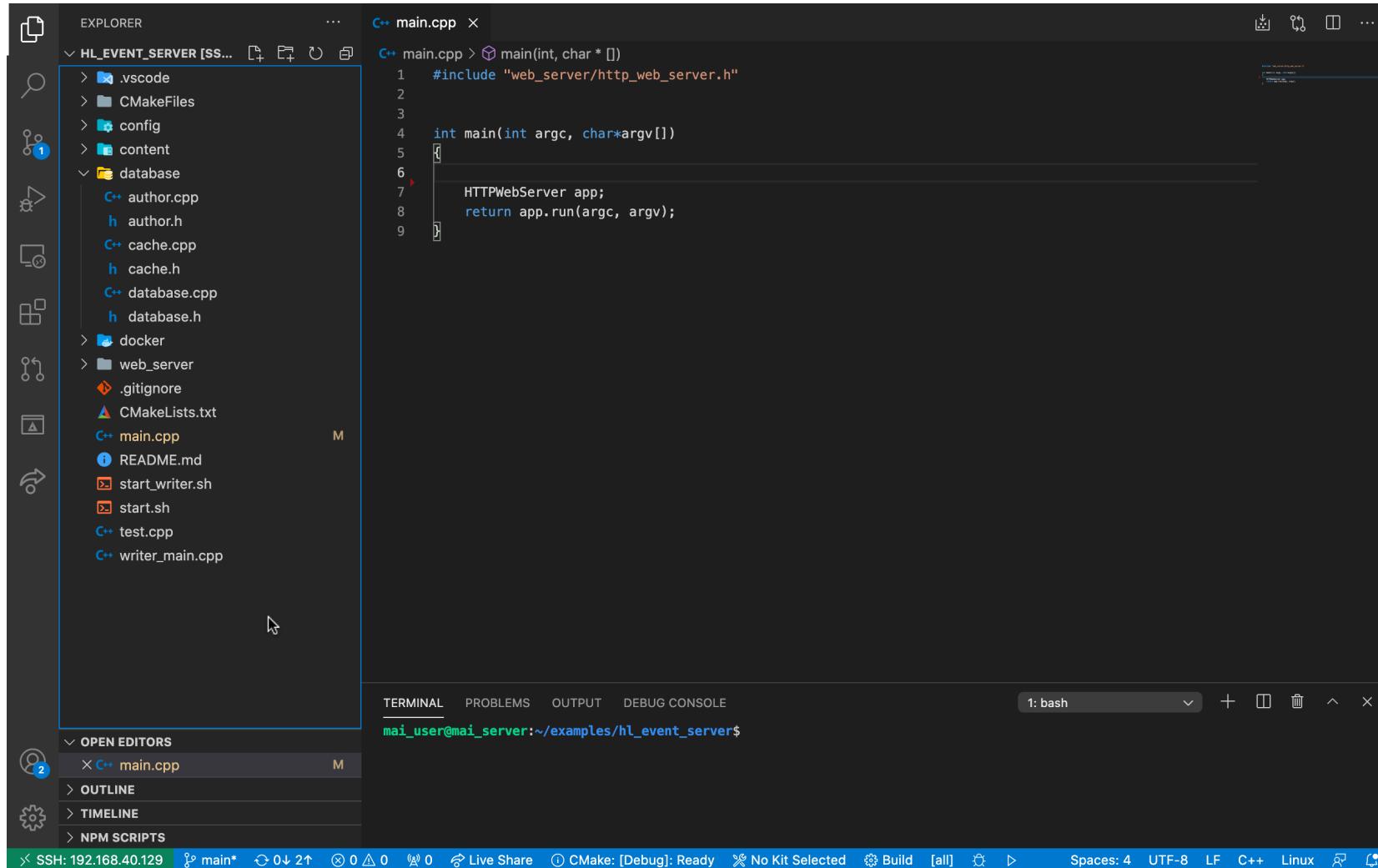


Самый популярный
редактор
<https://survey.stackoverflow.co/2022/>

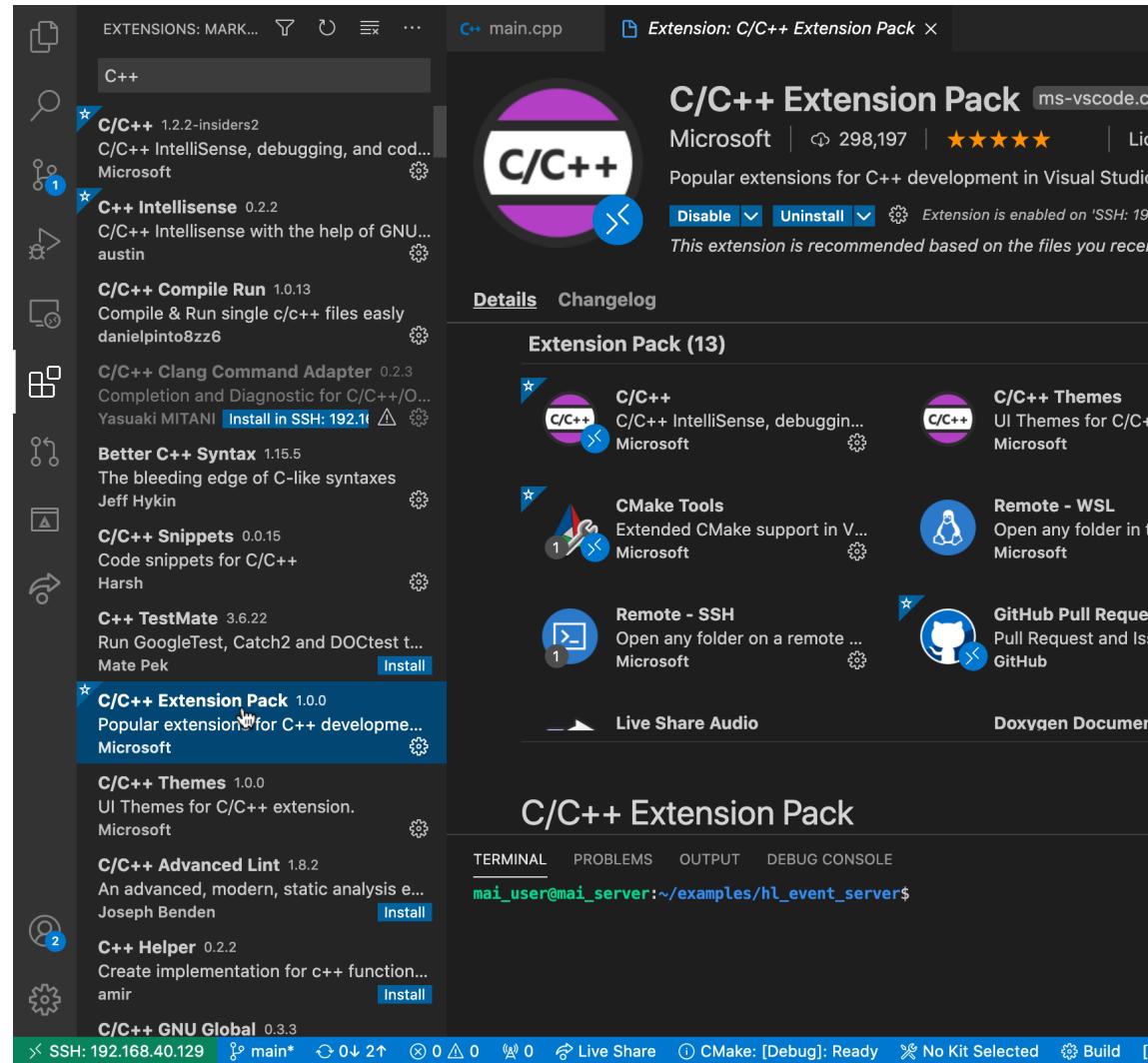


download & install
<https://code.visualstudio.com>

Основные элементы управления



<https://code.visualstudio.com/docs>



C++ extension pack



C++

мы будем использовать
любой доступный
компилятор GCC с
поддержкой стандарта C++ 17

Установка компилятора

1. Macos

необходимо установить Xcode и Command Line Tools

<https://developer.apple.com/download/all/?q=xcode>

2. Windows

необходимо установить или <https://visualstudio.microsoft.com/ru/downloads/> или

<https://www.mingw-w64.org/downloads/>

3. Ubuntu

установите gcc версии 12 или выше





cmake

КРОСС-
ПЛАТФОРМЕННАЯ
СБОРКА

ИНСТАЛЛЯЦИЯ

Linux

> sudo apt install cmake

или <https://cmake.org/download/>



простой сmake файл

```
cmake_minimum_required(VERSION 3.10)

# set the project name
project(MyProject)

# add the executable
add_executable(my_project tutorial.cpp)
```

Библиотеки

```
cmake_minimum_required(VERSION 2.8)
# Проверка версии CMake.

# Если версия установленной программы
# старее указаной, произайдёт аварийный выход.

project(hello_world) # Название проекта

set(SOURCE_EXE main.cpp)
# Установка переменной со списком исходников для исполняемого файла

set(SOURCE_LIB foo.cpp)
# Тоже самое, но для библиотеки

add_library(foo STATIC ${SOURCE_LIB})
# Создание статической библиотеки с именем foo

add_executable(main ${SOURCE_EXE})
# Создает исполняемый файл с именем main

target_link_libraries(main foo) # Линковка программы с библиотекой
```

Более сложный пример

```
cmake_minimum_required(VERSION 3.2)

project(hl_event_server C CXX)

SET (EXAMPLE_BINARY "event_server")

find_package(OpenSSL)
find_package(Threads)
find_package(Boost COMPONENTS filesystem system program_options regex REQUIRED)

include_directories(${Boost_INCLUDE_DIR})

add_executable(${EXAMPLE_BINARY} main.cpp
               config/config.cpp
               database/database.cpp
               database/cache.cpp
               database/author.cpp)

target_include_directories(${EXAMPLE_BINARY} PRIVATE "${CMAKE_BINARY_DIR}")
target_compile_options(${EXAMPLE_BINARY} PRIVATE -Wall -Wextra -pedantic -Werror )
target_link_libraries(${EXAMPLE_BINARY} PRIVATE
                     ${CMAKE_THREAD_LIBS_INIT}
                     ${Boost_LIBRARIES})

set_target_properties(${EXAMPLE_BINARY} PROPERTIES LINKER_LANGUAGE CXX)
set_target_properties(${EXAMPLE_BINARY} PROPERTIES CXX_STANDARD 17 CXX_STANDARD_REQUIRED ON)
```



Полное описание

[HTTPS://CMAKE.ORG/CM
AKE/HELP/LATEST/GUIDE/
TUTORIAL/INDEX.HTML](https://cmake.org/cmake/help/latest/guide/tutorial/index.html)



git

Git

РАБОТА С ИСХОДНЫМ КОДОМ

Инсталляция

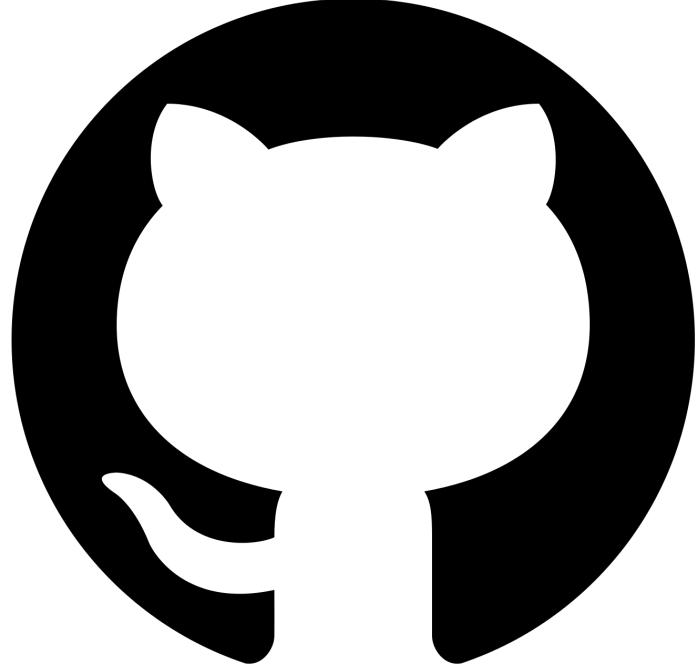
Linux

> sudo apt install git

Для windows

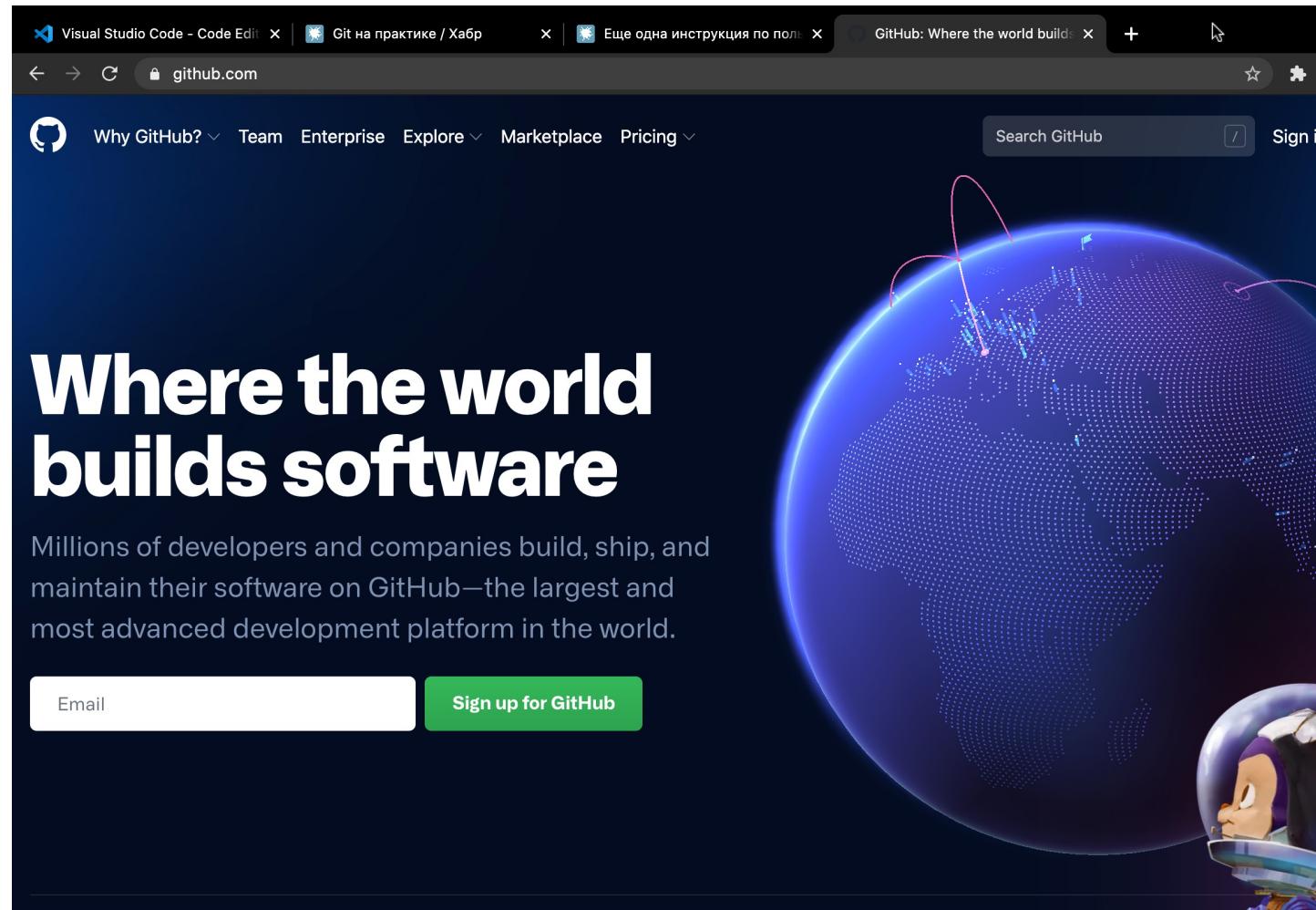
<https://git-scm.com/download/win>



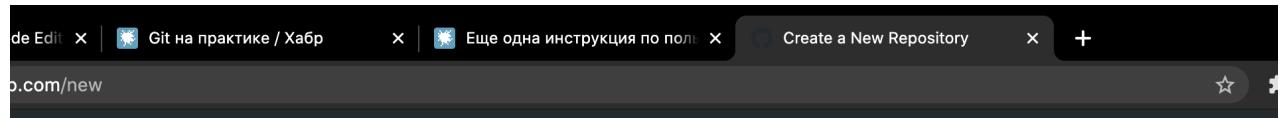


GitHub

РЕПОЗИТОРИЙ ИСХОДНОГО КОДА



Создаем аккаунт



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner *



Repository name *

/

Great repository names are short and memorable. Need inspiration? How about [crispy-octo-system](#)?

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.



Add a README file

This is where you can write a long description for your project. [Learn more](#).



Add .gitignore

Choose which files not to track from a list of templates. [Learn more](#).



Choose a license

A license tells others what they can and can't do with your code. [Learn more](#).

Создаем проект

Инициализируем локальный репозиторий

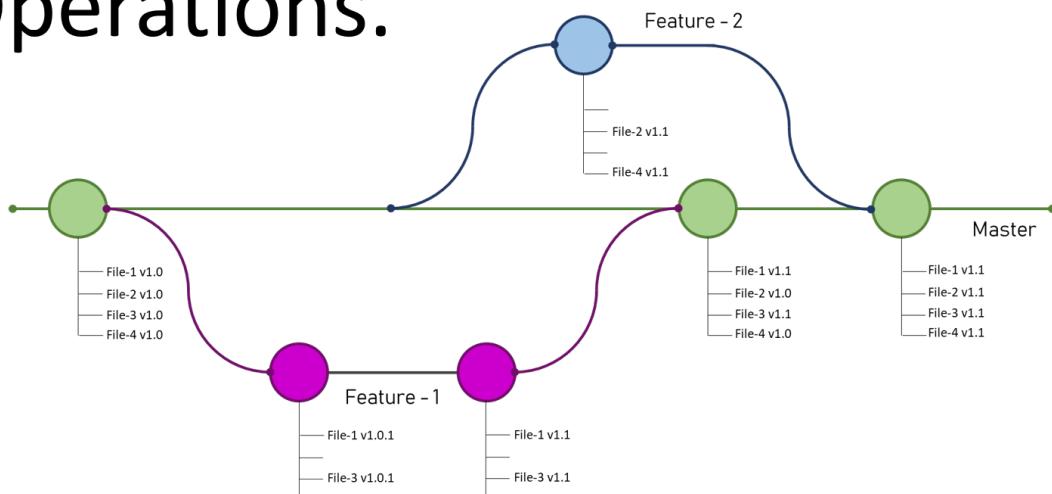
1. Вариант №1
> git init
2. Вариант №2
> git clone
https://github.com/DVDemon/hl_event_server.git

Имена по умолчанию

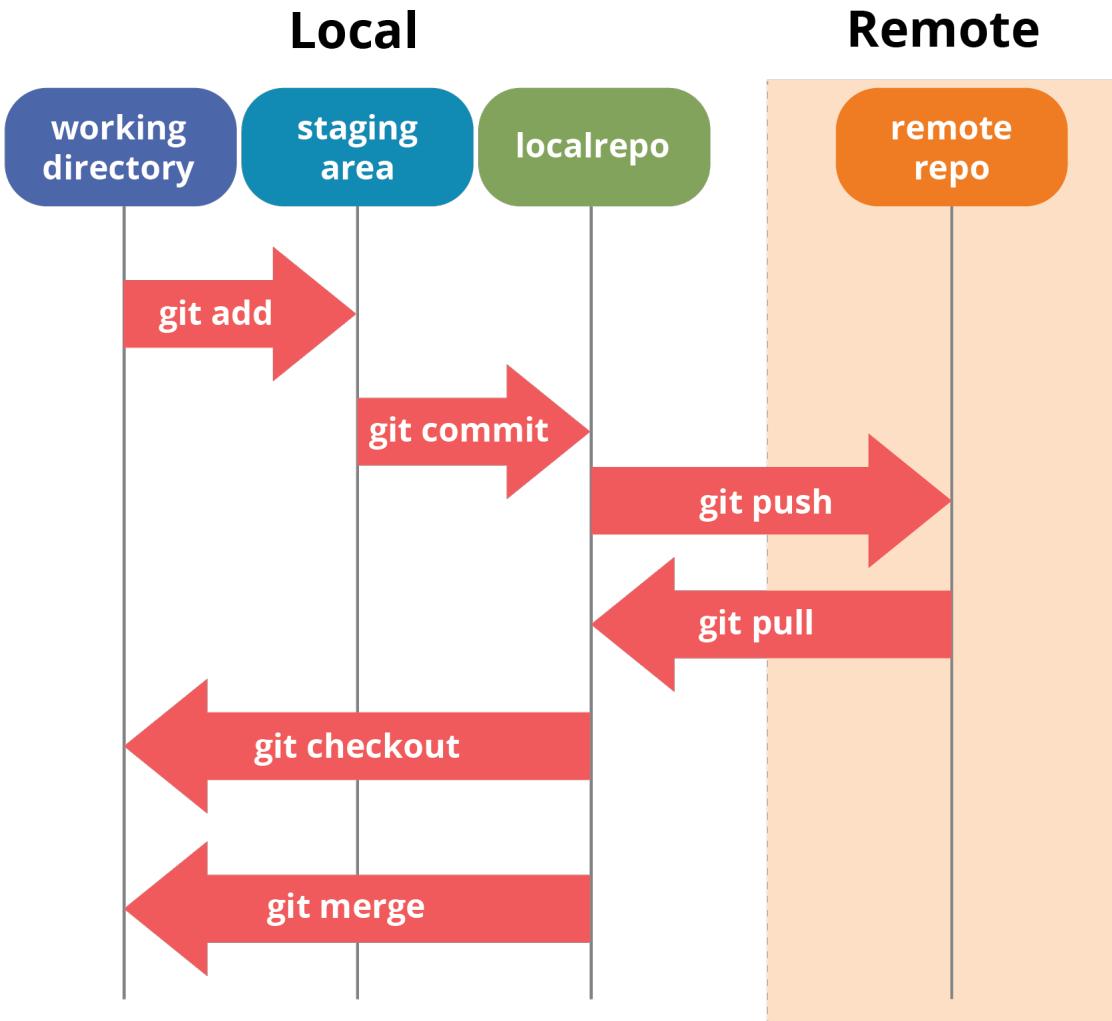
origin имя удаленного репозитория

master имя начальной ветки (branch)

GIT Branch and its Operations.



Концепция
веток



Репозиторий кода

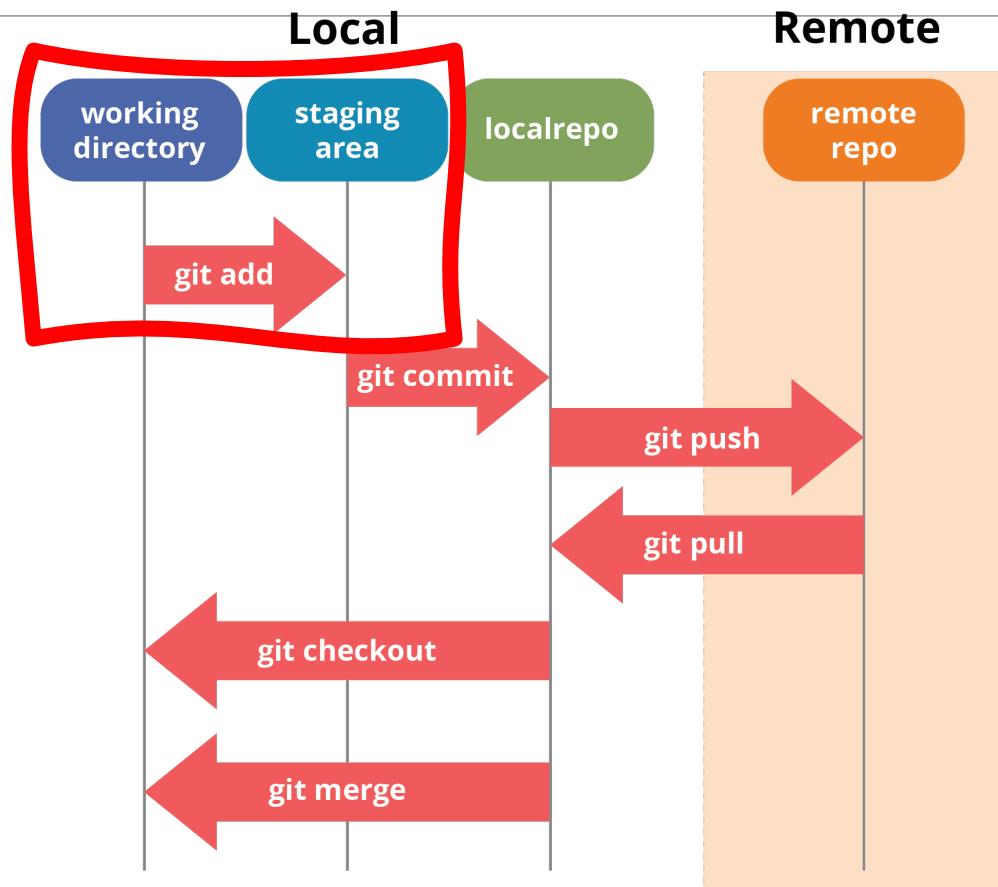
git add

`git add <file1> <file2>`

Добавить файлы file1 и file2 в staging.

`git add *.cpp`

Добавить все cpp файлы из текущей директории в staging



git status

git status

Посмотреть список изменений

```
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified: ..../main.cpp
        modified: ..../tests.cpp
```

```
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```



git commit

git commit

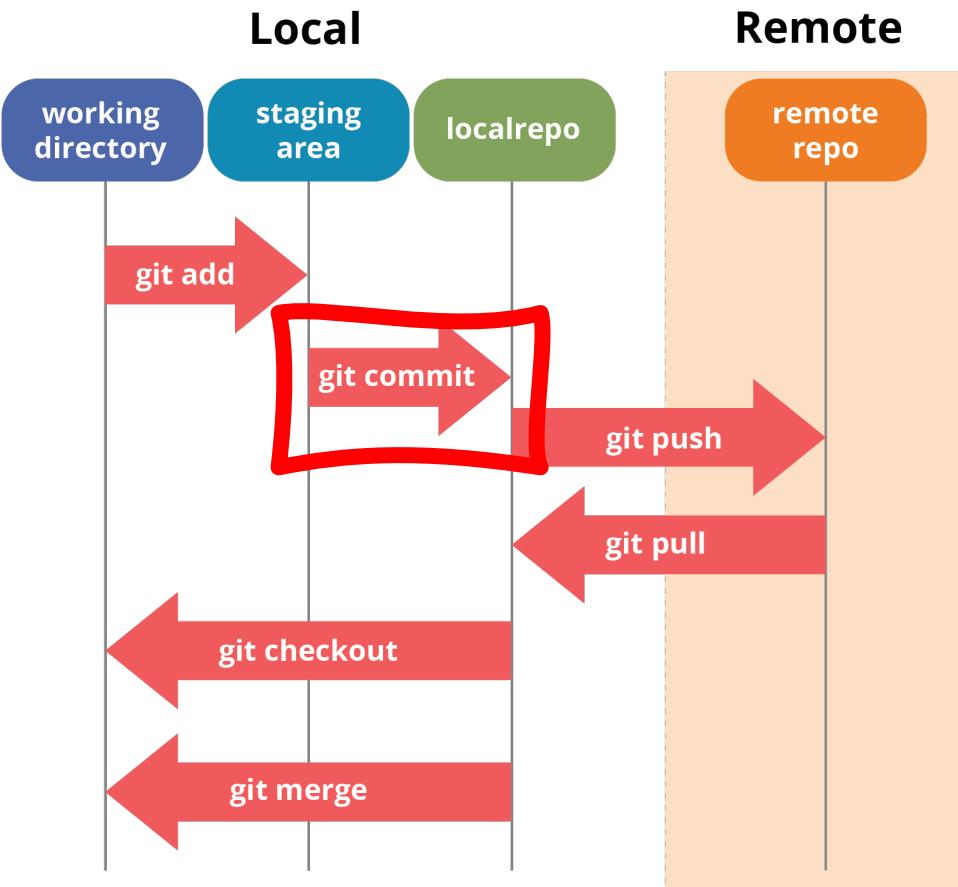
Сохраняем все изменения в локальном репозитории

git commit -m "Commit message"

Сохраняем все изменения в локальном репозитории и
добавляем комментарий
"Commit message"

git commit --amend

Модифицируем последний коммит.



git log

git log

Смотрим историю изменения репозитория

git log <filename>

Смотрим историю по файлу filename

```
commit b27553010af53e7b537da3a4d573886db56a883d (HEAD -> master)
Author: Dmitriy Dzyuba <ddzuba@yandex.ru>
Date: Sat Feb 27 17:36:23 2021 +0000
```

tests

```
commit f2837c71836fdb280df5eae239e904fed8033ae3 (origin/master)
Author: Dmitriy Dzyuba <ddzuba@yandex.ru>
Date: Sat Feb 13 17:26:09 2021 +0000
```

first commit



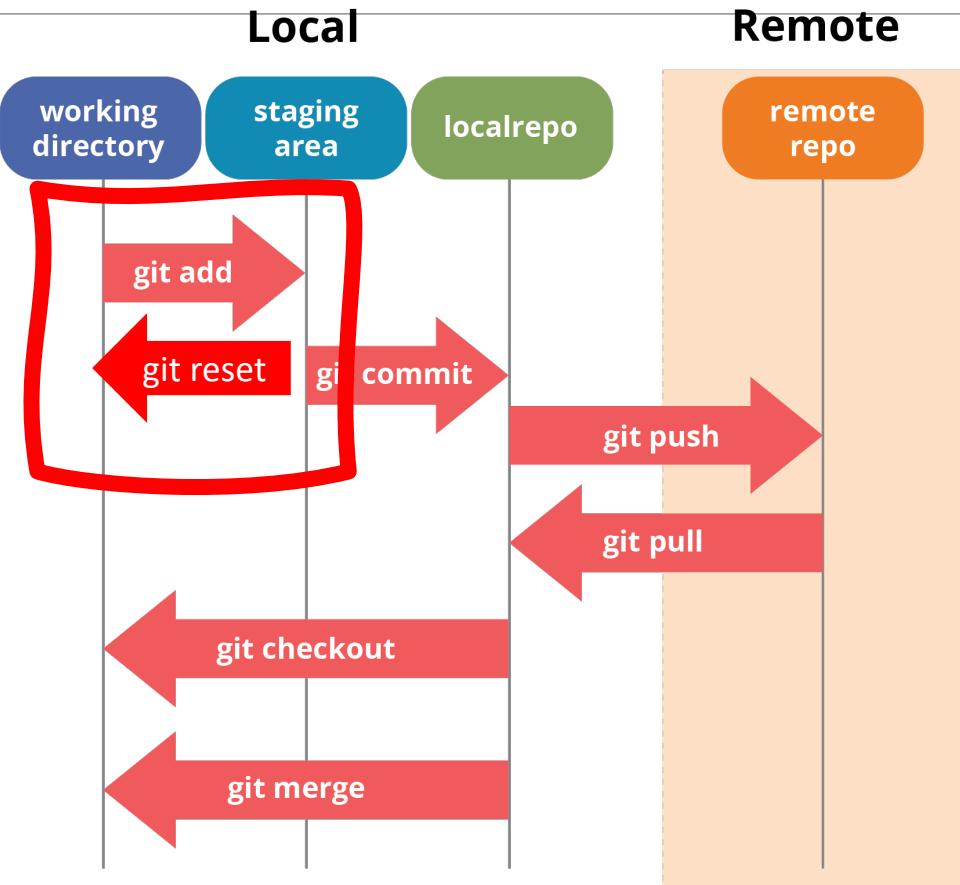
git reset

git reset

Удаляем все из staging (обратная операция к git add)

git reset <filename>

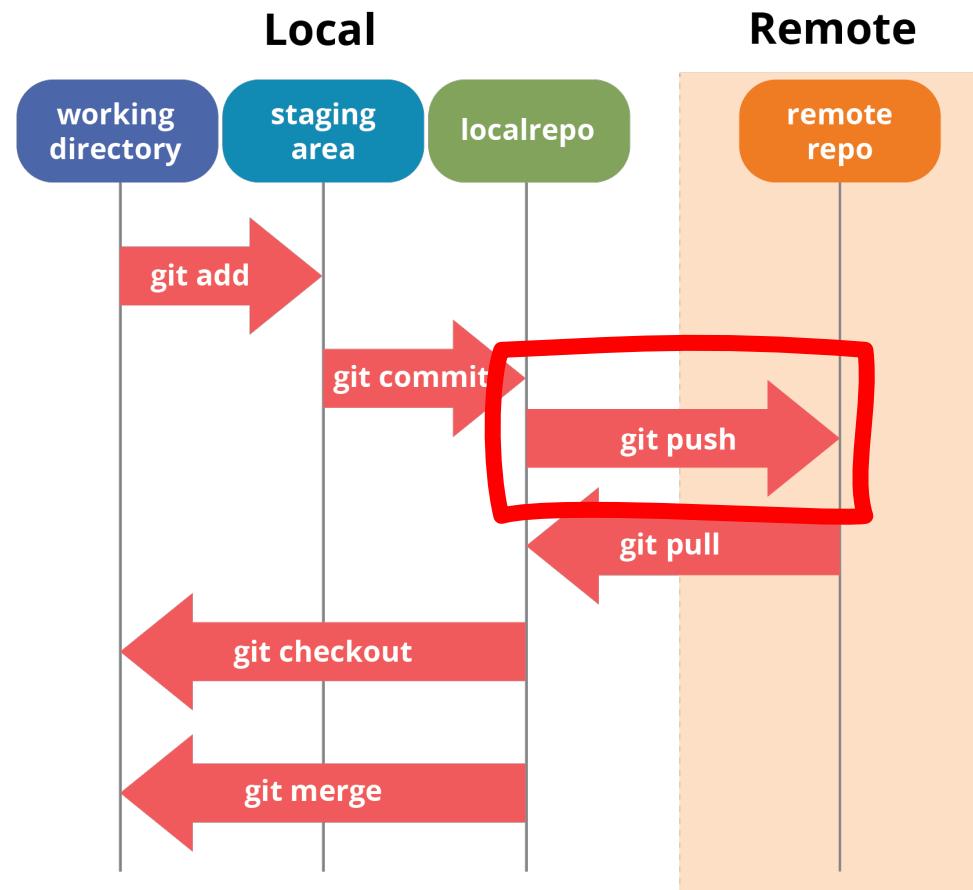
Удаляем все из staging по файлу filename



Работа с удаленным репозиторием

git push

git push	Отправляет изменения из текущей ветки в 'origin'.
git push <repo> <branch>	Отправляет изменения из текущей ветки в <repo> <branch>



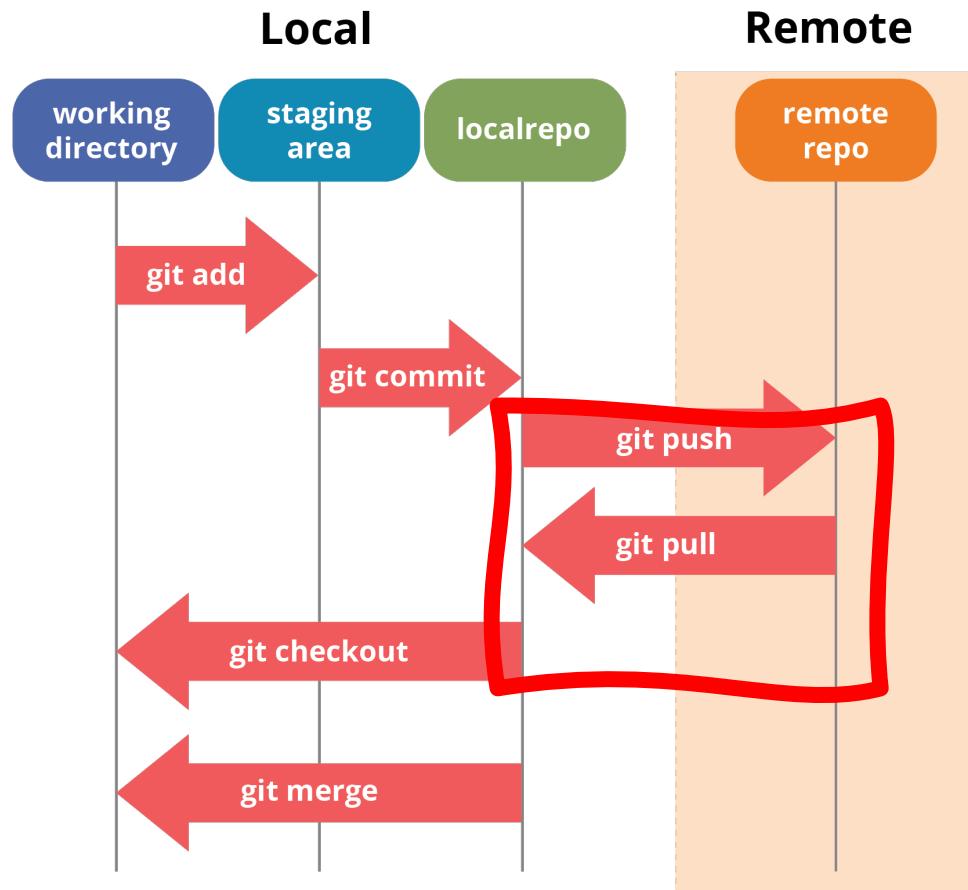
git pull

git pull

Загружает данные из удаленного репозитория в текущий.

git pull <repo> <branch>

Загружает только данные из определенной ветки



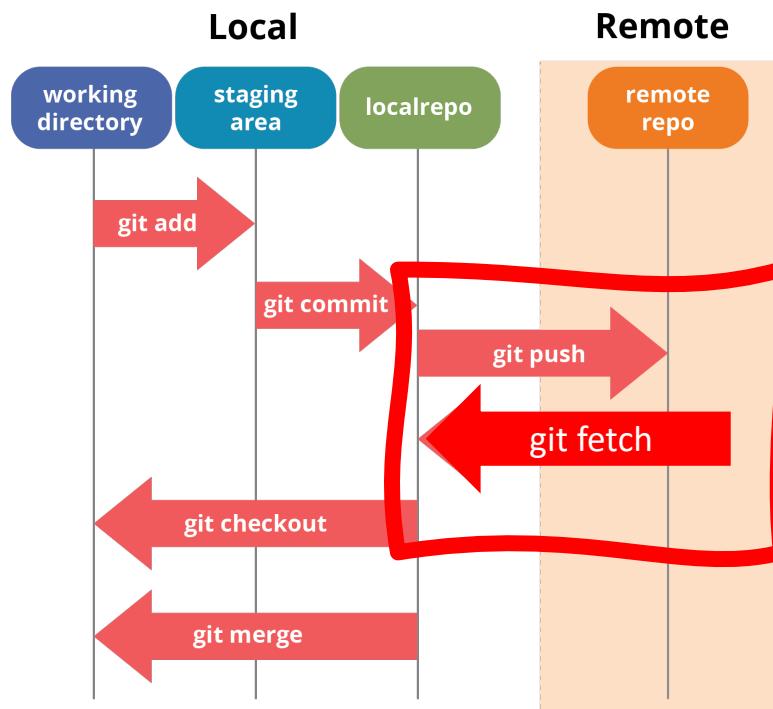
git fetch

git fetch

Скачивает состояние удаленного репозитория origin, но не меняет текущих файлов (как это делает git pull). Потом можно слить загруженные commit-ы с помощью merge

git fetch <repo> <branch>

Из репозитория <repo> и ветки <branch>.



branch	git branch	List branches.
	git branch <branch-name>	Create new branch <branch-name>
checkout	git checkout <branch-name>	Switch to editing branch <branch-name>
merge	git merge <branch-name>	Merge <branch-name> into current branch.

Работа с ветками

Модульные тесты

- ❑ Модульное тестирование, иногда блочное тестирование или юнит-тестирование ([англ. unit testing](#)) — процесс в [программировании](#), позволяющий проверить на корректность отдельные модули [исходного кода](#) программы, наборы из одного или более программных модулей вместе с соответствующими управляющими данными, процедурами использования и обработки.
- ❑ Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к [регрессии](#), то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

Библиотека Google Test

```
## Install gtest Linux
sudo apt-get install libgtest-dev
cd /usr/src/gtest/

# Или git clone https://github.com/google/googletest.git
sudo cmake -DBUILD_SHARED_LIBS=ON
sudo make
sudo cp *.so /usr/lib
```

Подключаем тесты

```
find_package(GTest REQUIRED)

add_executable(gtests tests.cpp)
target_link_libraries(gtests ${GTEST_LIBRARIES}
${CMAKE_THREAD_LIBS_INIT})

enable_testing()
add_test(gtests gtests)
```

Пишем тест

```
#include <gtest/gtest.h>

TEST(test_add_figure, basic_test_set)
{
    testing::internal::CaptureStdout();
    std::cout << "Hello world";
    std::string output = testing::internal::GetCapturedStdout();
    ASSERT_TRUE(output=="Hello world");
}

int main(int argc, char **argv) {
    testing::InitGoogleTest(&argc, argv);
    return RUN_ALL_TESTS();
}
```

Лабораторная работа №1

1. Установить компилятор C++
2. Установить Cmake
3. Установить Visual Studio Code
4. Установить Google Test
5. Создать аккаунт на GitHub
6. Создать программу на C++
7. Залить программу на GitHub

План занятия

Изучаем базовые конструкции языка C++

Изучаем Google Test

Типы данных

Математические и логические операторы

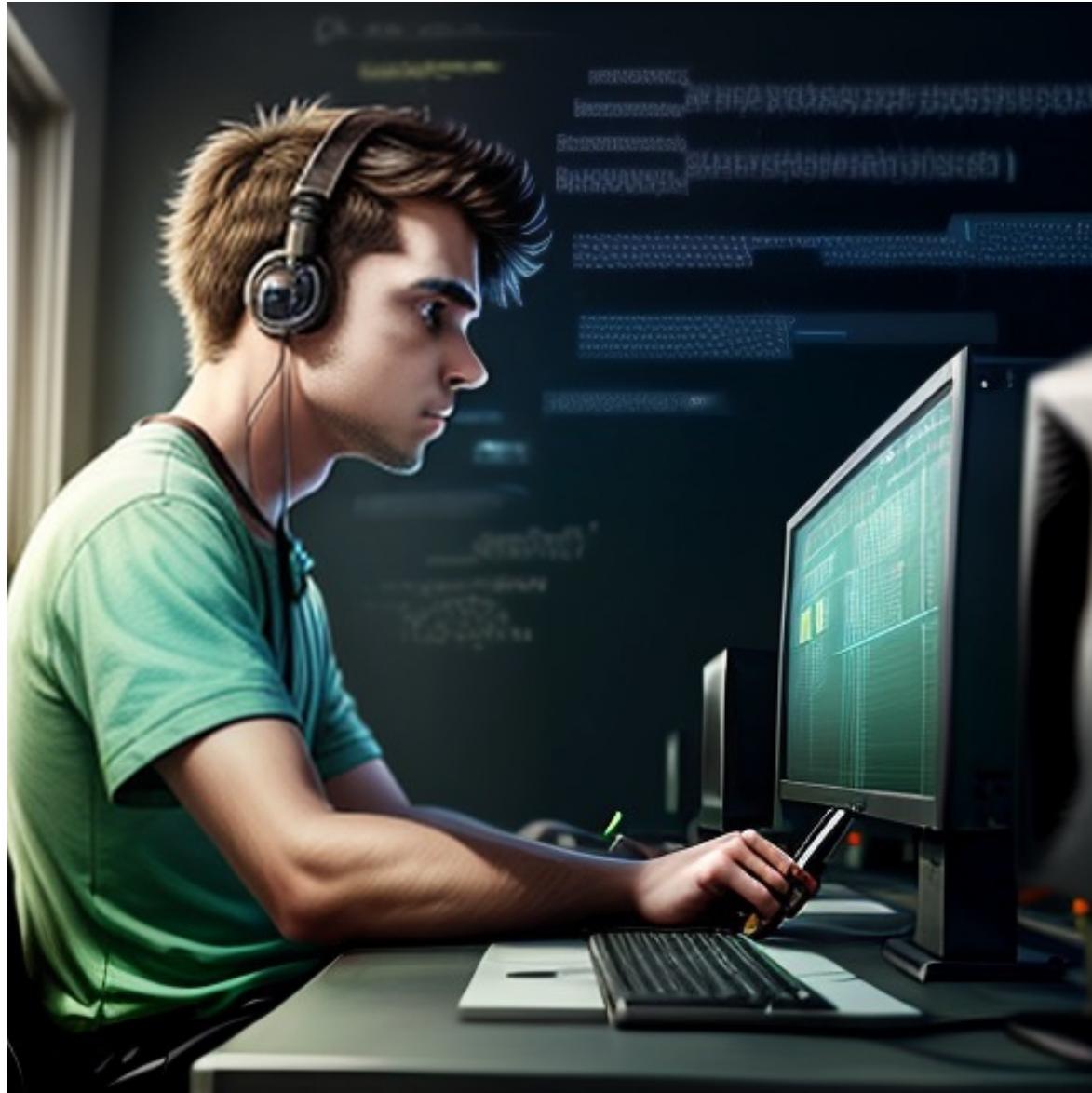
Константы

Приведения типов

Условия

Циклы





Переходим в
код



Спасибо!
НА СЕГОДНЯ ВСЕ
