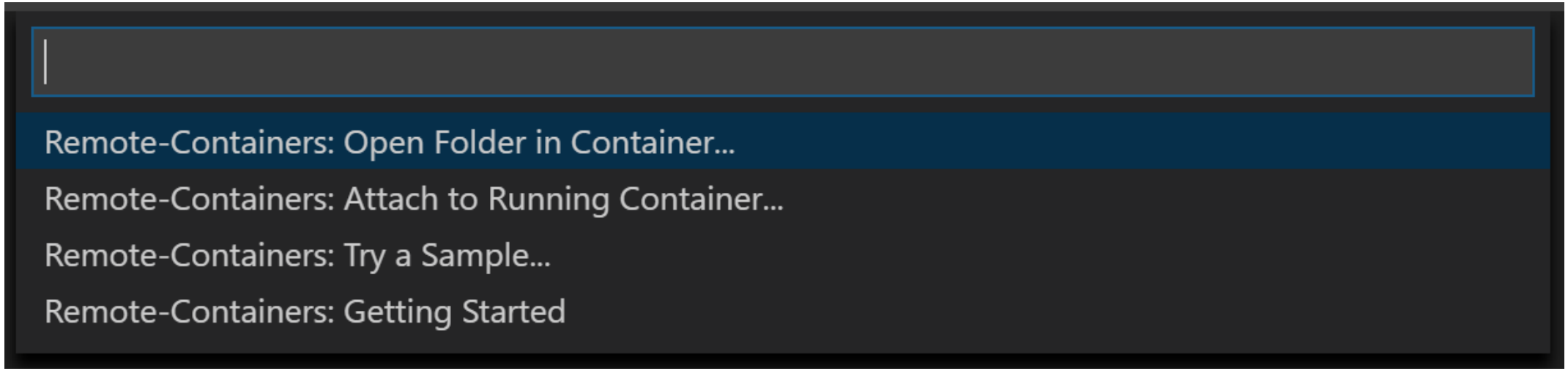
Шаблон для C++ проекта, бинарные числа

VS Code and Docker

1. Установите docker по ссылке: https://www.docker.com/get-started
2. Установите VS Code по ссылке: https://code.visualstudio.com/insiders/
3. Установите следующее расширения:
    i. https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.vscode-remote-extensionpack
    ii. https://marketplace.visualstudio.com/items?itemName=llvm-vs-code-extensions.vscode-clangd
    iii. https://marketplace.visualstudio.com/items?itemName=vadimcn.vscode-lldb

Как создать проект в контейнере:

1. Создаем папку для проекта, открываем ее в VS Code и затем Ctrl+Shift+P и выбираем следующую комманду:



2. Щелкаем Open, выбираем C++, затем Ubuntu 18.04
3. Перейдите во вкладку Extensions и убедитесь что все расширения также установлены в контейнере.

Git через ssh

1. `ssh-keygen -t ed25519 -C "korovaikon@gmail.com" # -C adds a comment`
2. `cat >> ~/.ssh/config` следующее:

```
Host github.com
      Hostname github.com
      User git
      IdentityFile ~/.ssh/id_ed25519
```

3. добавьте содержимое `cat ~/.ssh/id_ed25519.pub` в https://github.com/settings/keys
4. Сделайте чек-аут `git clone git@github.com:Krovatkin/pytorch_misc.git`

Чтобы не вводить passphrase постоянно

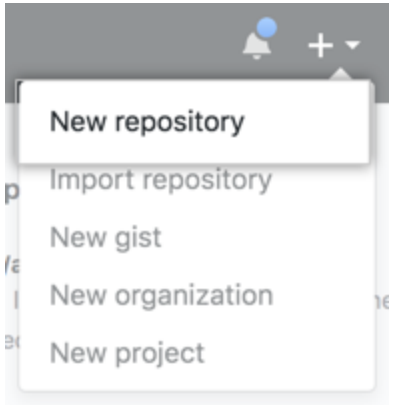
5. добавляем ключ в ssh-agent:

```
eval `ssh-agent -s`
ssh-add ~/.ssh/id_ed25519
```

Источник

Шаблон для проекта, 1:

1. Создаем новый репозиторий:



2. Проходим все шаги здесь

3. `git clone git@github.com:Krovatkin/yandex-4bit-workshop2.git`

Шаблон для проекта, 2:

4. К сожалению, git не может клонировать в папку с файлами, поэтому мы просто скопируем нужные файлы а папку удалим.

```
mv yandex-4bit-workshop2/.git yandex-4bit-workshop2/.gitignore yandex-4bit-workshop2/README .
rm -rf yandex-4bit-workshop2/
```

Шаблон для проекта, 3:

5. Создаем `CMakeLists.txt`

```
cmake_minimum_required(VERSION 3.10)
project(gtest_param)

add_subdirectory(googletest)

add_executable(my_test my_test.cpp)
set_property(TARGET my_test PROPERTY CXX_STANDARD 17)
```

Шаблон для проекта, 4:

6. Создаем `my_test.cpp`

```
#include "gtest/gtest.h"
TEST(IntegerTests, Addition) { ASSERT_EQ(2 + 2, 4); }
```

Шаблон для проекта, 5:

7. Клонируем **googletest** : `git clone https://github.com/google/googletest.git`
8. Компилируем и запускаем

```
mkdir build
cd build
cmake -DCMAKE_EXPORT_COMPILE_COMMANDS=ON -DCMAKE_BUILD_TYPE=Debug ..
cmake --build .
# run the executable
./my_test
```

Бинарные числа

- В бинарной системе счисления **(основание 2)**, цифры используются только **0 или 1**, в отличии от десятичной **(основание 10)**, где мы используем **0,1,2,3,4,5,6,7,8,9**

Разрядность

- Разрядность числа -- количество цифр

$$100_2 = 0 * 2^0 + 0 * 2^1 + 1 * 2^2 = 4_{10}$$

$$123_{10} = 3 * 10^0 + 2 * 10^1 + 1 * 10^2 = 3 * 0 + 2 * 10 + 1 * 100 = 123_{10}$$

Разрядность, 2

uint8_t

Разрядность: 8, позиции считаются с нуля

1	1	0	0	0	0	1	1
7	6	5	4	3	2	1	0

$$11000011_2 = 1 * 2^7 + 1 * 2^6 + 0 * 2^5 ... + 1 * 2^1 + 1 * 2^1 = 128 + 64 + 2 + 1 = 195$$

Разрядность и числа в C++

8 бит называют **байтом**.

```
sizeof(int32_t) == sizeof(int) == 4
```

```
Разрядность(int32_t) == sizeof(int) == 4 * 8 = 32
```

Сколько чисел может уместить 8-битное число?

2 * 2 * 2 * 2 ... итак **8 раз** = 256 или $[0, 255]$

2 умноженное на себя 8 раз, это степень 2^8

Дополнительный код (Two's complement)

или числа со знаком

4-битные положительные числа

4-битное число умещает 16 значений от $[0, 15]$ *без знака*

$0000_2 = 0_{10}$ - ноль он и в Африке НОЛЬ

$0001_2 = 1_{10}$ бит справа (самая маленькая позиция, 0)

самое большое положительное число:

$0111_2 = 7_{10}$

4-битные отрицательные числа

Бит в самой высокой позиции (то есть слева) **отведен под знак**

самое маленькое отрицательное число:

$1000_2 = -8_{10}$

$-8_{10} + -1_{10} =?$ нет не 9 😄

$$1111_2 + 0001_2 = 1\_0000_2 = 0000_2 = 0_{10}$$

Из отрицательного числа в десятичное

Начнем с отрицательного числа, $1001_2 = -7$
Инвертируем, 0110_2
И опять добавим +1 $0111_2 = 7$

7 + 1 = ?

волшебная арифметика: 7 + 1 = -8!!!!

$$0111_2 + 1_2 = 7_{10} + 1_{10} = -8_{10}$$

по другому это называется: overflow or wraparound поведение!

Простое правило:

- наименьшее отрицательное число (-8) + 1 = наименьшее положительное (0)
- наибольшее положительное число (7) + 1 = наибольшее отрицательное число (-1)

-1 самое интересное число:

$$1111_2 = -1_{10}$$

$$0_{10} - 1_{10} = 0_{10} + (-1_{10}) = 0000_2 + 1111_2 = 1111_2 = -1_{10}$$

Отрицание

как получить **-3** в 4-битном числе?

начнем с $0011_2 = 1 * 2^0 + 1 * 2^1 = 3_{10}$

инвертируем все биты в числе:

$!0011_2 = 1100_2$

добавим $+1_{10} = 0001_2$

$1100_2 + 0001_2 = 1101_2 = -3$

Еще один пример

$-(-8)_{10} = 1000_2$

Инвертируем:

$!1000_2 = 0111_2$

+1:

$0111_2 + 0001_2 = 1000_2 = -8$

Умножение НЕПРАВИЛЬНОЕ!

```
    1101
*   0010
    ----
    0000
   1101
  0000
 0000
 xxx0010
```

Умножение ПРАВИЛЬНОЕ!

```
          11111101
      *   00000010
          --------
          00000000
         11111101
        00000000
       00000000
      00000000
     00000000
+   00000000
    xxxxxx00001010
```

Расширение типов

Расширим 4-битное число до 8-бит

$$1111_2 = 15_{10}$$

добавляем 4 разряда слева

$$0000\_1111_2 = 15_{10}$$

```
uint4(15) + 1 = uint4(0) // not valid C++
uint8(8) + 1 = uint8(16) // not valid C++
```

Расширим 4-битное знаковое число до 8-бит

1. Смотрим на самый знаковый бит (самый левый)

$1111_2 = -1_{10}$ (бит равен 1)

2. Копируем этот бит в дополнительные разряды

$1111_2 -> 0000\_1111 -> 1111\_1111 = -1$

Gotcha 1, Implicit Widening

```
static_cast<uin8_t>(255) + static_cast<uint8_t>(255); // ?
```

Gotcha 1, Implicit Widening

```
static_cast<uin8_t>(255) + static_cast<uint8_t>(255) == 702;
static_cast<uint8_t>(static_cast<uin8_t>(255) + static_cast<uint8_t>(255)) == 254;
```

Gotcha 2, Implicit Conversion, одинаковый ранг

```
static_cast<int8_t>(-2) + static_cast<uint8_t>(1);
```

Gotcha 2, Implicit Conversion, одинаковый ранг

```
static_cast<int8_t>(-2) + static_cast<uint8_t>(1) == 255;
```

Gotcha 3, Implicit Conversion, разные ранги

```
static_cast<int16_t>(-2) + static_cast<uint8_t>(1) == -1;
```