

# Лабораторная работа №2

## UNIX C++ BUILD / IF / LOOP, PYTHON

Выполнил: Макаренко Александр, группа  
Z33434, 3 курс, 2024 г.

### Цель

Познакомить студента с принципами компиляции исходного кода. Составить программу с использованием циклов, условий и функций. Сравнить быстродействие между C++ и Python. Ознакомление с типами данных.

### Задачи

**1 [C++ EXPRESSION]** Создать и скомпилировать программу на C++

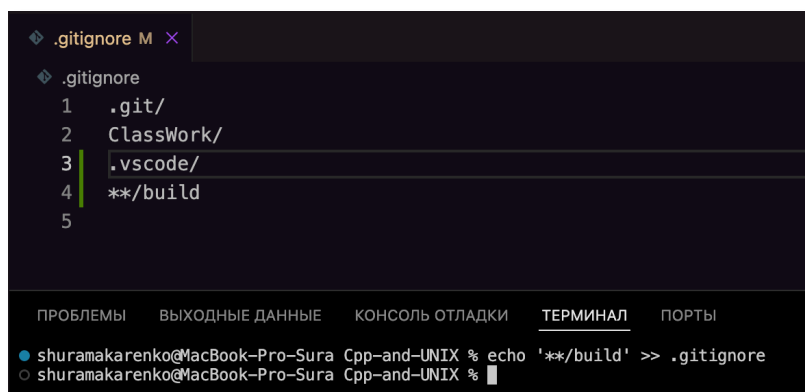
Результат сборки (компиляции) сохранять в папку `build`. Папку `build` сделать игнорируемой для GIT. Программа должна получать на вход число - это количество итераций для выполнения расчета. В рамках итерации выполнять следующее вычисление:

$$x^2 - x^2 + x^4 - x^5 + x + x.$$

Вычисление выполнять в виде отдельной от `main` функции, которая будет вызвана циклически из `main`. Фиксировать время выполнения программы, затрачиваемое на расчет выражения `n` раз (`n` задается в консоли перед вычислением). Предусмотреть дополнительный цикл на повторную итерацию запуска программы вычислений. Если было введено не число, то завершить выполнение программы.

### Решение

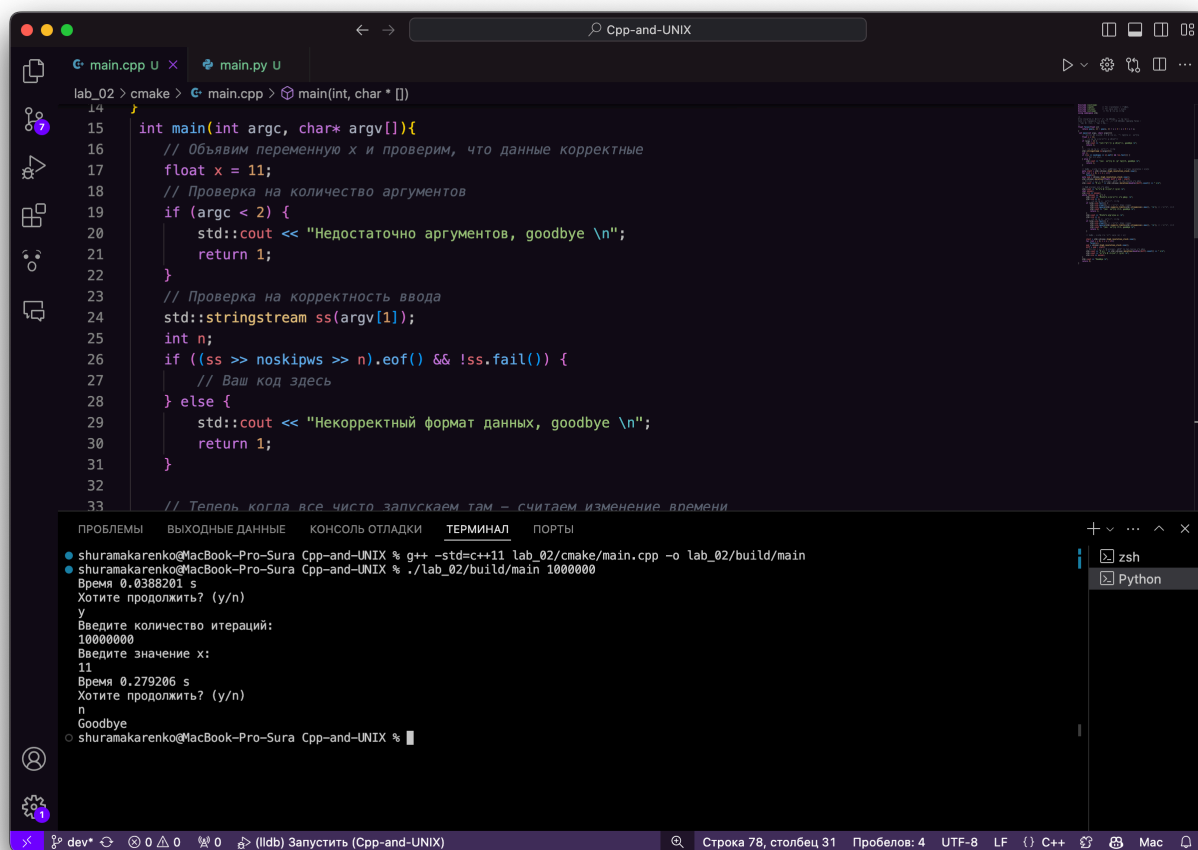
Для начала добавим все `build` папки в `.gitignore` и создадим скрипт для отката к текущей ревизии, отменяющим как изменения так и новые файлы.



```
.gitignore M X
.gitignore
1 .git/
2 ClassWork/
3 .vscode/
4 **/build
5

ПРОБЛЕМЫ Выходные данные КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX % echo '**/build' >> .gitignore
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX %
```

Теперь напишем код, сначала скомпилируем его и сохраним в папку build.



The screenshot shows a C++ IDE with a file named `main.cpp` open. The code is a C++ program that checks the number of command-line arguments and the format of the input. It uses `std::stringstream` to parse the input. The terminal window at the bottom shows the compilation and execution of the program. The compilation command is `g++ -std=c++11 lab_02/cmake/main.cpp -o lab_02/build/main`. The execution output shows the program running with 10,000,000 iterations and a value of 11, followed by a "Goodbye" message.

```
lab_02 > cmake > main.cpp > main(int, char * [])
14
15 int main(int argc, char* argv[]){
16     // Объявим переменную x и проверим, что данные корректные
17     float x = 11;
18     // Проверка на количество аргументов
19     if (argc < 2) {
20         std::cout << "Недостаточно аргументов, goodbye \n";
21         return 1;
22     }
23     // Проверка на корректность ввода
24     std::stringstream ss(argv[1]);
25     int n;
26     if ((ss >> noskipws >> n).eof() && !ss.fail()) {
27         // Ваш код здесь
28     } else {
29         std::cout << "Некорректный формат данных, goodbye \n";
30         return 1;
31     }
32
33     // Теперь, когда все чисто запускаем там — считаем изменение времени
```

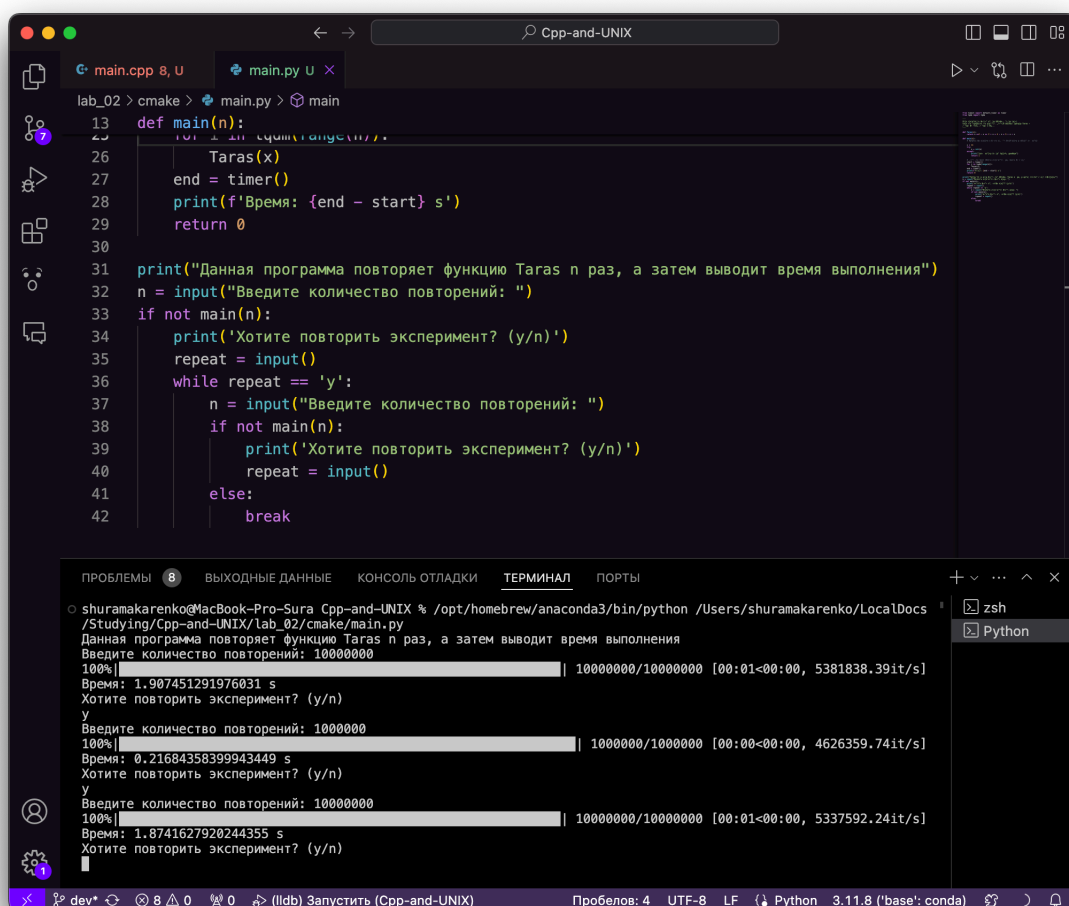
ПРОБЛЕМЫ Выходные данные КОНСОЛЬ ОТЛАДКИ **ТЕРМИНАЛ** ПОРТЫ

```
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX % g++ -std=c++11 lab_02/cmake/main.cpp -o lab_02/build/main
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX % ./lab_02/build/main 10000000
Время 0.0388201 s
Хотите продолжить? (y/n)
y
Введите количество итераций:
10000000
Введите значение x:
11
Время 0.279206 s
Хотите продолжить? (y/n)
n
Goodbye
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX %
```

Строка 78, столбец 31 Пробелов: 4 UTF-8 LF {} C++ Mac

## 2. [PYTHON EXPRESSION] Создать и скомпилировать программу на Python 3

Результат сборки (компиляции) сохранять в папку build. Папку build сделать игнорируемой для GIT. Программа должна получать на вход число – это количество итераций для выполнения расчета. В рамках итерации выполнять следующее вычисление:  $x^2 - x^2 + x^4 - x^5 + x + x$ . Вычисление выполнять в виде отдельной от main функции, которая будет вызвана циклически из main. Фиксировать время выполнения программы, затрачиваемое на расчет выражения n раз (n задается в консоли перед вычислением). Предусмотреть дополнительный цикл на повторную итерацию запуска программы вычислений. Если было введено не число, то завершить выполнение программы.



The screenshot shows a code editor with a Python script and its terminal output. The script defines a function `Taras(x)` and a `main` function that takes an input `n` and repeats the calculation `n` times. The terminal output shows the program running with `n=1000000` and displaying the execution time and iteration count for each run.

```
13 def main(n):
14     for i in range(1, n+1):
15         Taras(x)
16     end = timer()
17     print(f'Время: {end - start} s')
18     return 0
19
20 print("Данная программа повторяет функцию Taras n раз, а затем выводит время выполнения")
21 n = input("Введите количество повторений: ")
22 if not main(n):
23     print('Хотите повторить эксперимент? (y/n)')
24     repeat = input()
25     while repeat == 'y':
26         n = input("Введите количество повторений: ")
27         if not main(n):
28             print('Хотите повторить эксперимент? (y/n)')
29             repeat = input()
30         else:
31             break
```

Terminal Output:

```
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX % /opt/homebrew/anaconda3/bin/python /Users/shuramakarenko/LocalDocs/
/Studying/Cpp-and-UNIX/lab_02/cmake/main.py
Данная программа повторяет функцию Taras n раз, а затем выводит время выполнения
Введите количество повторений: 1000000
100% | 1000000/1000000 [00:01:00:00, 5381838.39it/s]
Время: 1.907451291976031 s
Хотите повторить эксперимент? (y/n)
y
Введите количество повторений: 1000000
100% | 1000000/1000000 [00:00:00:00, 4626359.74it/s]
Время: 0.21684358399943449 s
Хотите повторить эксперимент? (y/n)
y
Введите количество повторений: 1000000
100% | 1000000/1000000 [00:01:00:00, 5337592.24it/s]
Время: 1.8741627920244355 s
Хотите повторить эксперимент? (y/n)
```

### 3. [SAVE] Результат всех вышеперечисленных шагов сохранить в репозиторий

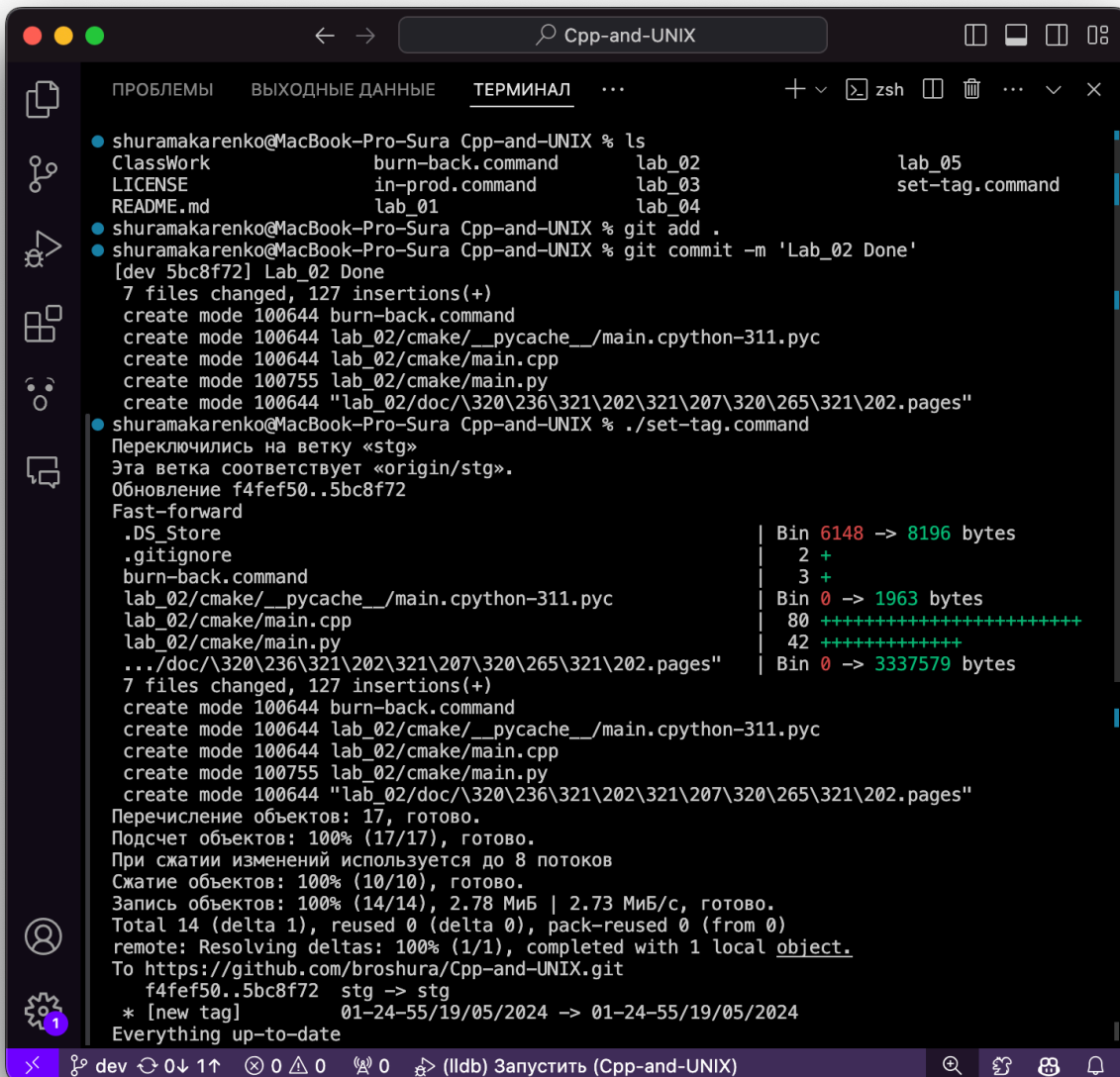
(+ отчет по данной ЛР в папку doc)

Фиксацию ревизий производить строго через ветку dev. С помощью скриптов

накатить ревизии на stg и на prd. Скрипты разместить в корне репозитория. Также

создать скрипты по возврату к виду текущей ревизии (даже если в папке имеются

несохраненные изменения + новые файлы).



```
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX % ls
ClassWork      burn-back.command  lab_02          lab_05
LICENSE        in-prod.command    lab_03          set-tag.command
README.md      lab_01             lab_04

shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX % git add .
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX % git commit -m 'Lab_02 Done'
[dev 5bc8f72] Lab_02 Done
7 files changed, 127 insertions(+)
create mode 100644 burn-back.command
create mode 100644 lab_02/cmake/___pycache___/main.cpython-311.pyc
create mode 100644 lab_02/cmake/main.cpp
create mode 100755 lab_02/cmake/main.py
create mode 100644 "lab_02/doc/\320\236\321\202\321\207\320\265\321\202.pages"

shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX % ./set-tag.command
Переключились на ветку «stg»
Эта ветка соответствует «origin/stg».
Обновление f4fef50..5bc8f72
Fast-forward
 .DS_Store                      | Bin 6148 -> 8196 bytes
 .gitignore                    | 2 +
 burn-back.command              | 3 +
 lab_02/cmake/___pycache___/main.cpython-311.pyc | Bin 0 -> 1963 bytes
 lab_02/cmake/main.cpp          | 80 ++++++
 lab_02/cmake/main.py           | 42 ++++++
 .../doc/\320\236\321\202\321\207\320\265\321\202.pages" | Bin 0 -> 3337579 bytes
7 files changed, 127 insertions(+)
create mode 100644 burn-back.command
create mode 100644 lab_02/cmake/___pycache___/main.cpython-311.pyc
create mode 100644 lab_02/cmake/main.cpp
create mode 100755 lab_02/cmake/main.py
create mode 100644 "lab_02/doc/\320\236\321\202\321\207\320\265\321\202.pages"
Перечисление объектов: 17, готово.
Подсчет объектов: 100% (17/17), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (10/10), готово.
Запись объектов: 100% (14/14), 2.78 Миб | 2.73 Миб/с, готово.
Total 14 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/broshura/Cpp-and-UNIX.git
 f4fef50..5bc8f72 stg -> stg
 * [new tag]      01-24-55/19/05/2024 -> 01-24-55/19/05/2024
Everything up-to-date
```

PROБЛЕМЫ Выходные данные ТЕРМИНАЛ ... + - zsh

```
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX % ./in-prod.command
Переключились на ветку «prd»
Эта ветка соответствует «origin/prd».
Обновление f4fef50..5bc8f72
Fast-forward
 .DS_Store | Bin 6148 -> 8196 bytes
 .gitignore | 2 +
 burn-back.command | 3 +
 lab_02/cmake/___pycache___/main.cpython-311.pyc | Bin 0 -> 1963 bytes
 lab_02/cmake/main.cpp | 80 ++++++
 lab_02/cmake/main.py | 42 ++++++
 .../doc/\320\236\321\202\321\207\320\265\321\202.pages" | Bin 0 -> 3337579 bytes
7 files changed, 127 insertions(+)
create mode 100644 burn-back.command
create mode 100644 lab_02/cmake/___pycache___/main.cpython-311.pyc
create mode 100644 lab_02/cmake/main.cpp
create mode 100755 lab_02/cmake/main.py
create mode 100644 "lab_02/doc/\320\236\321\202\321\207\320\265\321\202.pages"
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/broshura/Cpp-and-UNIX.git
 f4fef50..5bc8f72 prd -> prd
Переключились на ветку «dev»
Ваша ветка опережает «origin/dev» на 1 коммит.
(используйте «git push», чтобы опубликовать ваши локальные коммиты)
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX %
```

dev 0 ↓ 1 ↑ 0 0 0 (lldb) Запустить (Cpp-and-UNIX)

\$ time\_reverse.command x

```
1 #!/bin/bash
2 git stash -u
3
```

PROБЛЕМЫ Выходные данные ТЕРМИНАЛ ... + - zsh

```
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX % chmod +x time_reverse.command
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX % ./time_reverse.command
Рабочий каталог и состояние индекса сохранены WIP on dev: 5bc8f72 Lab_02 Done
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX %
```

dev 0 ↓ 1 ↑ 0 0 0 (lldb) Запустить (Cpp-and-UNIX) Shell Script

The screenshot shows a VS Code editor window with a terminal open. The terminal output shows the execution of `time_reverse.command` and `git stash -u`. The output indicates that the current branch is `dev` and it is ahead of `origin/dev` by 1 commit. It also shows the files that are not tracked by git, including `time_reverse.command`. The terminal output is as follows:

```
$ time_reverse.command U
$ time_reverse.command
1 #!/bin/bash
2 git stash -u
3

shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX % chmod +x time_reverse.command
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX % ./time_reverse.command
Рабочий каталог и состояние индекса сохранены WIP on dev: 5bc8f72 Lab_02 Done
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX % git stash pop
Текущая ветка: dev
Ваша ветка опережает «origin/dev» на 1 коммит.
(используйте «git push», чтобы опубликовать ваши локальные коммиты)

Изменения, которые не в индексе для коммита:
(используйте «git add/rm <файл>...», чтобы добавить или удалить файл из индекса)
(используйте «git restore <файл>...», чтобы отменить изменения в рабочем каталоге)
удалено:      burn-back.command
изменено:     "lab_02/doc/\320\236\321\202\321\207\320\265\321\202.pages"

Неотслеживаемые файлы:
(используйте «git add <файл>...», чтобы добавить в то, что будет включено в коммит)
time_reverse.command

индекс пуст (используйте «git add» и/или «git commit -a»)
Отброшено refs/stash@{0}: (f4633785f50f38949118585890683a7278c307ac)
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX %
```

## Выводы

1. Получили, что C++ справляется с поставленной задачей примерно в 7 раз лучше.
2. Заполняемая анимирующая полоска (tqdm) замедляется питон примерно на 30%, без него результаты отличались в 5 раз.
3. Можно использовать jit-компиляцию для ускорения участков кода - таким образом удалось достичь отношения в 3 раза.