

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ  
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «Обчислювальної техніки та програмування»

Звіт з лабораторної роботи №3  
Тема: «Розробка лінійних програм»

Виконав:

ст. гр. КІТ-120в Львов Артем

Перевірив:

Бульба С.С.

Харків 2020

## **Лабораторна робота №3. Розробка лінійних програм.**

### **Розробник:**

Львов Артем Сергійович

Студент групи КІТ-120В

07.12.2020

### **Загальне завдання:**

Розробка лінійних програм

### **Обчислення варіанту індивідуального завдання:**

За формулою  $N_t = ((N_j - 1) \% C) + 1$ :

$$N_t = ((12 - 1) \% 4) + 1 = 4;$$

### **Індивідуальне завдання:**

Дана сума грошей в гривнях. Перевести гривні в долари, євро, російські рублі. Курси валют (долари, євро, російські рублі) задати у вигляді констант.

### **Функціональне призначення:**

Програмі задається певна сума грошей в гривнях, після завершення роботи в змінних `usd`, `eur`, `rub` можна буде побачити суму відповідно у американських доларах, євро та російських рублях.

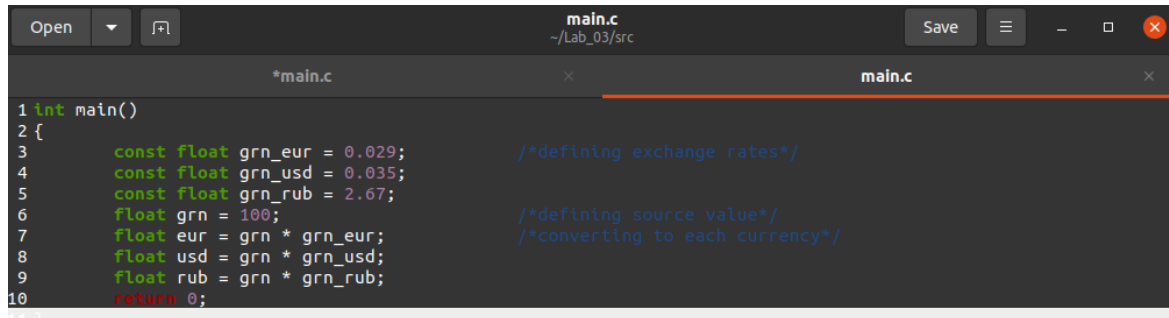
### **Виконання роботи:**

#### **1. Створення репозиторію:**

Створюю новий репозиторій `Lab_03`, імпортую туди репозиторій Давидова В.В та клоную до локального репозиторію командою `git clone`.

## 2.Написання коду:

Пишу код програми у файлі Lab\_03/src/main.c

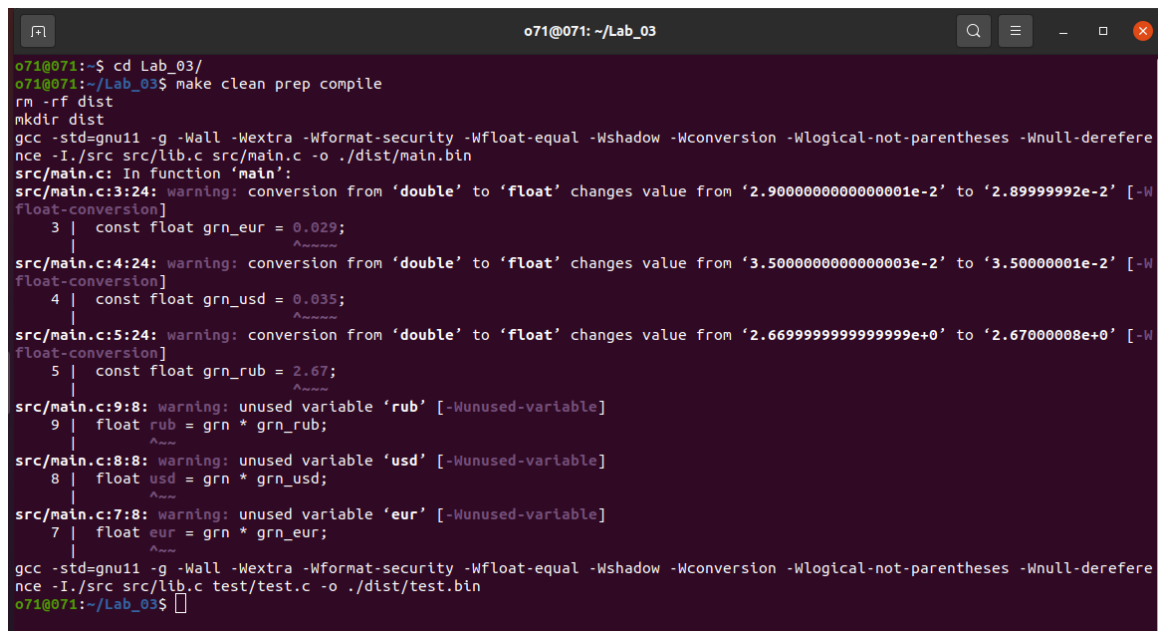


```
1 int main()
2 {
3     const float grn_eur = 0.029;           /*defining exchange rates*/
4     const float grn_usd = 0.035;
5     const float grn_rub = 2.67;
6     float grn = 100;                       /*defining source value*/
7     float eur = grn * grn_eur;             /*converting to each currency*/
8     float usd = grn * grn_usd;
9     float rub = grn * grn_rub;
10    return 0;
```

Рисунок 2 – код програми

### 2.1. Компіляція проекту:

Компілюю проект за допомогою команди make clean prep compile:



```
o71@o71: ~/Lab_03
o71@o71:~$ cd Lab_03/
o71@o71:~/Lab_03$ make clean prep compile
rm -rf dist
mkdir dist
gcc -std=gnu11 -g -Wall -Wextra -Wformat-security -Wfloat-equal -Wshadow -Wconversion -Wlogical-not-parentheses -Wnull-dereference -I./src src/lib.c src/main.c -o ./dist/main.bin
src/main.c: In function 'main':
src/main.c:3:24: warning: conversion from 'double' to 'float' changes value from '2.9000000000000001e-2' to '2.89999992e-2' [-Wfloat-conversion]
    3 |     const float grn_eur = 0.029;
      |                      ~~~~~
src/main.c:4:24: warning: conversion from 'double' to 'float' changes value from '3.5000000000000003e-2' to '3.50000001e-2' [-Wfloat-conversion]
    4 |     const float grn_usd = 0.035;
      |                      ~~~~~
src/main.c:5:24: warning: conversion from 'double' to 'float' changes value from '2.6699999999999999e+0' to '2.67000008e+0' [-Wfloat-conversion]
    5 |     const float grn_rub = 2.67;
      |                      ~~~~~
src/main.c:9:8: warning: unused variable 'rub' [-Wunused-variable]
    9 |     float rub = grn * grn_rub;
      |         ~~~~~
src/main.c:8:8: warning: unused variable 'usd' [-Wunused-variable]
    8 |     float usd = grn * grn_usd;
      |         ~~~~~
src/main.c:7:8: warning: unused variable 'eur' [-Wunused-variable]
    7 |     float eur = grn * grn_eur;
      |         ~~~~~
gcc -std=gnu11 -g -Wall -Wextra -Wformat-security -Wfloat-equal -Wshadow -Wconversion -Wlogical-not-parentheses -Wnull-dereference -I./src src/lib.c test/test.c -o ./dist/test.bin
o71@o71:~/Lab_03$
```

Рисунок 2.1 – компіляція проекту

## 2.2. Перевірка за допомогою nemiver:

Перевіряю роботу проекту командою `nemiver ./dist/main.bin`

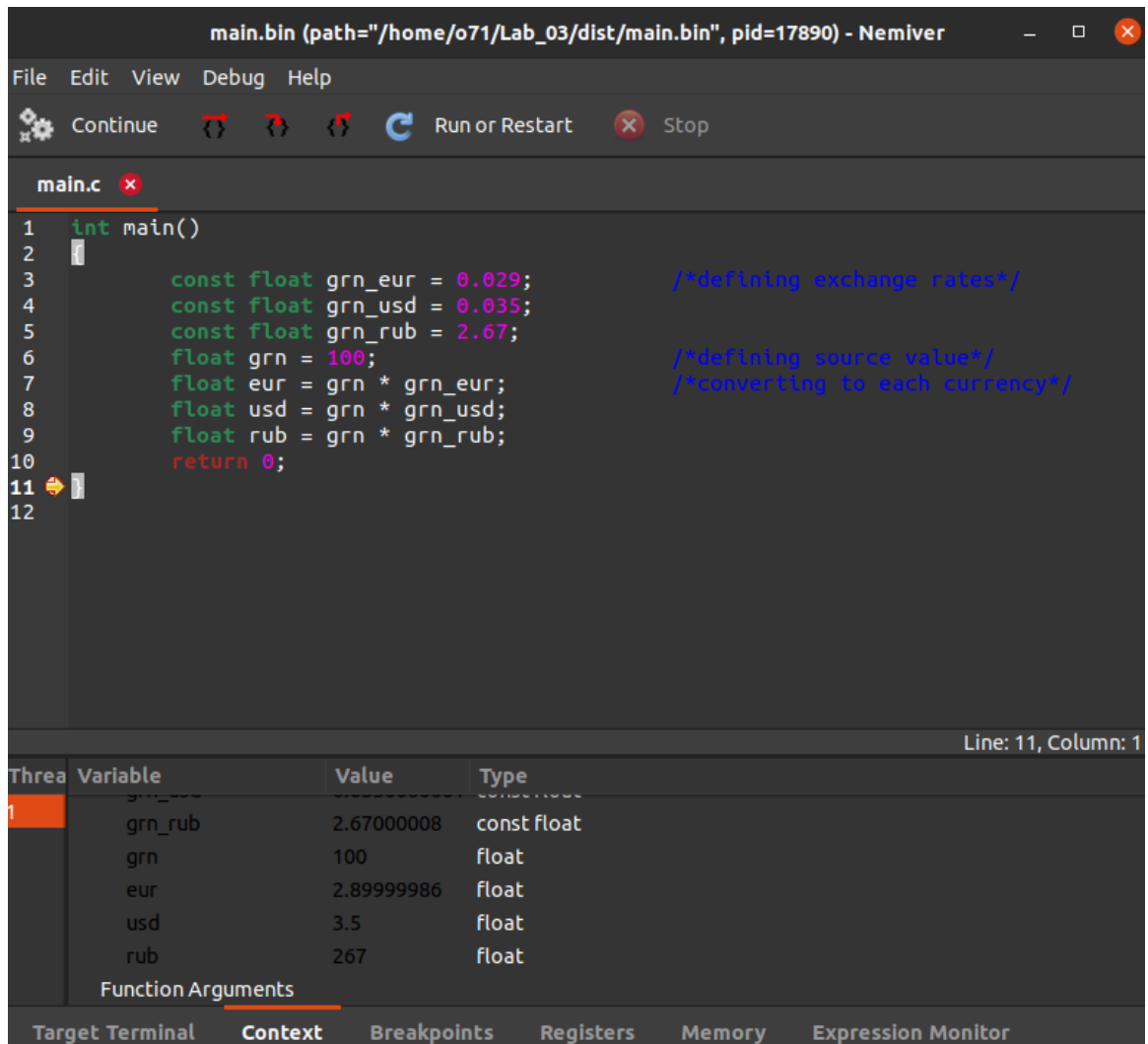
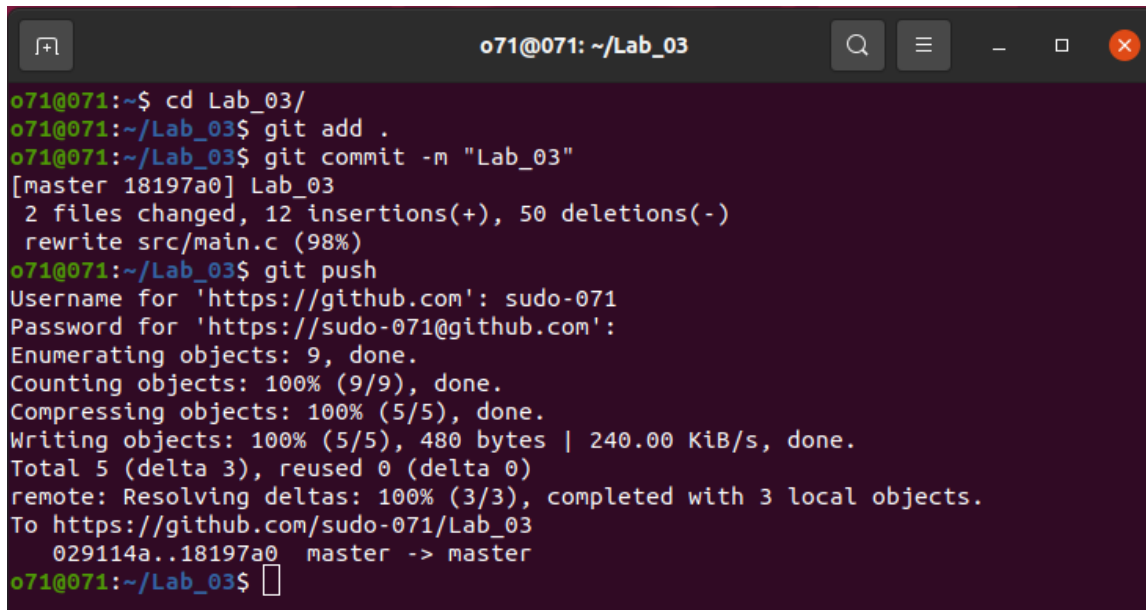


Рисунок 2.2 – вікно nemiver

### 3. Завантаження проекту на GitHub:

A terminal window with a dark background and light-colored text. The window title is 'o71@071: ~/Lab\_03'. The terminal shows a series of commands and their outputs. The commands are: 'cd Lab\_03/', 'git add .', 'git commit -m "Lab\_03"', and 'git push'. The outputs include commit details like '2 files changed, 12 insertions(+), 50 deletions(-)' and 'rewrite src/main.c (98%)', followed by the push progress: 'Enumerating objects: 9, done.', 'Counting objects: 100% (9/9), done.', 'Compressing objects: 100% (5/5), done.', 'Writing objects: 100% (5/5), 480 bytes | 240.00 KiB/s, done.', 'Total 5 (delta 3), reused 0 (delta 0)', 'remote: Resolving deltas: 100% (3/3), completed with 3 local objects.', and the final push result: 'To https://github.com/sudo-071/Lab\_03 029114a..18197a0 master -> master'.

```
o71@071:~$ cd Lab_03/
o71@071:~/Lab_03$ git add .
o71@071:~/Lab_03$ git commit -m "Lab_03"
[master 18197a0] Lab_03
 2 files changed, 12 insertions(+), 50 deletions(-)
 rewrite src/main.c (98%)
o71@071:~/Lab_03$ git push
Username for 'https://github.com': sudo-071
Password for 'https://sudo-071@github.com':
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 480 bytes | 240.00 KiB/s, done.
Total 5 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/sudo-071/Lab_03
 029114a..18197a0 master -> master
o71@071:~/Lab_03$
```

Рисунок 3 – Виконання команд git add, git commit, git push

### 4.Висновки:

При виконанні лабораторної роботи навчився писати прості лінійні програми. Написав програму для швидкої конвертації гривні в американські долари, євро та російські рублі

## **Відповіді на питання:**

1. Типова програма мовою C складається з наступних розділів:

1.1 – модулі (Підключаються через команду `#include`, можна підключати як вбудовані бібліотеки, так і власні створені файли з функціями як частини багатофайлової структури проекту)

(Всі подальші пункти виконуються в рамках основної функції `main`)

1.2 – Об'явлення глобальних змінних, констант і масивів

1.3 – Об'явлення функцій

2. Лінійним називають найпростіший алгоритм, дії в якому виконуються одна за одною

3.

3.1 `a += 1`

3.2 `a ++`

3.3 `a = a + 1`

3.4 `++a`

4. Проект – певна файлова структура що містить файли, необхідні для виконання програми та документацію

5. Скомпілювати програму

6. В залежності від компілятора, компіляція проекту виконується різними командами (Наприклад `gcc -o "файл програми"` при використанні `gcc` або `make` при використанні `make`)

7. `git add "назва файлу"` – `git commit` – `git push`

8. При виявленні помилки за допомогою відладника, виконати зміни в файлі у текстовому редакторі, та скомпілювати його знову, після виправлення. В цьому і полягає процес дабагінгу

9. Після компіляції, програму можна запустити командою в терміналі `./"Назва програми"`

10.Робота у віконних програмах відбувається в основному за допомогою графічного інтерфейсу, і певні функції програми запускаються при натисненні кнопки, в свою чергу функції в консольних програмах запускаються при введенні певної команди в термінал. Віконні програми простіші у використанні і більш дружні і зручні для користувача, але консольні дають більший контроль над тим, що саме робить програма і до чого звертається

11.Як очевидно з назви, змінні можуть змінювати своє значення в процесі виконання програми, в свою чергу константи мають сталі значення

12.При постфіксному записі операції інкременту (наприклад `res = src ++`) змінна `res` спочатку прийме значення змінної `src`, після чого збільшиться на одиницю

При префіксному ж записі тієї самої (`res = ++ src`), спочатку значення `src` збільшиться на 1, і тільки після цього запишеться до змінної `res`