Лабораторна робота №7. Функції

Індивідульне завдання на оцінку “добре” номер 2.

1. Вимоги

1) Розробник

- Князькін Владислав Ігорович
- студент групи КІТ-320
  07.12.20

2) Загальне завдання

Переробити програми, що були розроблені під час виконання лабораторних робіт з тем “Масиви” та “Цикли” таким чином, щоб використовувалися функції для обчислення результату. Функції повинні задовільняти їхню причетність — уникати дублювання коду. Тому, для демонстрації роботи, ваша програма(функція *main()*) повинна викликати декілька раз розроблену функцію з різними вхідними даними.

Слід звернути увагу: параметри одного з викликів функції повинні бути згенерованя за допомогою генератора псевдовипадкових чисел rand().

3**) Індивідуальне завдання**

У масиві цілих чисел визначити кількість елементів, що більше попереднього та наступного.

2. Хід роботи

1. Зробимо у раніше створеному репозиторії нову під-директорію *lab07.*
   valadon@valadon-VirtualBox:~ PROGRAMMING-KNIAZKIN/naxxramass/PROGRAMMING-KNIAZKIN/naxxramass/PROGRAMMING-KNIAZKIN $ mkdir lab07
2. Створимо папку для нашого завдання *task01_6,* у якому створимо папку *src* та відкриємо і запустимо у ньому *nano.*
   valadon@valadon-VirtualBox:~ PROGRAMMING-KNIAZKIN/naxxramass/PROGRAMMING-KNIAZKIN/naxxramass/PROGRAMMING-KNIAZKIN/lab07/ $ mkdir task01_6
   valadon@valadon-VirtualBox:~ PROGRAMMING-KNIAZKIN/naxxramass/PROGRAMMING-KNIAZKIN/naxxramass/PROGRAMMING-KNIAZKIN/lab07/task01_6 $ mkdir src
   valadon@valadon-VirtualBox:~ PROGRAMMING-KNIAZKIN/naxxramass/PROGRAMMING-KNIAZKIN/naxxramass/PROGRAMMING-KNIAZKIN/lab07/task01_6/src $ nano
3. Записуємо наш код програми, дотримуючись всіх відступів та закриття дужок.

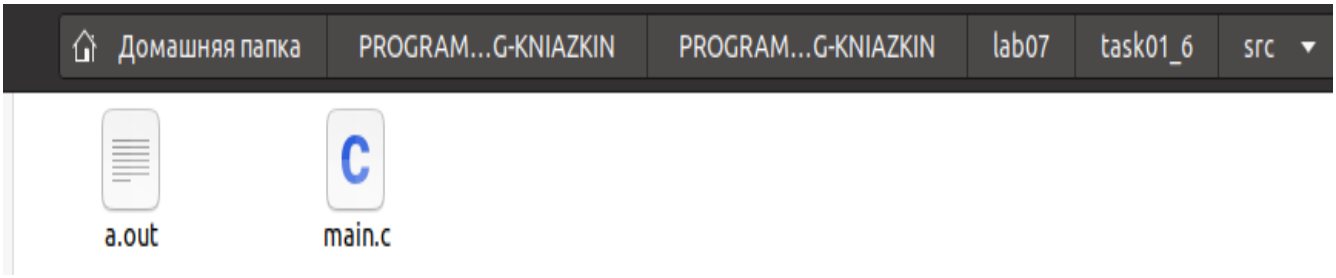
   Запишемо нашу програму:

```
#include <stdlib.h>    //* підключаємо бібліотеки stdlib.h та time.h
#include <time.h>
int get_count_prev(int b, int c);        //* додаємо функції get_count_prev та get_count_next
int get_count_next(int b, int c);
```

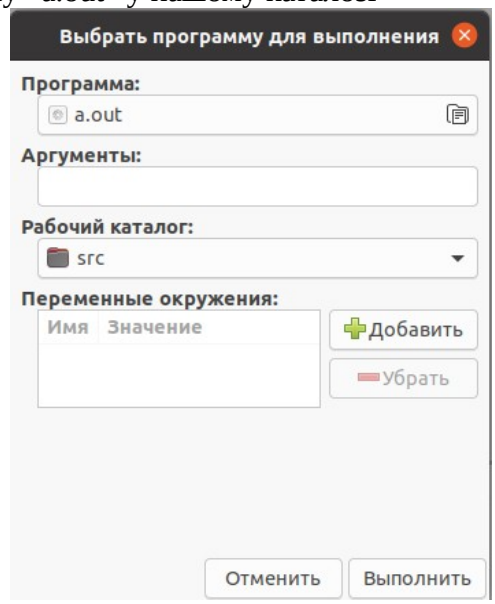
```
int main() {     //* створимо функцію main()
srand(time(0));          //*  завдяки time.h, використовуємо цей прийом, який при різних запусків rand() буде встановлювати різні значення
int x = 0 +  rand() % 3;          //* змінні координатів елемента масиву
int y = 0 +  rand() % 3;
int values[4][4] = { -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7};          //* створений власноруч массив
int number = values[x][y];     //* змінна елемента масиву
int prev = number - 1;//* змінні попереднього та наступного чисел вище вказаного елемента масиву
int next = number + 1;
int num_max = values[3][3]; //* максимальний елемент масиву по порядку
int num_min = values[0][0];  //* мінімальний елемент масиву по порядку
int res_prev_count = get_count_prev(prev, num_min);       //* змінні результатів кількості попередніх та наступних чисел
int res_next_count = get_count_next(next, num_max);
}

int get_count_prev(int b, int c){        //* функція для обчислення кількості попередніх чисел
        int a = c - b;    //* формула обчислення вище вказанної операції
        if(a < 0){
        a = -1 * a;
        }else{
        a = a * 1;
        }
        return a;         //* повертаємо значення a
}

int get_count_next(int b, int c){        //* функція для обчислення кількості наступних чисел
        int a = c - b;    //* формула обчислення вище вказанної операції
        if(a < 0){
        a = -1 * a;
        }else{
        a = a * 1;
        }
        return a;         //* повертаємо значення a
}
```

Збережемо наш код та підпишемо його *main.c.*

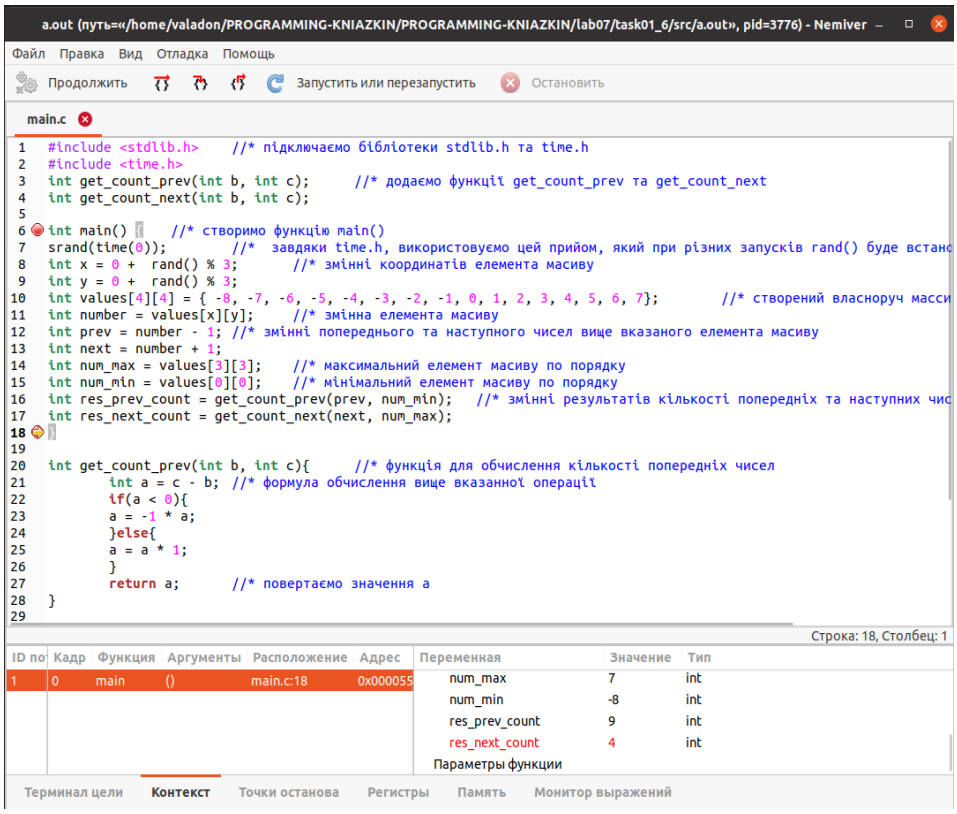
4. Повертаємось до терміналу. Завдяки команді (у каталозі src):
   gcc -g main.c //* Зробимо виконувальні файли для наших *main.c*
   Вони автоматично назвуться a.out



5. Запускаємо відлагодник (nemiver), щоб перевірити, як працює наша програма. Щоб запустити програму, треба запустити її через виконувальний файл:
   1. Відкриваємо “Файл”
   2. Знаходимо фунцію “Завантажити виконувальний файл...”
   3. Вибираємо програму “a.out” у нашому каталозі

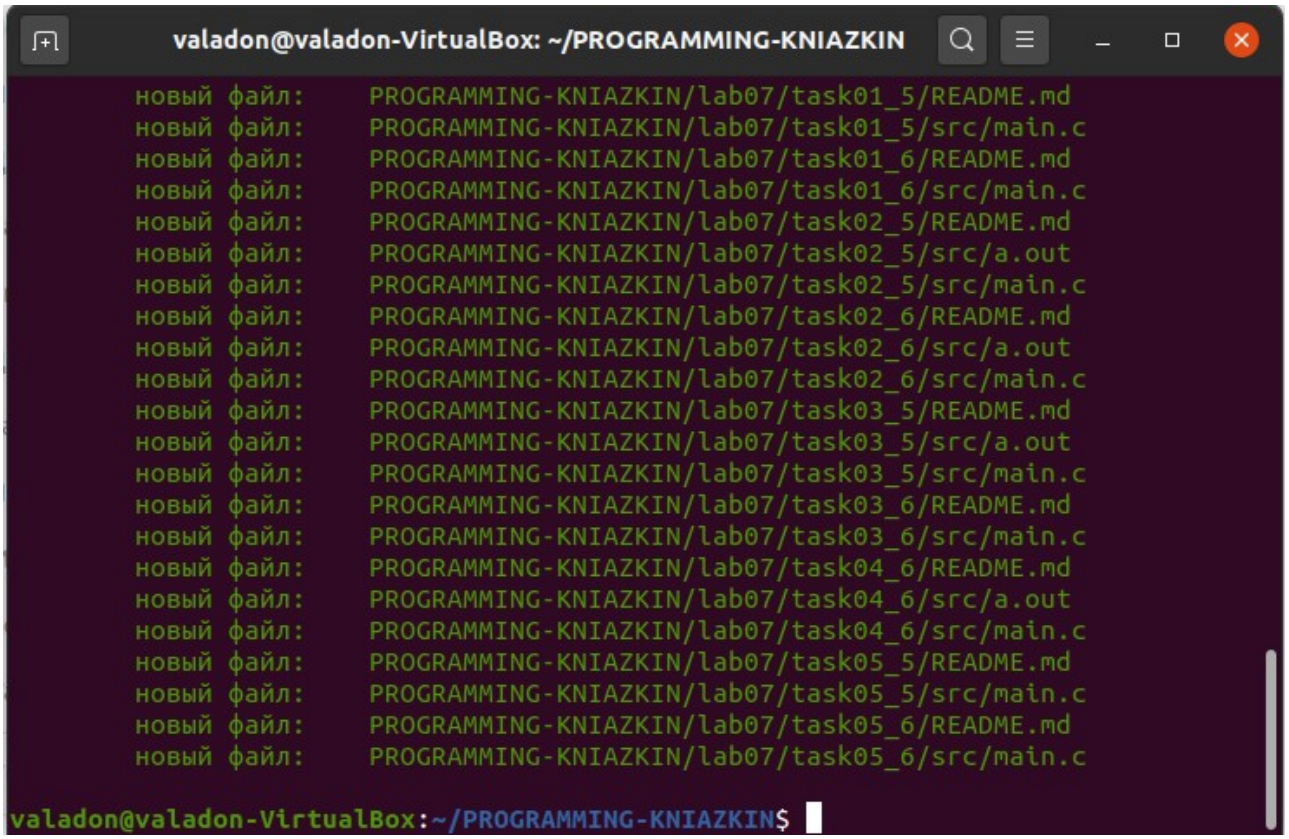


   4. Натискаємо “Виконати”.
   Тепер можна побачити програму, яку ми написали раніше:
   Завдяки функції “Step over”, переводимо нашу “стрілку” до *останнього рядка* ,щоб перевірити результат команди.



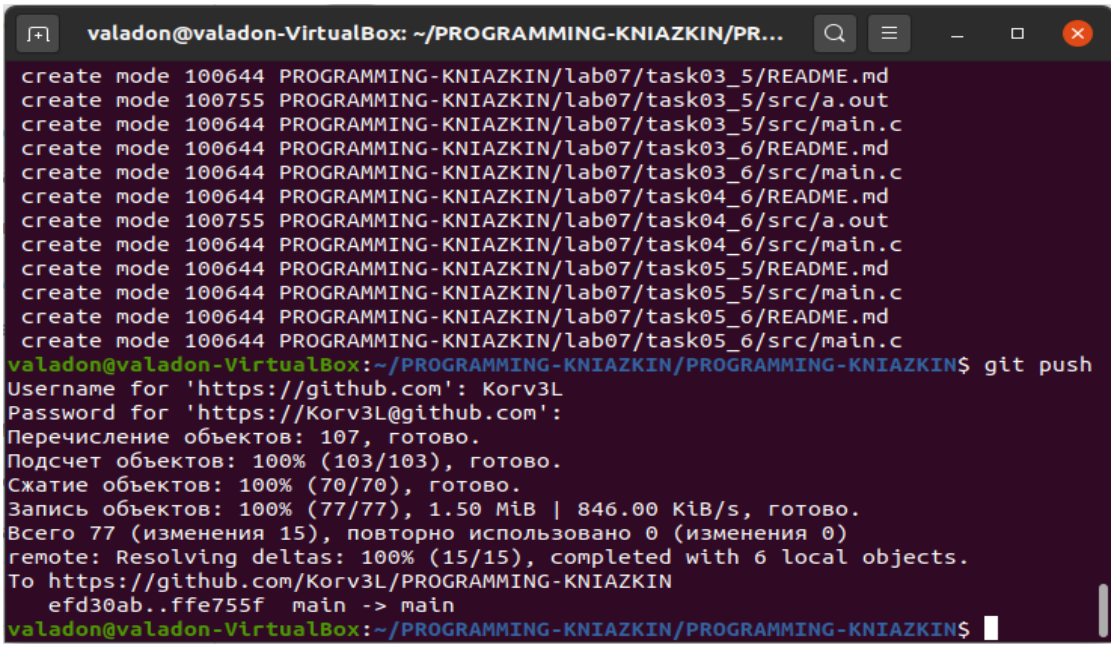
6. Бачимо, що результат вірний. Видалимо a.out. Збережемо зміни у нашій директорії на *github* через команди *git*:

valadon@valadon-VirtualBox:~ PROGRAMMING-KNIAZKIN/naxxramass $ git add .
valadon@valadon-VirtualBox:~ PROGRAMMING-KNIAZKIN/naxxramass $ git status



valadon@valadon-VirtualBox:~ PROGRAMMING-KNIAZKIN/naxxramass $ git commit -m “Create lab07”

valadon@valadon-VirtualBox:~ PROGRAMMING-KNIAZKIN/naxxramass $ git push



3. **Способи використання**
   Щодо функцій, вони допомагають скоротити код та уникнути його повторень. Щодо програми, завдяки їй можна визначити наступне та попереднє числа заданого цілого числа, а також кількість елементів що більше та менше заданого. На жаль, я не можу представити , де можна використовувати аналоги цієї програми у реальному світі.
4. **Висновок**
   Завдяки цій лабораторній роботі, я навчився переробляти програми таким чином, щоб використовувалися функції для обчислення результату, на язику C, у системі Linux, додавати до них коментарі, перевіряти програму на дієздатність у відлагоднику (nemiver), та відправляти зміни у свій репозиторій на github.