

# Лабораторна робота №7. Функції

## Індивідуальне 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 згенеровані 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
масив
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
}

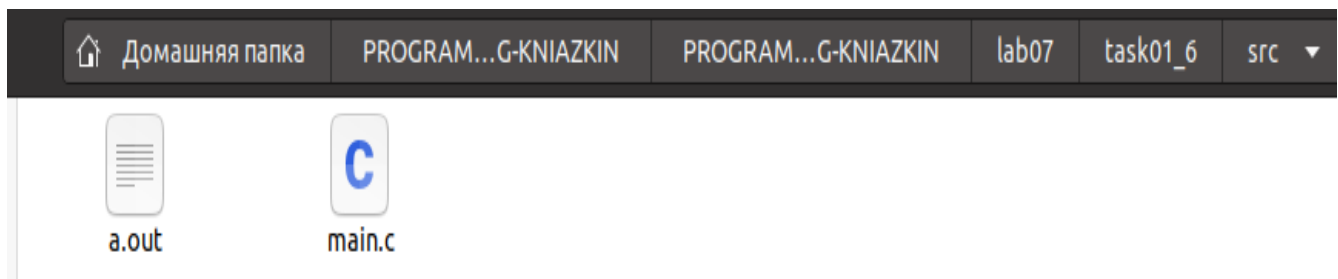
int get_count_next(int b, int c){    /* функція для обчислення кількості наступних чисел
    int a = c - b;    /* формула обчислення вище вказанної операції
    if(a < 0){
        a = -1 * a;
    }else{
        a = a * 1;
    }
    return a;    /* повертаємо значення a
}

```

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

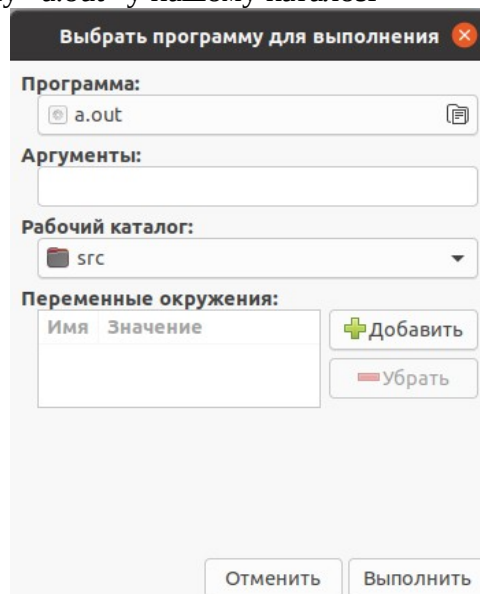
#### 4. Повертаємось до терміналу. Завдяки команді (у каталозі *src*):

```
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 функцію “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.

a.out (путь=«/home/valadon/PROGRAMMING-KNIAZKIN/PROGRAMMING-KNIAZKIN/lab07/task01\_6/src/a.out», pid=3776) - Nemiver

Файл Правка Вид Отладка Помощь

Продолжить [иконка] [иконка] [иконка] [иконка] Запустить или перезапустить [иконка] Остановить [иконка]

main.c

```

1 #include <stdlib.h>    /* підключаємо бібліотеки stdlib.h та time.h
2 #include <time.h>
3 int get_count_prev(int b, int c);    /* додаємо функції get_count_prev та get_count_next
4 int get_count_next(int b, int c);
5
6 int main() {           /* створимо функцію main()
7     srand(time(0));    /* завдяки time.h, використовуємо цей прийом, який при різних запусках rand() буде встановлювати різні значення
8     int x = 0 + rand() % 3;    /* змінні координатів елемента масиву
9     int y = 0 + rand() % 3;
10    int values[4][4] = { -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7 };    /* створений власноруч масив
11    int number = values[x][y];    /* змінна елемента масиву
12    int prev = number - 1;    /* змінні попереднього та наступного чисел вище вказаного елемента масиву
13    int next = number + 1;
14    int num_max = values[3][3];    /* максимальний елемент масиву по порядку
15    int num_min = values[0][0];    /* мінімальний елемент масиву по порядку
16    int res_prev_count = get_count_prev(prev, num_min);    /* змінні результатів кількості попередніх та наступних чисел
17    int res_next_count = get_count_next(next, num_max);
18 }
19
20 int get_count_prev(int b, int c) {    /* функція для обчислення кількості попередніх чисел
21     int a = c - b;    /* формула обчислення вище вказаної операції
22     if(a < 0) {
23         a = -1 * a;
24     } else {
25         a = a * 1;
26     }
27     return a;    /* повертаємо значення a
28 }
29

```

Строка: 18, Столбец: 1

ID по	Кадр	Функция	Аргументы	Расположение	Адрес	Переменная	Значение	Тип
1	0	main	()	main.c:18	0x000055	num_max	7	int
						num_min	-8	int
						res_prev_count	9	int
						res_next_count	4	int
						Параметры функции		

Терминал цели    **Контекст**    Точки останова    Регистры    Память    Монитор выражений

6. Бачи  
мо,  
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

новый файл: PROGRAMMING-KNIAZKIN/lab07/task01_5/README.md
новый файл: PROGRAMMING-KNIAZKIN/lab07/task01_5/src/main.c
новый файл: PROGRAMMING-KNIAZKIN/lab07/task01_6/README.md
новый файл: PROGRAMMING-KNIAZKIN/lab07/task01_6/src/main.c
новый файл: PROGRAMMING-KNIAZKIN/lab07/task02_5/README.md
новый файл: PROGRAMMING-KNIAZKIN/lab07/task02_5/src/a.out
новый файл: PROGRAMMING-KNIAZKIN/lab07/task02_5/src/main.c
новый файл: PROGRAMMING-KNIAZKIN/lab07/task02_6/README.md
новый файл: PROGRAMMING-KNIAZKIN/lab07/task02_6/src/a.out
новый файл: PROGRAMMING-KNIAZKIN/lab07/task02_6/src/main.c
новый файл: PROGRAMMING-KNIAZKIN/lab07/task03_5/README.md
новый файл: PROGRAMMING-KNIAZKIN/lab07/task03_5/src/a.out
новый файл: PROGRAMMING-KNIAZKIN/lab07/task03_6/README.md
новый файл: PROGRAMMING-KNIAZKIN/lab07/task03_6/src/main.c
новый файл: PROGRAMMING-KNIAZKIN/lab07/task04_6/README.md
новый файл: PROGRAMMING-KNIAZKIN/lab07/task04_6/src/a.out
новый файл: PROGRAMMING-KNIAZKIN/lab07/task04_6/src/main.c
новый файл: PROGRAMMING-KNIAZKIN/lab07/task05_5/README.md
новый файл: PROGRAMMING-KNIAZKIN/lab07/task05_5/src/main.c
новый файл: PROGRAMMING-KNIAZKIN/lab07/task05_6/README.md
новый файл: PROGRAMMING-KNIAZKIN/lab07/task05_6/src/main.c

valadon@valadon-VirtualBox:~/PROGRAMMING-KNIAZKIN$
```

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

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

```
valadon@valadon-VirtualBox: ~/PROGRAMMING-KNIAZKIN/PR...

create mode 100644 PROGRAMMING-KNIAZKIN/lab07/task03_5/README.md
create mode 100755 PROGRAMMING-KNIAZKIN/lab07/task03_5/src/a.out
create mode 100644 PROGRAMMING-KNIAZKIN/lab07/task03_5/src/main.c
create mode 100644 PROGRAMMING-KNIAZKIN/lab07/task03_6/README.md
create mode 100644 PROGRAMMING-KNIAZKIN/lab07/task03_6/src/main.c
create mode 100644 PROGRAMMING-KNIAZKIN/lab07/task04_6/README.md
create mode 100755 PROGRAMMING-KNIAZKIN/lab07/task04_6/src/a.out
create mode 100644 PROGRAMMING-KNIAZKIN/lab07/task04_6/src/main.c
create mode 100644 PROGRAMMING-KNIAZKIN/lab07/task05_5/README.md
create mode 100644 PROGRAMMING-KNIAZKIN/lab07/task05_5/src/main.c
create mode 100644 PROGRAMMING-KNIAZKIN/lab07/task05_6/README.md
create mode 100644 PROGRAMMING-KNIAZKIN/lab07/task05_6/src/main.c
valadon@valadon-VirtualBox:~/PROGRAMMING-KNIAZKIN/PROGRAMMING-KNIAZKIN$ git push
Username for 'https://github.com': Korv3L
Password for 'https://Korv3L@github.com':
Перечисление объектов: 107, готово.
Подсчет объектов: 100% (103/103), готово.
Сжатие объектов: 100% (70/70), готово.
Запись объектов: 100% (77/77), 1.50 MiB | 846.00 KiB/s, готово.
Всего 77 (изменения 15), повторно использовано 0 (изменения 0)
remote: Resolving deltas: 100% (15/15), completed with 6 local objects.
To https://github.com/Korv3L/PROGRAMMING-KNIAZKIN
 efd30ab..ffe755f  main -> main
valadon@valadon-VirtualBox:~/PROGRAMMING-KNIAZKIN/PROGRAMMING-KNIAZKIN$
```

### **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 мові С, у системі Linux, додавати до них коментарі, перевіряти програму на дієздатність у відлагоднику (nemiver), та відправляти зміни у свій репозиторій на github.