Лабораторна робота №8. Вступ до блок-схем алгоритмів.
Лабораторна робота №9. Вступ до документації коду(частина 1).
Лабораторна робота №10. Вступ до документації проекту.

Індивідульне завдання на оцінку “добре” номер 2.

1. Вимоги

1) Розробник

- Князькін Владислав Ігорович
- студент групи КІТ-320
  11.12.20

2) Загальне завдання

Розробити повноцінний звіти для лабораторної роботи “Функції”, що присвячена функціям у двох форматів (+їх презентація у PDF форматі)

3**) Індивідуальні 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().

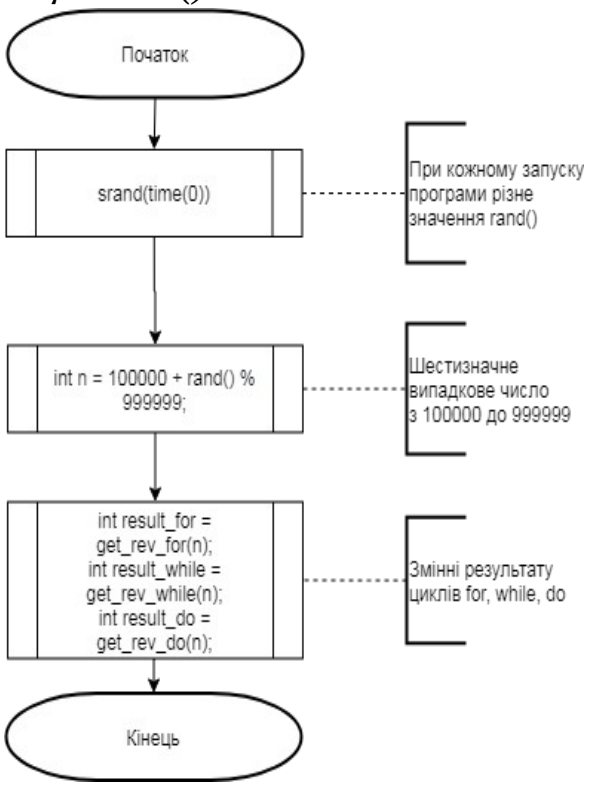
2. Хід роботи

1. Зробимо у раніше створеному репозиторії нову під-директорію *lab08_09_10.*

   valadon@valadon-VirtualBox:~ PROGRAMMING-KNIAZKIN $ mkdir lab08_09_10
2. Скопіюємо роботи з лабораторних робіт №5 та №6, *task02_5 task02_6* аналогічно.

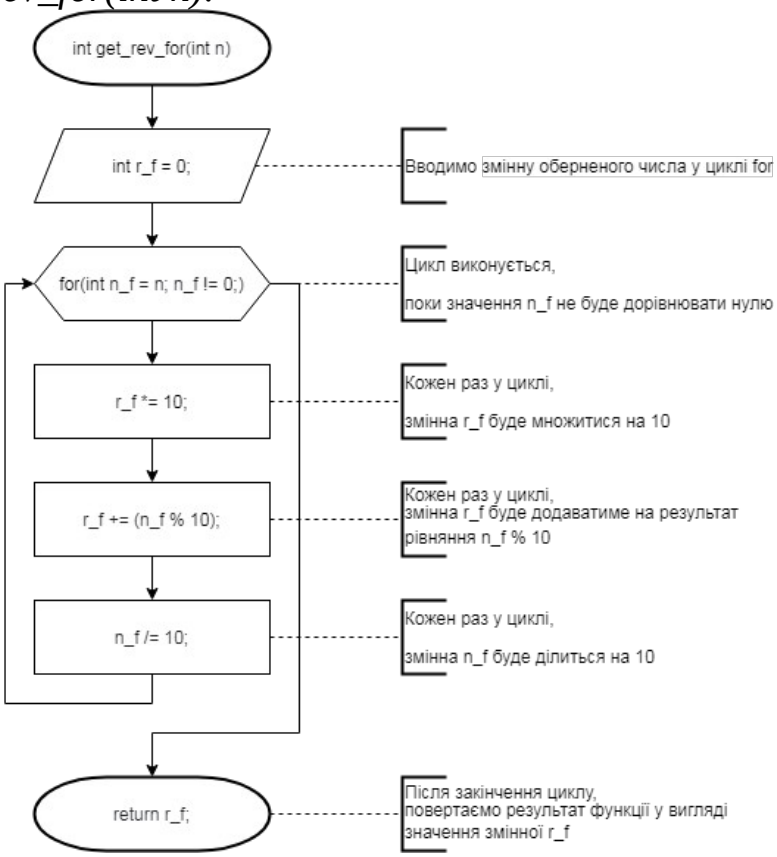
   valadon@valadon-VirtualBox:~ PROGRAMMING-KNIAZKIN/ $ cp -r lab07/task02_5 lab08_09_10
   valadon@valadon-VirtualBox:~ PROGRAMMING-KNIAZKIN/ $ cp -r lab07/task02_6 lab08_09_10

3. Перейменовуємо роботи у *lab02_5*, *lab02_6.* Переглянемо наші програми у вигляді блок-схем алгоритмів.
   Спочатку — розберемо lab*02_5:*
   Функція main():

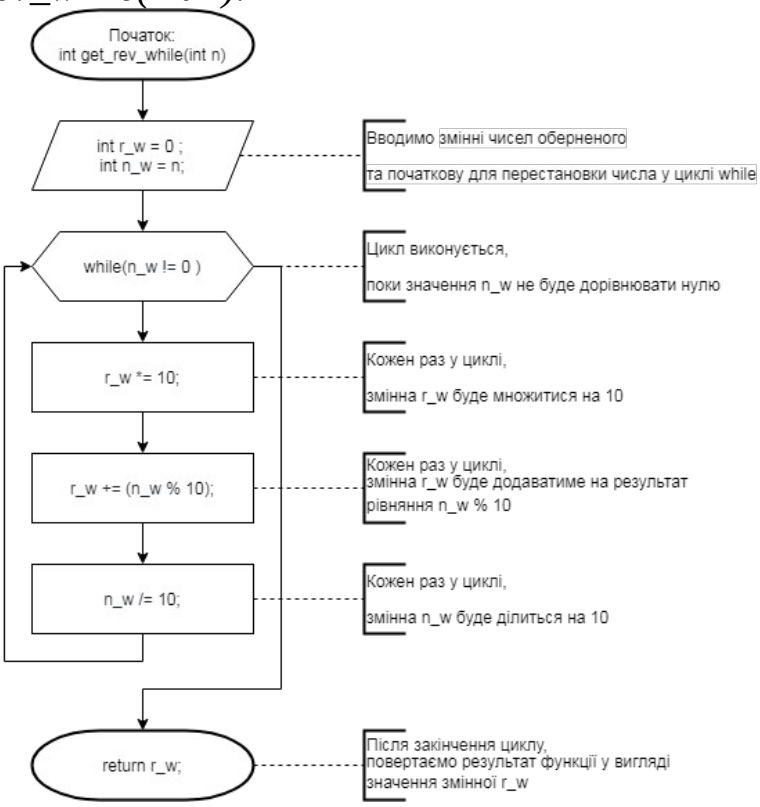


Усі три функції, а саме get_rev_for, get_rev_while та get_rev_do, виконують одну й теж саму дію, але розроблені вони на основі відомих для нас, мовою програмування С, циклів For, While_Do та Do_While відповідно. Розглянемо кожну функцію на основі кожного з вище перерахованих циклів, у виді блок-схем алгоритмів.

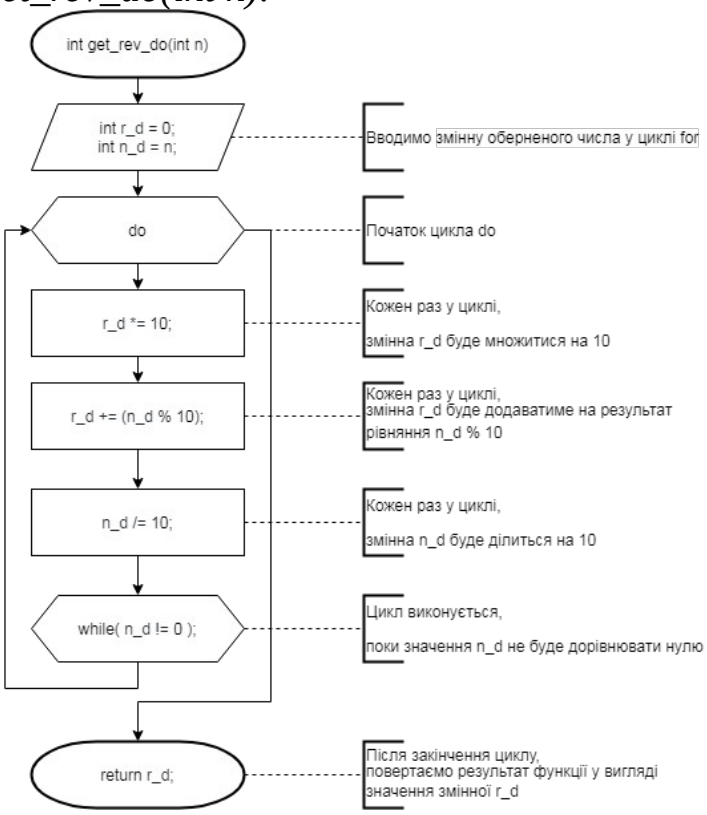
Функція int get_rev_for(int n):



Функція int get_rev_while(int n):

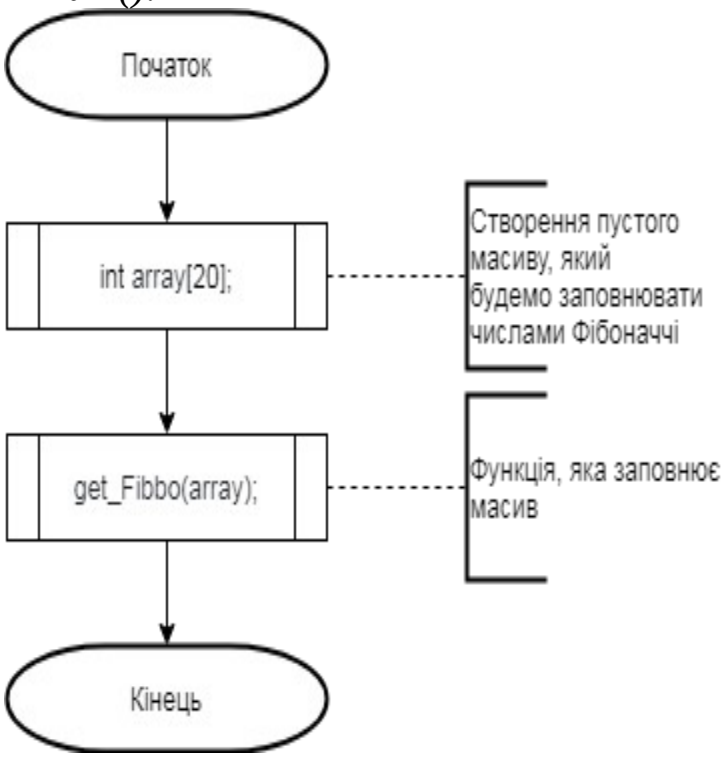


Функція int get_rev_do(int n):



Розберемо lab02_6:

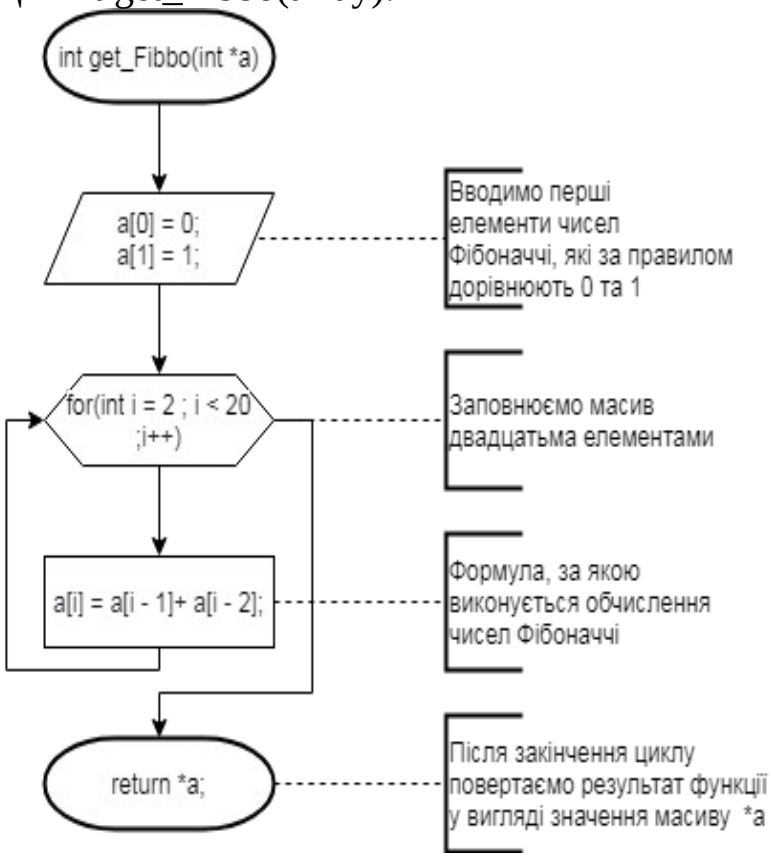
Функція main():



У цій програмі, ми створюємо функцію get_Fibbo, яка має

заповнювати наш масив, за формулою для розрахунку елементів цілих чисел Фібоначчі.

Функція int get_Fibbo(array):



4. Повертаємось до терміналу, щоб перевірити якість роботи наших програм. Звернемо увагу на те, що у відлагоднику ми не бачимо дії власних функцій, але бачимо результат, завдяки діям *return*, які повертають результат функцій до змінних, які ми вказали.
   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.

Програма task02_5:

main.c

```
1  #include <stdlib.h>
2  #include <time.h>
3  int get_rev_for(int n);
4  int get_rev_while(int n);
5  int get_rev_do(int n);
6  //*
7  int main(){
8          srand(time(0));
9          int n = 100000 + rand() % 999999;
10
11         int result_for = get_rev_for(n);
12         int result_while = get_rev_while(n);
13         int result_do = get_rev_do(n);
14         }
15 //*
16 int get_rev_for(int n){
17         int r_f = 0;    //* змінна оберненого числа у циклі for
18         for(int n_f = n; n_f != 0;){
19         r_f *= 10;
20         r_f += (n_f % 10);
21         n_f /= 10;
22         }
23         return r_f;
24 }
25 //*
26 int get_rev_while(int n){
27         int r_w = 0;    //* змінна оберненого числа у циклі while
28         int n_w = n;    //* змінна початкового числа для перестановки числа
29         while(n_w != 0 ){
30         r_w *= 10;
```

ID по	Кадр	Функция	Аргументы	Расположение	А
1	0	main	()	main.c:14	0

Переменная	Значение	Тип
Локальные переменные		
n	564902	int
result_for	209465	int
result_while	209465	int
result_do	209465	int
Параметры функции		

Програма task02_6:

main.c

```
1  #include <stdlib.h>
2  #include <time.h>
3  int get_Fibbo(int *a); //* функція для заповнення масиву числами Фібоначчі
4
5  int main()     // * створюємо функцію
6  int array[20];         // * створюємо масив з 20 елементів
7  get_Fibbo(array);
8  }
9
10
11 int get_Fibbo(int *a){
12 a [0] = 0;      // * оголошуємо початкове значення нульового елемента масиву
13 a [1] = 1;      // * оголошуємо початкове значення першого елемента масиву
14 for (int i = 2; i <20; i++) {  // * створюємо цикл для заповнення масиву цілими числами Фібоначчі
15 a[i] = a[i - 1]+ a[i - 2];     // * записуємо формулу для обчислення чисел Фібоначчі
16 }       // * кінець циклу
17 return *a;      // * повертаємо результат програми
18 }       // * кінець функції
```

ID по	Кадр	Функция	Аргументы	Расположение	А
1	0	main	()	main.c:8	0

Переменная	Значение	Тип
Локальные переменные		
array	[20]	int [20]
0	0	int
1	1	int
2	1	int
3	2	int
4	3	int
5	5	int
6	8	int
7	13	int
8	21	int
9	34	int
10	55	int
11	89	int
12	144	int
13	233	int
14	377	int
15	610	int
16	987	int
17	1597	int

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$ git add .
valadon@valadon-VirtualBox:~/PROGRAMMING-KNIAZKIN$ git status
На ветке main
Ваша ветка базируется на «origin/main», но вышестоящий репозиторий исчез.
  (для исправления запустите «git branch --unset-upstream»)

Изменения, которые будут включены в коммит:
  (use "git restore --staged <file>..." to unstage)
        новый файл:    lab08_09_10/DoxyFile
        изменено:      lab08_09_10/Makefile
        удалено:       lab08_09_10/task02_5/Makefile
        новый файл:    lab08_09_10/task02_5/src/a.out
        новый файл:    lab08_09_10/task02_6/src/a.out

Изменения, которые не в индексе для коммита:
  (используйте «git add <файл>…», чтобы добавить файл в индекс)
  (use "git restore <file>..." to discard changes in working directory)
  (сделайте коммит или отмените изменения в неотслеживаемом или измененном содер
жимом в подмодулях)
```

valadon@valadon-VirtualBox:~ PROGRAMMING-KNIAZKIN/naxxramass $ git commit -m “Create lab08_09_10”

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

```
valadon@valadon-VirtualBox:~/PROGRAMMING-KNIAZKIN$ git commit -m "Create lab08_0
9_10"
[main 41417d4] Create lab08_09_10
 5 files changed, 2581 insertions(+), 33 deletions(-)
 create mode 100644 lab08_09_10/DoxyFile
 delete mode 100644 lab08_09_10/task02_5/Makefile
 create mode 100755 lab08_09_10/task02_5/src/a.out
 create mode 100755 lab08_09_10/task02_6/src/a.out
valadon@valadon-VirtualBox:~/PROGRAMMING-KNIAZKIN$ git push
Username for 'https://github.com': Korv3L
Password for 'https://Korv3L@github.com':
Перечисление объектов: 18, готово.
Подсчет объектов: 100% (18/18), готово.
Сжатие объектов: 100% (11/11), готово.
Запись объектов: 100% (11/11), 34.56 KiB | 631.00 KiB/s, готово.
Всего 11 (изменения 3), повторно использовано 0 (изменения 0)
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To https://github.com/Korv3L/PROGRAMMING-KNIAZKIN
   8002f46..41417d4  main -> main
error: update_ref failed for ref 'refs/remotes/origin/main': cannot lock ref 're
fs/remotes/origin/main': unable to resolve reference 'refs/remotes/origin/main':
 reference broken
```

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.
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.