Звіт

Лабораторна робота №07

Тема: Функції.

Розробник: студент Клименко Станіслава Олександрівна, група 120-а.

Перевірив: асистент Челак Віктор Володимирович.

Індивідуальне завдання: Реалізувати функцію, що визначає скільки серед заданої послідовності чисел таких пар, у котрих перше число більше за наступне.

Опис програми 1:

- *Функціональне призначення*: Визначення чи є чилсло досконалим. Якщо сумма всіх дільників числа буде дорівнювати самому числу, то число досконале.

- *Опис логічної структури*:

* Функція ``main``. Оголошує два массиви. Перший массив це наш масив з числами, що перевіряють на досконаліcть, а другий це масив відповідей, що покаже нам досконале число, чи ні. Викликає функцію ``perfect_numbers``.

* Функція ``perfect_numbers``. Оголошуємо значення суми дільників нашого числа. Цикл допомагає нам знайти значення суми дільників, а наступні наші дії заповнюють наш масив відповіді залежно від порівняння числа та суми його дільників.(рис.1)

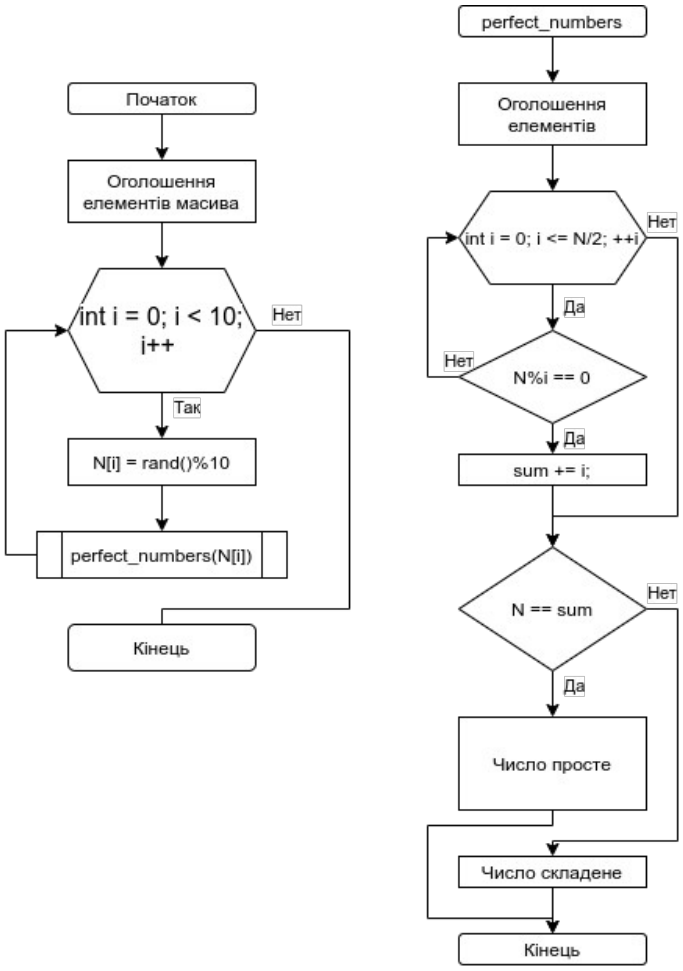

Рисунок 1 — блок-схема програми

Варіанти використання программи 1

- Поставимо точку зупину наприкінці нашої головної функції, та побачимо результат перевірки нашого числа на досконалість.(рис.2)

```c
#include <stdio.h>
#include<math.h>

/**
 * @file ex1.c виконання перевірки числа на досконалість
 */

/**
 * перевірка числа на простоту за допогою флагів
 * @param N наше задане число
 */

char perfect_numbers (int N);

/**
головна функція  {задати фунцію, що кожного разу дає нове псевдовипадкове число,
                  оголосити число,
                  виконати перевірку на досконалість}
*/

int main (){
    srand(time(NULL));
    int N[10];
    char ans[10];
    for (int i = 0; i < 10; i++){
        N[i] = rand()%10;
        ans[i] = perfect_numbers(N[i]);
    }
}

char perfect_numbers (int N){
    int sum = 0;
    for (int i = 1;i<=N/2;++i){
        if(N%i == 0){
            sum +=i;
        }
    }
    if(N == sum){
        return 'Y';
    }
    else {
        return 'N';
    }
}
```

Рисунок 2 — готова програма з постановкою точки зупину

Опис программи 2

- *Функціональне призначення*: Заповнення масиву простими числаами, що не повторюються.

- *Опис логічної структури*:

* Функція ``main``. Оголошує розміри масиву і самі масиви. Викликає функцію ``generation_array``.

* Функція ``generation_array``. Оголошуємо значення числа ,яке ми перевіряємо на простоту, а після заносимо у масив. Перший цикл відповідає за внесення числа у масив, другий за перевірку числа на простість.(рис.3)

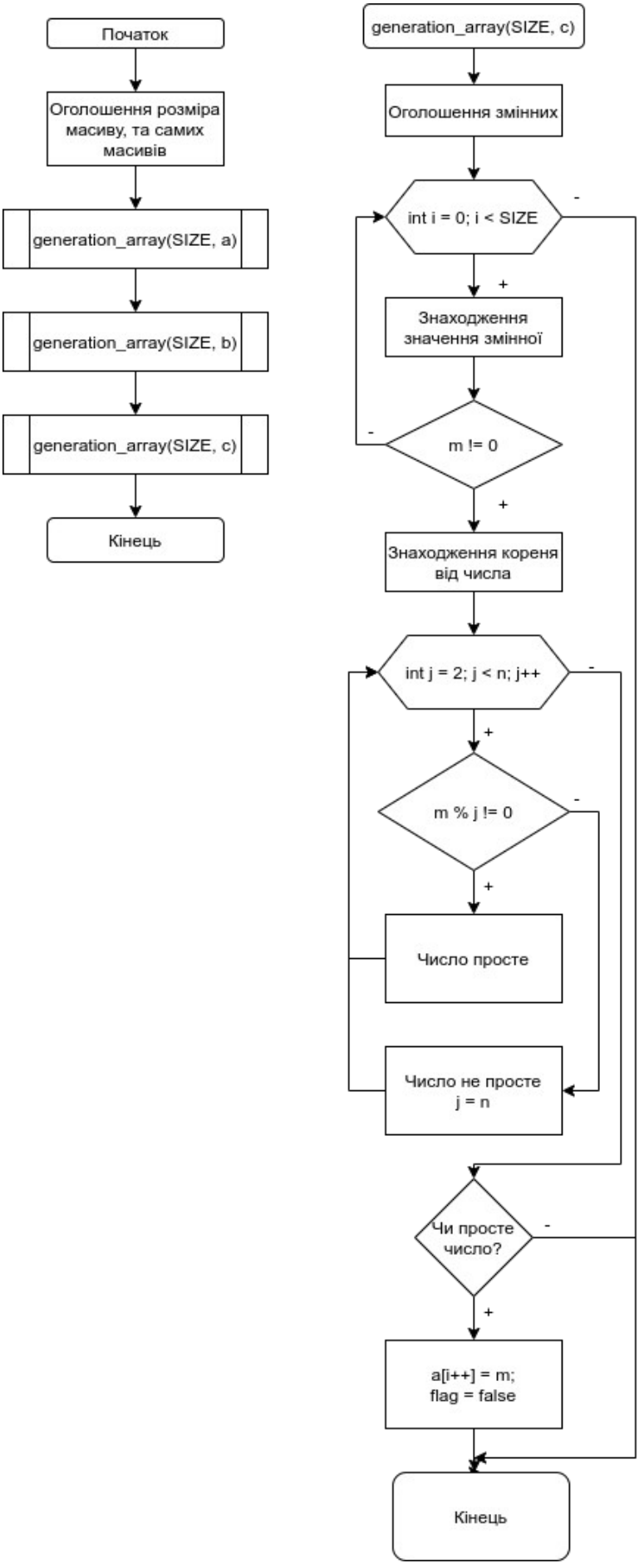

Рисунок 3 — блок-схема програми

Варіанти використання программи 2

- Ми можемо змінювати розмір масиву залежно від нашого бажання ,цей масив буде заповнюватися простими числами. Поставимо точку зупину наприкінці нашої головної функції, та побачимо вже запонений масив.(рис.4)

```
int main() {
    srand(time(NULL));
    int SIZE = 2;
    int a[SIZE];
    int b[SIZE];
    int c[SIZE];

    generation_array(SIZE, a);
    generation_array(SIZE, b);
    generation_array(SIZE, c);
    return 0;
}

void generation_array (int SIZE, int a[]){
    int m;
    bool flag = false;
    for (int i = 0; i < SIZE;) {
        m = rand() % 100;
        if (m !=0){
            const double eps=0.01;
            double n=0;
            while(n * n <=m){
                n+=eps;
            };
            for (int j = 2; j < n; j++) {
                if (m % j != 0) {
                    flag = true;
                }
                else{
                    flag = false;
                    j = n;
                    }
            }
            if (flag)
            {
                a[i++] = m;
                flag = false;
            }
        }
    }
    return ;

}
```

Рисунок 4 — готова програма з постановкою точки зупину

Опис программи 3

- *Функціональне призначення*: Реалізування фукції, що буде визначати кількість пар у яких перше число більше за наступне.

- *Опис логічної структури*:

* Функція ``main``. Оголошує змінну, що відповідає на наше функціональне призначення, та показує кількість пар, у котрих друге число меньше за попередне. Викликає функцію ``function``.

* Функція ``function``. Оголошуємо перемінну, що відповідає за наше функціональне призначення.Змінну, завдяки якій ми зможемо порівн.ювати значення з попереднім значенням. ``result`` - змінюється після перевірки на те, чи задовільняє вана нашій умові.(рис.5)

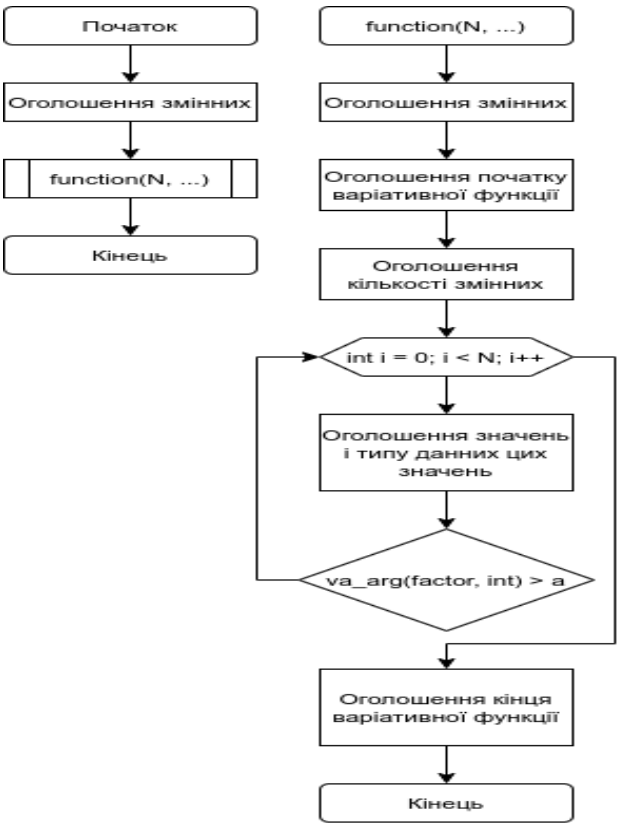

Рисунок 5 — блок-схема програми

Варіанти використання программи 3:

- Ми можемо дізнатися, скільки таких пар, де друге число меньше за попередне. Поставивши точку зупину наприкінці нашої головної функції ми побачиму змінну ,та чому вона дорівнює, що відповідає за те, скільки у нас пр чисел, що відповідають умові завдання.(рис.6)

```c
#include <stdarg.h>

/**
 * @file ex3.c реалізація функції, що визначає, скільки серед заданої пос
 *
*/

/**
 * реалізація варіативної функції, що повертає значення кількості пар, що
 * @param N значення, що відповідає за кількість значень викликаної функц
 */

int function(int N, ...);

/**
головна функція  { Оголошення змінної та виклик варіативної функції }

*/

int main(){
    int result = function(12, 17, 0, -34, 8, 5, 10, -9, 99, 10, 13);
    return 0;
}

int function(int N, ...){

    int a = 0;
    int result = 0;
    va_list factor;
    va_start (factor, N);
    for (int i = 1; i < N; i++){
        va_arg(factor, int);
        if(va_arg(factor, int) > a){
            result++;
        }
        a = va_arg(factor, int);

    }
    va_end(factor);
    return (result);
}
```

Рисунок 6 — готова програма з постановкою точки зупину

Висновок:

Для виконання лабораторної роботи ми навчились створювати та реалізовувати алгоритми функції, створювати схеми алгоритмів, та оформлювати документацію.