Міністерство освіти і науки України
Національний технічний університет
«Харківський політехнічний інститут»
Кафедра «Обчислювальна техніка та програмування»

ЗВІТ

Про виконання лабораторної роботи № 18, 19
«Динамічні масиви. Динамічні списки»

Керівник: викладач
Бульба С. С.

Виконавець: студент гр. КІТ-120в
Бабенко А. П.

Харків 2021

Лабораторна робота №18, 19. Динамічні масиви. Динамічні списки

1. Вимоги

1.1 Розробник

- Бабенко Антон Павлович;
- Студент групи КІТ-120в;
- 20 лютого 2021;

1.2 Загальне завдання

- розробити функцію, яка реалізує вставку в рядок “s” другий рядок “s2” в “i”-у позицію рядка “s”. Наприклад, `insert(“abrakadabra”, “TEXT2”, 4)` повинна створити рядок “abraTEXT2kadabra”;
- розробити функцію видалення з рядка “s” усіх символів з індексами в заданому діапазоні. Наприклад, `reduce(“abrakadabra”, 4, 8)` повинна створити рядок “abrara” (без підрядка kadab).
- за допомогою функцій memcpy, memset створити функції додання та видалення елементів з динамічного масиву вашої прикладної області
- додати модульні тести, що демонструють коректність розроблених функцій

1.3 Індивідуальне завдання

- виконати завдання на “добре”, але замість односпрямованого списку треба використовувати доспрямований;
- Реалізувати сортування вмісту списку за одним з критеріїв. При цьому обов’язково забезпечити, щоб обмін місцями об’єктів здійснювався шляхом обміну їх покажчиків.

1.4 Функціональне призначення

Програма №1 призначена для додавання та видалення елементів з

динамічного масиву за допомогою функцій memset та memcpy. Програма №2 призначена для сортування двоспрямованого списку за одним критерієм використовуючи перестановку покажчиків.

2 Виконання роботи

2.1 Написання коду, що додає та видаляє елементи з динамічного масиву за допомогою функцій memset та memcpy . Зображено на рис.1.

```
file * remove_struct(file *files, int N)
{
    printf("\nВведите номер структуры, которую нужно удалить(0-%d): \n", N - 1);
    int to_remove;
    scanf("%d", &to_remove);
    memset(files + to_remove, NULL, sizeof(file));
    for (int i = to_remove; i < N - 1; i++)
    {
        memcpy(files + i, files + i + 1, sizeof(file));
    }

    files = (file *)realloc(files, sizeof(file) * (N - 1));

    return files;
}

file * add_struct(file *files, int N)
{
    files = (file *)realloc(files, sizeof(file) * (N + 1));

    printf("\nВведите номер структуры, которую нужно добавить(0-%d): \n", N);
    int to_add;
    scanf("%d", &to_add);
    for (int i = N; i > to_add; i--)
    {
        memcpy(files + i, files + i - 1, sizeof(file));
    }

    files[to_add].isVisible = random_bool();
    sprintf(files[to_add].filename, "add");
    files[to_add].size = random_float();
    files[to_add].access.write = random_bool();
    files[to_add].access.read = random_bool();
    files[to_add].access.execute = random_bool();
    char ext_arr[10][5] = {"txt", "docx", "pdf", "mp3", "avi", "mp4", "mkv", "exe", "bat", "jar"};
    sprintf(files[to_add].extension, ext_arr[randomer() - 1]);

    return files;
}
```

Рисунок 1 – код програми

2.2 Виконання опису функцій. Зображення опису зі сторінок документації doxygen на рис. 2.

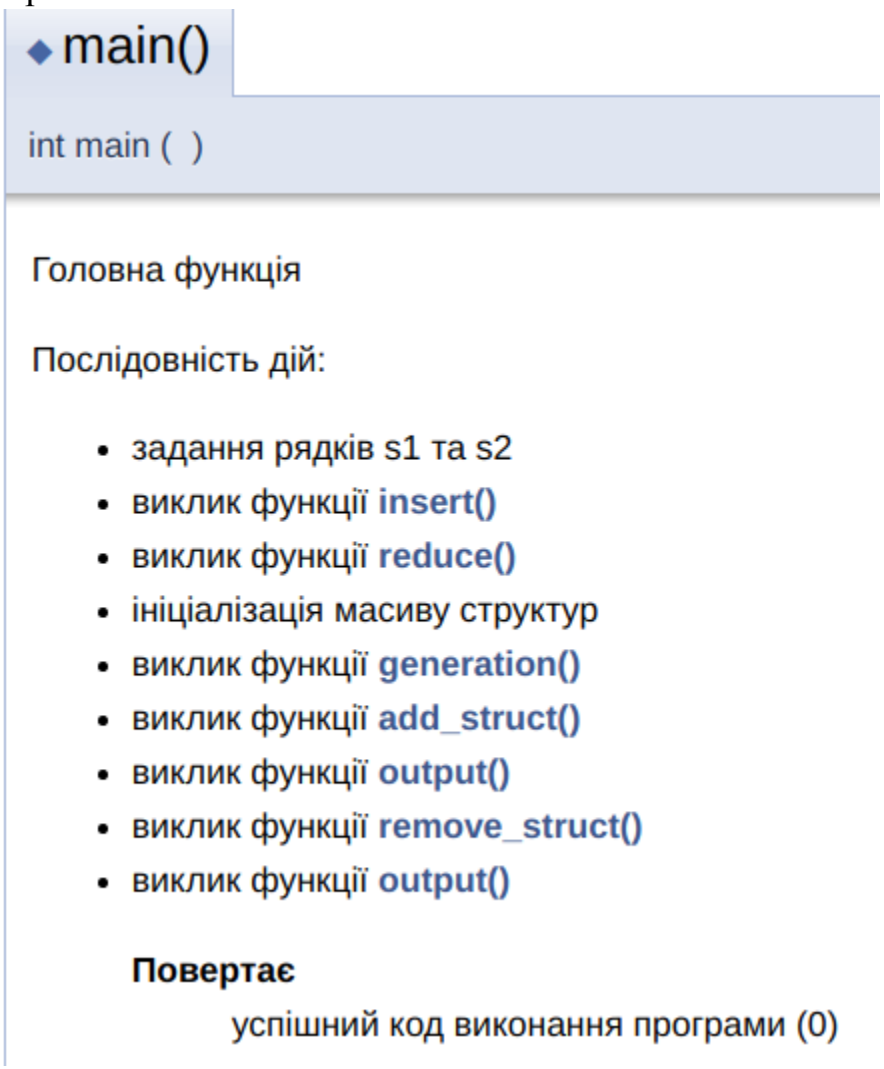

Рисунок 2 – опис функції

2.3 Ставимо точку зупину, проходимо по файлу і бачимо зміну значення в масиві в момент проходження по програмі. Зображено на рис.3.

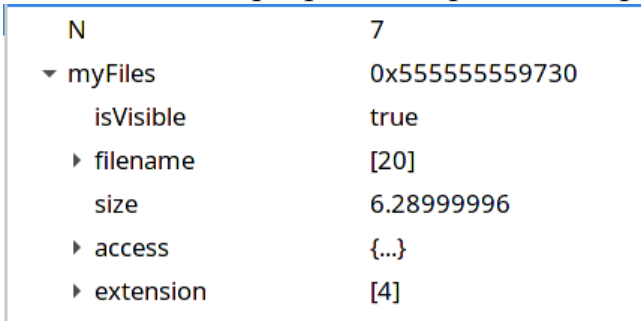

Рисунок 3 – файл у відлагоднику

2.4 Створення блок-схеми програми і графу викликів функції main. Зображення блок-схеми на рис.4-5.

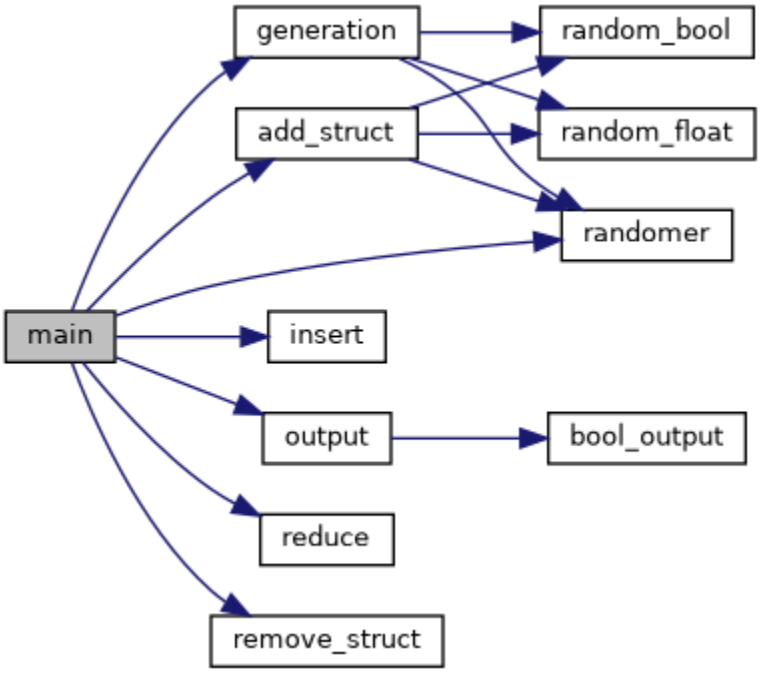

Рисунок 4 – граф викликів функції

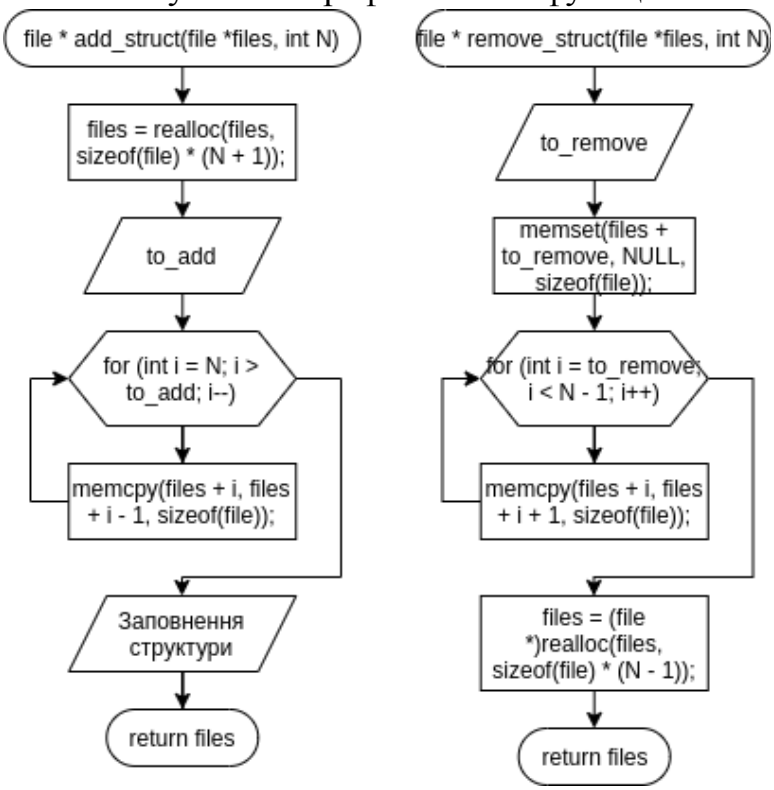

Рисунок 5 – блок-схема

2.5 Написання коду для сортування двоспрямованого списку за одним критерієм використовуючи перестановку покажчиків. Зображено на рис.6.

```
void sort_by_criterion(DblLinkedList *list)
{
    int criterion = 0;
    printf("\nPick criterion for sorting: \nVisibility[
    scanf("%d", &criterion);

    Node *node = list->head;
    Node *n1, *n2, *n3, *n4;

    bool flag;

    while (node->next != NULL)
    {
        switch (criterion)

        if (flag)
        {
            n1 = node->prev;
            n2 = node;
            n3 = node->next;
            n4 = node->next->next;

            if (n1 == NULL)
                list->head = n3;
            else
                n1->next = n3;

            n3->next = n2;
            n2->next = n4;

            n3->prev = n1;
            n2->prev = n3;

            if (n4 != NULL)
                n4->prev = n2;

            if (n1 != NULL)
                node = n1;
        }
        else
            node = node->next;
    }
}
```

Рисунок 6 – код програми

2.6 Створення блок-схеми програми. Зображено на рис.7.

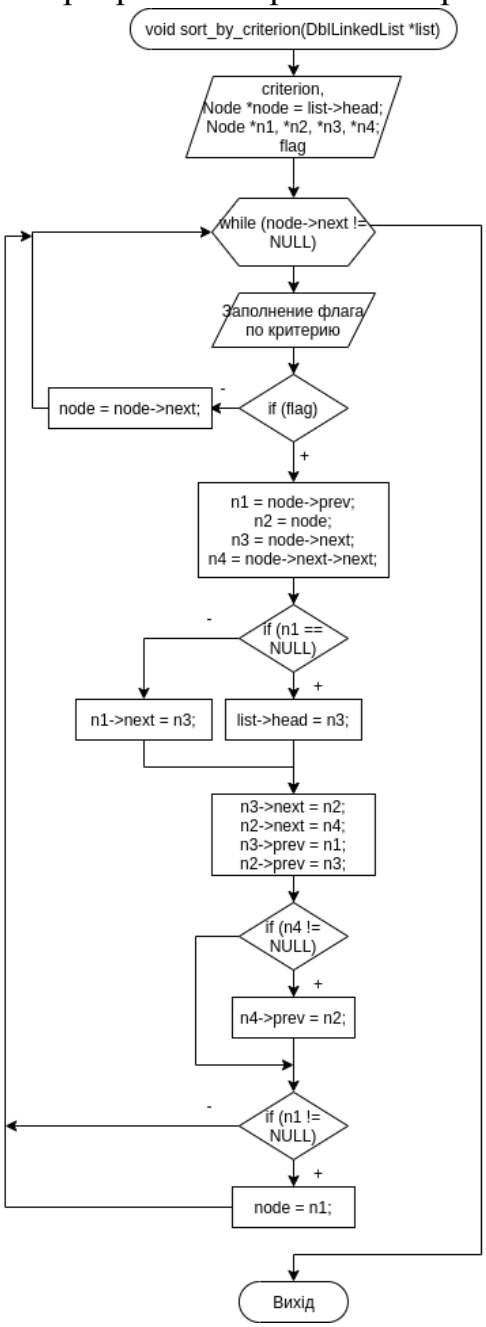

Рисунок 7 – блок-схема

2.7 Виконання опису функцій. Зображення опису зі сторінок документації doxygen на рис. 8.

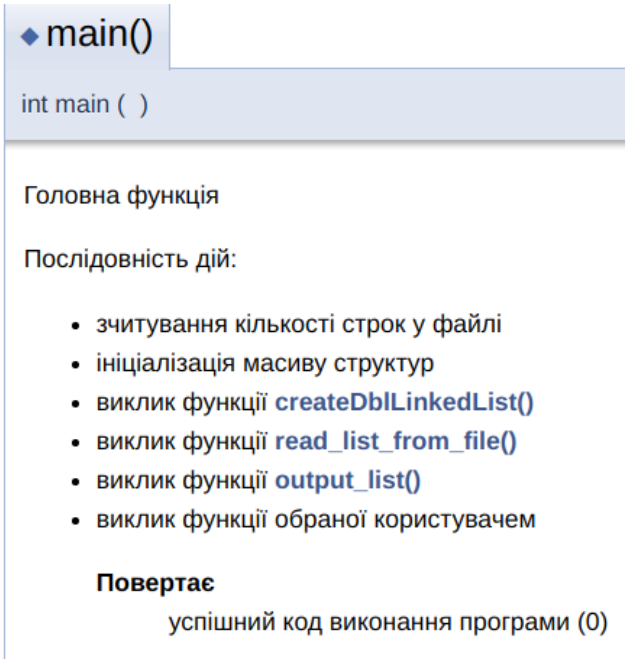

Рисунок 8 – опис функції

2.8 Створення графу викликів функції main. Зображення на рис.9.

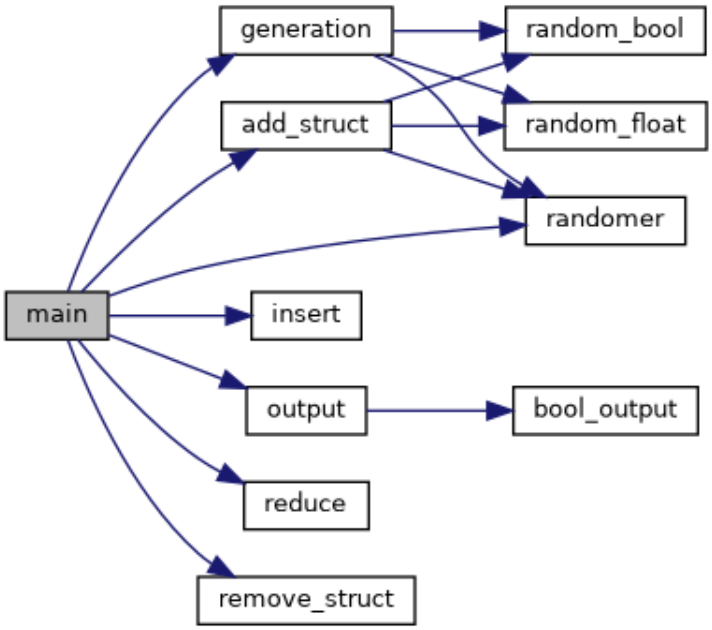

Рисунок 9 – граф викликів функції

Висновки

При виконанні даної лабораторної роботи було набуто навичок розробки програм із динамічними масивами та динамічними списками.