**Лабораторна робота № 23. ООП. Вступ до ООП**

1. ВИМОГИ
   1. Розробник

**** Радєвич Владислав Романович;

**** студент групи КІТ – 320;

**** 20.05.2021 р.

* 1. Загальне завдання

Для предметної галузі з розділу "Розрахункове завдання/Індивідуальні завдання" розробити два класи: - клас, що відображає сутність "базового класу", у тому числі: - конструктор за замовчуванням, копіювання та конструктор з аргументами (реалізація конструкторів продемонстрована за допомогою списків ініціалізацій); - деструкторі - клас, що має у собі динамічний масив об'єктів базового класу та має в собі методи додавання, видалення елемента, отримання елемента по індексу (або ідентифікатору), вивід усіх елементів на екран..

1. ОПИС ПРОГРАМИ
   1. Функціональне призначення

Програма призначена для сортування структури даних заданих птахів з файлу та виведення результату у файл або на екран, використовуючи динамічні списки.

* 1. Опис логічної структури
     1. Основна функція

int main

*Призначення*: головна функція.

*Схема алгоритму функції* подана на рис. 1.

*Опис роботи:* демонструє роботу заданого динамічного списку елементів структури та методів оперування ним, викликаючи функція для роботи з динамічним списком.

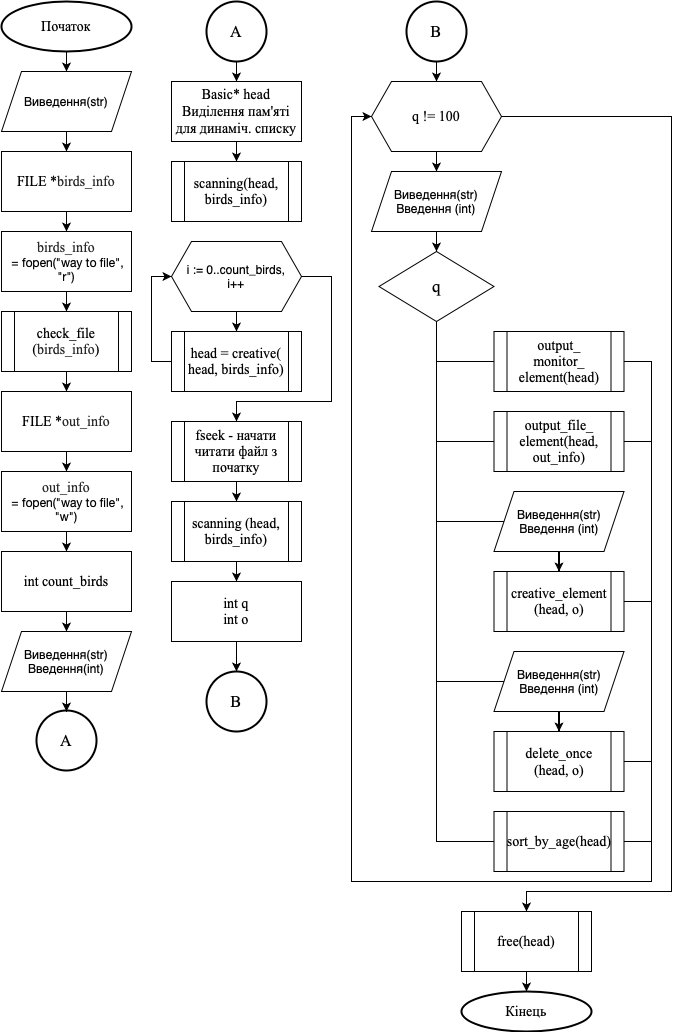


Рисунок 1 – Схема алгоритму функції main

* + 1. Функція перевірки файлу

void check\_file (FILE \*fmatrix);

*Призначення*: перевірка місцезнаходження файлу по вказаному шляху.

*Схема алгоритму функції* подана на рис. 2

*Опис роботи*: функція перевіряє знаходження файлу з даними для структурами по заданій директорії.

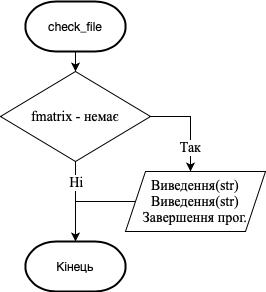


Рисунок 2 – Схема алгоритму функції check\_file

* + 1. Функція сканування даних

void scanning (Basic\* p\_bird, int count\_birds, FILE \*birds\_info);

*Призначення*: сканування данні з файлу, в якому заздалегідь написані данні для структури.

*Схема алгоритму функції* подана на рис. 3.

*Опис роботи*: функція сканує та переносить данні з заданого файлу в задану структуру за допомогою циклу.

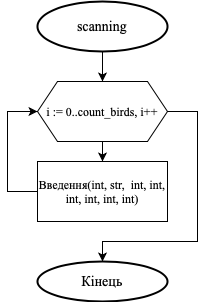


Рисунок 3 – Схема scanning

* + 1. Функція виводу у файл

void output\_file (Basic\* p\_bird, int count\_birds, FILE\* out\_info);

*Призначення:* вивід даних у файл за певної директорією.

*Схема алгоритму функції* подана на рис. 4

*Опис роботи*: оброблення усіх полей структури та вивід усіх даних, що містяться в усіх структурах в певний файл.

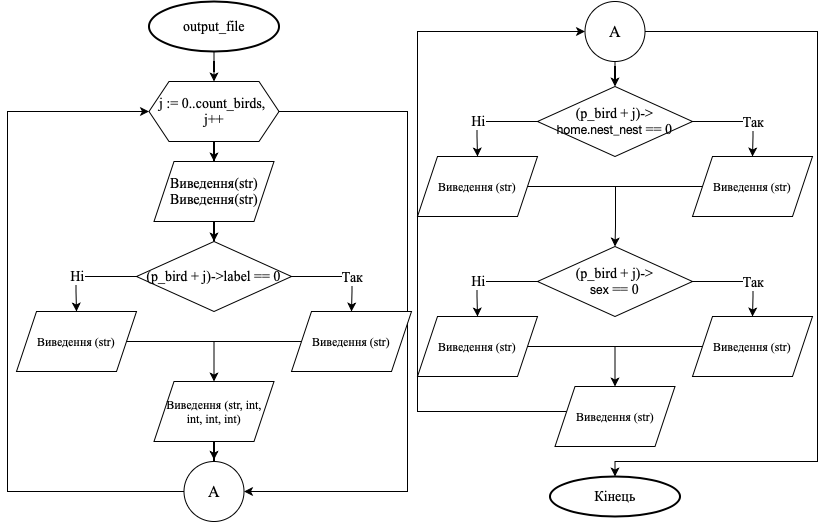


Рисунок 4 – схема алгоритму функції output\_file

* + 1. Функція виводу даних на екран

void output\_monitor (Basic\* p\_bird, int count\_birds);

*Призначення:* вивід окремих даних кожної структури з масиву структур на екран.

*Схема алгоритму функції* подана на рис. 5

*Опис роботи*: оброблення та вивід усіх даних кожної структури з зазначеного масиву структур.

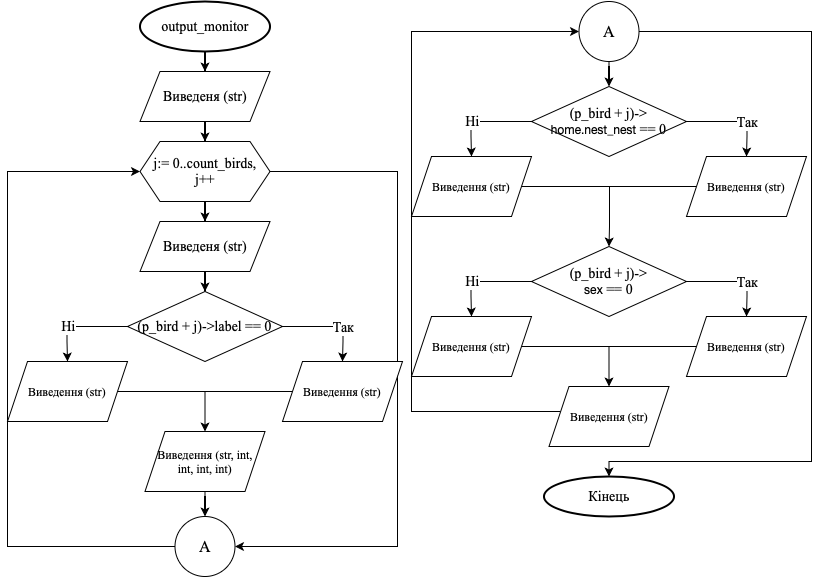


Рисунок 5 – схема алгоритму функції output\_monitor

* + 1. Функція вводу даних з файлу для списку

Basic\* creative(Basic\* head, FILE \*birds\_info);

*Призначення:* перепис даних з заданого файлу до динамічного списку елементів структури.

*Схема алгоритму функції* подана на рис. 6

*Опис роботи:* за допомогою функції scanning записуються дані з файлу до динамічного списку.



Рисунок 6 – смеха алгоритму функції creative

* + 1. Функція створення одного елементу

void creative\_element(Basic\* head, int position);

*Призначення:* додавання елементу структури до динамічного списку.

*Схема алгоритму функції* подана на рис. 7

*Опис роботи:* функція додає, у певну позицію, до заданого динамічного списку ще один елемент структури.

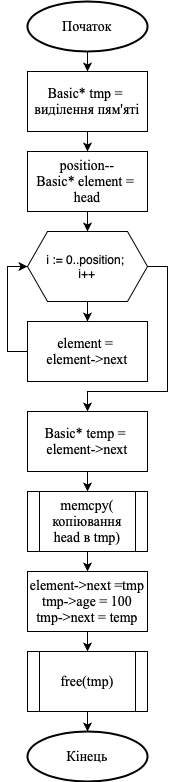


Рисунок 7 – смеха алгоритму функції creative\_element

* + 1. Функція видалення елементу

void delete\_once (Basic\* head, int n);

*Призначення:* видалення елементу структури з динамічного списку.

*Схема алгоритму функції* подана на рис. 8

*Опис роботи:* функція видаляє певний елемент структури з динамічного списку елементів структури.

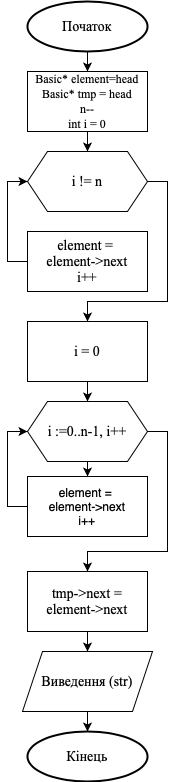


Рисунок 8 – смеха алгоритму функції delete\_once

* + 1. Функція виводу у файл списку

void output\_file\_element (Basic\* head, FILE \*out\_info);

*Призначення:* виведення у файл динамічний список.

*Схема алгоритму функції* подана на рис. 9

*Опис роботи:* виведення у заданий файл динамічного списки елементів структури.

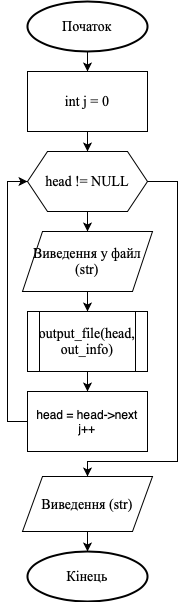


Рисунок 9 – смеха алгоритму функції output\_file\_element

* + 1. Функція виводу на екран для списку

void output\_monitor\_element (Basic\* head);

*Призначення:* виведення на екран динамічний список.

*Схема алгоритму функції* подана на рис. 10

*Опис роботи:* виведення на екран динамічного списки елементів структури.

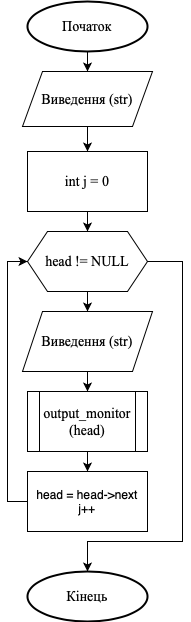


Рисунок 10 – смеха алгоритму функції output\_monitor\_element

* + 1. Функція сортування для списку

void sort\_by\_age (Basic\* head);

*Призначення:* сортування динамічного списку.

*Схема алгоритму функції* подана на рис. 11

*Опис роботи:* сортування за відповідним критерієм (в цьому випадку за роками життя) динамічного списку елементів структури від меншого до більшого. Сортування виконується методом «бульбашки».

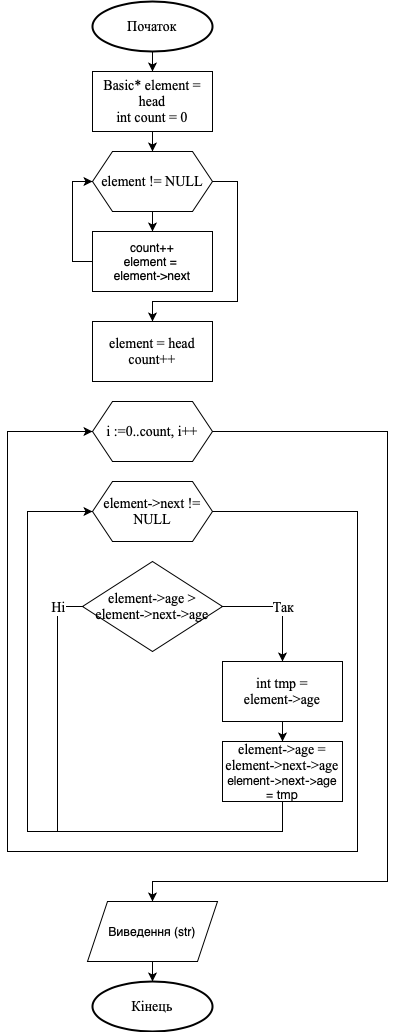


Рисунок 11 – смеха алгоритму функції sort\_by\_age

* 1. Структура проекту

.

├── 1

│   ├── src

│   │   ├── lib.c

│   │   ├── lib.h

│   │   └── main.c

│   └── test

│   └── test.c

├── 2

│   ├── src

│   │   ├── data.c

│   │   ├── data.h

│   │   ├── list.c

│   │   ├── list.h

│   │   └── main.c

│   └── test

│   └── test.c

├── Doxyfile

├── Makefile

├── README.md

└── doc

├── Radievych18\_19.pdf

└── assets

├── creative.png

├── creative\_element.png

├── delete\_once.png

├── flowchart\_\ output\_file.png

├── flowchart\_check\_file.png

├── flowchart\_output\_monitor.png

├── flowchart\_scanning.png

├── main.png

├── output\_file\_element.png

├── output\_monitor\_element.png

└── sort\_by\_age.png

1. Варіанти використання

Цю програму можна використовувати за для перепису усіх зареєстрованих птахів в окремий файл на комп’ютері або для заповнення списку, заздалегідь давши про них певну інформацію.

Результат роботи з doxygen продемонстровано на рисунку 12, рисунку 13 та рисунку 14, виконання модульних тестів на рисунку 15 та демонстрація відсутності витоків пам’яті на рисунку 16..

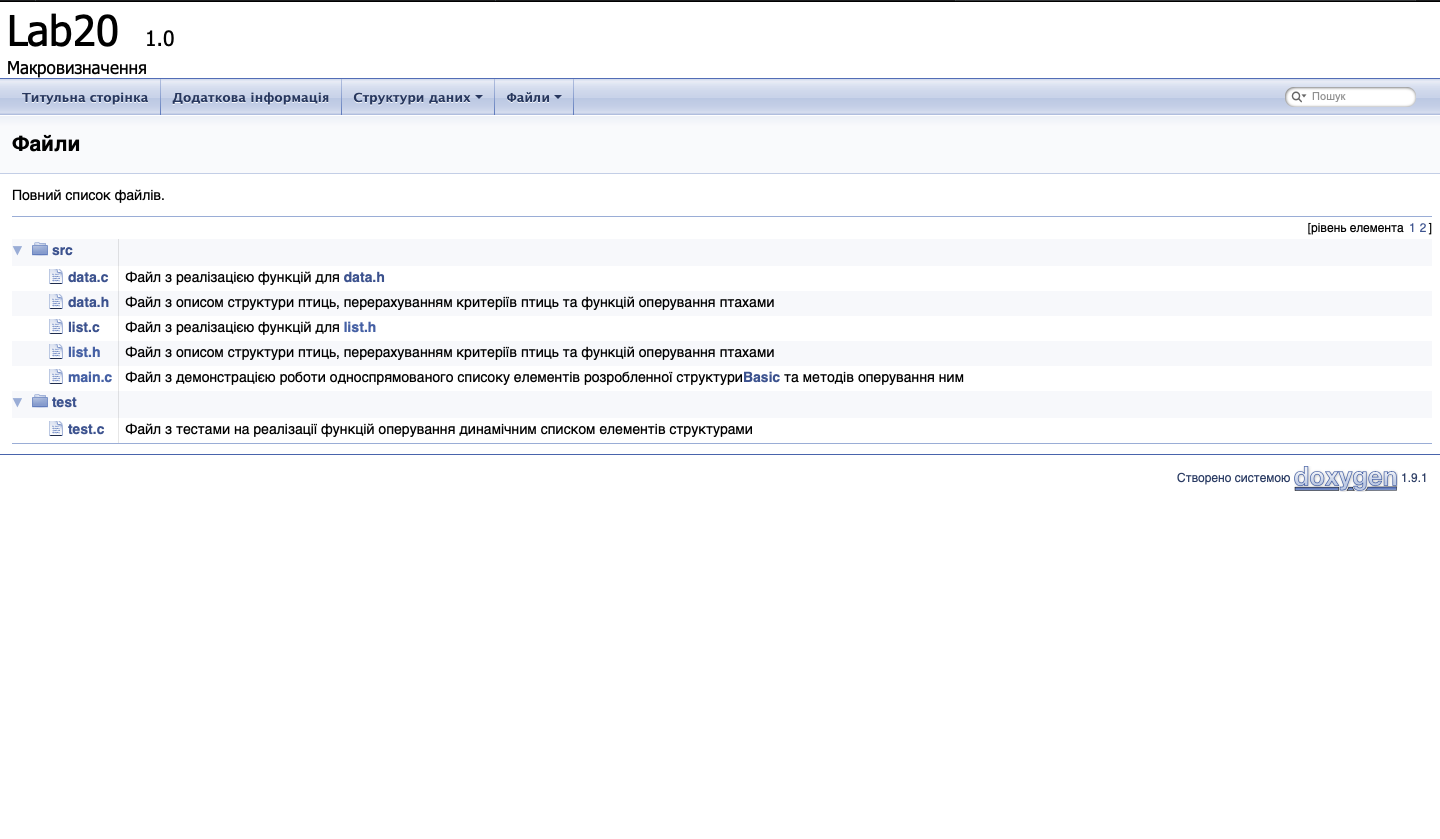


Рисунок 12 – робота з doxygen

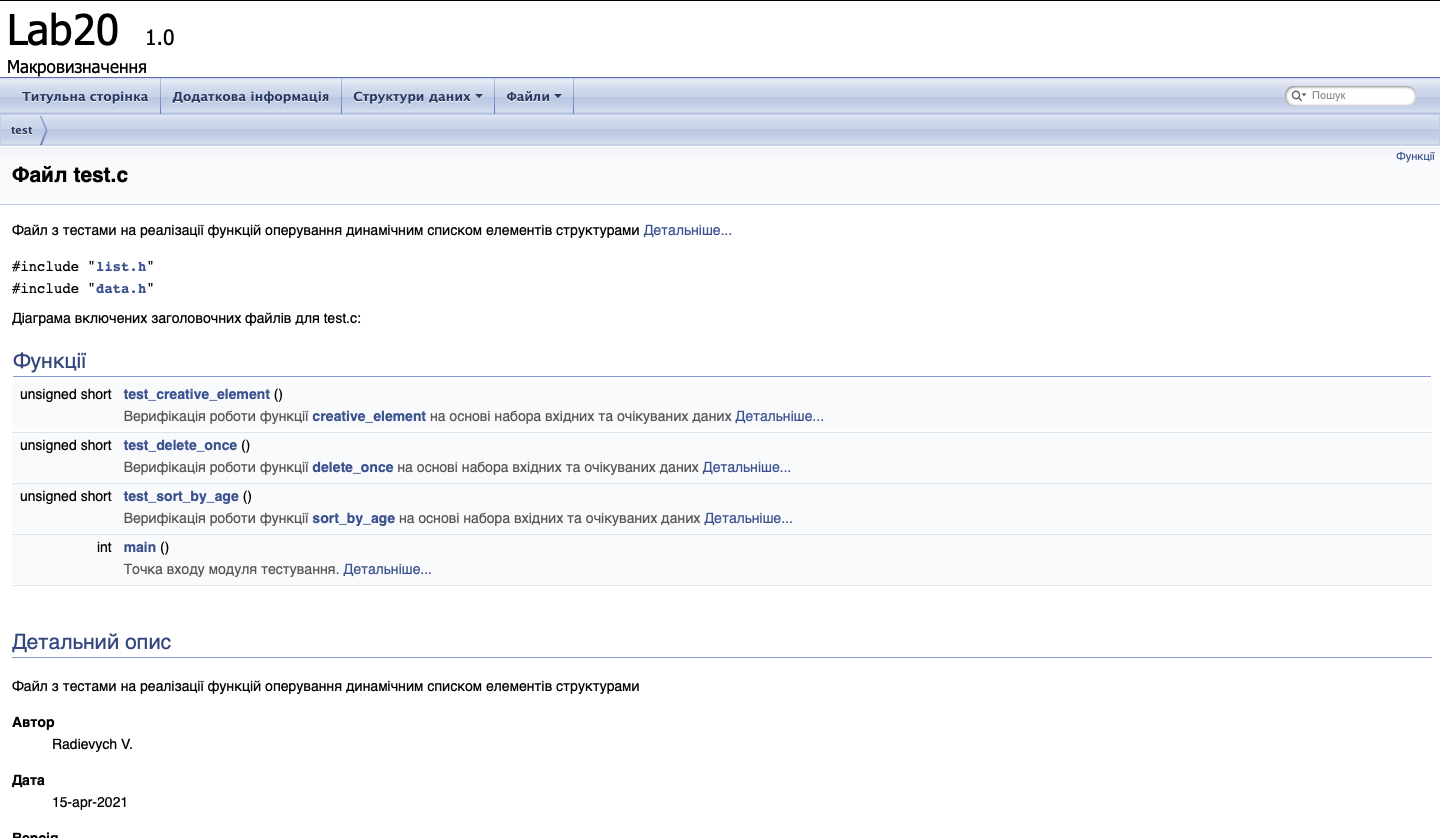


Рисунок 13 – робота з doxygen

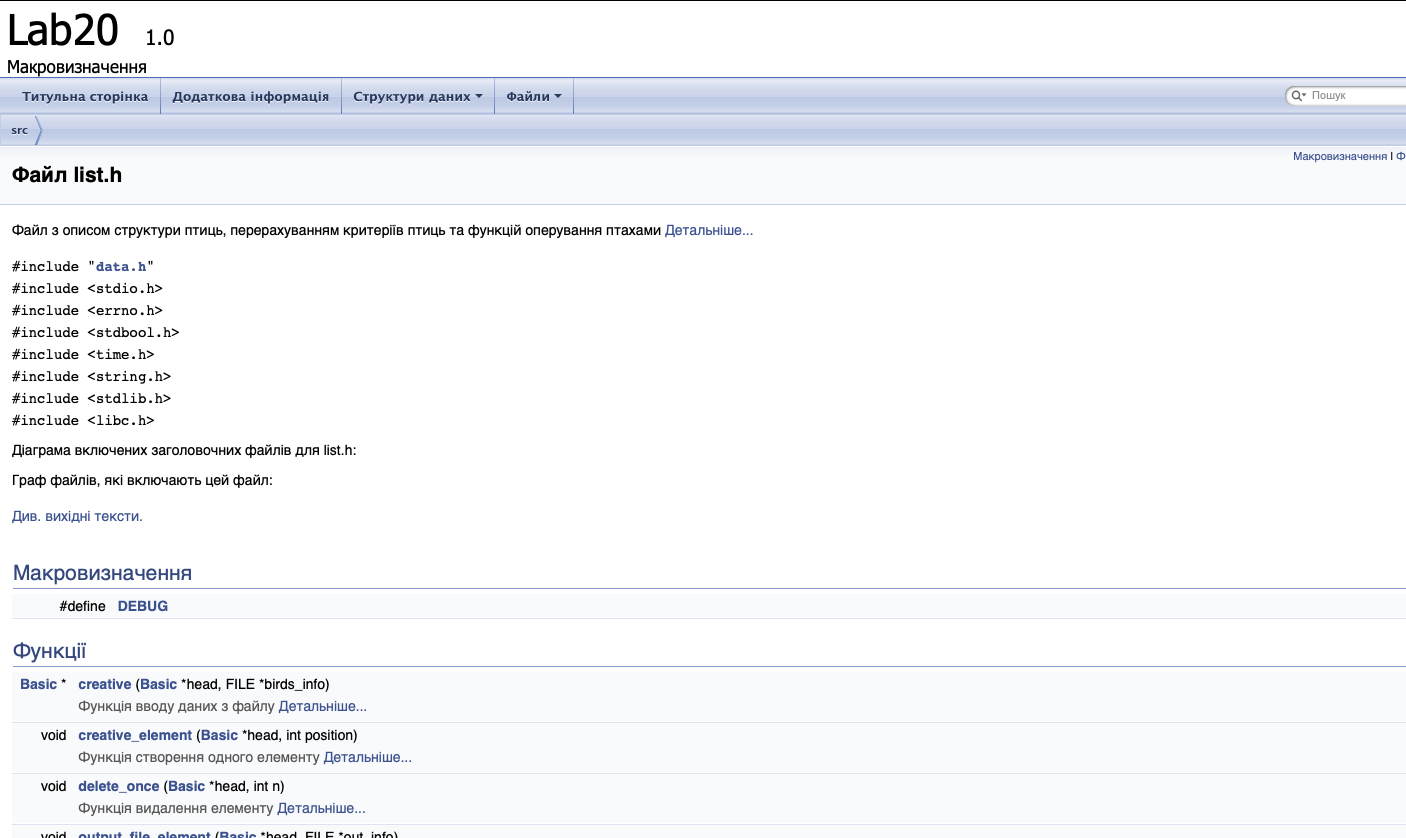


Рисунок 15 – робота з doxygen

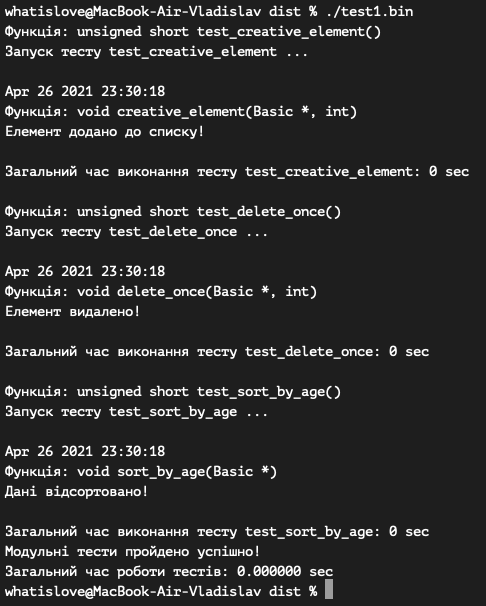


Рисунок 15 – робота з модульними тестами

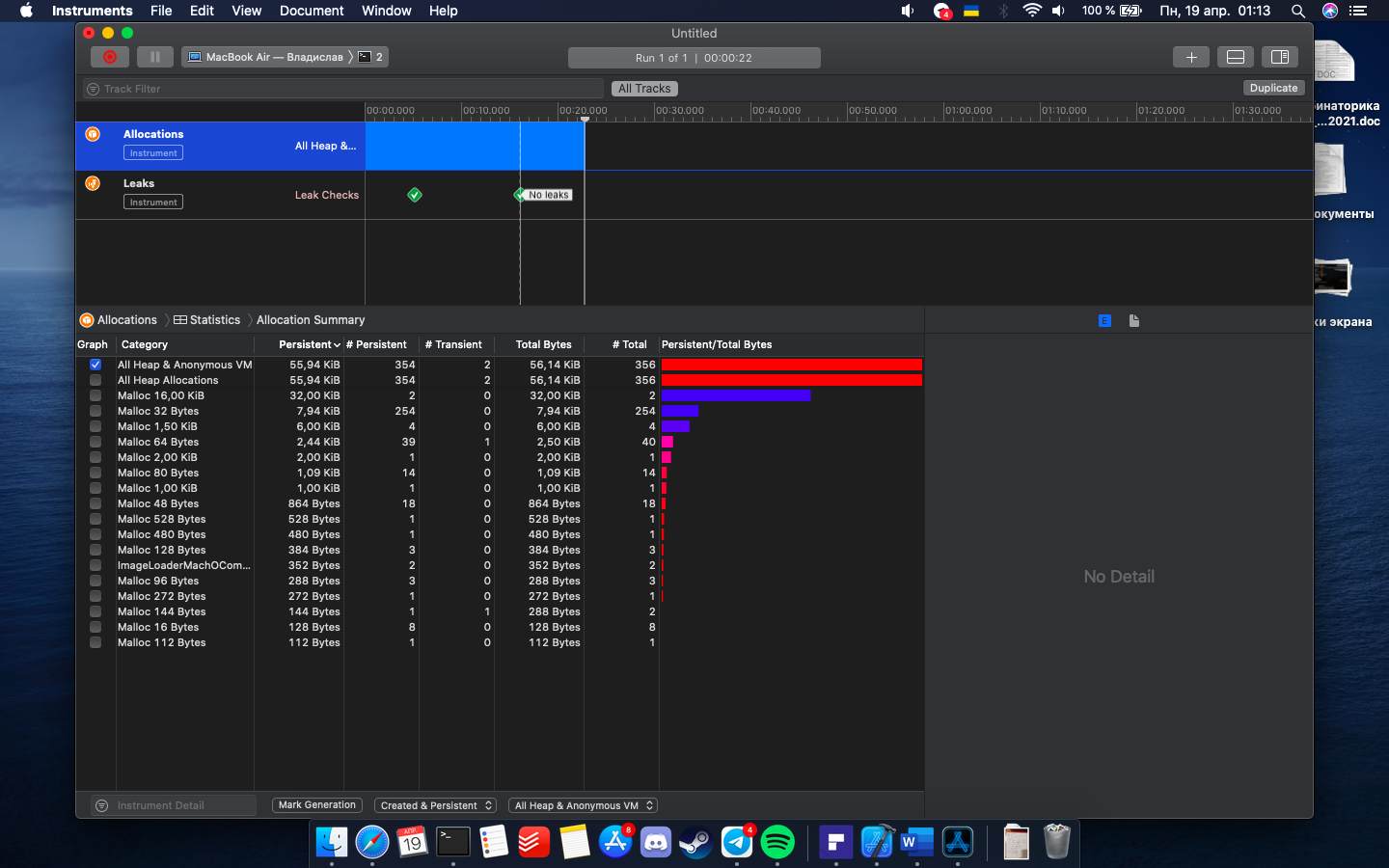


Рисунок 16 – демонстрація відсутності витоків пам’яті

1. Висновок

При виконанні даної лабораторної роботи я закріпив набуті мною навички, створення та взаємодію з динамічними масивами та динамічними структурами.

Посилання на GitHub, де знаходяться усі програми: <https://github.com/KotKHPI/Programming_Radievych>