Лабораторна робота № 26. ООП. Спадкування

1 ВИМОГИ

1.1 Розробник

- Радєвич Владислав Романович;
- студент групи КІТ – 320;
- 20.05.2021 р.

1.2 Загальне завдання

Поширити Модернізувати попередню лабораторну роботу шляхом: додавання класів-спадкоємцев з розділу "Розрахункове завдання / Індивідуальні завдання", котрі будуть поширювати функціонал "базового класу" відповідно до індивідуального завдання та додавання ще класу-списку для кожного класу-спадкоємцю, що буде керувати лише елементами стосовного класу-спадкоємця.

2 ОПИС ПРОГРАМИ

2.1 Функціональне призначення

Програма призначена для роботи з структури даних заданих птахів, використовуючи динамічні списки, використання заданих методів для роботи з класами та методами роботи з файлом.

2.2 Опис логічної структури

2.2.1 Основна функція

```
int main
```

Призначення: головна функція.

Схема алгоритму функції подана на рис. 1.

Опис роботи: демонструє роботу заданого динамічного списку елементів класу та методів оперування ним, методи роботи інших класів та вводом даних з файлу та виведення даних у файл.

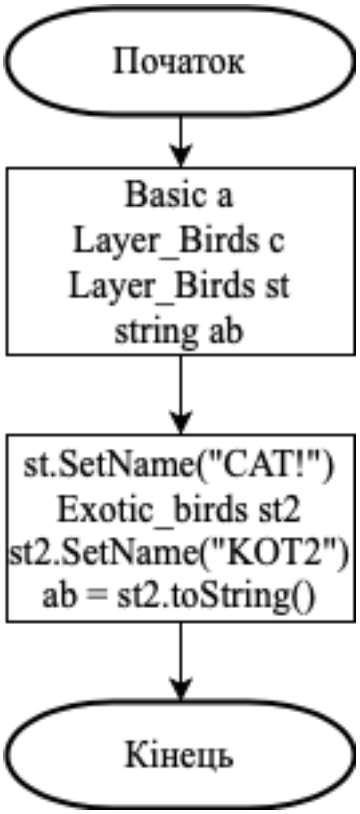

Рисунок 1 – Схема алгоритму функції main

2.2.1 Клас «список»

`class List`

Призначення: створення динамічного списку, в якому будуть міститися елементи базового класу.

Властивості класу:

`Basic** birds` – динамічний масив об'єктів базового класу;

`int count` – кількість об’єктів базового класу в масиві;

Методи класу:

`List(): count(0)` – конструктор за замовчуванням

`List(int count1)` - конструктор класу, виділяє певну кількість пам’яті для певної кількості елементів.

Параметри:

`int count1` – кількість об’єктів базового класу для яких буде виділено пам’ять у динамічному масиві.

Блок-схема показана на рисунку 2

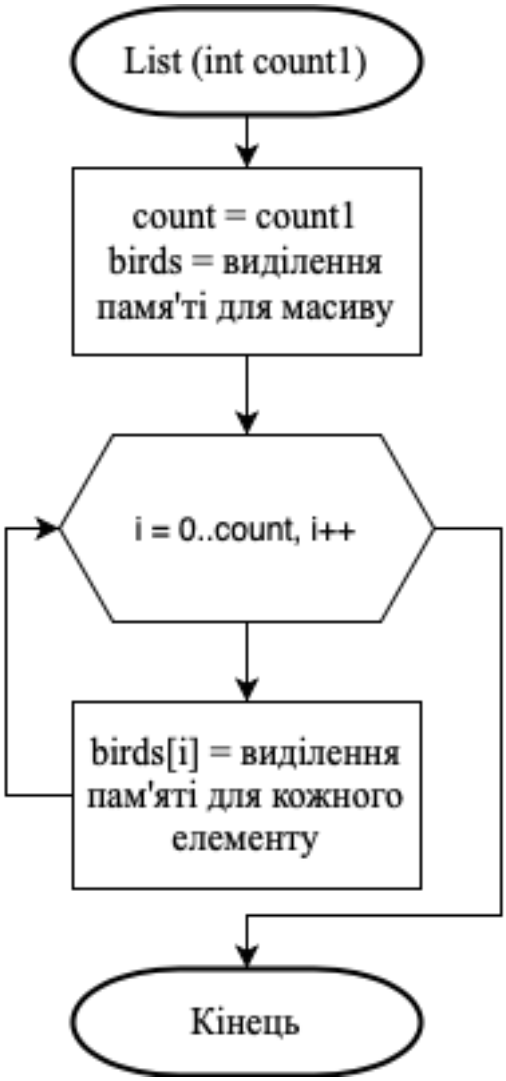

Рисунок 2 – блок-смеха конструктору List(int count1)

`void Paste(const Basic &other, int position)` – метод копіювання об’єкту базового класу на певну позицію в масиві.

Параметри:

`Basic &other` – об’єкт базового класу, що містить в собі інформацію про об’єкт;

`int position` – індекс, який буде присвоєно `Basic &other` у динамічному масиві;

`int GetCount() const` – метод отримання кількості об’єктів в масиві.

`Basic& GetBird (int index)`- метод отримання об’єкту масиву за його індексом.

Параметри:

`int index` – індекс елементу у динамічному масиві;

`void AddBird(Basic &other)` - метод додавання елементу базового класу в кінець масиву з об’єктами

Параметри:

`Basic &other` – об’єкт базового класу, який буде додано в кінець динамічного масиву;

Блок-схема показана на рисунку 3

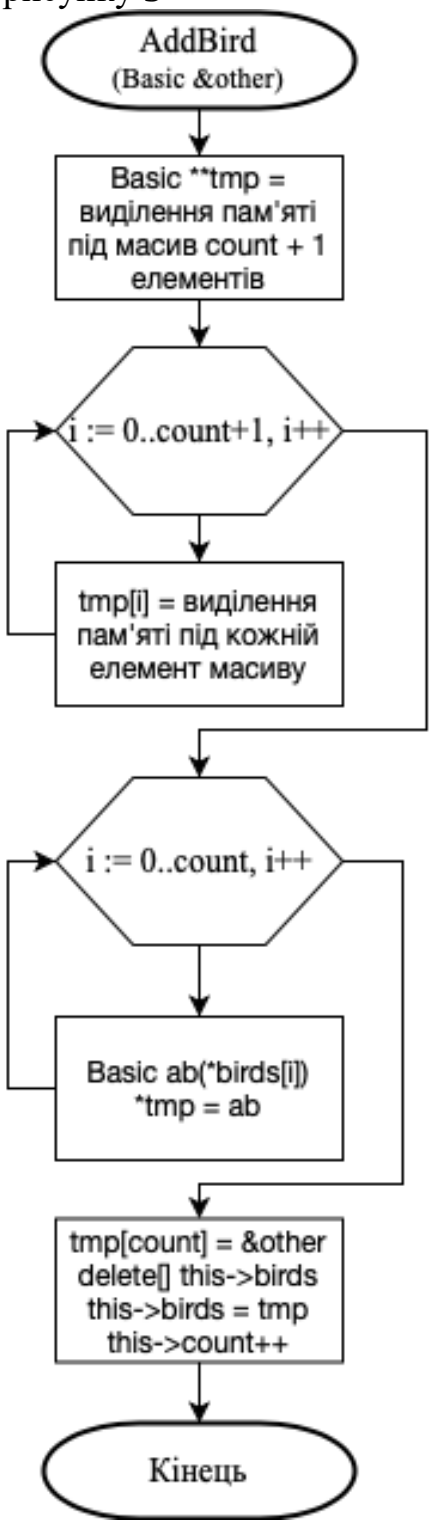

Рисунок 3 – блок-схема методу AddBird

`void RemoveBird(int index)` - метод видалення елементу базового класу, з масиву об’єктів, за його індексом

Параметри:

`int index` – індекс елементу в динамічному масиві, який буде видалено

Блок-схема показана на рисунку 4

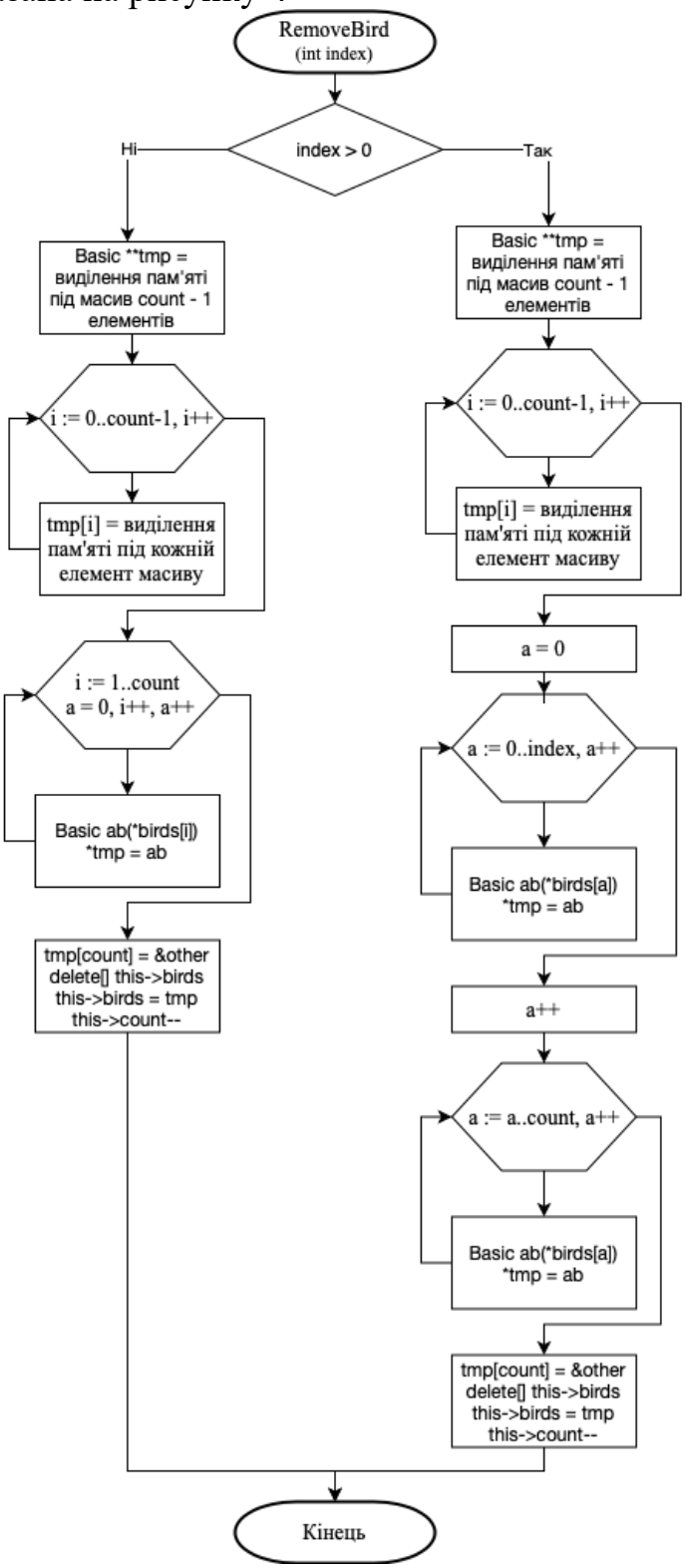

Рисунок 4 – блок-схема методу RemoveBird

`void ShowAll()` – метод виводу на екран усі елементів динамічного масиву об’єктів
Блок-схема показана на рисунку 5

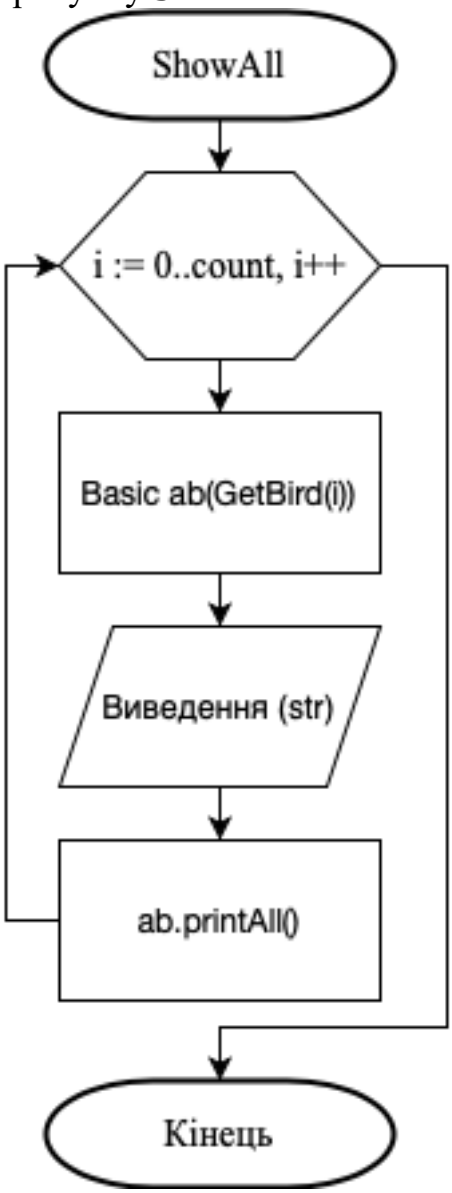

Рисунок 5 – блок-схема методу ShowAll

`void FindPercentage()` – метод знаходження відсоткового відношення чоловіків до жінок, де враховується усі елементи масиву.
Блок-схема показана на рисунку 6

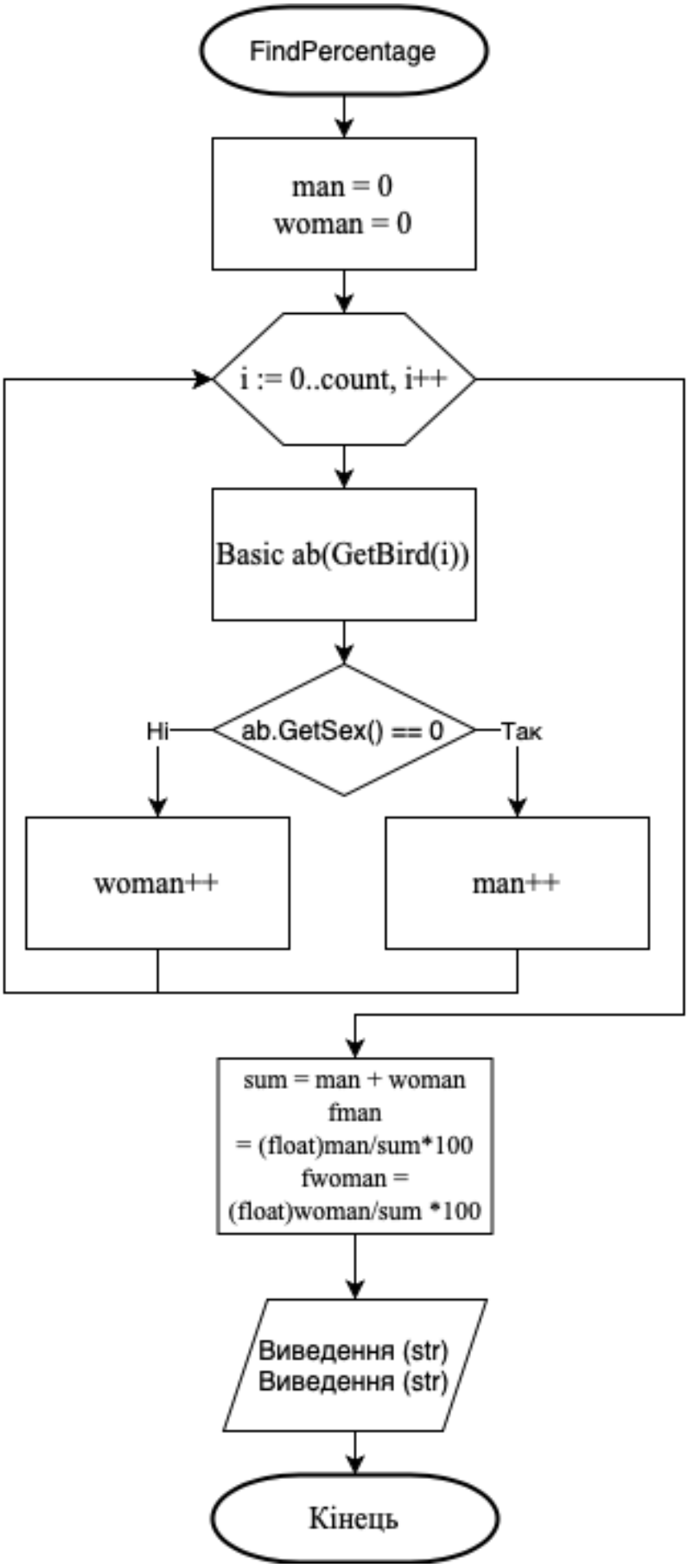

Рисунок 6 – блок-схема методу FindPercentage

`void readFromFile(std::string fileName)` – метод зчитування інформації про характеристики об’єкта з файлу.
Параметри:
`std::string fileName` – шлях до файлу з якого відбудеться зчитування;
Блок-схема показана на рисунку 7

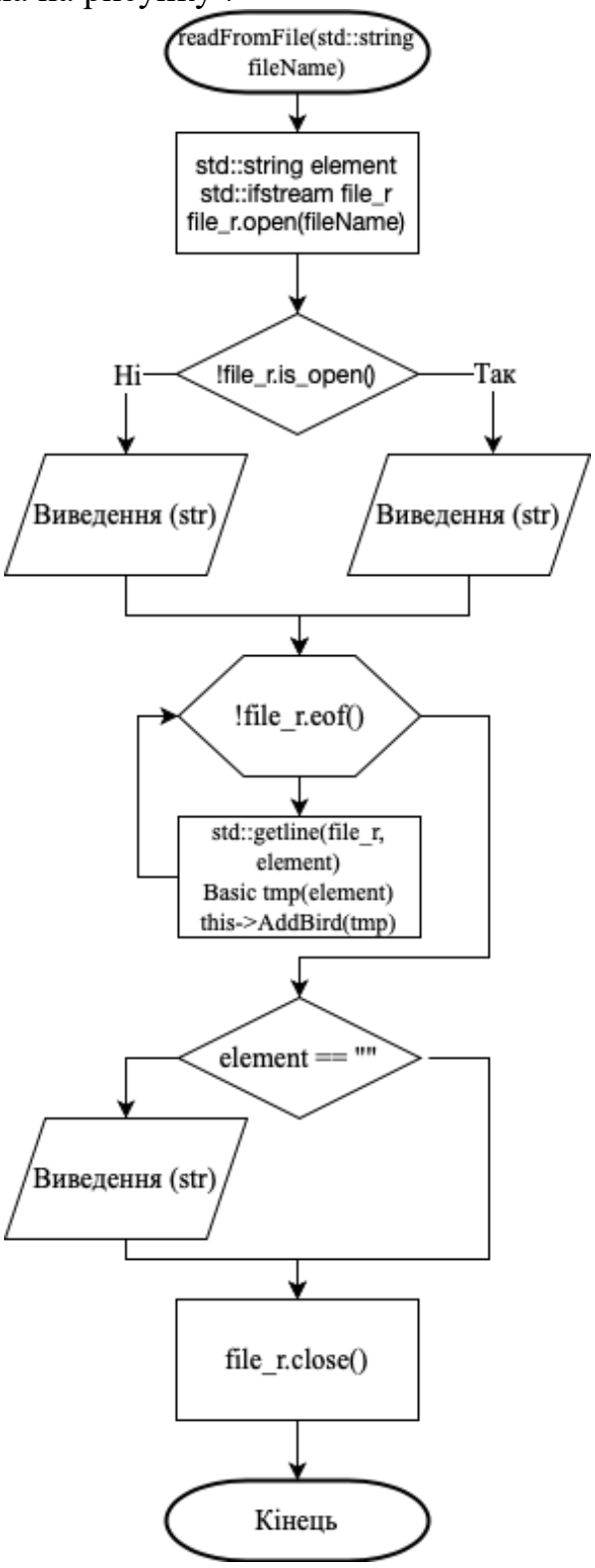

Рисунок 7 – блок-схема методу readFromFile

`void writeToFile(std::string fileName)` – метод записування інформації про характеристики об’єкта до файлу.

Параметри:

`std::string fileName` – шлях до файлу з якого відбудеться зчитування;

Блок-схема показана на рисунку 8

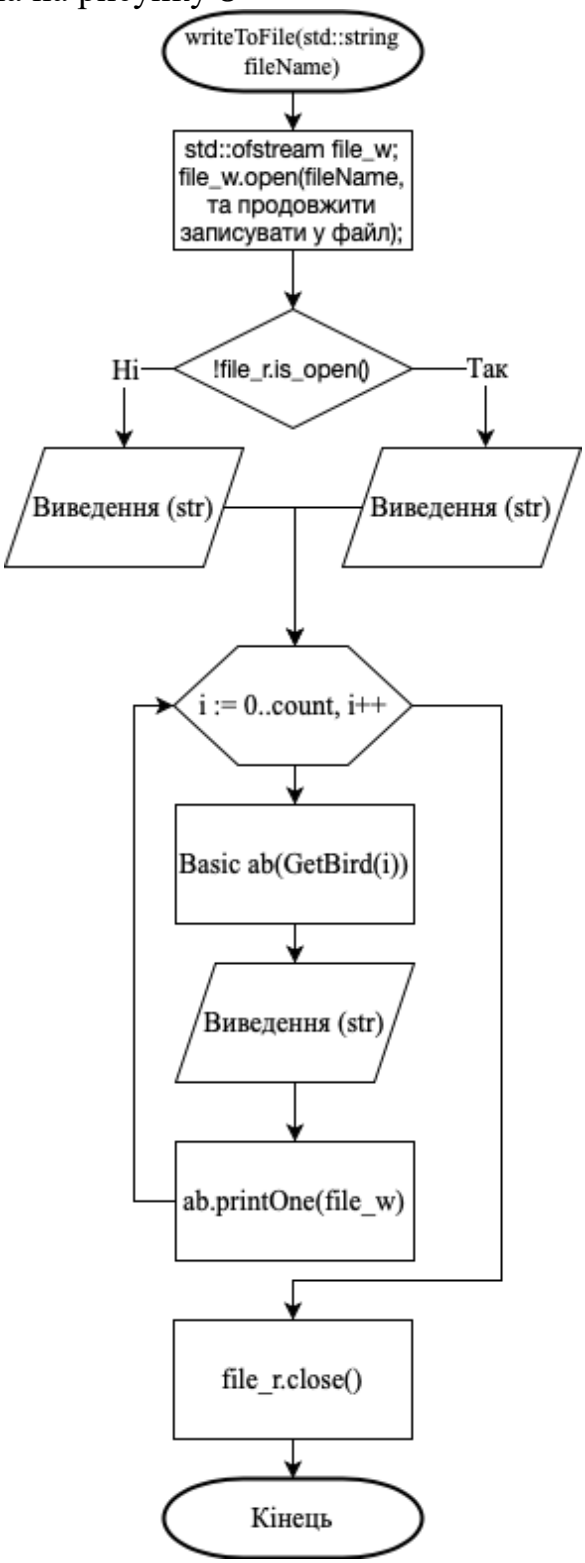

Рисунок 8 – блок-схема методу writeToFile

`virtual ~List()` – деструктор, який звільняє виділену пам’ять під динамічний масив об’єктів

2.2.2 Клас «список типу Layer_Birds»

`List_Layer_Birds`

Призначення: створення списку об’єктів, похідних від базового класу, типу Layer_Birds

Властивості:

`Layer_Birds** birds` – динамічний масив об'єктів похідного від базового класу, типом Layer_Birds;

`int count` – кількість об’єктів даного типу в масиві;

Методи класу:

`List_Layer_Birds(): count(0)` – конструктор за замовчуванням

`List_Layer_Birds (int count1)` - конструктор класу, виділяє певну кількість пам’яті для певної кількості елементів.

Параметри:

`int count1` – кількість об’єктів базового класу для яких буде виділено пам’ять у динамічному масиві.

Блок-схема показана на рисунку 2

`void Paste(const Basic &other, int position)` – метод копіювання об’єкту базового класу на певну позицію в масиві.

Параметри:

`Basic &other` – об’єкт базового класу, що містить в собі інформацію про об’єкт;

`int position` – індекс, який буде присвоєно `Basic &other` у динамічному масиві;

`int GetCount() const` – метод отримання кількості об’єктів в масиві.

`Basic& GetBird (int index)`- метод отримання об’єкту масиву за його індексом.

Параметри:

`int index` – індекс елементу у динамічному масиві;

`void AddBird(Basic &other)`- метод додавання елементу базового класу в кінець масиву з об’єктами

Параметри:

`Basic &other` – об’єкт базового класу, який буде додано в кінець динамічного масиву;
Блок-схема показана на рисунку 3

`void RemoveBird(int index)`- метод видалення елементу базового класу, з масиву об’єктів, за його індексом
Параметри:
`int index` – індекс елементу в динамічному масиві, який буде видалено
Блок-схема показана на рисунку 4

`void ShowAll()` – метод виводу на екран усі елементів динамічного масиву об’єктів
Блок-схема показана на рисунку 5

`void FindPercentage()` – метод знаходження відсоткового відношення чоловіків до жінок, де враховується усі елементи масиву.
Блок-схема показана на рисунку 6

`void readFromFile(std::string fileName)` – метод зчитування інформації про характеристики об’єкта з файлу.
Параметри:
`std::string fileName` – шлях до файлу з якого відбудеться зчитування;
Блок-схема показана на рисунку 7

`void writeToFile(std::string fileName)` – метод записування інформації про характеристики об’єкта до файлу.
Параметри:
`std::string fileName` – шлях до файлу з якого відбудеться зчитування;
Блок-схема показана на рисунку 8

`virtual ~List()` – деструктор, який звільняє виділену пам’ять під динамічний масив об’єктів

2.2.3 Клас «список типу Layer_Birds»

`List_Layer_Birds`

Призначення: створення списку об’єктів, похідних від базового класу, типу Layer_Birds

Властивості:

`Layer_Birds** birds` – динамічний масив об'єктів похідного від базового класу, типом Layer_Birds;

`int count` – кількість об’єктів даного типу в масиві;

Методи класу:

`List_Layer_Birds(): count(0)` – конструктор за замовчуванням

`List_Layer_Birds (int count1)` - конструктор класу, виділяє певну кількість пам’яті для певної кількості елементів.

Параметри:

`int count1` – кількість об’єктів базового класу для яких буде виділено пам’ять у динамічному масиві.

Блок-схема показана на рисунку 2

`void Paste(const Layer_Birds &other, int position)` – метод копіювання об’єкту базового класу на певну позицію в масиві.

Параметри:

`Layer_Birds &other` – об’єкт, що містить в собі інформацію про об’єкт;

`int position` – індекс, який буде присвоєно `Layer_Birds &other` у динамічному масиві;

`int GetCount() const` – метод отримання кількості об’єктів в масиві.

`Basic& GetBird (int index)`- метод отримання об’єкту масиву за його індексом.

Параметри:

`int index` – індекс елементу у динамічному масиві;

`void AddBird(Layer_Birds &other)`- метод додавання елементу базового класу в кінець масиву з об’єктами

Параметри:

`Layer_Birds &other` – об’єкт, який буде додано в кінець динамічного масиву;

Блок-схема показана на рисунку 3

`void RemoveLayer_Birds(int index)`- метод видалення елементу базового класу, з масиву об’єктів, за його індексом

Параметри:

`int index` – індекс елементу в динамічному масиві, який буде видалено

Блок-схема показана на рисунку 4

`void ShowAll()` – метод виводу на екран усі елементів динамічного масиву об’єктів
Блок-схема показана на рисунку 5

`void FindPercentage()` – метод знаходження відсоткового відношення чоловіків до жінок, де враховується усі елементи масиву.
Блок-схема показана на рисунку 6

`void readFromFile(std::string fileName)` – метод зчитування інформації про характеристики об’єкта з файлу.
Параметри:
`std::string fileName` – шлях до файлу з якого відбудеться зчитування;
Блок-схема показана на рисунку 7

`void writeToFile(std::string fileName)` – метод записування інформації про характеристики об’єкта до файлу.
Параметри:
`std::string fileName` – шлях до файлу з якого почнеться зчитування;
Блок-схема показана на рисунку 8

`virtual ~List()` – деструктор, який звільняє виділену пам’ять під динамічний масив об’єктів

2.2.4 Клас «список типу Exotic_birds»

`List_Layer_Birds`

Призначення: створення списку об’єктів, похідних від базового класу, типу Exotic_birds

Властивості:

`Exotic_birds** birds` – динамічний масив об'єктів похідного від базового класу, тип Exotic_birds;

`int count` – кількість об’єктів даного типу в масиві;

Методи класу:

`List_Exotic_birds(): count(0)` – конструктор за замовчуванням
`List_Exotic_birds (int count1)` - конструктор класу, виділяє певну кількість пам’яті для певної кількості елементів.
Параметри:

`int count1` – кількість об’єктів базового класу для яких буде виділено пам’ять у динамічному масиві.
Блок-схема показана на рисунку 2

`void Paste(const Exotic_birds &other, int position)` – метод копіювання об’єкту типу Exotic_birds на певну позицію в масиві.
Параметри:
`Exotic_birds &other` – об’єкт, що містить в собі інформацію про об’єкт;
`int position` – індекс, який буде присвоєно `Exotic_birds &other` у динамічному масиві;
`int GetCount() const` – метод отримання кількості об’єктів в масиві.

`Exotic_birds& GetBird (int index)`- метод отримання об’єкту масиву за його індексом.
Параметри:
`int index` – індекс елементу у динамічному масиві;

`void AddBird(Exotic_birds &other)`- метод додавання елементу похідного від базового класу, а саму типу Exotic_birds, в кінець масиву з об’єктами
Параметри:
`Exotic_birds &other` – об’єкт, який буде додано в кінець динамічного масиву;
Блок-схема показана на рисунку 3

`void RemoveExotic_birds (int index)`- метод видалення елементу похідного від базового класу, а саму типу Exotic_birds, з масиву об’єктів, за його індексом
Параметри:
`int index` – індекс елементу в динамічному масиві, який буде видалено
Блок-схема показана на рисунку 4

`void ShowAll()` – метод виводу на екран усі елементів динамічного масиву об’єктів
Блок-схема показана на рисунку 5

`void FindPercentage()` – метод знаходження відсоткового відношення чоловіків до жінок, де враховується усі елементи масиву.
Блок-схема показана на рисунку 6

`void readFromFile(std::string fileName)` — метод зчитування інформації про характеристики об’єкта з файлу.
Параметри:
`std::string fileName` — шлях до файлу з якого почнеться зчитування;
Блок-схема показана на рисунку 7

`void writeToFile(std::string fileName)` — метод записування інформації про характеристики об’єкта до файлу.
Параметри:
`std::string fileName` — шлях до файлу з якого відбудеться зчитування;
Блок-схема показана на рисунку 8

`virtual ~List()` — деструктор, який звільняє виділену пам’ять під динамічний масив об’єктів

2.2.5 Клас «базовий»

`List Basic`

Призначення: створення об’єкту базового класу, який відповідає критеріям птаха.

Властивості класу:

`enum Yes_no label` - чи окільцьована птаха;
`std::string name` — назва виду птаха;
`int age` - вік птаха (в місяцях);
`Feature home` — клас характеристик домівки для птаха;
`enum Sex sex` — стать птаха;

Методи класу:

`Basic(): label(Так), name("Птиця"), age(0), sex(Чоловіча)` — конструктор за замовчуванням

`Basic(Yes_no label1, std::string name1, int age1, Feature home1, Sex sex1)` — конструктор класу, який приймає інформація про об’єкт
Параметри:
`Yes_no label1` - чи окільцьована птаха;
`std::string name1` — назва виду птаха;
`int age1` — вік птаха
`Feature home1` - клас характеристик житла для птаха;
`Sex sex1` — стать птаха;

`void SetLabel (Yes_no x)` – метод присвоєння значення чи окільцьована птаха чи ні
Параметри:
`Yes_no x` – чи окільцьована птаха;

`Yes_no GetLabel() const` – отримання значення чи окільцьована птаха

`void SetName (std::string n)` – метод присвоєння об'єктові, як називається вид птаха
Параметри:
`char n[15]` – назва виду птаха

`void SetSex (Sex x) const` – встановлення значення статі птаха
Параметри:

`Sex x` – стать птаха;

`Sex GetSex() const` - отримання значення статі птаха

`void SetAge(int x)` – встановлення значення віку птаха
Параметри:

`int x` – вік птаха;

`int GetAge()const` – отримання значення віку птаха

`Feature GetHome()const` – отримання значення характеристик дому птаха

`void printOne(std::ofstream &file)` – метод виведення інформація про об'єкт базового класу
Параметри:
`std::ofstream &file` – файл в який буде записно інформацію про об'єкт базового класу
Блок-схема показана на рисунку 11

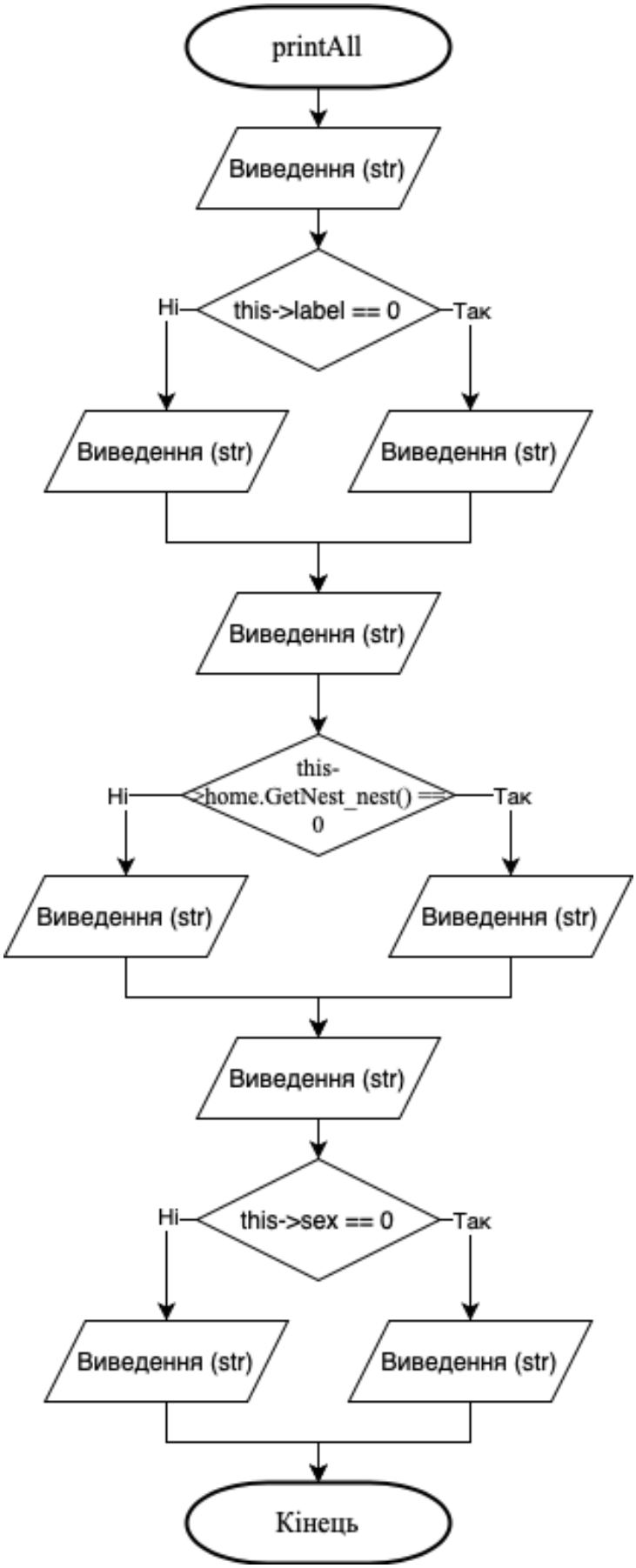

Рисунок 11 – блок-схема методу printOne

`Basic (std::string tmp)` – конструктор який приймає інформацію про об’єкт базового класу через строку
Параметри:
`std::string tmp` – строка в якій знаходиться інформація про об’єкт;
Блок-схема показана на рисунку 12

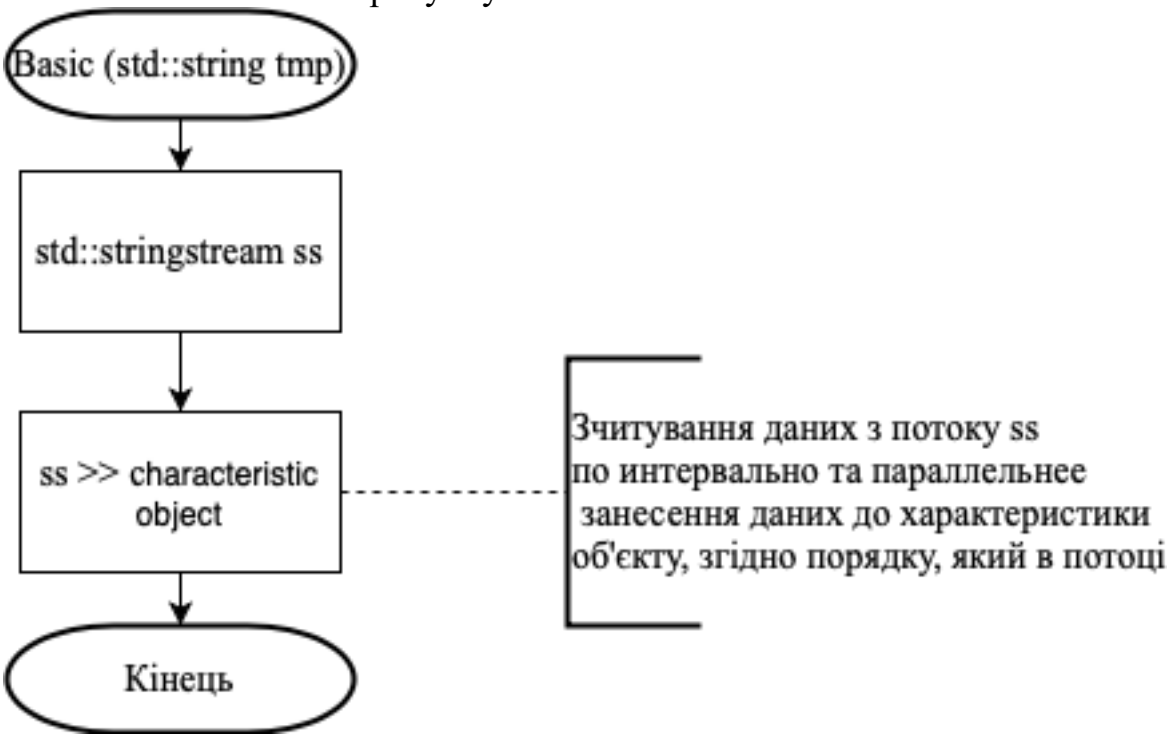

Рисунок 12 – блок схема конструктору зі строкою

`std::string toString()` – метод який об’єкт базового класу перетворює на строку, в який буде інформація про цей об’єкт
Блок-схема показана на рисунку 13

Рисунок 13 – блок схема методу toString

`Basic (const Basic& other)` – конструктор копіювання інших об’єктів базового класу

Параметри:

`const Basic& other` – об’єкт базового класу;

`virtual ~Basic()` – деструктор базового класу

2.2.6 Клас «домівка птаха»

`class Feature`

Призначення: створення об’єкту, який відповідає характеристиками домівки птаха.

Властивості класу:

`int square` – площа домівки (в см^2);
`int height` – висота домівки;
`int number_of_feeders` – кількість годівниць;
`enum Yes_no nest_nest` – наявність гнізда;

Методи класу:

`Feature(): square(0), height(0), number_of_feeders(0), nest_nest(Так)` – конструктор за замовчуванням

`Feature(int square1, int height1, int number_of_feeders1, Yes_no nest_nest1)` - конструктор класу, який приймає інформація про об’єкт домівку

Параметри:

`int square1` – площа домівки (в см^2);
`int height1` – висота домівки;
`int number_of_feeders1` – кількість годівниць;
`enum Yes_no nest_nest1` – наявність гнізда;

`void SetSquare (int x)` – встановлення значення віку птаха

Параметри:

`int x` – площа домівки;

`int GetHeight ()const` – отримання значення площі домівки птаха

`void SetHeight (int x)` – встановлення висоти домівки птаха

Параметри:

`int x` – площа домівки;

`int GetHeight ()const` – отримання значення висоти домівки птаха

`void SetNumber_of_feeders (int x)` – встановлення кількості годівниць для птаха

Параметри:

`int x` –кількість годівниць;

`int GetNumber_of_feeders ()const` – отримання значення кількості годівниць для птаха

`void SetNest_nest (int x)` – встановлення значення наявності гнізда

Параметри:

`Yes_no x` – наявність гнізда;

`int GetNest_nest ()const` – отримання значення наявності гнізда

`Feature (const Feature &other)` – конструктор копіювання

Параметри:

`Feature &other` – об'єкт класу домівка птаха;

`virtual ~ Feature ()` – деструктор класу домівка птаха

2.2.7 Клас «перелітні птахи»

`Layer_Birds`

Призначення: створення об'єкту, який є спадкоємцем базового класу, який відповідає критеріям, окрім базового класу, ще місяцем прильоту та відльоту птаха.

Властивості класу:

`enum Month take_off` – місяць, коли птах відлітає у вирій;
`enum Month rifle` – місяць, коли птах прилітає з вирію;

Методи класу:

`Layer_Birds(): Basic(), take_off(Січень), rifle(Січень)` – конструктор за замовчуванням

`Layer_Birds(Basic &other, enum Month take_off, enum Month refle): Basic(other), take_off(take_off), rifle(refle)` – конструктор класу, який приймає інформація про об'єкт

Параметри:

`Basic &other -` об'єкт базового класу, який має інформацію про себе;

`enum Month take_off` – місяць, коли птах відлітає у вирій;
`enum Month rifle` – місяць, коли птах прилітає з вирію;

`void SetTake_off(enum Month take_off)` – метод присвоєння значення місяць, в який птах відлітає;

Параметри:

`enum Month take_off` – місяць, коли птах відлітає у вирій;

`Month GetTake_off() const` – отримання значення місяць, в який птах відлітає;

`void SetRifle(enum Month rifle)` – метод присвоєння об’єктові місяць в який птах прилітає;

Параметри:

`enum Month rifle` – місяць, коли птах прилітає з вирію;

`Month GetRifle() const` - отримання значення місяця, коли птах прилітає;

`void printOne(std::ofstream &file)` – метод виведення інформація про об’єкт похідного класу від базового класу, а саме Layer_Birds;

Параметри:

`std::ofstream &file` – файл в який буде записно інформацію про об’єкт об’єкт похідного класу від базового класу, а саме Layer_Birds;

Блок-схема показана на рисунку 11

`Layer_Birds (std::string tmp)` – конструктор який приймає інформацію про об’єкт похідного класу від базового класу, а саме Layer_Birds, через строку

Параметри:

`std::string tmp` – строка в якій знаходиться інформація про об’єкт;

Блок-схема показана на рисунку 12

`std::string toString()` – метод, який перетворює об’єкт похідного класу від базового класу, а саме Layer_Birds, на строку, в який буде інформація про цей об’єкт

Блок-схема показана на рисунку 13

`virtual ~Layer_Birds()` – деструктор об’єкт похідного класу від базового класу, а саме Layer_Birds;

2.2.8 Клас «екзотичні птахи»

`Exotic_birds`

Призначення: створення об’єкту, який є спадкоємцем базового класу, який відповідає критеріям, окрім базового класу, ще максимально і мінімально комфортною температурою для птаха.

Властивості класу:

`int max_temp` – максимально комфортна температура для птаха;
`int min_temp` – мінімально комфортна температура для птаха;

Методи класу:

`Exotic_birds(): max_temp(0), min_temp(0)` – конструктор за замовчуванням

`Exotic_birds(Basic &other, int min_temp, int max_temp): Basic(other), min_temp(min_temp), max_temp(max_temp)` – конструктор класу, який приймає інформація про об’єкт
Параметри:
`Basic &other` - об’єкт базового класу, який має інформацію про себе;
`int max_temp` – максимально комфортна температура для птаха;
`int min_temp` – мінімально комфортна температура для птаха

`void SetMin_temp(int min_temp)` – метод присвоєння значення мінімальної температури для птаха;
Параметри:
`int min_temp` – мінімально комфортна температура для птаха;

`int GetMin_temp() const` – отримання значення мінімальної температури для птаха;

`void SetMax_temp(int max_temp)` – метод присвоєння об’єктові максимальної температури для птаха;
Параметри:
`int max_temp` – максимально комфортна температура для птаха;

`int GetMax_temp() const` - отримання значення місяця максимальної температури для птаха;

`void printOne(std::ofstream &file)` – метод виведення інформація про об’єкт похідного класу від базового класу, а саме Exotic_birds;
Параметри:
`std::ofstream &file` – файл в який буде записно інформацію про об’єкт об’єкт похідного класу від базового класу, а саме Exotic_birds;

Блок-схема показана на рисунку 11

`Exotic_birds (std::string tmp)` – конструктор який приймає інформацію про об’єкт похідного класу від базового класу, а саме Exotic_birds, через строку
Параметри:
`std::string tmp` – строка в якій знаходиться інформація про об’єкт;
Блок-схема показана на рисунку 12

`std::string toString()` – метод, який перетворює об’єкт похідного класу від базового класу, а саме Exotic_birds, на строку, в який буде інформація про цей об’єкт
Блок-схема показана на рисунку 13

`virtual ~Exotic_birds()` – деструктор об’єкт похідного класу від базового класу, а саме Exotic_birds;

2.3 Структура проекту

```
.
├── Doxyfile
├── Makefile
├── README.md
├── doc
│   ├── Radievych26.pdf
│   └── assets
│       ├── main-2.drawio
│       └── main.png
├── src
│   ├── data.cpp
│   ├── data.h
│   ├── list.cpp
│   ├── list.h
│   └── main.cpp
└── test
    └── test.cpp
```

3 ВАРІАНТИ ВИКОРИСТАННЯ

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

Результат роботи з doxygen продемонстровано на рисунку 14, рисунку 15 та рисунку 16, виконання модульних тестів на рисунку 17 та рисунок 18 – демонстрація відсутності витоків пам'яті

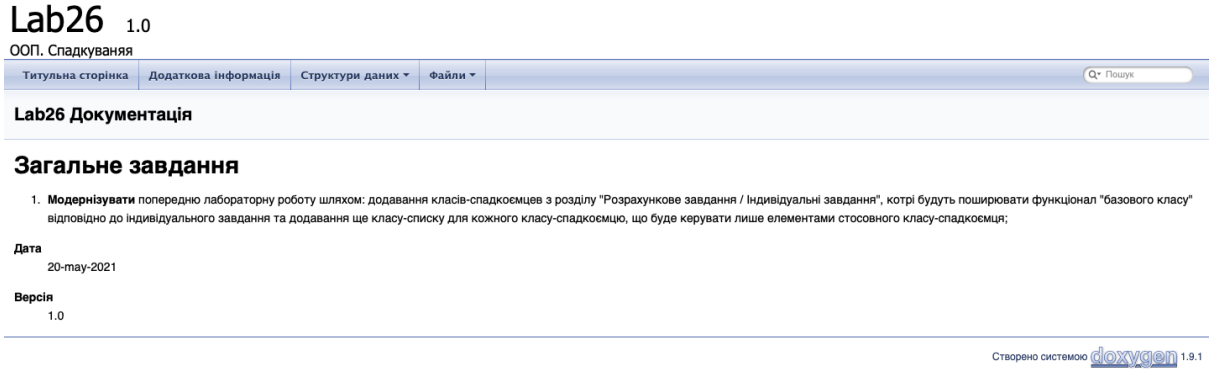

Рисунок 14 – робота з doxygen

Рисунок 15 – робота з doxygen

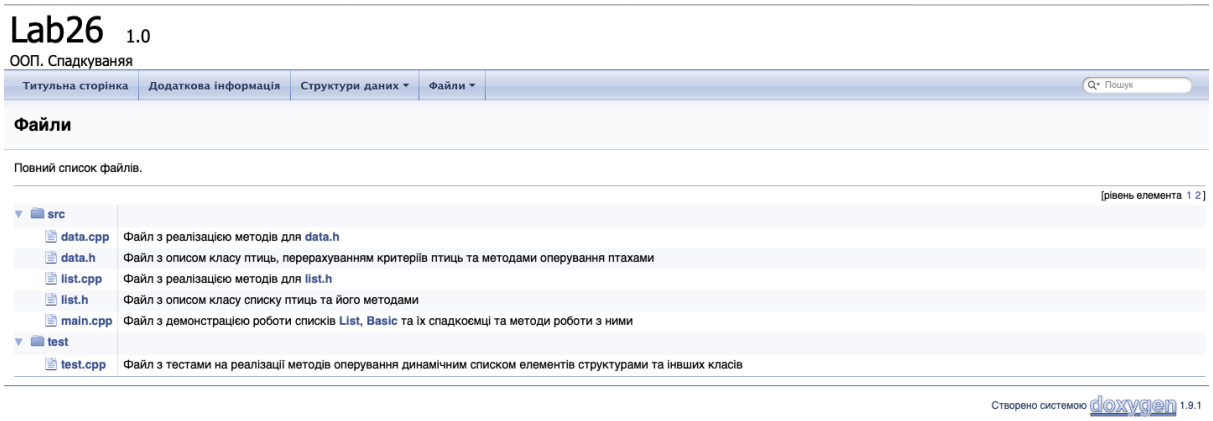

Рисунок 16 – робота з doxygen

```
whatislove@MacBook-Air-Vladislav dist % ./test1.bin
Запуск тесту test_AddLayer_bird ...
Запуск тесту test_AddExotic_bird ...
Запуск тесту test_RemoveLayer_bird ...
Запуск тесту test_RemoveExotic_birds ...
Запуск тесту test_Layer_bird_by_string ...
Запуск тесту test_Exotic_birds_by_string ...
Запуск тесту test_Layer_bird_to_string ...
Запуск тесту test_Exotic_birds_to_string ...
Модульні тести пройдено успішно!
whatislove@MacBook-Air-Vladislav dist %
```

Рисунок 17 – робота з модульними тестами

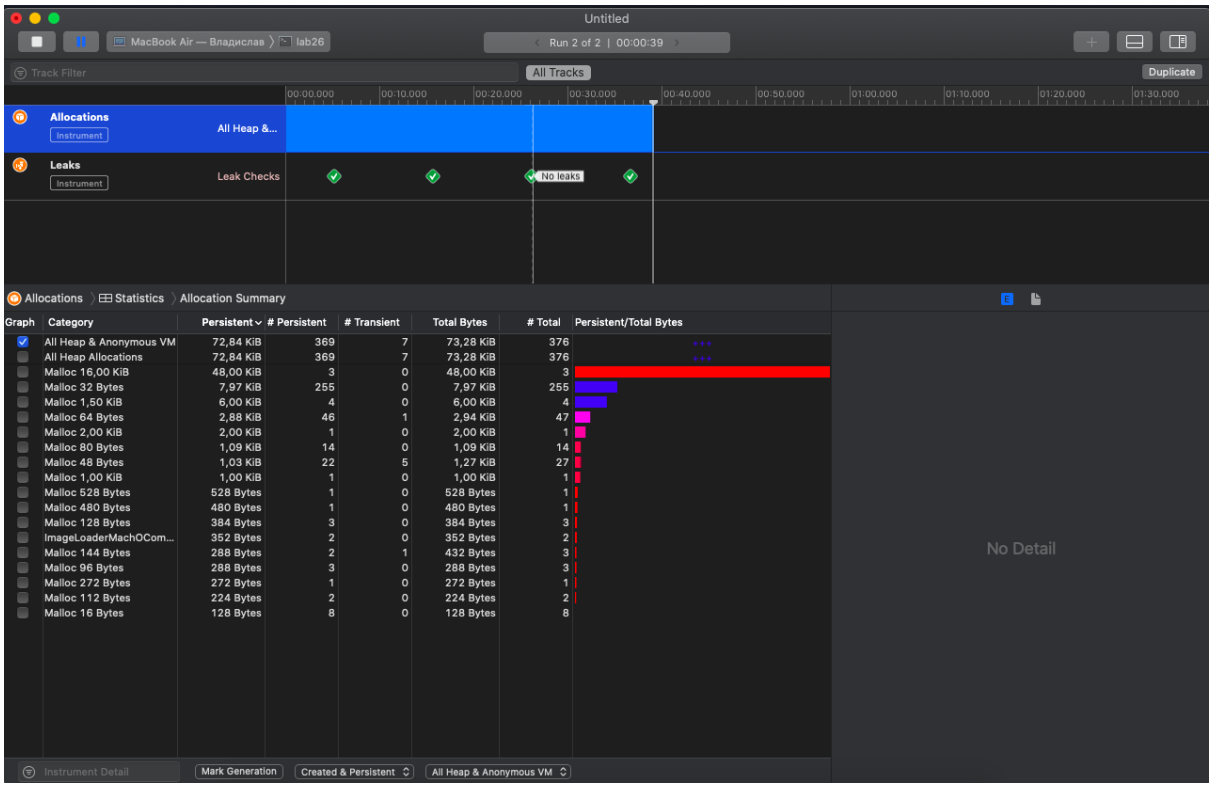

Рисунок 18 - демонстрація відсутності витоків пам'яті

4 ВИСНОВОК

При виконанні даної лабораторної роботи я закріпив набуті мною навички та ознайомився з принципами ООП.

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