Лабораторна робота № 27. ООП. Поліморфізм

1 ВИМОГИ

1.1 Розробник

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

1.2 Загальне завдання

Модернізувати попередню лабораторну роботу шляхом: базовий клас зробити абстрактним. Додати абстрактні методи; розроблені класи-списки поєднуються до одного класу таким чином, щоб він міг працювати як з базовим класом, так і з його спадкоємцями. При цьому серед полів класу-списку повинен бути лише один масив, що містить усі типи класів ієрархії. Оновити методи, що працюють з цим масивом; у функціях базового класу та класів-спадкоємців обов'язкове використання ключових final та override.

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.

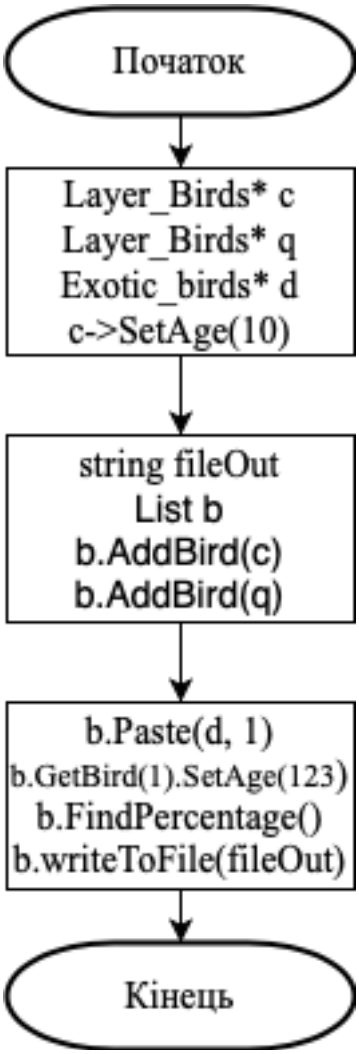

Рисунок 1 – Схема алгоритму функції main

2.2.1 Клас «список»

`class List`

Призначення: створення динамічного списку, в якому будуть міститися елементи базового класу та похідних від нього класів.

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 замовчування.

`void Paste(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 або його нащадка, який буде додано в кінець динамічного масиву;
Блок-схема показана на рисунку 2

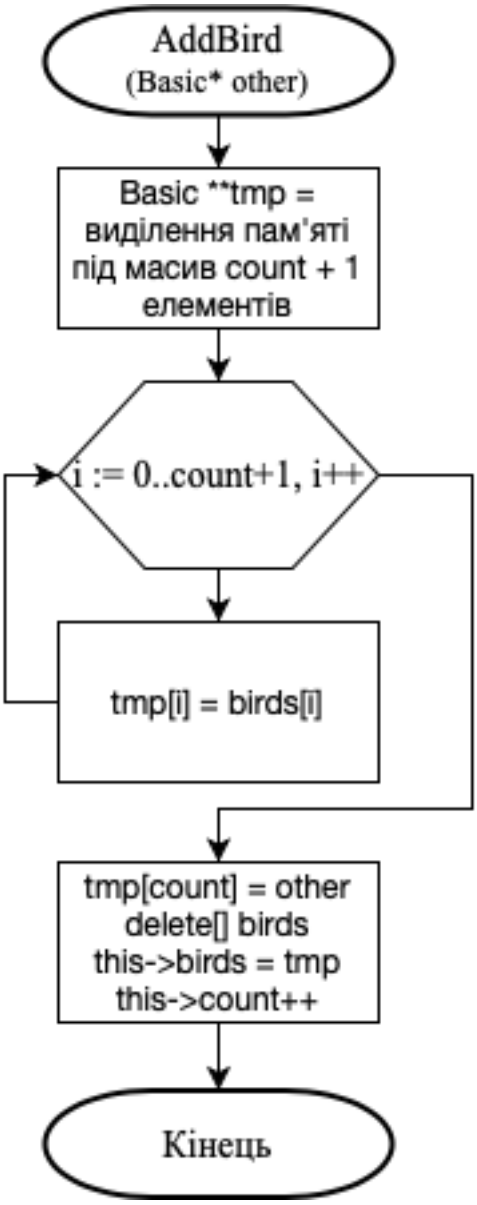

Рисунок 2 – блок-схема методу AddBird

`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 3

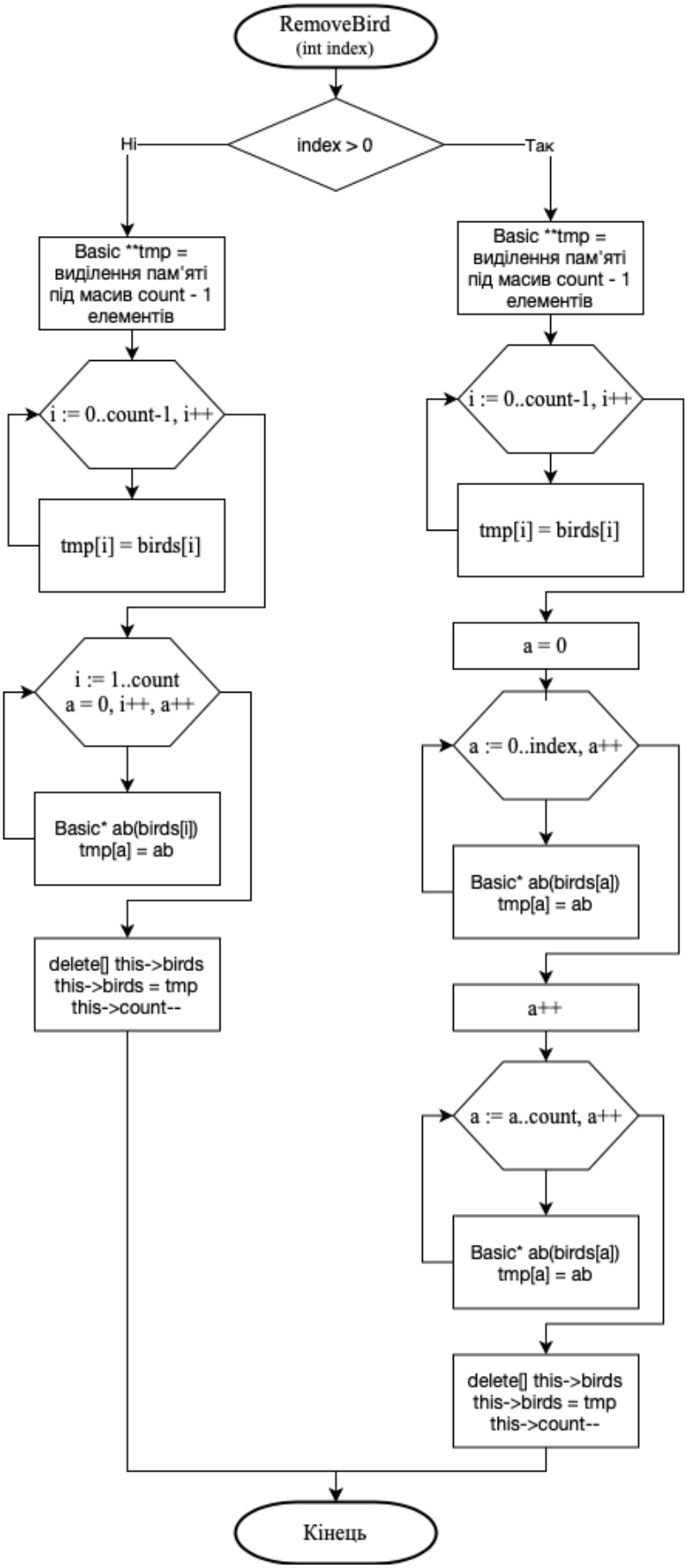

Рисунок 3 – блок-схема методу RemoveBird

`int FindPercentageMan()` – метод знаходження відсоткового відношення чоловіків до жінок, де враховується усі елементи масиву та повертає відсоток птахі чоловічої статі серед усіх.
Блок-схема показана на рисунку 4

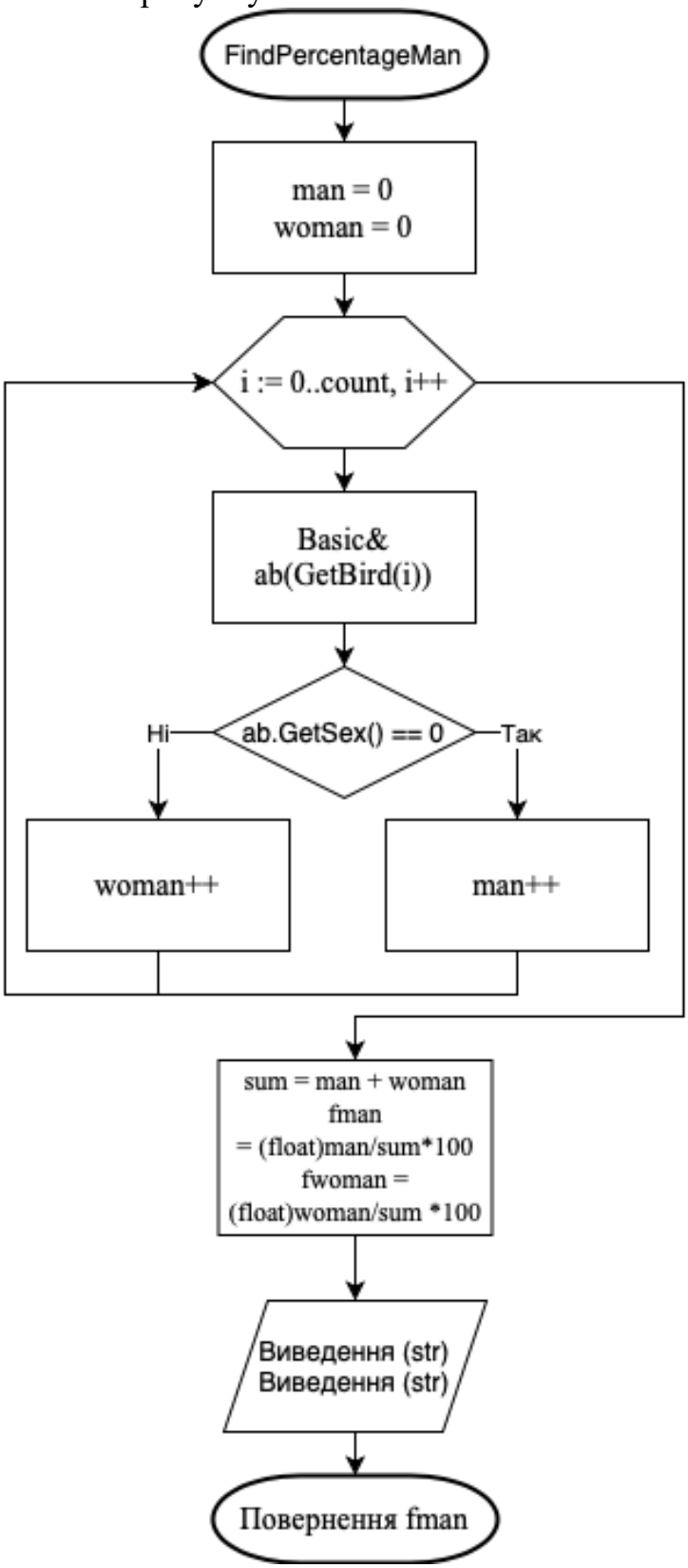

Рисунок 4 – блок-схема методу FindPercentageMan

`int FindPercentageWoman()` – метод знаходження відсоткового відношення чоловіків до жінок, де враховується усі елементи масиву та повертає відсоток птахі жіночої статі серед усіх.
Блок-схема показана на рисунку 5

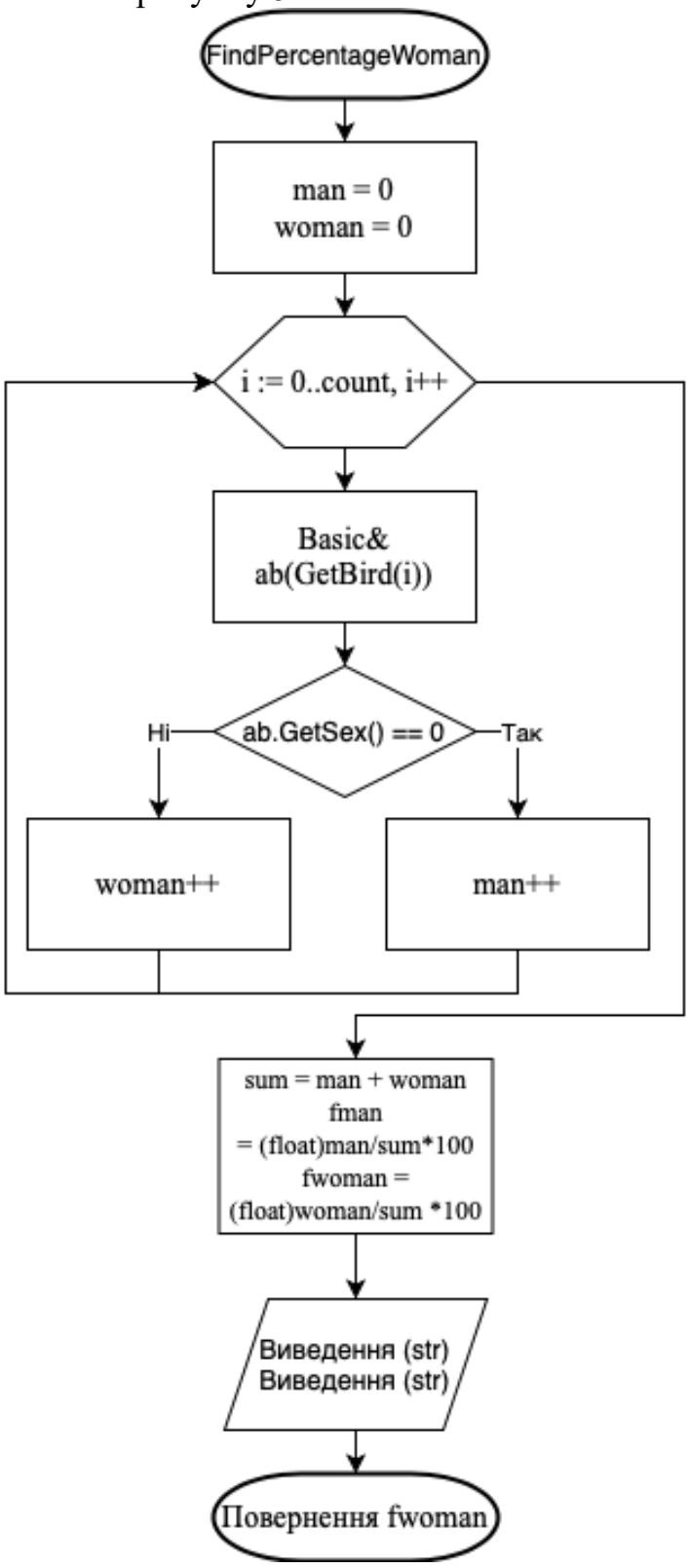

Рисунок 5 – блок-схема методу FindPercentageWoman

`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 6

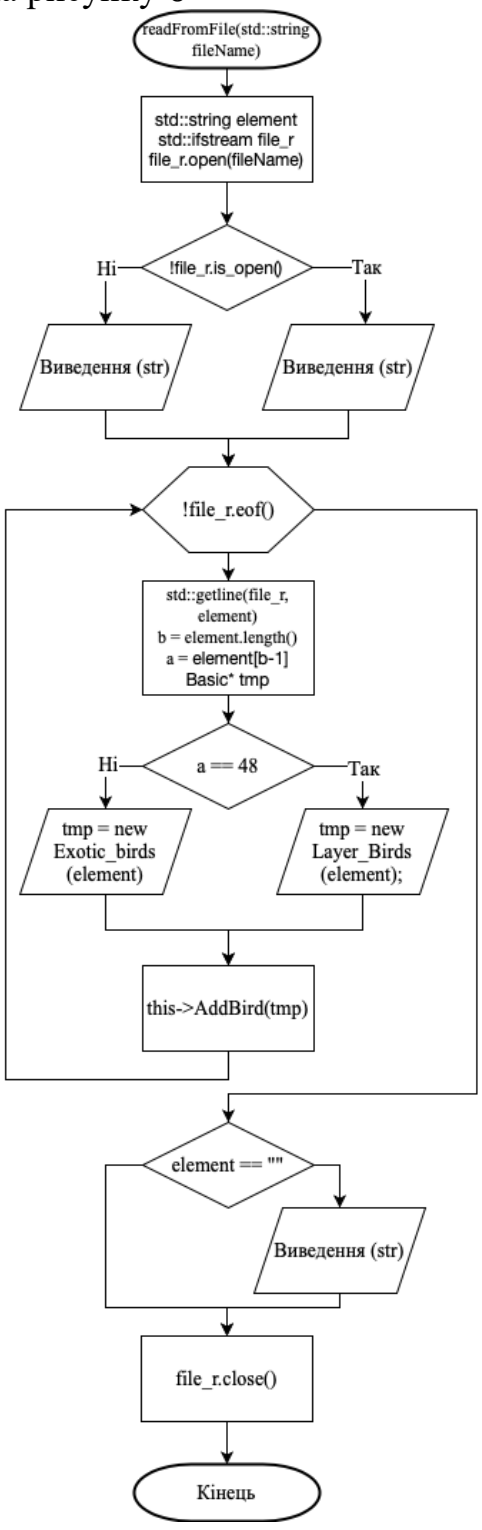

Рисунок 6 – блок-схема методу readFromFile

`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 7

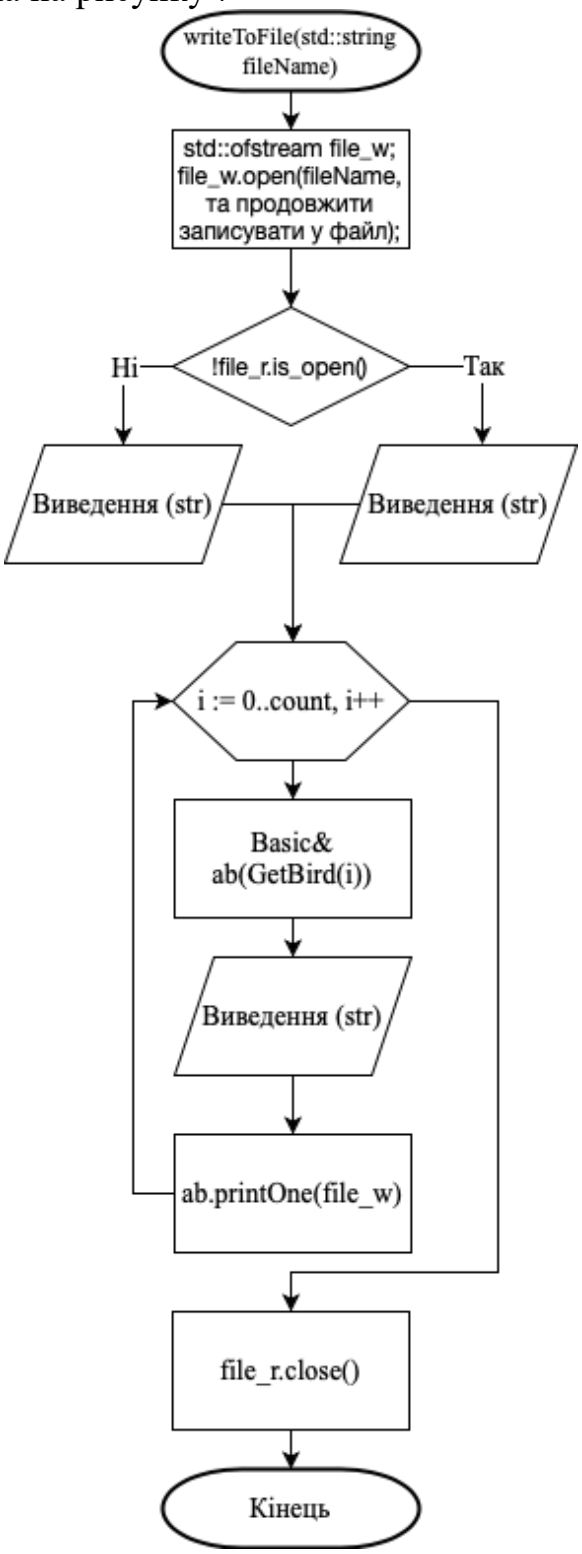

Рисунок 7 – блок-схема методу writeToFile

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

2.2.2 Клас «базовий»

`List Basic`

Призначення: даний клас абстрактний, але при цьому відповідає базовим 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` – є абстрактним методом

`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 8

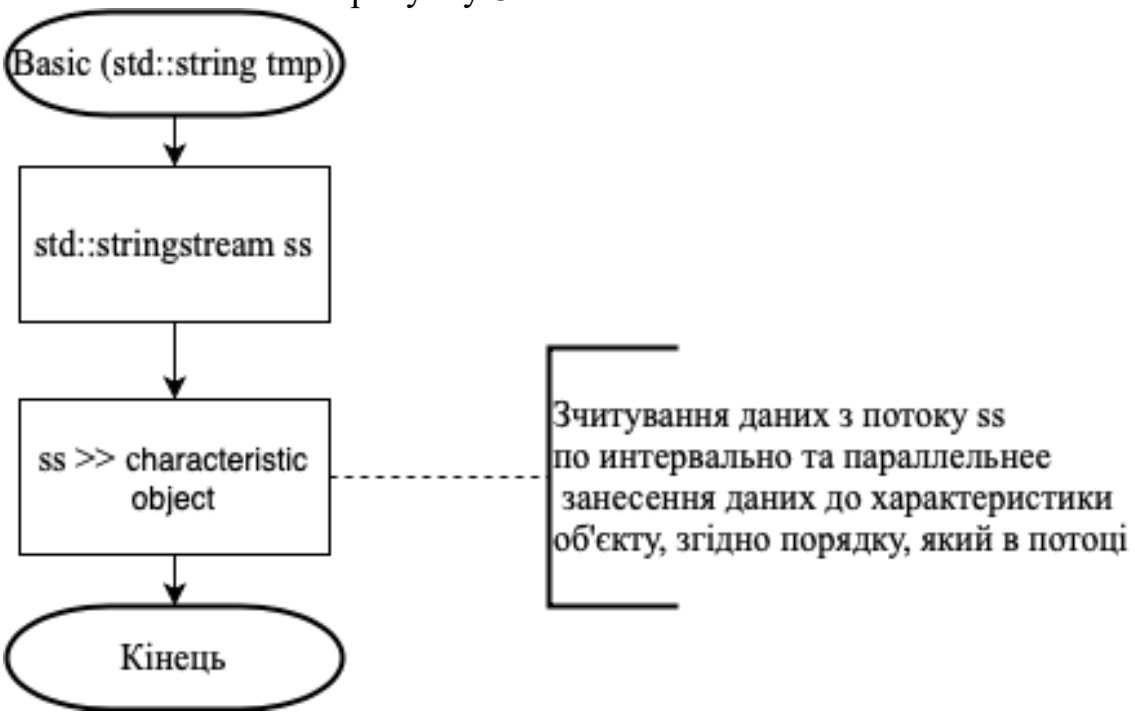

Рисунок 8 – блок схема конструктору зі строкою

`std::string toString()` – є абстрактним методом.

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

Параметри:

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

`virtual ~Basic()` – деструктор базового класу

2.2.3 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

2.2.4 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 9

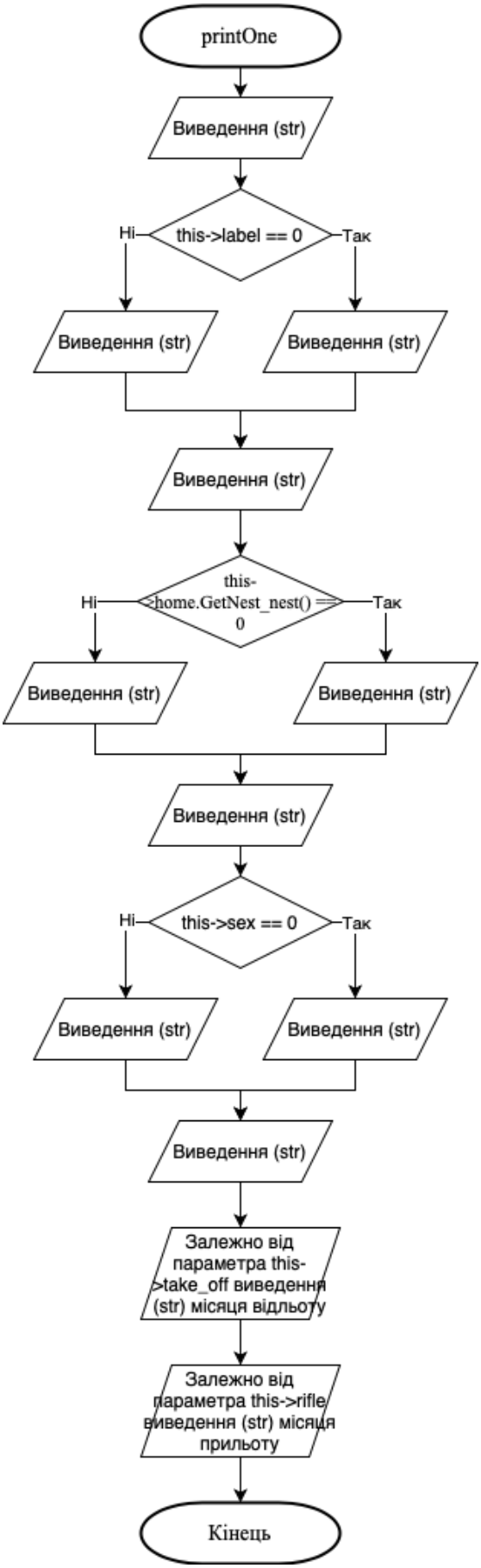

Рисунок 9 – блок-схема методу printOne для Layer_Birds

`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 10

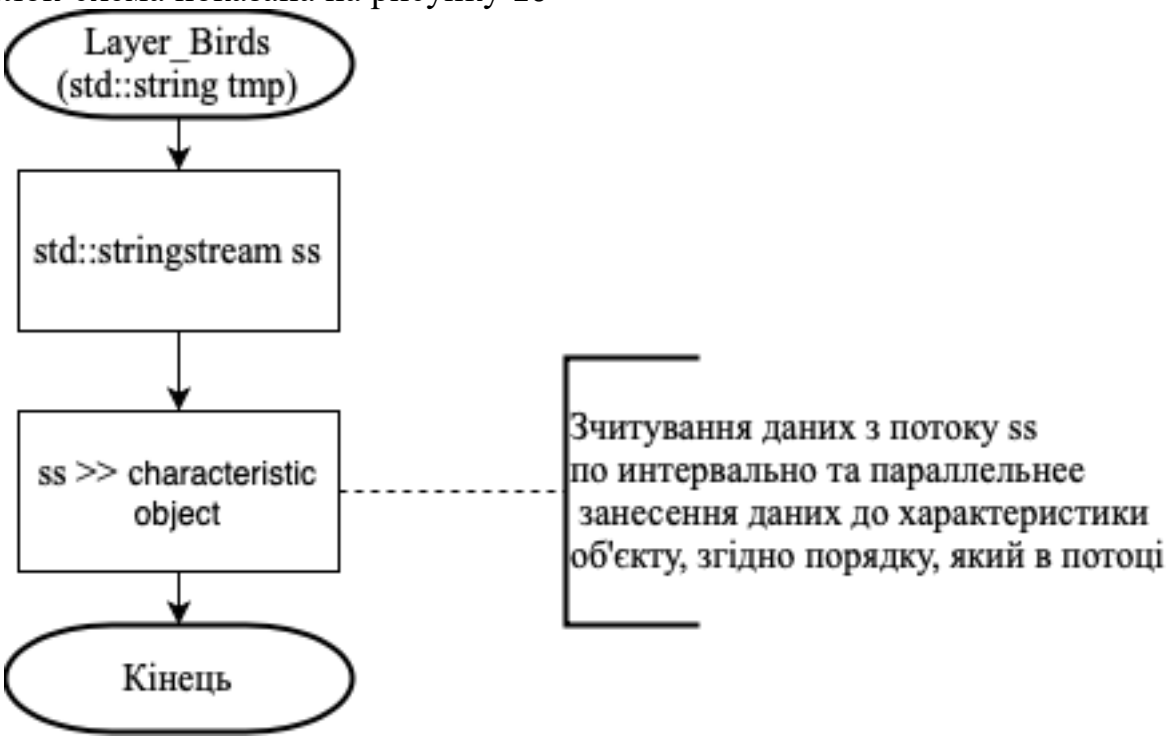

Рисунок 10 – блок-схема конструктора для Layer_Birds

`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 11

Рисунок 11 – блок-схема методу toString для Layer_Birds

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

2.2.5 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 12

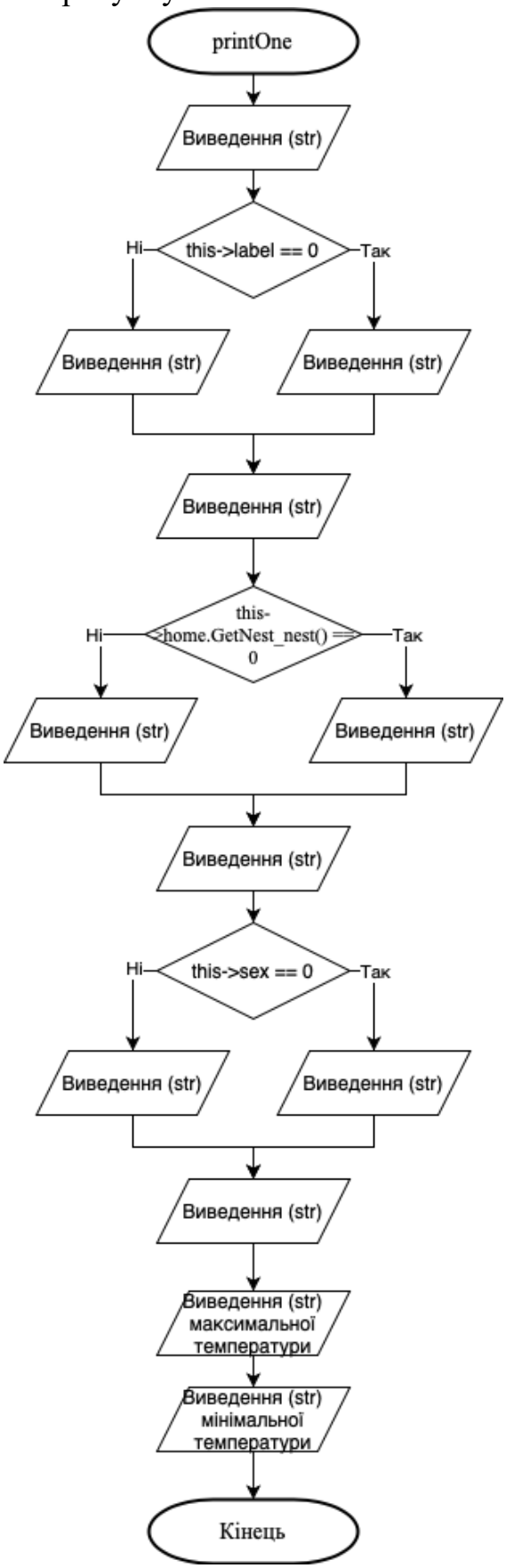

Рисунок 12 – блок-схема методу printOne для Exotic_birds

`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 13

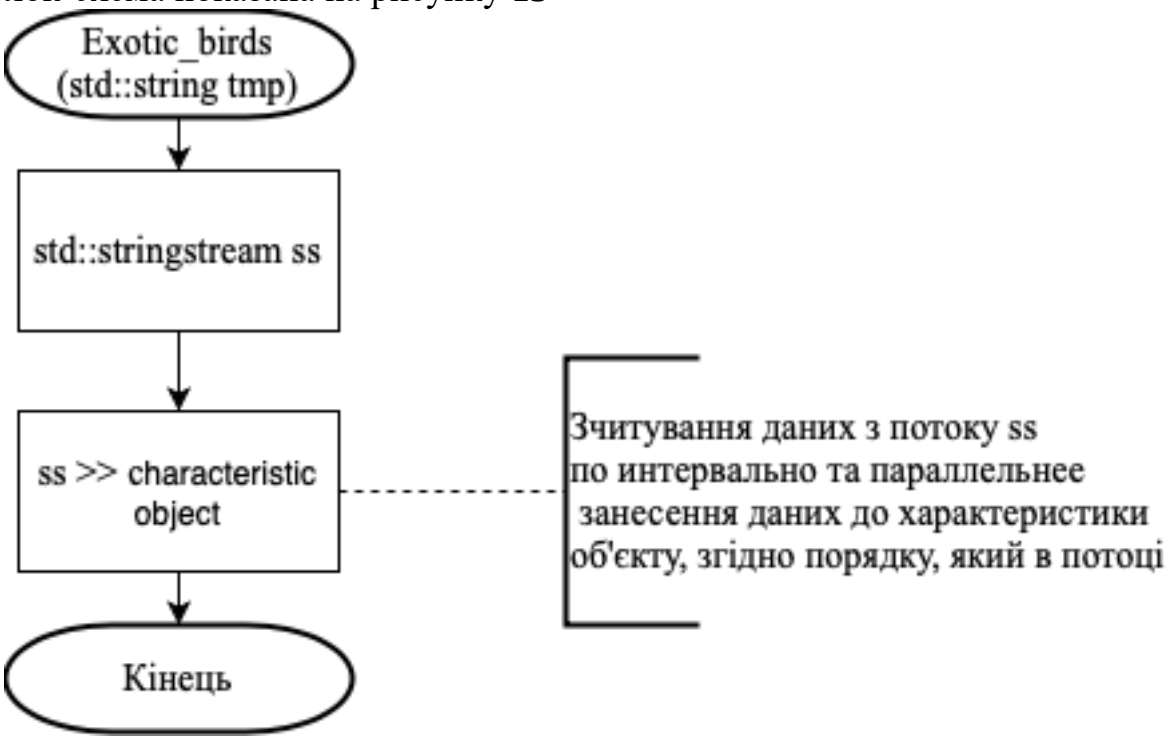

Рисунок 13 – блок-схема конструктора для Exotic_birds

`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 14

Рисунок 14 – блок-схема методу toString для Exotic_birds

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

2.3 Структура проекту

```
.
├── Doxyfile
├── Makefile
└── README.md
```

```
├── doc
│   ├── Radievych27.pdf
│   └── assets
│       ├── (Basic)toString.drawio
│       ├── (Basic)toString.png
│       ├── (Exotic)ToString.drawio
│       ├── (Exotic)ToString.png
│       ├── (Exotic)printOne.drawio
│       ├── (Exotic)printOne.png
│       ├── (Layer)ToString.drawio
│       ├── (Layer)ToString.png
│       ├── (Layer)printOne.drawio
│       ├── (Layer)printOne.png
│       ├── (List)AddBirds.drawio
│       ├── (List)AddBirds.png
│       ├── (List)FindPe...MAN.drawio
│       ├── (List)FindPe...MAN.png
│       ├── (List)FindPe...WOman.drawio
│       ├── (List)FindPe...WOman.png
│       ├── (List)ReadFromFile.drawio
│       ├── (List)ReadFromFile.png
│       ├── (List)RemoveBirds.drawio
│       ├── (List)RemoveBirds.png
│       ├── (List)WriteToFile.drawio
│       ├── (List)WriteToFile.png
│       ├── Basic(string).drawio
│       ├── Basic(string).png
│       ├── Exotic(string).drawio
│       ├── Exotic(string).png
│       ├── Layer(string).drawio
│       ├── Layer(string).png
│       ├── main27.drawio
│       └── main27.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 15, рисунку 16 та рисунку 17, виконання модульних тестів на рисунку 18 та рисунок 19 – демонстрація відсутності витоків пам'яті

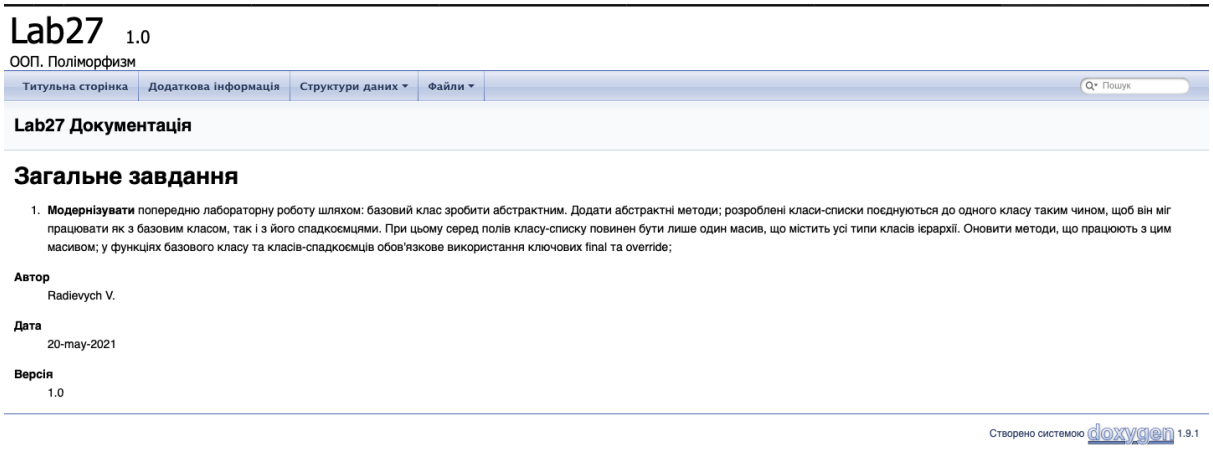

Рисунок 15 – робота з doxygen

Рисунок 16 – робота з doxygen

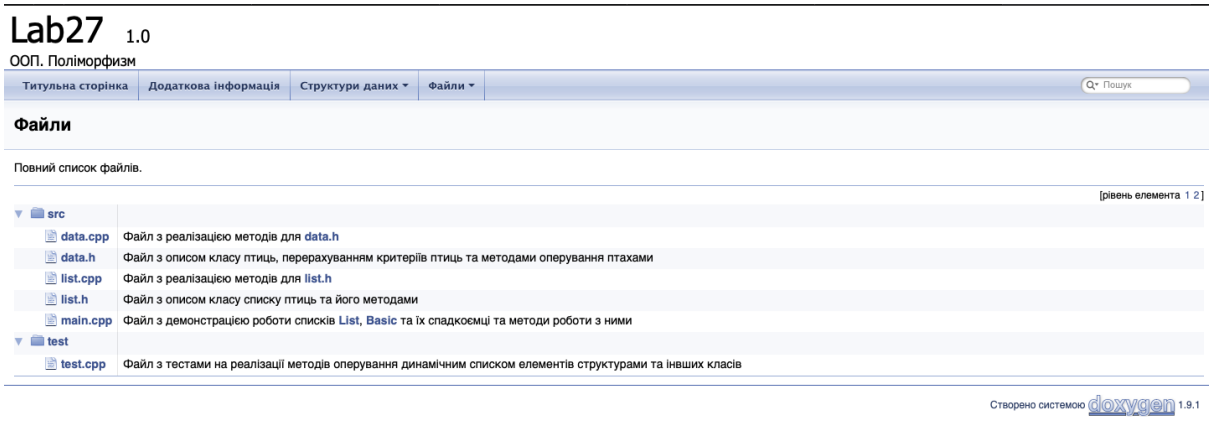

Рисунок 17 – робота з doxygen

```
[whatislove@MacBook-Air-Vladislav lab27 % cd dist
[whatislove@MacBook-Air-Vladislav dist % ./test1.bin
Запуск тесту test_AddBird ...
Запуск тесту test_RemoveBird ...
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 ...
Запуск тесту test_FindPercentageMan ...
Запуск тесту test_FindPercentageWoman ...
Модульні тести пройдено успішно!%
```

Рисунок 18 – робота з модульними тестами

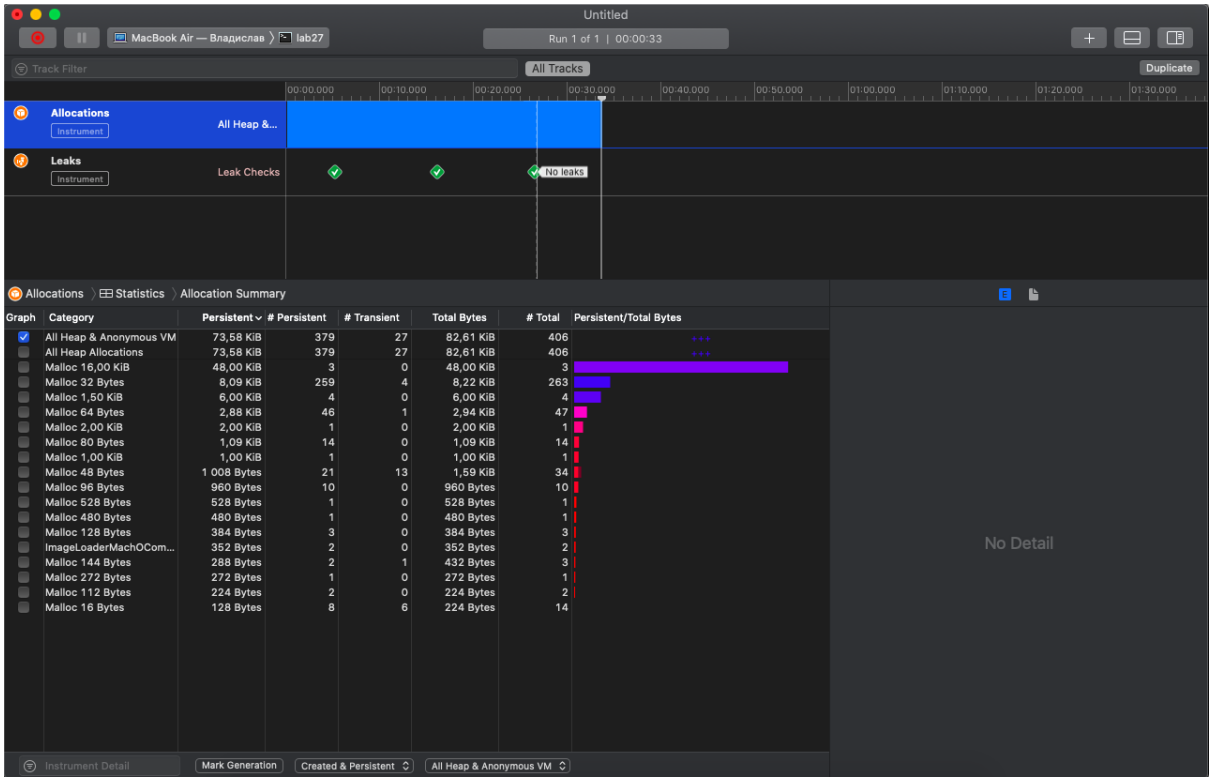

Рисунок 19 - демонстрація відсутності витоків пам'яті

4 ВИСНОВОК

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

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