МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
"ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ"

Факультет (відділення) КІТ

Кафедра (предметна, циклова комісія) Обчислювальна техніка та програмування

Спеціальність Кібербезпека

Освітня програма Кібербезпека

ПОЯСНЮВАЛЬНА ЗАПИСКА
до розрахункового завдання

на тему «Розробка інформаційно-довідкової системи»

Виконав студент 1 курсу, групи 320

Радєвич Владислав Романович

Керівник Давидов В’ячеслав Вадимович

Харків 2021

ЗМІСТ

ВСТУП

Метою виконання розрахункового завдання є закріпити отриманні знання з дисципліни «Основи безпечного програмування» шляхом виконання типового комплексного завдання. Під час виконання даного завдання було показано набутий рівень знань з даної дисципліни , вміння працювати з науково-технічною літературою, самостійно вирішувати поставлені індивідуальні завдання, забезпечуючи його ефективність та розробляти відповідні документи. Розробка таких видів роботи є одним з видів тренінгу, тематика якого відповідає сучасному стану та перспективам розвитку комп'ютерних технологій. Дана робота вимагає знання алгоритмів обробки як чисельних, так і нечисельних даних. Головною технологією програмування є об'єктно-орієнтовне програмування, яке дозволяє розкласти проблему на пов'язані між собою завдання. Кожне з них стає самостійним об'єктом, що містить свої власні коди та дані, які стосуються цього об'єкту.

ПРИЗНАЧЕННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Дане розроблене індивідуальне завдання можна використовувати, наприклад, у відділі орнітології, де почався перепис усіх зареєстрованих птахів. Базовими характеристиками для птаха, якого потрібно занести у реєстр, є чи окільцьована птаха, назва виду, вік птаха, тип домівки птаха, стать птаха. Та окрім, базових відомостей, також є додаткові характеристики для окремих класів птахів, а саме для перелітних птахів та екзотичних птахів. Перелітні птахи мають додаткові характеристики, окрім базових, як, місяць відльоту у вирій та місяць прильоту з вирію. Екзотичні птахи мають додаткові характеристики, окрім базових, як, мінімальна комфортна для життя температура та максимальна комфортна для життя температура.

Таким реєстром птахів можна маніпулювати змінюючи деякі дані певних птиць та можна структуровано оформити дані про усіх птахів в окремі файли.

ПОСТАНОВКА ЗАВДАННЯ ДО РОЗРОБКИ

Для розробки інформаційно-довідкової системи необхідно було, обрати індивідуальне завдання, номер якого обираються за номером студента в журналі групи, в моєму випадку мій номер в журналі 23, тому індивідуальне завдання було обрано під номер 23, прикладна галузь «Птахи». Також, для приладної галузі потрібно розробити розгалужену ієрархію класів, що у описана у завданні та складається з одного базового класу та двох спадкоємців. Класи повинні мати перевантажені оператори введення-виведення даних та порівняння.

При цьому усі класи повинні мати конструктори та деструктори; якщо функція не змінює поля класу, вона не має бути декларована як константа; рядки повинні бути типу string; при перевантажені функції треба використовувати ключове слово override; програмний код усіх класів має бути 100% doxygen-документований; код повинен бути написаний максимально з використанням C++11 стандартом та стандартної бібліотеки шаблонів.

ОПИС ВХІДНИХ ТА ВИХІДНИХ ДАНИХ

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

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)` – 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;

- `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

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 1

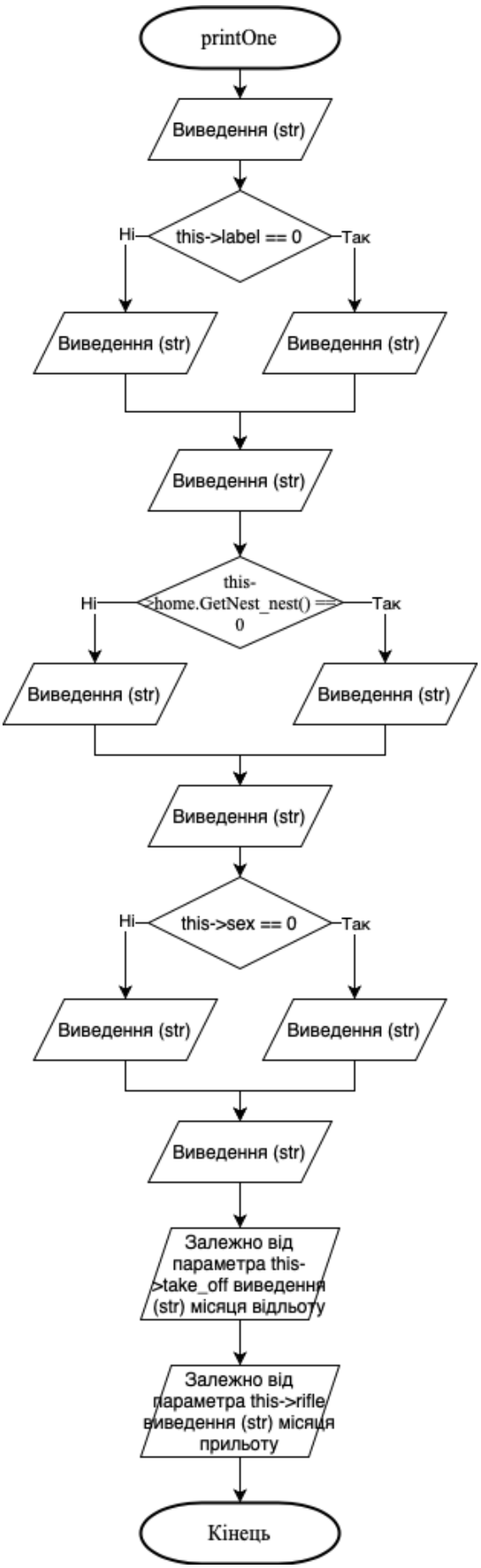

Рисунок 1 – блок-схема методу printOne для Layer_Birds

- `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;

- `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 2

Рисунок 2 – блок-схема методу toString для Layer_Birds

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

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 життя птаха.

Властивості класу:

`int max_temp` – максимально комфортна температура для життя птаха;

`int min_temp` – мінімально комфортна температура для життя 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 3

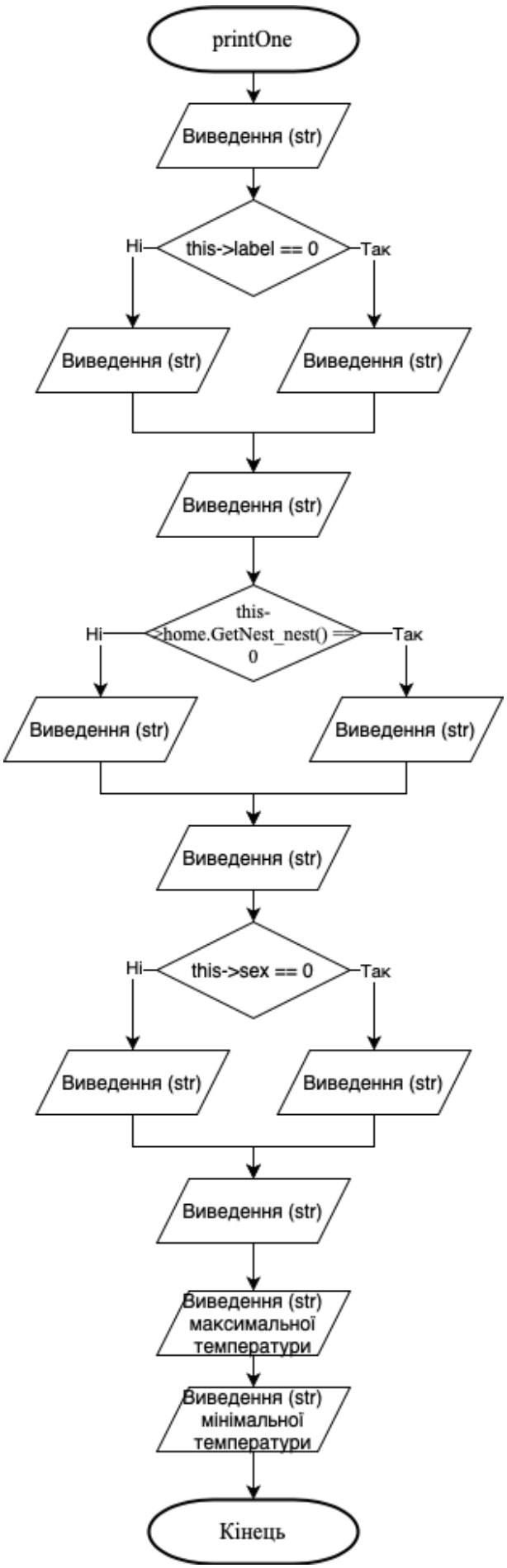

Рисунок 3 – блок-схема методу printOne для Exotic_birds

- `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;

- `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 4

Рисунок 4 – блок-схема методу toString для Exotic_birds

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

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 спадкоємець, який буде додано в кінець динамічного масиву;

Блок-схема показана на рисунку 5

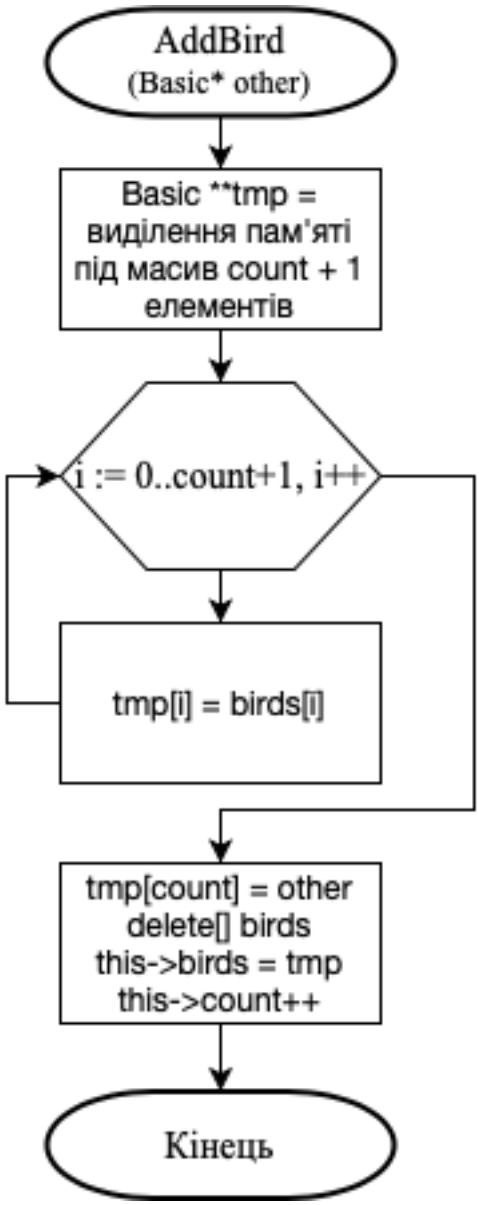

Рисунок 5 – блок-схема методу AddBird

- `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 6

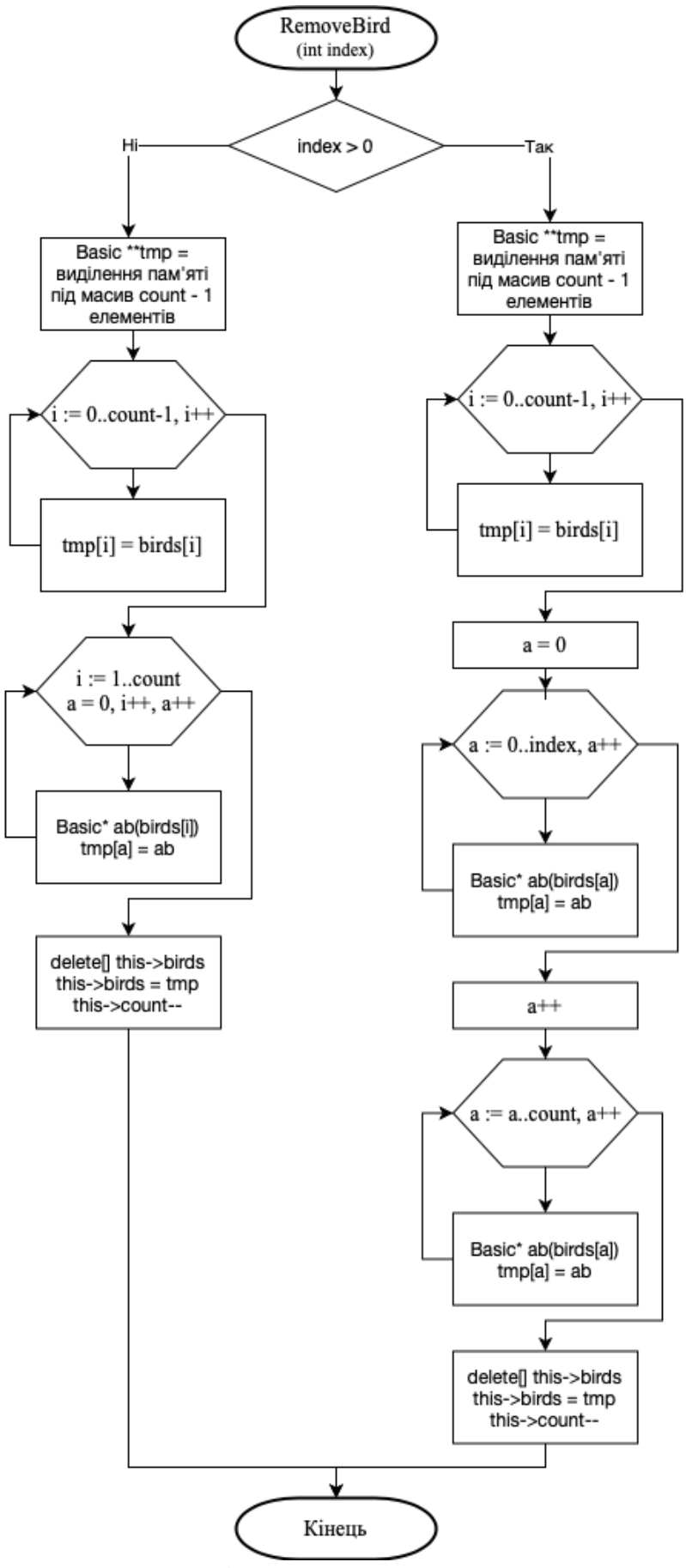

Рисунок 6 – блок-схема методу RemoveBird

- `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 7

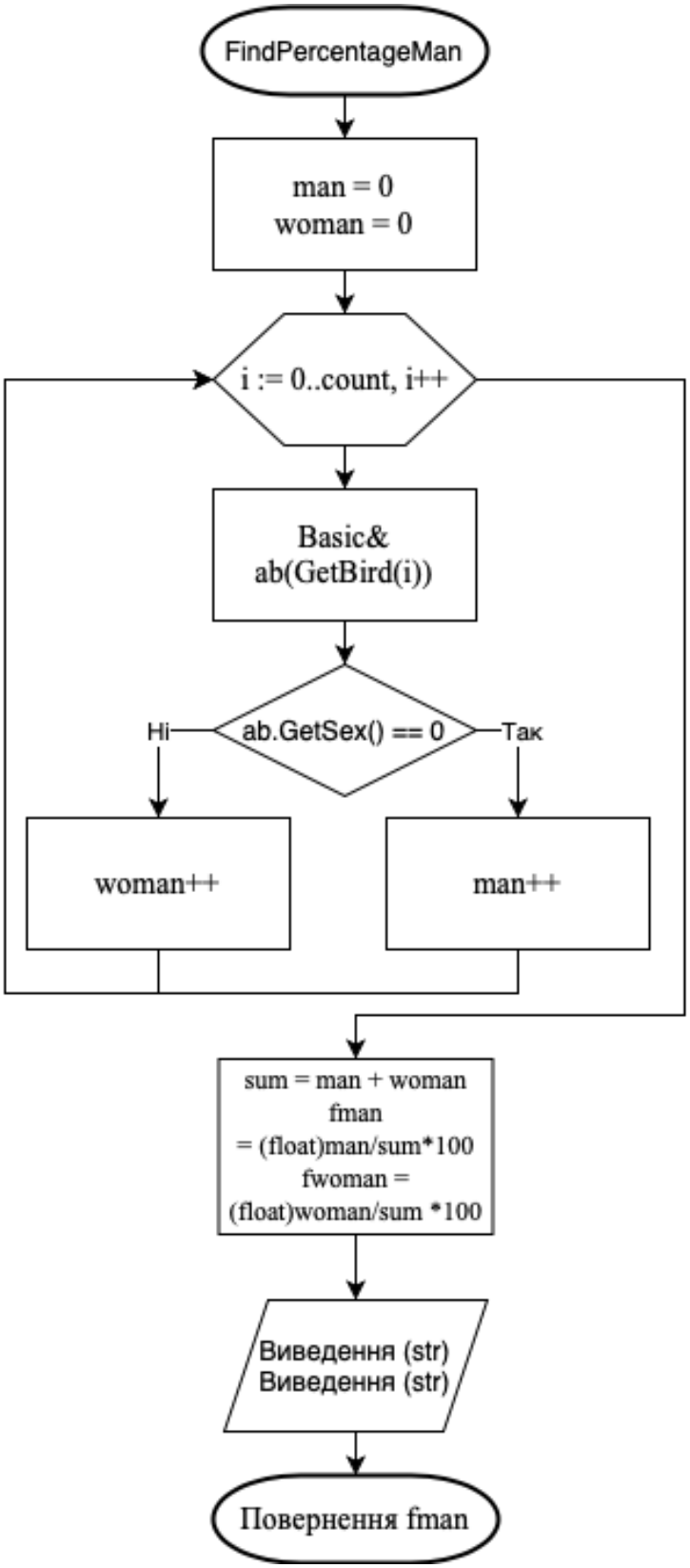

Рисунок 7 – блок-схема методу FindPercentageMan

- `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 8

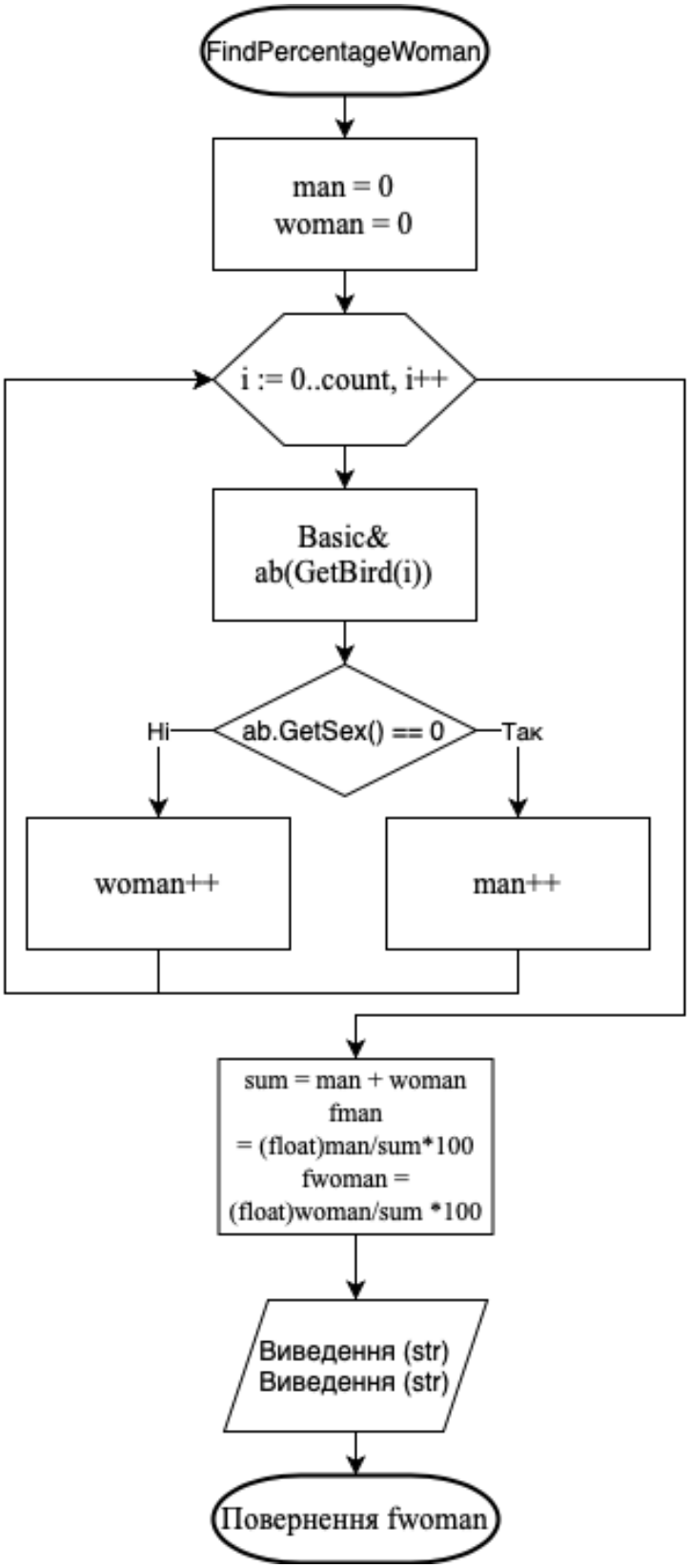

Рисунок 8 – блок-схема методу FindPercentageWoman

- `void readFromFile(std::string fileName)` – метод зчитування інформації про характеристики об’єкта з файлу. Але для правильної роботи методу необхідно в кінці рядка вказати тип птиці – якщо перелітна птаха – 0, якщо екзотична – 1.

Параметри:

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

Блок-схема показана на рисунку 9

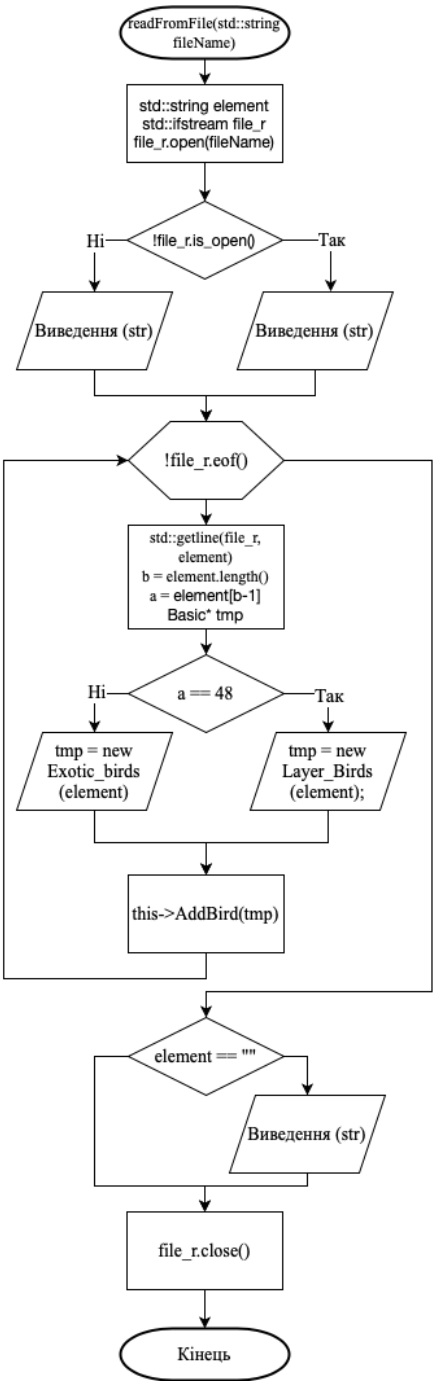

Рисунок 9 – блок-схема методу readFromFile

- `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 10

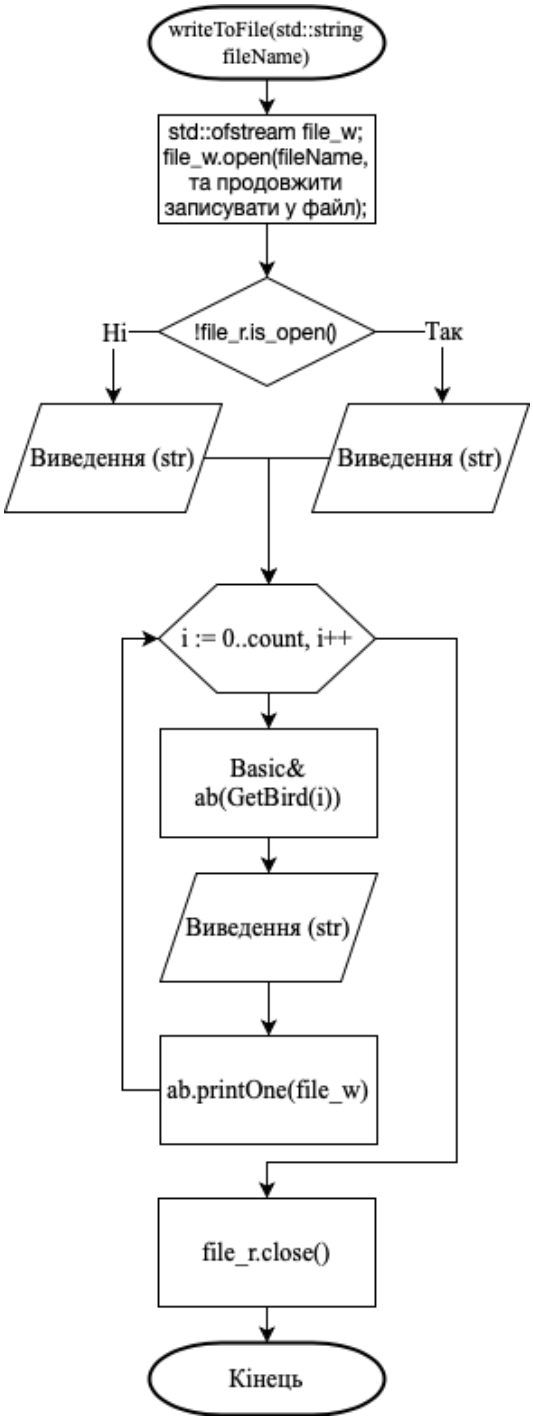

Рисунок 10 – блок-схема методу writeToFile

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

ОПИС СКЛАДУ ТЕХНІЧНИХ ТА ПРОГРАМНИХ ЗАСОБІВ

1.1.1. Клас «базовий»

`List Basic`

Методи класу:

- `Basic (std::string tmp)` – конструктор який приймає інформацію про об’єкт базового класу через строку. Цей конструктор також реалізований в класах Layer_Birds та Exotic_birds за тим самим алгоритмом.

Параметри:

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

Алгоритм:

При виклику цього конструктора дані, які було добавлені, а саме строку типу string, поміщаються в потоковий клас stringstream, наче в буфер зберігання даних. Потім завдяки перевантаженим операторам вводу-виводу вводимо дані з буферу до класу. При виводі з потоку дані виводиться до пробілу.

1.1.2. Клас «перелітні птахи» та «екзотичні птахи»

`Layer_Birds && Exotic_birds`

Методи класу:

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

Алгоритм:

На початку створюється головна строка, яка буде повертатися та містити в собі інформацію про об’єкт. Потім потрібно створити ще дві строки для зручного занесення даних в головну строку та необхідна строку класу stringstream. Беручи інформацію з об'єкта занесемо її перевантажними операторами вводу-виводу в строку класу stringstream, розділяючи дані пробілом. Після чого, в залежності від кількості характеристик, запускаємо

цикл, який буде передавати дані з stringstream в дві допоміжні строки. Потім поєднуємо допоміжні строки за записуємо результат в головну строку, очищаємо stringstream та повертаємо значення головної строки.
Блок-схема показана на рисунку 2 (Layer_Birds) та 4 (Exotic_birds)

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

Алгоритм:

По порядку дані об'єкту записуються в файл та оформлюються додатковим текстом, який конкретно вказує на кожну характеристику об'єкта. Та залежно від значення деяких характеристик об'єкта виводиться оброблене значення об'єкту.
Блок-схема показана на рисунку 1 (Layer_Birds) та 3 (Exotic_birds)

1.1.3. Клас «список»

`class List`

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

Методи класу:

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

Параметри:

`Basic* other` – об'єкт базового класу або його спадкоємець, який буде додано в кінець динамічного масиву;

Алгоритм:

Спочатку виділяємо пам’ять тимчасовому списку спираючись на кількість елементів у списку, але на один більше, потім переписуємо значення з старого списку в новий завдяки циклу. Об’єкт, який повинен був добавитися ставимо на останню позицію в списку. Видаляємо пам’ять старого списку та присвоюємо пам’ять нового та добавляємо загальну кількість елементів на один.

Блок-схема показана на рисунку 5

- `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

Алгоритм:

Залежно від номеру елементу, який буде видалено будуть відбуватися різні дії.

Якщо потрібно видалити не нулевий елемент списку, то спочатку виділяємо пам’ять тимчасовому списку спираючись на кількість елементів у списку, але на один менше, потім переписуємо значення з старого списку в новий завдяки циклу, але переписуємо до індексу елемента, який потрібно видалити. Потім пропускаємо цей елемент та продовжуємо переписування. Видаляємо пам’ять старого списку та присвоюємо пам’ять нового та відбавляємо загальну кількість елементів на один.

Якщо потрібно видалити нулевий елемент списку, то то спочатку виділяємо пам’ять тимчасовому списку спираючись на кількість елементів у списку, але на один менше, потім переписуємо значення з старого списку в новий завдяки циклу, але починаєм переписувати з першого елементу списку. Видаляємо пам’ять старого списку та присвоюємо пам’ять нового та відбавляємо загальну кількість елементів на один.

Блок-схема показана на рисунку 6

- `int FindPercentageMan() && int FindPercentageWoman()` – метод знаходження відсоткового відношення чоловіків до жінок та навпаки, де враховується усі елементи масиву та повертає відсоток птахі чоловічої статі серед усіх та навпаки.

Алгоритм:

Створюємо змінні, які відображають кількість птиць чоловічої та жіночої статі. Завдяки циклу, продивляємося всі елементи списку та в залежності від статі птиці добавляємо в значення змінних одиницю. Потім знаходимо суму усіх птахів. За для знаходження відсотка усіх птиць чоловічої статі, потрібно кількість птиць чоловічої статі поділити на загальну кількість птиць та помножити на 100. Аналогічно, для знаходження відсотка усіх птиць жіночої статі, , потрібно кількість птиць жіночої статі поділити на загальну кількість птиць та помножити на 100. Залежно від викликаного методи, методи будуть повертати різні значення FindPercentageMan – відсоток птиць чоловічої статі, FindPercentageWoman – відсоток птиць жіночої статі.

Блок-схема показана на рисунку 7 та 8

- `void readFromFile(std::string fileName)` – метод зчитування інформації про характеристики об’єкта з файлу. Але для правильної роботи методу необхідно в кінці рядка вказати тип птиці – якщо перелітна птаха – 0, якщо екзотична – 1.

Параметри:

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

Алгоритм:

Створюємо змінні типу string в яку буде записуватися строки з файлу та ifstream, після чого відкриваємо файл. Робимо перевірку, чи відкрився файл за допомогою методу is_open. Далі робимо цикл, до поки

метод eof() не поверне результат false, наступним, потрібно створити дві зміні, одна буде приймати довжину рядка у файлі, а інша буде приймати передостанній символ у строчці та створюємо вказівник тимчасову зміну типу Basic. Залежно від передостаннього символу в строчці файлу буде виділятися пам’ять для класу Layer_Birds – якщо передостанній символ 0 та Exotic_birds – якщо передостанній символ 1. Після завершення циклу, викликається метод AddBird(). Якщо строка типу string порожня, то файл пустий, тоді виводимо повідомлення про це на екран. Та закриваємо файл. Блок-схема показана на рисунку 9

- `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;

Алгоритм:

Створюємо змінну типу ifstream, після чого відкриваємо файл та ставимо додаткову функцію, щоб у файл дані дописувалися, а не перезаписувалися за допомогою методу ofstream::app. Робимо перевірку, чи відкрився файл за допомогою методу is_open. Запускаємо цикл на кількість елементів у заданому списку, потім за допомогою методу GetBird отримаємо елементи по індексу, після чого виводимо у файл номер птаха та виводимо елемент за допомогою методу printOne. Після чтого закриваємо файл
Блок-схема показана на рисунку 10

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. https://purecodecpp.com/archives/2751
2. https://www.cplusplus.com/reference/sstream/stringstream/
3. https://code-live.ru/post/cpp-class-inheritance/
4. https://ravesli.com/urok-163-virtualnye-funktsii-i-polimorfizm/
5. https://ravesli.com/urok-126-druzhestvennye-funktsii-i-klassy/

ДОДАТОК А

Код програми

- Файл data.h

```
/**
 * @file data.h
 * @brief Файл з описом класу птиць, перерахуванням критеріїв птиць та методами оперування
птахами
 *
 * @author Radievych V.
 * @date 20-may-2021
 * @version 1.0
 */

#pragma once
#include <cstdio>
#include <cstring>
#include <iostream>
#include <string>
#include <sstream>
#include <fstream>

/**
 * Так або ні
 */
enum Yes_no {
    Так, //0
    Ні //1
};

/**
 * Стать птиці
 */
enum Sex {
    Чоловіча,
    Жіноча
};

/**
 * Місяці року
 */
enum Month {
    Січень,
    Лютий,
    Березень,
    Квітнь,
    Травень,
    Червень,
    Липень,
    Серпень,
    Вересень,
    Жовтень,
    Листопад,
    Грудень
};

/**
 * Клас домівки птаха
 */
class Feature {
private:
    int square; /**< площа домівки, см^2 */
    int height; /**< висота домівки, см */
    int number_of_feeders; /**< кількість годівниць */
    enum Yes_no nest_nest; /**< наявність гнізда */
public:
    Feature(): square(0), height(0), number_of_feeders(0), nest_nest(Так) { } //данные по
умолчанию (конструктор)

    Feature(int square1, int height1, int number_of_feeders1, Yes_no nest_nest1){
        square = square1;
        height = height1;
        number_of_feeders = number_of_feeders1;
```

```
        nest_nest = nest_nest1;
    }

    void SetSquare (int x) {
        square = x;
    }
    int GetSquare () const{
        return square;
    }

    void SetHeight (int x) {
        height = x;
    }
    int GetHeight() const{
        return height;
    }

    void SetNumber_of_feeders (int x) {
        number_of_feeders = x;
    }
    int GetNumber_of_feeders () const {
        return number_of_feeders;
    }

    void SetNest_nest (Yes_no x) {
        nest_nest = x;
    }
    Yes_no GetNest_nest() const {
        return nest_nest;
    }
    Feature (const Feature &other) { //копирование
        this->square = other.square;
        this->height = other.height;
        this->number_of_feeders = other.number_of_feeders;
        this->nest_nest = other.nest_nest;
    }

    virtual ~Feature() { // :/ Bay-bay!

    }
};

/**
 * Базовий клас "Птах" (абстрактний)
 */
class Basic {
protected:
    enum Yes_no label; /**< чи окольцьована птаха */
    std::string name; /**< назва виду*/
    int age; /**< вік патаха, місяців*/
    Feature home; /**< структура домівки птаха (@link  Feature) */
    enum Sex sex; /**< стать птаха */
public:
    Basic(): label(Так), name("Птиця"), age(0), sex(Чоловіча) { } //конструктор1

    Basic(Yes_no label1, std::string name1, int age1, Feature home1, Sex sex1) { //конструктор2
        label = label1;
        name = name1;
        age = age1;
        home = home1;
        sex = sex1;
    }

    /**
     * Конструктор вводу інформації про об'єкт через строку
     */
    Basic (std::string tmp) {
        std::stringstream ss;
        ss << tmp;
        int label1;

        ss >> label1;
        if (label1 == 0) {
            this->label = Так;
        } else {
            this->label = Ні;
        }

        ss >> this->name;
        ss >> this->age;
```

```
    int sq = 0;
    ss >> sq;
    this->home.SetSquare(sq);

    int h = 0;
    ss >> h;
    this->home.SetHeight(h);

    int num = 0;
    ss >> num;
    this->home.SetNumber_of_feeders(num);

    int nest = 0;
    ss >> nest;
    if (nest == 0) {
        this->home.SetNest_nest(Так);
    } else {
        this->home.SetNest_nest(Hi);
    }

    int s = 0;
    ss >> s;
    if (s == 0) {
        this->sex = Чоловіча;
    } else {
        this->sex = Жіноча;
    }
}

void SetLabel (Yes_no x){
    label = x;
}
Yes_no GetLabel() const{
    return label;
}

void SetName (std::string n) {
    name = n;
}

/**
 * Метод запису в рядок-інформацію
 */
virtual std::string toString() = 0;

void SetSex (Sex x){
    sex = x;
}
Sex GetSex() const{
    return sex;
}

void SetAge(int x) {
    age = x;
}
int GetAge()const {
    return age;
}

Feature GetHome() {
    return home;
}

virtual void printOne(std::ofstream &file) = 0;

/**
 * Конструктор копіювання
 */
Basic (Basic& other){ //копирование
    this->label = other.label;
    this->name = other.name;
    this->age = other.age;
    this->home = other.home;
    this->sex = other.sex;
}

virtual ~Basic() { // :/

}
```

```
};

/**
 * Клас "перелітні птахи"
 */
class Layer_Birds : public Basic {
private:
    enum Month take_off; /**< місяць коли відлітає птах у вирій */
    enum Month rifle; /**< місяць коли прилітає птах з вирію */
public:
    Layer_Birds(): Basic(), take_off(Січень), rifle(Січень) {}
    Layer_Birds(Basic &other, enum Month take_off, enum Month refle): Basic(other),
take_off(take_off), rifle(refle){ }

    void SetTake_off(enum Month take_off) {
        this->take_off = take_off;
    }
    Month GetTake_off() const {
        return this->take_off;
    }

    void SetRifle(enum Month rifle) {
        this->rifle = rifle;
    }
    Month GetRifle() const {
        return this->rifle;
    }

    /**
     * Конструктор вводу інформації про об'єкт через строку
     */
    Layer_Birds (std::string tmp) {
        std::stringstream ss;
        ss << tmp;
        int label1;

        ss >> label1;
        if (label1 == 0) {
            this->label = Так;
        } else {
            this->label = Ні;
        }

        ss >> this->name;
        ss >> this->age;

        int sq = 0;
        ss >> sq;
        this->home.SetSquare(sq);

        int h = 0;
        ss >> h;
        this->home.SetHeight(h);

        int num = 0;
        ss >> num;
        this->home.SetNumber_of_feeders(num);

        int nest = 0;
        ss >> nest;
        if (nest == 0) {
            this->home.SetNest_nest(Так);
        } else {
            this->home.SetNest_nest(Ні);
        }

        int s = 0;
        ss >> s;
        if (s == 0) {
            this->sex = Чоловіча;
        } else {
            this->sex = Жіноча;
        }
        int q = 0;
        ss >> q;
        switch (q) {
            case 0:
                this->take_off = Січень;
                break;
            case 1:
```

```
            this->take_off = Лютий;
            break;
        case 2:
            this->take_off = Березень;
            break;
        case 3:
            this->take_off = Квітнь;
            break;
        case 4:
            this->take_off = Травень;
            break;
        case 5:
            this->take_off = Червень;
            break;
        case 6:
            this->take_off = Липень;
            break;
        case 7:
            this->take_off = Серпень;
            break;
        case 8:
            this->take_off = Вересень;
            break;
        case 9:
            this->take_off = Жовтень;
            break;
        case 10:
            this->take_off = Листопад;
            break;
        case 11:
            this->take_off = Грудень;
            break;
    }
    int w = 0;
    ss >> w;
    switch (w) {
        case 0:
            this->rifle = Січень;
            break;

        case 1:
            this->rifle = Лютий;
            break;
        case 2:
            this->rifle = Березень;
            break;
        case 3:
            this->rifle = Квітнь;
            break;
        case 4:
            this->rifle = Травень;
            break;
        case 5:
            this->rifle = Червень;
            break;
        case 6:
            this->rifle = Липень;
            break;
        case 7:
            this->rifle = Серпень;
            break;
        case 8:
            this->rifle = Вересень;
            break;
        case 9:
            this->rifle = Жовтень;
            break;
        case 10:
            this->rifle = Листопад;
            break;
        case 11:
            this->rifle = Грудень;
            break;
    }

}

/**
 * Метод запису в рядок-інформацію
 *
```

```
     * Метод записує в рядок-інформацію всі даніх елементу типу Layer_birds
     */
    std::string toString() override final;

    /**
     * Метод виведення у файл
     *
     * Метод виводе у файл оформлений елемент класу (@link Layer_Birds)
     */
    void printOne(std::ofstream &file) override final;

    /**
     * Конструктор копіювання
     */
    Layer_Birds (Layer_Birds &other) : Basic(other), take_off(other.take_off), rifle(other.rifle)
{ }

    virtual ~Layer_Birds() {

    }
};

/**
 * Клас "екзотичні птахи"
 */
class Exotic_birds : public Basic {
private:
    int max_temp; /**< максимально комфортна температура для птаха */
    int min_temp; /**< манімально комфортна температура для птаха */
public:
    Exotic_birds(): max_temp(0), min_temp(0) {}
    Exotic_birds(Basic &other, int min_temp, int max_temp): Basic(other), min_temp(min_temp),
max_temp(max_temp) { }

    void SetMin_temp(int min_temp) {
        this->min_temp = min_temp;
    }
    int GetMin_temp() const{
        return this->min_temp;
    }

    void SetMax_temp(int max_temp) {
        this->max_temp = max_temp;
    }
    int GetMax_temp() const{
        return this->max_temp;
    }

    /**
     * Конструктор вводу інформації про об'єкт через строку
     */
    Exotic_birds (std::string tmp) {
        std::stringstream ss;
        ss << tmp;
        int label1;

        ss >> label1;
        if (label1 == 0) {
            this->label = Так;
        } else {
            this->label = Ні;
        }

        ss >> this->name;
        ss >> this->age;

        int sq = 0;
        ss >> sq;
        this->home.SetSquare(sq);

        int h = 0;
        ss >> h;
        this->home.SetHeight(h);

        int num = 0;
        ss >> num;
        this->home.SetNumber_of_feeders(num);

        int nest = 0;
        ss >> nest;
```

```
        if (nest == 0) {
            this->home.SetNest_nest(Так);
        } else {
            this->home.SetNest_nest(Hi);
        }

        int s = 0;
        ss >> s;
        if (s == 0) {
            this->sex = Чоловіча;
        } else {
            this->sex = Жіноча;
        }

        ss >> this->min_temp;
        ss >> this->max_temp;
    }

    /**
     * Метод запису в рядок-інформацію
     *
     * Метод записує в рядок-інформацію всі даніх елементу типу Exotic_birds
     */
    std::string toString() override final;

    /**
     * Метод виведення у файл
     *
     * Метод виводе у файл оформлений елемент класу (@link Exotic_birds)
     */
    void printOne(std::ofstream &file) override final;


    /**
     * Метод виведення у файл
     *
     * Метод виводе у файл оформлений елемент класу (@link Exotic_birds)
     */
    void printOne(std::ofstream &file) override final;

    /**
     * Конструктор копіювання
     */
    Exotic_birds (Exotic_birds &other) : Basic(other), min_temp(other.min_temp),
max_temp(other.max_temp) { }

    virtual ~Exotic_birds() {

    }
};
```

- Файл data.cpp

```
/**
 * @file data.cpp
 * @brief Файл з реалізацією методів для data.h
 *
 * @author Radievych V.
 * @date 20-may-2021
 * @version 1.0
 */
#include "data.h"

std::string Layer_Birds::toString() {
    std::string s_info;
    std::string s_info1;
    std::string s_info2;
    std::stringstream ss;

    ss << this->label << " ";

    ss << this->name << " " << this->age <<" " << this->home.GetSquare()<< " " << this -
>home.GetHeight()<< " " << this->home.GetNumber_of_feeders()<< " ";

    ss << this->home.GetNest_nest() << " ";

    ss << this->sex << " ";

    ss << this->GetTake_off() << " ";
    ss << this->GetRifle() << " ";
```

```
    ss >> s_info;
    s_info += " ";
    for (int i = 0; i < 4; i++) {
        ss >> s_info1;
        ss >> s_info2;
        s_info += s_info1 + " " + s_info2 + " ";
    }
    ss >> s_info1;
    s_info += s_info1;

    ss.str(std::string());

    return s_info;
}

void Layer_Birds::printOne(std::ofstream &file) {
    file << "Чи окольцована птаха: ";
    if (this->label == 0) {
        file << "Так\n";
    } else {
        file << "Ні\n";
    }

    file << "Назва птаха: " << this->name << "\n"; file << "Вік птаха: " << this->age <<"
місяців\n"; file << "Площа домівки: " << this->home.GetSquare() << " см^2\n"; file << "Висота
домівки: " << this->home.GetHeight() << " см.\n"; file << "Кількість годівниць: " << this-
>home.GetNumber_of_feeders() << "\n";

    file << "Наявність гнізда: ";
    if (this->home.GetNest_nest() == 0) {
        file << "Є гніздо\n";
    } else {
        file << "Немає гнізда\n";
    }

    file << "Стать: ";
    if (this->sex == 0) {
        file << "Чоловіча\n";
    } else {
        file << "Жіноча\n";
    }

    file << "Місяць відльоту: ";
    switch (this->take_off) {
        case 0:
            file << "Січень\n";
            break;
        case 1:
            file << "Лютий\n";
            break;
        case 2:
            file << "Березень\n";
            break;
        case 3:
            file << "Квітнь\n";
            break;
        case 4:
            file << "Травень\n";
            break;
        case 5:
            file << "Червень\n";
            break;
        case 6:
            file << "Липень\n";
            break;
        case 7:
            file << "Серпень\n";
            break;
        case 8:
            file << "Вересень\n";
            break;
        case 9:
            file << "Жовтень\n";
            break;
        case 10:
            file << "Листопад\n";
            break;
        case 11:
            file << "Грудень\n";
```

```
            break;
    }

    file << "Місяць прильту: ";
    switch (this->rifle) {
        case 0:
            file << "Січень\n\n";
            break;
        case 1:
            file << "Лютий\n\n";
            break;
        case 2:
            file << "Березень\n\n";
            break;
        case 3:
            file << "Квітнь\n\n";
            break;
        case 4:
            file << "Травень\n\n";
            break;
        case 5:
            file << "Червень\n\n";
            break;
        case 6:
            file << "Липень\n\n";
            break;
        case 7:
            file << "Серпень\n\n";
            break;
        case 8:
            file << "Вересень\n\n";
            break;
        case 9:
            file << "Жовтень\n\n";
            break;
        case 10:
            file << "Листопад\n\n";
            break;
        case 11:
            file << "Грудень\n\n";
            break;
    }
}

std::string Exotic_birds::toString() {
    std::string s_info;
    std::string s_info1;
    std::string s_info2;
    std::stringstream ss;

    ss << this->label << " ";

    ss << this->name << " " << this->age <<" " << this->home.GetSquare()<< " " << this -
>home.GetHeight()<< " " << this->home.GetNumber_of_feeders()<< " ";

    ss << this->home.GetNest_nest() << " ";

    ss << this->sex << " ";

    ss << this->min_temp << " ";
    ss << this->max_temp << " ";

    ss >> s_info;
    s_info += " ";
    for (int i = 0; i < 4; i++) {
        ss >> s_info1;
        ss >> s_info2;
        s_info += s_info1 + " " + s_info2 + " ";
    }
    ss >> s_info1;
    s_info += s_info1;

    ss.str(std::string());

    return s_info;
}

void Exotic_birds::printOne(std::ofstream &file) {
    file << "Чи окольцована птаха: ";
    if (this->label == 0) {
```

```
        file << "Так\n";
    } else {
        file << "Ні\n";
    }

    file << "Назва птаха: " << this->name << "\n"; file << "Вік птаха: " << this->age <<"
місяців\n"; file << "Площа домівки: " << this->home.GetSquare() << " см^2\n"; file << "Висота
домівки: " << this->home.GetHeight() << " см.\n"; file << "Кількість годівниць: " << this-
>home.GetNumber_of_feeders() << "\n";

    file << "Наявність гнізда: ";
    if (this->home.GetNest_nest() == 0) {
        file << "Є гніздо\n";
    } else {
        file << "Немає гнізда\n";
    }

    file << "Стать: ";
    if (this->sex == 0) {
        file << "Чоловіча\n";
    } else {
        file << "Жіноча\n";
    }

    file << "Мінімальна комфортна температура: " << this->min_temp << "\n";

    file << "Максимальна комфортна температура: " << this->max_temp << "\n\n";
}
```

- Файл list.h

```
/**
 * @file list.h
 * @brief Файл з описом класу списку птиць та його методами
 *
 * @author Radievych V.
 * @date 20-may-2021
 * @version 1.0
 */

#pragma once
#include "data.h"
#include <cerrno>
#include <cstdlib>
#include <fstream>

/**
 * Клас списку птиць та його методи
 */
class List {
private:
    Basic** birds;
    int count;
public:
    List(): count(0) {
        this->birds = new Basic*[count];
    }

    int GetCount() const{
        return count;
    }

    /**
     * Метод додавання елементу
     *
     * Метод добавляє елемент типу Basic до списку птиці, а саме у кіцень списку
     */
    void AddBird(Basic* other);

    /**
     * Метод видалення елементу
     *
     * Метод відаляє елемент за його інкексом зі списку з птицями
     */
    void RemoveBird(int index);

    /**
     * Метод отримання елементу масиву за індексом
     */
    Basic& GetBird (int index) {
```

```
        return *birds[index];
    }

    /**
     * Метод знаходження відсоток статі
     *
     * Метод знаходе відсоткове відношення усі елементів чоловічої та жіночої статі та повертає
відсоток чоловічої статі
     */
    int FindPercentageMan();

    /**
     * Метод знаходження відсоток статі
     *
     * Метод знаходе відсоткове відношення усі елементів чоловічої та жіночої статі та повертає
відсоток жіночої статі
     */
    int FindPercentageWoman();

    /**
     * Метод читання з файлу
     *
     * Метод читає дані для класу Basic з файлу, шлях якого повинен міститися в строці
     */
    void readFromFile(std::string fileName);

    /**
     * Метод запису в файл
     *
     * Метод записує дані про всі елементи класу List до файлу, шлях якого повинен міститися в
строці
     */
    void writeToFile(std::string fileName);

    virtual ~List() {
        for (int i = 0; i < count; i++) {
            delete birds[i];
        }
        delete[] birds;
        }
};
```

- Файл list.cpp

```
ф /**
 * @file list.cpp
 * @brief Файл з реалізацією методів для list.h
 *
 * @author Radievych V.
 * @date 20-may-2021
 * @version 1.0
 */
#include "list.h"

void List::AddBird(Basic* other) {
    Basic** tmp = new Basic*[this->count+1];
    for (int i = 0; i < count; i++) {
        tmp[i] = birds[i];
    }

    tmp[count] = other;
    delete[] birds;
    this->birds = tmp;
    this->count++;
}

void List::RemoveBird(int index) {
    if (index > 0) {
```

```
        Basic** tmp = new Basic*[this->count - 1];
        for (int i = 0; i < count - 1; i++) {
            tmp[i] = birds[i];
        }

        int a = 0;
        for (a = 0; a < index; a++) {
            Basic* ab(birds[a]);
            tmp[a] = ab;
        }
        a++;
        while(a < count) {
            Basic* ab(birds[a]);
            tmp[a] = ab;
            a++;
        }
        delete[] this->birds;
        this->birds = tmp;
        this->count--;

    } else {

        Basic** tmp = new Basic*[this->count - 1];
        for (int i = 0; i < count - 1; i++) {
            tmp[i] = birds[i];
        }
        for (int i = 1, a = 0; i < count; i++, a++){
            Basic* ab(birds[i]);
            tmp[a] = ab;
        }
        delete[] this->birds;
        this->birds = tmp;
        this->count--;
    }
}

int List::FindPercentageMan() {
    int man = 0;
    int woman = 0;
    for (int i = 0; i < count; i++) {
        Basic& ab(GetBird(i));
        if (ab.GetSex() == 0) {
            man++;
        } else {
            woman++;
        }
    }
    int sum = man + woman;
    int fman = (float)man/sum * 100;
    std::cout << "Відсоток птахів чоловічої статі : " << fman << "%\n";
    int fwoman = (float)woman/sum * 100;
    std::cout << "Відсоток птахів жіночої статі : "<< fwoman << "%\n";
    return fman;
```

```
}

int List::FindPercentageWoman() {
    int man = 0;
    int woman = 0;
    for (int i = 0; i < count; i++) {
        Basic& ab(GetBird(i));
        if (ab.GetSex() == 0) {
            man++;
        } else {
            woman++;
        }
    }
    int sum = man + woman;
    int fman = (float)man/sum * 100;
    std::cout << "Відсоток птахів чоловічої статі : " << fman << "%\n";
    int fwoman = (float)woman/sum * 100;
    std::cout << "Відсоток птахів жіночої статі : "<< fwoman << "%\n";
    return fwoman;
}

void List::readFromFile(std::string fileName) {
    std::string element;
    std::ifstream file_r;
    file_r.open(fileName);
    if (!file_r.is_open()) {
        std::cout << "Помилка відкриття файлу" << std::endl;
    } else {
        std::cout << "Файл відкрито!" << std::endl;
    }
    while(!file_r.eof()) {
        std::getline(file_r, element);
        int b = element.length();
        int a = element[b-1];
        Basic *tmp;
        if (a == 48) {
            tmp = new Layer_Birds(element);
        } else {
            tmp = new Exotic_birds(element);
        }
        this->AddBird(tmp);
    }
    if (element == ""){
        std::cout << "Файл пустий!" << std::endl;
    }
    file_r.close();
}

void List::writeToFile(std::string fileName) {
    std::ofstream file_w;
    file_w.open(fileName, std::ofstream::app);
    if (!file_w.is_open()) {
        std::cout << "Помилка відкриття файлу" << std::endl;
```

```
        } else {
            std::cout << "Файл відкрито!" << std::endl;
        }
        for (int i = 0; i < count; i++) {
            Basic& ab(GetBird(i));
            file_w << "Птах № " << i + 1 << "\n";
            ab.printOne(file_w);
    }
        file_w.close();
}
```