

Лабораторна робота № 23. ООП. Вступ до ООП

1 ВИМОГИ

1.1 Розробник

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

1.2 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..

2 ОПИС ПРОГРАМИ

2.1 Функціональне призначення

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

2.2 Опис логічної структури

2.2.1 Основна функція

`int main`

Призначення: головна функція.

Схема алгоритму функції подана на рис. 0.

Опис роботи: демонструє роботу заданого динамічного списку елементів класу та методів оперування ним, а також методи роботи інших класів.

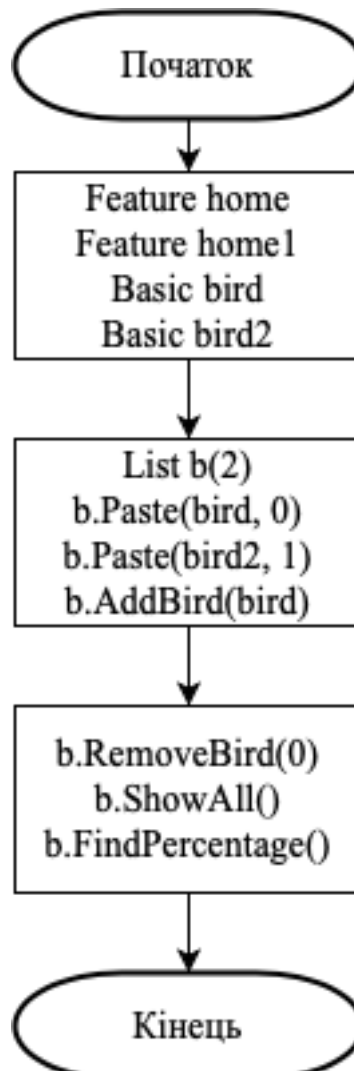


Рисунок 0 – Схема алгоритму функції main

2.2.2 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 1

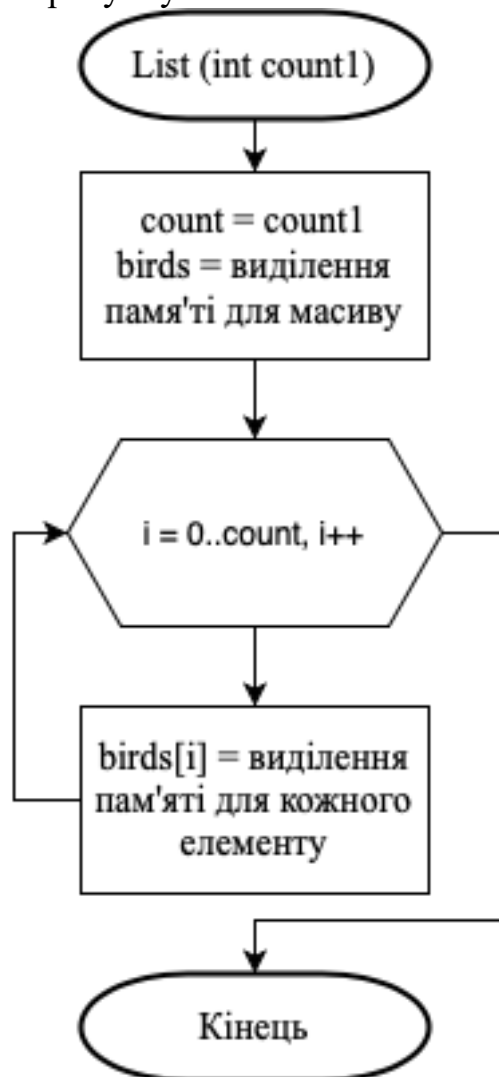


Рисунок 1 – блок-схема конструктору `List(int count1)`

`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;

Блок-схема показана на рисунку 2

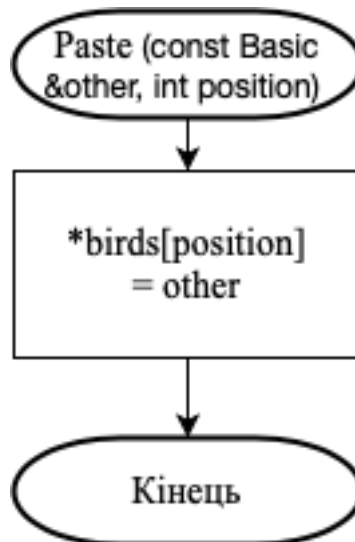


Рисунок 2 – блок-схема методу Paste

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

Блок-схема показана на рисунку 3

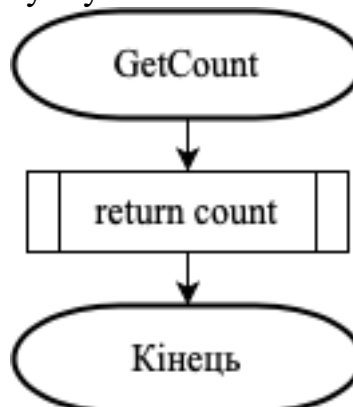


Рисунок 3 – блок-схема методу GetCount

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

Параметри:

`int index` – індекс елементу у динамічному масиві;

Блок-схема показана на рисунку 4

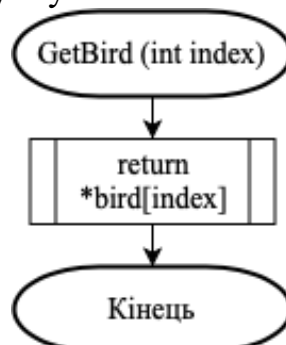


Рисунок 4 – блок-схема методу GetBird

`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 5

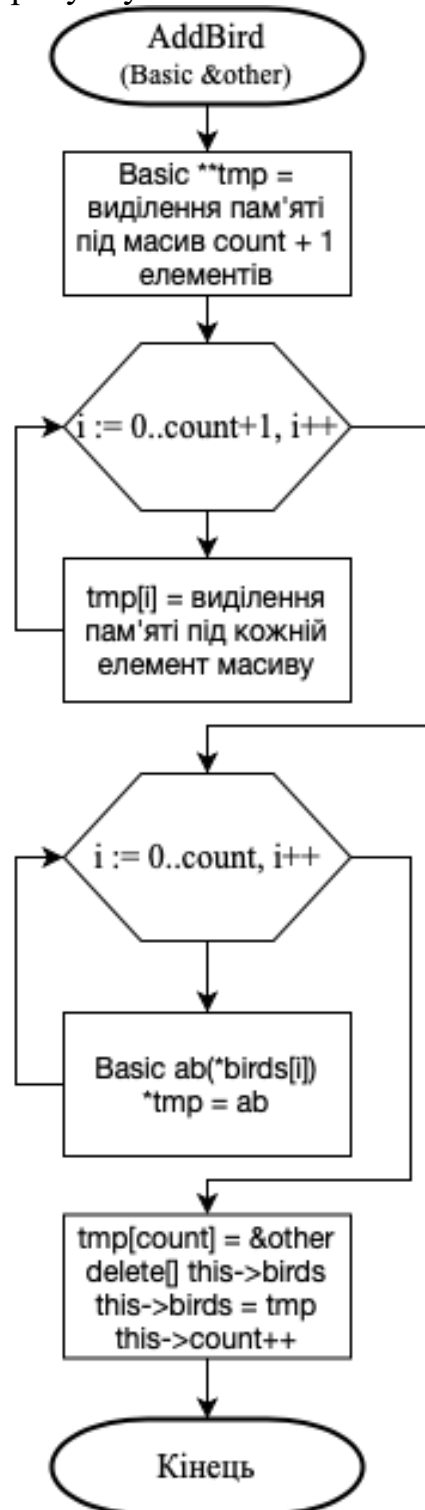


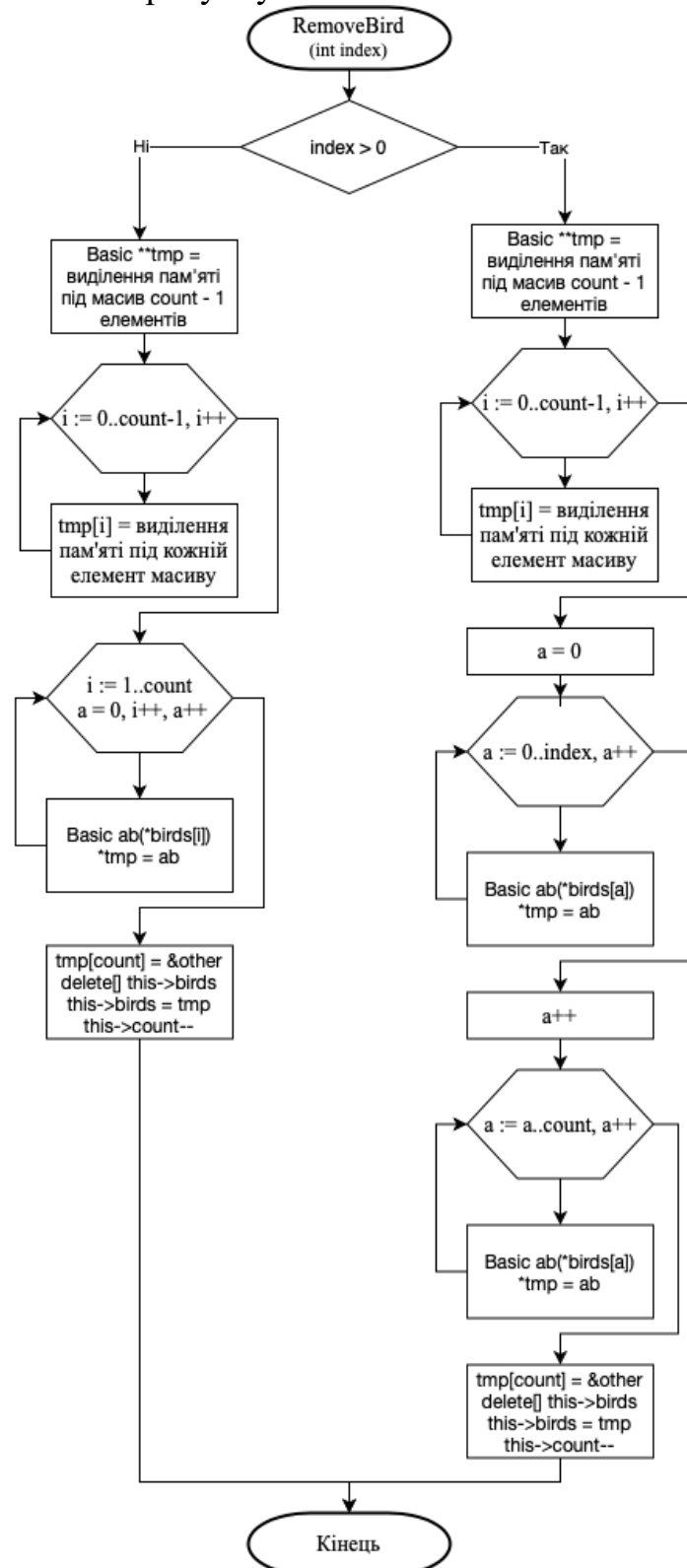
Рисунок 5 – блок-схема методу AddBird

`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



Рисунк 6 – блок-схема методу RemoveBird

`void ShowAll()` – метод виводу на екран усі елементів динамічного масиву об'єктів

Блок-схема показана на рисунку 7

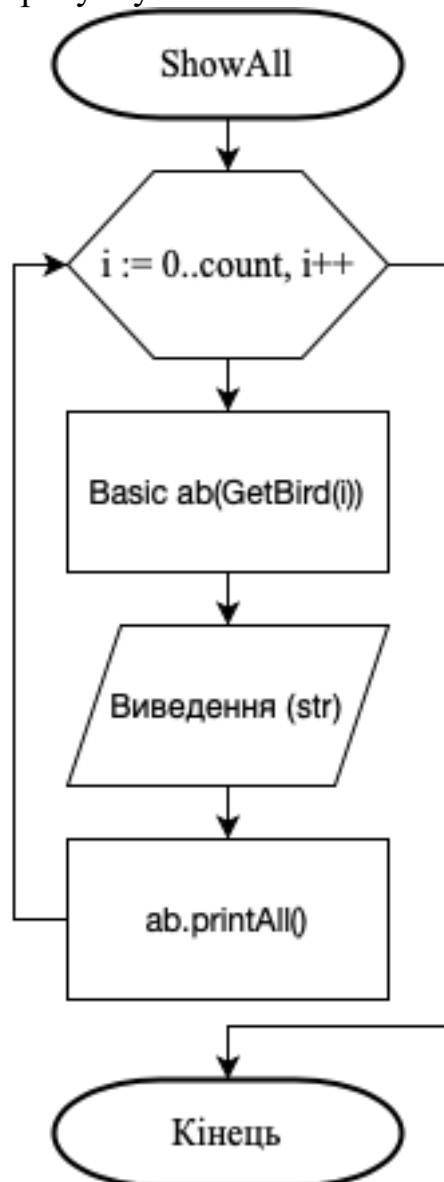


Рисунок 7 – блок-схема методу ShowAll

`void FindPercentage()` – метод знаходження відсоткового відношення чоловіків до жінок, де враховується усі елементи масиву. Блок-схема показана на рисунку 8

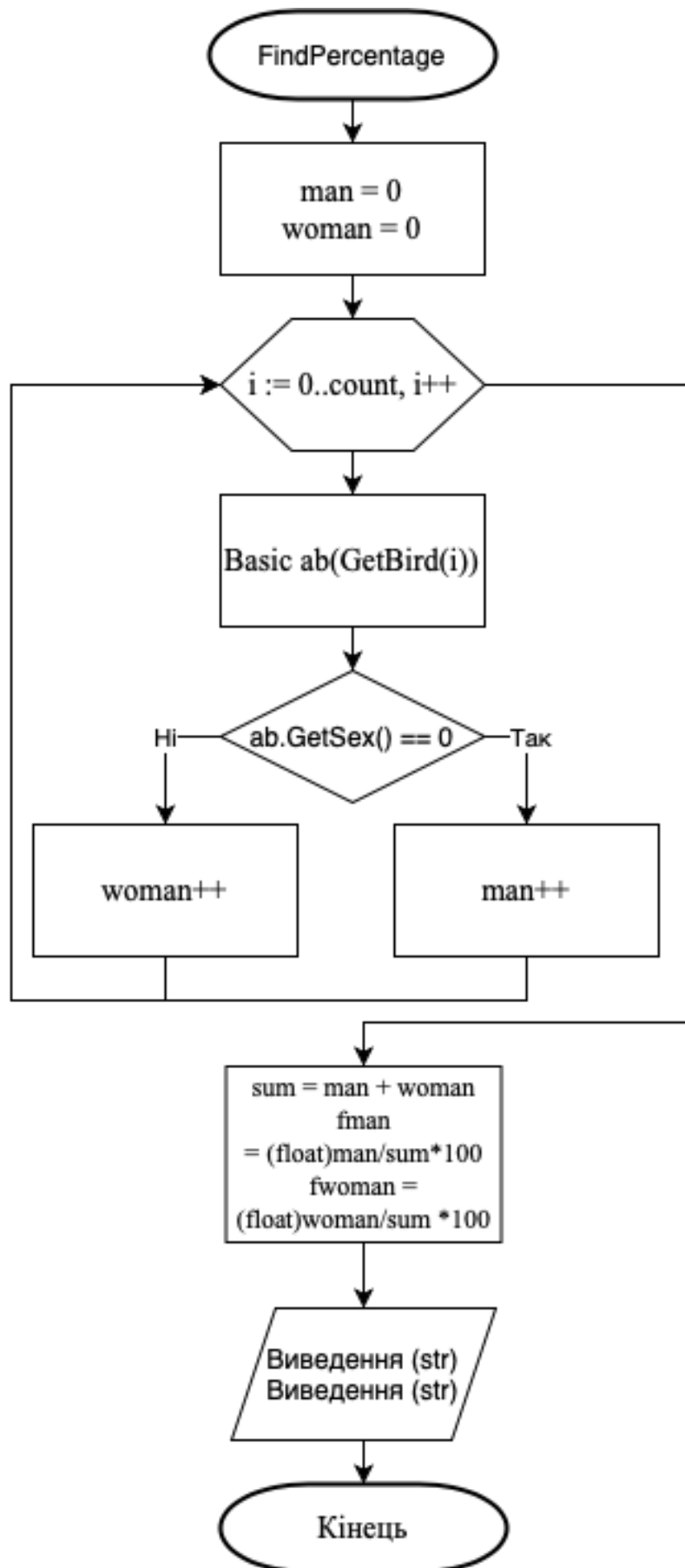


Рисунок 8 – блок-схема методу FindPercentage

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

2.2.3 Клас «базовий»

List Basic

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

Властивості класу:

```
enum Yes_no label - чи окільцьована птаха;  
char name[15] – 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, char name1[], int age1, Feature home1, Sex sex1)` – конструктор класу, який приймає інформація про об'єкт

Параметри:

```
Yes_no label1 - чи окільцьована птаха;  
char name1[] – назва виду птаха;  
int age1 – вік птаха  
Feature home1 - клас характеристик житла для птаха;  
Sex sex1 – стать птаха;
```

Блок-схема показана на рисунку 9

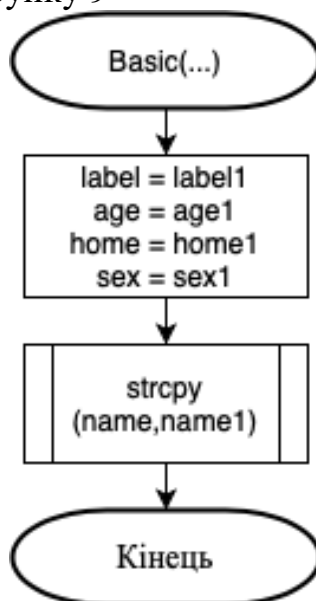


Рисунок 9 – блок-схема конструктору Basic

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

Параметри:

`Yes_no x` – чи окільцьована птаха;

Блок-схема показана на рисунку 10

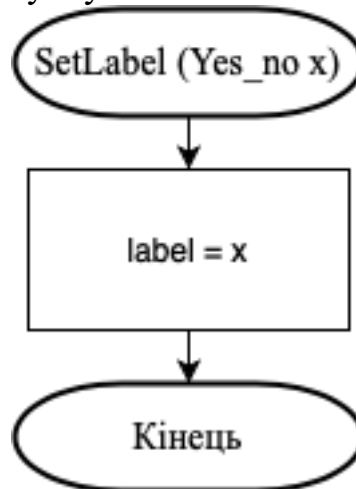


Рисунок 10 – блок-схема методу `SetLabel`

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

Блок-схема показана на рисунку 11

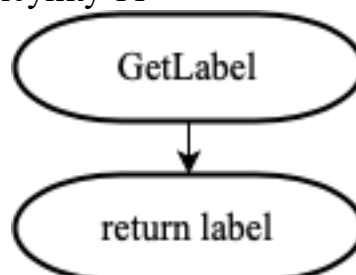


Рисунок 11 – блок-схема методу `GetLabel`

`void SetName (char n[15])` – метод присвоєння об'єктові, як називається вид птаха

Параметри:

`char n[15]` – назва виду птаха

Блок-схема показана на рисунку 12

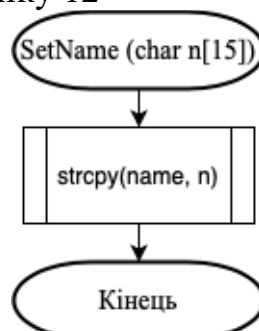


Рисунок 12 – блок-схема методу `SetName`

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

Параметри:

`Sex x` – стать птаха;

Блок-схема показана на рисунку 13

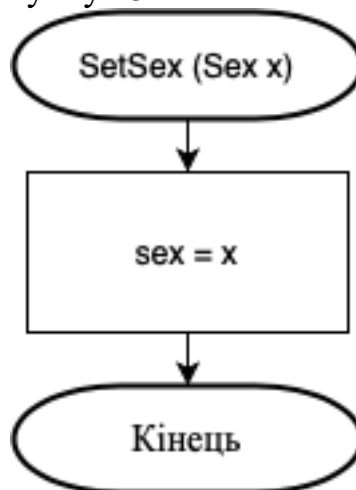


Рисунок 13 – блок-схема методу `SetSex`

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

Блок-схема показана на рисунку 14



Рисунок 14 – блок-схема методу `GetSex`

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

Параметри:

`int x` – вік птаха;

Блок-схема показана на рисунку 15

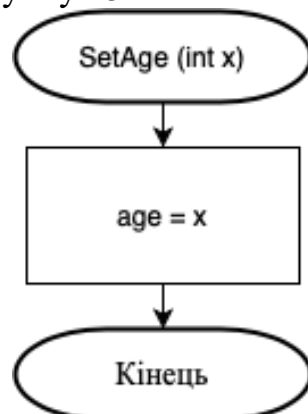


Рисунок 15 – блок-схема методу `SetAge`

`int GetAge() const` – отримання значення віку птаха
Блок-схема показана на рисунку 16

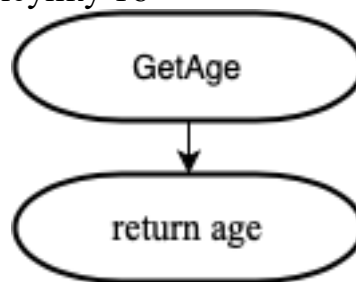


Рисунок 16 – блок-схема методу `GetAge`

`Feature GetHome() const` – отримання значення характеристик дому птаха
Блок-схема показана на рисунку 17



Рисунок 17 – блок-схема методу `GetHome`

`void printAll() const` – метод виведення інформація про об'єкт базового класу
Блок-схема показана на рисунку 18

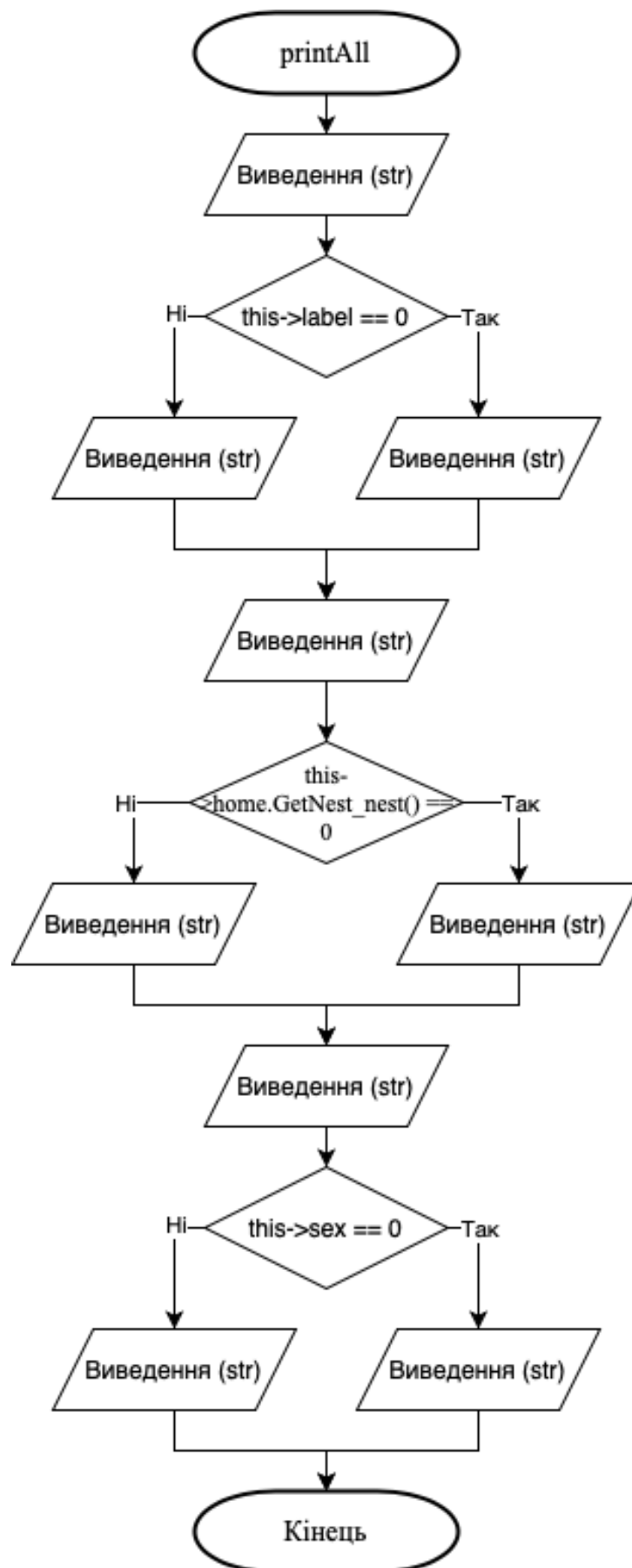


Рисунок 18 – блок-схема методу printAll

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

Параметри:

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

Блок-схема показана на рисунку 19

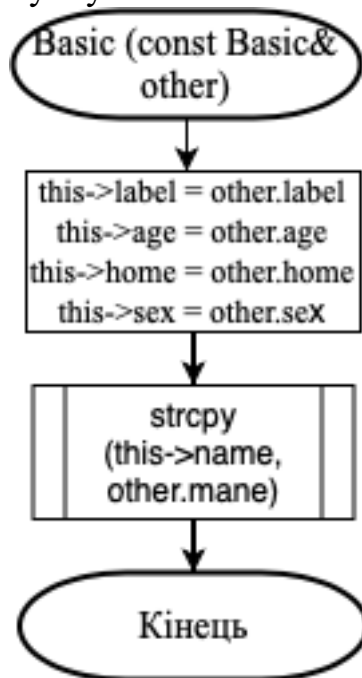


Рисунок 19 – блок-схема конструктору копіювання

`virtual ~Basic()` – деструктор базового класу

2.2.4 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 см²);
`int height1` – висота домівки;
`int number_of_feeders1` – кількість годівниць;
`enum Yes_no nest_nest1` – наявність гнізда;

Блок-схема показана на рисунку 20

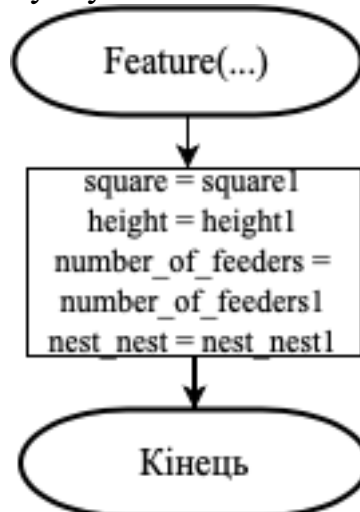


Рисунок 20 – блок-схема конструктору

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

Параметри:

`int x` – площа домівки;

Блок-схема показана на рисунку 21

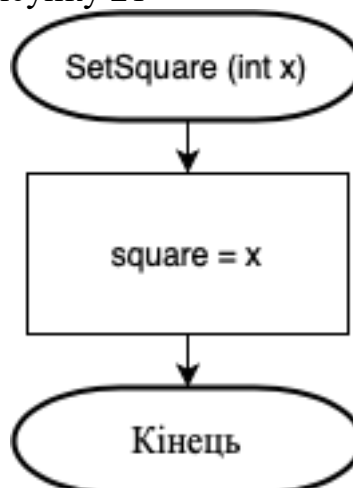


Рисунок 21 – блок-схема методу SetSquare

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

Блок-схема показана на рисунку 22

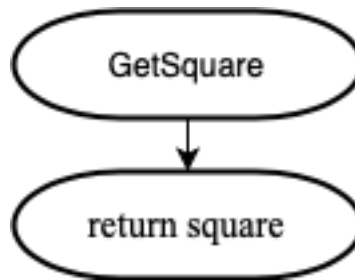


Рисунок 22 – блок-схема методу `GetSquare`

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

Параметри:

`int x` – площа домівки;

Блок-схема показана на рисунку 23

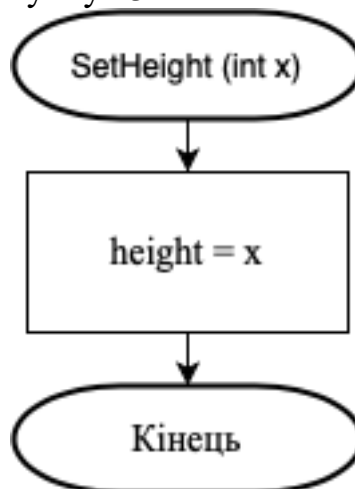


Рисунок 23 – блок-схема методу `SetHeight`

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

Блок-схема показана на рисунку 24

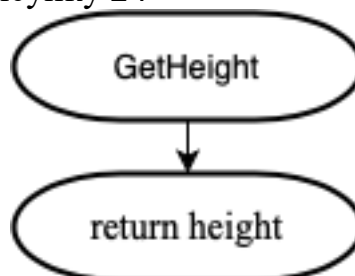


Рисунок 24 – блок-схема методу `GetHeight`

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

Параметри:

`int x` – кількість годівниць;

Блок-схема показана на рисунку 25

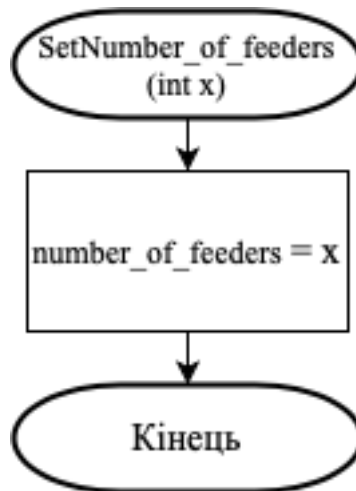


Рисунок 25 – блок-схема методу `SetNumber_of_feeders`

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

Блок-схема показана на рисунку 26

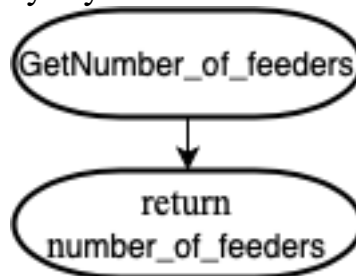


Рисунок 26 – блок-схема методу `GetNumber_of_feeders`

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

Параметри:

`Yes_no x` – наявність гнізда;

Блок-схема показана на рисунку 27

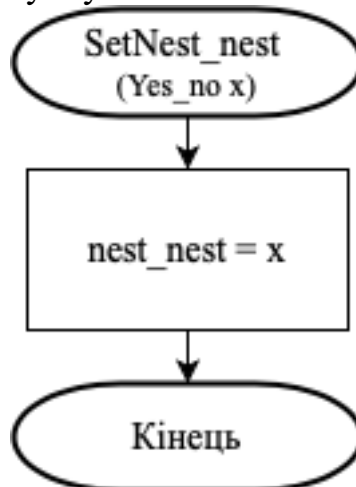


Рисунок 27 – блок-схема методу `SetNest_nest`

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

Блок-схема показана на рисунку 28

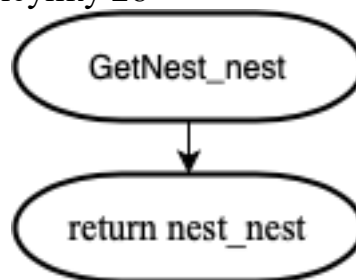


Рисунок 28 – блок-схема методу `GetNest_nest`

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

Параметри:

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

Блок-схема показана на рисунку 29

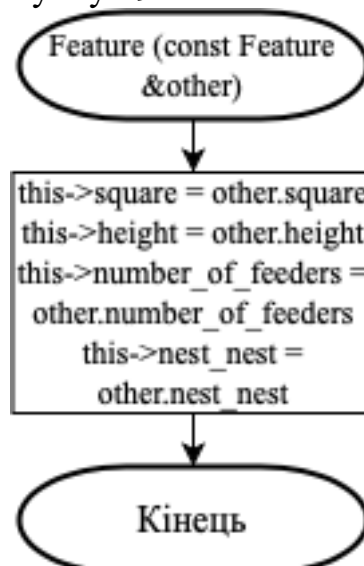


Рисунок 29 – блок-схема конструктору копіювання

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

2.3 Структура проекту

```
.
├── Doxyfile
├── Makefile
├── README.md
├── doc
│   ├── Radievych23.pdf
│   ├── assets
│   │   ├── (Basic)Basic.drawio
│   │   ├── (Basic)Basic.png
│   │   ├── (Basic)GetAge.drawio
│   │   ├── (Basic)GetAge.png
│   │   ├── (Basic)GetHome.drawio
│   │   └── (Basic)GetHome.png
```



```
└─ test
   └─ test.cpp
```

3 ВАРІАНТИ ВИКОРИСТАННЯ

Цю програму можна використовувати за для перепису усіх зареєстрованих птахів в програму та оперувати таким списком птахів, заздалегідь давши про них певну інформацію.

Результат роботи з doxygen продемонстровано на рисунку 30, рисунку 31 та рисунку 32, виконання модульних тестів на рисунку 33 та демонстрація відсутності витоків пам'яті на рисунку 34.

Lab23 1.0
ООП. Вступ до ООП

Титульна сторінкаДодаткова інформаціяСтруктури даних▼Файли▼

Q* Пошук

Lab23 Документація

Загальне завдання

1. **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 ініціалізації); - 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;

Автор
Radievych V.

Дата
20-май-2021

Версія
1.0

Створено системою **doxygen** 1.9.1

Рисунок 30 – робота з doxygen

Lab23 1.0
ООП. Вступ до ООП

Титульна сторінкаДодаткова інформаціяСтруктури даних▼Файли▼

Q* Пошук

Файли

Повний список файлів.

[рівень елемента 1 2]

▼ src

data.cpp

Файл з реалізацією функцій для data.h

data.h

Файл з описом класу птахів, перерахуванням критеріїв птахів та методами оперування птахами

list.cpp

Файл з реалізацією функцій для list.h

list.h

Файл з описом класу списку птахів та його методами

main.cpp

Файл з демонстрацією роботи списків ((Basic) та методи роботи з ними

▼ test

test.cpp

Файл з тестами на реалізації функцій оперування динамічним списком елементів структурами

Створено системою **doxygen** 1.9.1

Рисунок 31 – робота з doxygen

```
#include "list.h"
```

Діаграма включених заголовочних файлів для main.cpp:

Функції

```
int main ()
```

Головна функція. Детальніше...

Детальний опис

Файл з демонстрацією роботи списків ((Basic) та методи роботи з ними.

Автор

Radievych V.

Дата

20-may-2021

Версія

1.0

Опис функцій

main()

```
int main ()
```

Головна функція.

Послідовність дій:

- створення даних для класу (створення класу (List)робота з методами класу (List), що має в собі методи додавання, видалення елемента, отримання елемента по індексу (або ідентифікатору), вивід усіх елементів на екран успішний код повернення з програми (0)

Граф всіх викликів цієї функції:

Рисунок 32 – робота з doxygen

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

Рисунок 33 – робота з модульними тестами

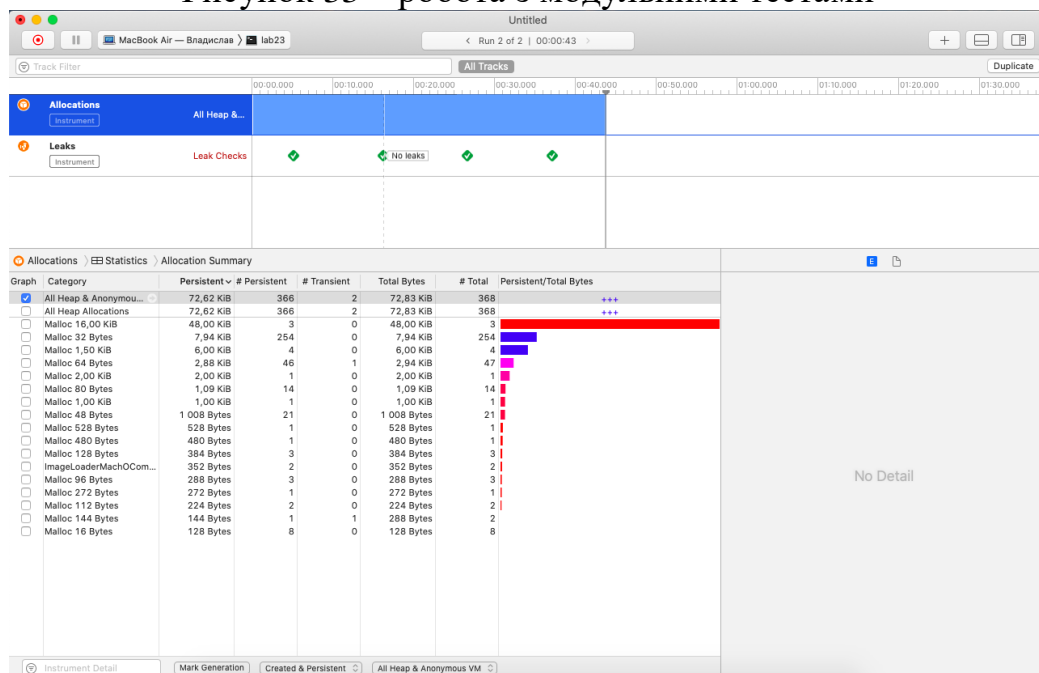


Рисунок 34 – демонстрація відсутності витоків пам'яті

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