НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ ТА ПРОГРАМУВАННЯ»

Розрахункове завдання з дисципліни

«Програмування ч.2»

Пояснювальна записка

КІТ.120А.23-01 90 01-1 -ЛЗ

Розробники

Виконав:

студент групи КІТ-120А

\_\_\_\_\_\_\_\_\_**\_**\_**\_\_\_\_**/ Старовойтов Н.А./

Перевірив:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Давидов В.В./

Харків – 2021

**Вступ**

Інформаційною системою називають сукупність організаційних і технічних засобів для збереження та обробки інформації з метою забезпечення інформаційних потреб користувачів.

Використання інформаційно-довідкової системи дозволяє вирішити проблему необхідністі накопичення великих об’ємів професійно цінної інформації.

Крім цього, інформаційно-довідкові системи надають можливість оперувати даними, забезпечують зручний і швидкий пошук необхідних відомостей та полегшують процес обробки статистичних даних.

**Призначення та галузь застосування**

Розроблена інформаційна система має колекцію птахів та методи роботи з нею. Серед базових методів є ті, що дають змогу: додати птаха до колекції; видалити певного птаха за індексом з колекції; очистити колекцію; отримати птаха по індексу; зчитати список птахів із файлу; вивести інформацію про конкретного птаха на екран; вивести весь список на екран або у файл.

Серед методів, що займаються обробкою інформації, можна виділити: пошук птаха з найдовшею зимівлею, знаходження відношення самок до самців у колекції та знаходження середнього віку всіх птахів, що не мають кільця.

Також є можливість сортування колекції залежно від заданого користувачем напрямку та за вказанимм критерієм.

Дану інформаційну систему можна застосовувати в роботі організацій, що займаються вивченням птахів або захистом їх рідкісних видів. Також розроблена інформаційна система може бути корисною для підприємців, що займаються розводом птахів – від фермерів до постачальників елітних порід та видів.

**Постановка завдання до розробки**

***Тема:*** Розробка інформаційно-довідкової системи

***Мета:*** Закріпити отримані знання з дисципліни «Програмування» шляхом виконання типового комплексного завдання.

***Загальне завдання***

1. З розділу ”Розрахункове завдання / Індивідуальні завдання”, відповідно до варіанта завдання, обрати прикладну галузь;
2. Для прикладної галузі розробити розгалужену ієрархію класів, що описана у завдані та складається з одного базового класу та двох спадкоємців. Класи повинні мати перевантажені оператори введення-виведення даних та порівняння;
3. Розробити клас-список List.[h/cpp], що буде включати до себе масив (STL-колекцію) вказівників до базового класу. А також базові методи роботи з списком: а) очистка списку б) відображення списку в) додання/видалення/отримання/оновлення елементу;
4. Розробити клас-контролер Controller.[h/cpp], що буде включати колекцію розроблених класів, та наступні методи роботи з цією колекцією: а) читання даних з файлу та ix запис у контейнер (STL-контейнер); б) запис даних з контейнера у файл; в) сортування елементів у контейнері за вказаними критеріями: поле та напрям сортування, які задаються користувачем з клавіатури; г) пошук елементів за вказаними критеріями (три критерія, що присутні у кожному варіанті);
5. Розробити клас Menu.[h/cpp], який має відображати діалогове меню для демонстрації реалізованих функцій класу контролера;
6. Оформити схеми алгоритмів функцій класів контролера (за необхідністю), тесту-контролера та діалогового меню;
7. Оформити документацію: пояснювальну записку.

***Додаткові вимоги на оцінку «відмінно»:***

* виконати перевірку вхідних даних за допомогою регулярних виразів.
* критерій для пошуку та сортування задавати у вигляді функтора;
* розробити клас-тестер контролеру ControllerTest.срр, основною метою якого буде перевірка коректності роботи класу-контролера.

***Індивідуальне завдання***

Варіант 23. "Птахи"

* Поля базового класу:
  + Чи окольцьована птаха (наприклад: так, ні)
  + Назва виду (наприклад: журавель, гусак)
  + Вік птаха, місяців (наприклад: 2, 6, 8)
  + Тип домівки птаха (структура, що містить площу у кв.см, висоту у см домівки птаха, а також кількість годівниць та наявність гнізда)
  + Стать птаха (один з переліку: чоловіча, жіноча)
* Спадкоємець 1 - Перелітні птахи. Додаткові поля:
  + Місяць відльоту у вирій (один з переліку: січень, лютий, березень, ... , грудень)
  + Місяць прильоту з вирію (один з переліку: січень, лютий, березень, ... , грудень)
* Спадкоємець 2 - Рюкзак з тканини. Додаткові поля:
  + Мінімальна комфортна для життя температура, градусів Цельсію (наприклад: -5, +10, +15)
  + Максимальна комфортна для життя температура, градусів Цельсію (наприклад: +5, +20, +40)
* Методи роботи з колекцією:

1. Знайти відсоткове відношення самок до самців у відділі
2. Знайти середній вік усіх не окольцьованих птахів
3. Знайти птаха із найдовшою зимівлею

**Опис вхідних та вихідних даних**

Під час запуску програми, необхідно ввести ім’я файлу, звідки будуть взяті вхідні дані. В файлі повинні бути наступні дані: першим повинно бути слово «Yes» або «No», що позначає окольцьованість птаха, далі назва (ім’я) птаха, його вік у місяцях, потім площа домівки, висота домівки, кількість годівниць, чи є домівка гніздом («Yes» або «No») та стать птаха («Male» або «Female»).

В залежності від типу птаха, після вищеназваних параметрів можуть вказуватися додаткові: місяць відльоту у вирій та місяць прильоту з вирію, або ж мінімальна та максимальна комфортні температури. При цьому явно вказувати тип птаха не потрібно.

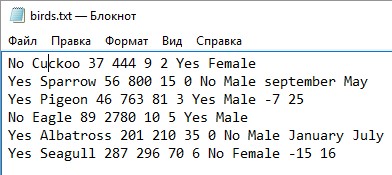


Рисунок 1 — Приклад вхідного файлу

До вихідного файлу дані заносяться в тому ж порядку та форматі, що і у вхідному файлі.

**Опис складу технічних та програмних засобів**

Програма виводить меню можливих дій с колекцією, та в залежності від отриманих від користувача даних виконує методи із загального та індивідуального завдань.

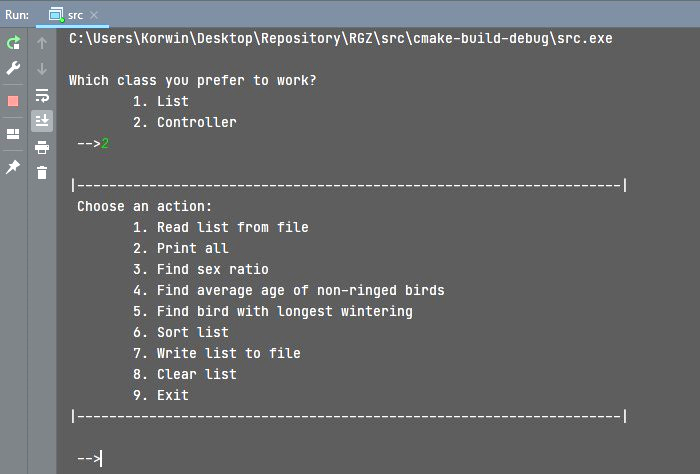


Рисунок 2 — Діалогове меню

Далі буде наведений опис декількох методів, що можуть бути викликані користувачем:

Метод Call\_Dialog\_Menu класа Menu в залежності від вибору користувача, викликає відповідні методи для роботи з класом CList або класом Controller.

Метод Find\_Sex\_Ratio класа Controller знаходить відношення кількості самок до самців у списку та виводить його у консоль.

Метод Read\_From\_File класа Controller зчитує дані з файла, та, у випадку коректності вхідних даних (перевірка за допомогою метода Regex\_Check), формує птаха та заносить його у список (за допомогою метода Add\_Bird).

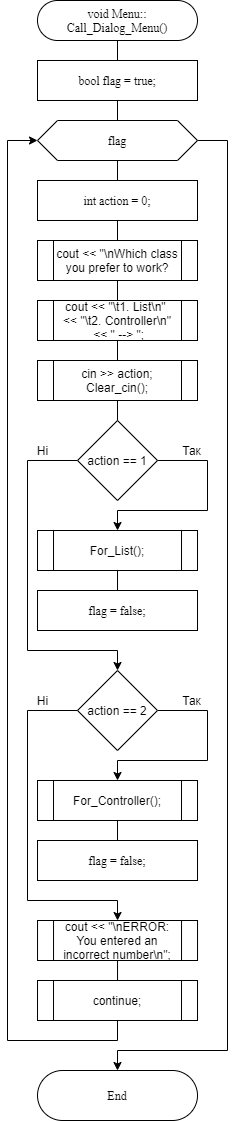
****

Рисунок 3 — Метод Call\_Dialog\_Menu класа Menu

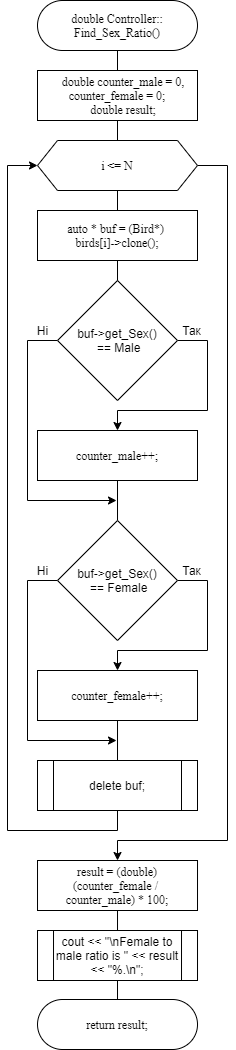
****

Рисунок 4 — Метод Find\_Sex\_Ratio класа Controller

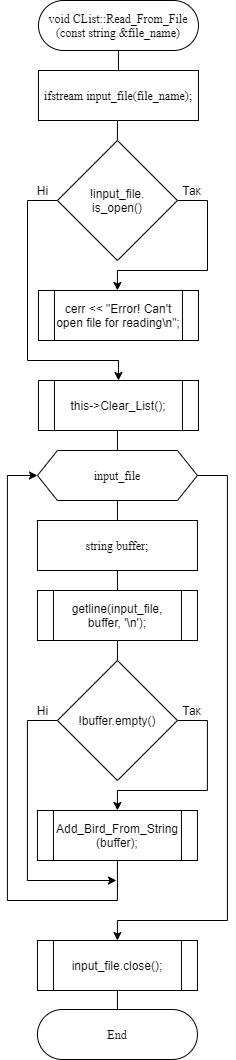
****

Рисунок 5 — Метод Read\_From\_File класа Controller

**Список джерел інформації**

* <https://prog-cpp.ru/>
* <https://ravesli.com/>
* <https://coderoad.ru/>

**Висновки**

Під час виконання даного розрахункового завдання було закріплено отримані знання з дисципліни «Програмування» та отримано практичні навички виконання типового комплексного завдання.

**Додаток А. Реалізація методів Find\_Bird\_With\_Longest\_Wintering(), Find\_Average\_Age\_of\_not\_LOTR() та Sort()**

**Virtual\_Bird\* Controller::Find\_Bird\_With\_Longest\_Wintering() const{**

**int buffer\_result = 0, result = 0, number\_of\_result = 0;**

**bool flag = false;**

**for(int i = 0; i <= N; i++){**

**if (birds[i]->GetType() == 'M') {**

**flag = true;**

**auto \* buf = (Migrant\*) birds[i]->clone();**

**buffer\_result = buf->get\_arrival() - buf->get\_departure();**

**if (buffer\_result < 0) buffer\_result += 12;**

**if (buffer\_result > result) {**

**result = buffer\_result;**

**number\_of\_result = i;**

**}**

**delete buf;**

**}**

**}**

**if (!flag) {**

**cout << "\nNo migrant birds in list\n";**

**return nullptr;**

**}**

**auto \* buf = (Migrant\*) birds[number\_of\_result]->clone();**

**cout << "\nBird with longest wintering is:\n " << buf << endl;**

**delete buf;**

**return birds[number\_of\_result];**

**}**

**double Controller::Find\_Average\_Age\_of\_not\_LOTR() const{**

**double counter\_age = 0, counter\_amount = 0;**

**double result;**

**for(int i = 0; i <= N; i++){**

**auto \* buf = (Bird\*) birds[i]->clone();**

**if (!buf->get\_LOTR()) {**

**counter\_age += buf->get\_age();**

**counter\_amount++;**

**}**

**delete buf;**

**}**

**result = (double) (counter\_age / counter\_amount);**

**cout << "\nAverage age of non-LOTR birds is " << result << " months.\n";**

**return result;**

**}**

**void Controller::Sort(bool direction, int criterion) {**

**Functor functor;**

**functor.set\_criterion(criterion);**

**std::sort(birds.begin(), birds.end(), functor);**

**if (direction) std::reverse(birds.begin(), birds.end());**

**}**

**Додаток Б. Реалізація методу For\_Controller()**

**int Menu::For\_Controller() const {**

**Controller List;**

**while (true) {**

**int action;**

**cout << "\n|--------------------------------------------------------------------|\n";**

**cout << " Choose an action: \n";**

**cout << "\t1. Read list from file\n";**

**cout << "\t2. Print all\n";**

**cout << "\t3. Find sex ratio\n";**

**cout << "\t4. Find average age of non-ringed birds\n";**

**cout << "\t5. Find bird with longest wintering\n";**

**cout << "\t6. Sort list\n";**

**cout << "\t7. Write list to file\n";**

**cout << "\t8. Clear list\n";**

**cout << "\t9. Exit\n";**

**cout << "|--------------------------------------------------------------------|\n\n";**

**cout << " --> ";**

**cin >> action;**

**switch (action) {**

**case 1:{**

**cout << "Enter input file name\n";**

**cout << " --> ";**

**string filename;**

**cin >> filename;**

**List.Read\_From\_File(filename);**

**cout << "\nList successfully filled\n";**

**break;}**

**case 2:**

**cout << "\nList:\n";**

**List.Print\_All();**

**break;**

**case 3:**

**List.Find\_Sex\_Ratio();**

**break;**

**case 4:**

**List.Find\_Average\_Age\_of\_not\_LOTR();**

**break;**

**case 5:**

**List.Find\_Bird\_With\_Longest\_Wintering();**

**break;**

**case 6:{**

**cout << "\nDo you want to invert sort direction? (Yes/No)\n";**

**cout << " --> ";**

**string answer;**

**cin >> answer;**

**cout << " Choose your criterion: \n";**

**cout << "\t1. Ringed or not ringed\n";**

**cout << "\t2. Name\n";**

**cout << "\t3. Age\n";**

**cout << "\t4. Space of home\n";**

**cout << "\t5. Height of home\n";**

**cout << "\t6. Count of feeders in home\n";**

**cout << "\t7. Is home a nest\n";**

**cout << "\t8. Sex\n";**

**cout << "\t9. Departure month\n";**

**cout << "\t10. Arrival month\n";**

**cout << "\t11. Minimal living temperature\n";**

**cout << "\t12. Maximal living temperature\n";**

**cout << "\t13. I'm afraid and want to exit\n";**

**cout << " --> ";**

**int criterion = 1;**

**bool direction = false;**

**cin >> criterion;**

**Clear\_cin();**

**if (answer == "Yes" || answer == "yes") direction = true;**

**if (criterion == 13) continue;**

**List.Sort(direction, criterion - 1);**

**cout << "\nList successfully sorted\n";**

**break;}**

**case 7:{**

**cout << "Enter output file name\n";**

**cout << " --> ";**

**string filename;**

**cin >> filename;**

**List.Write\_To\_File(filename);**

**cout << "\nList successfully writed to output file\n";**

**break;}**

**case 8:**

**List.Clear\_List();**

**cout << "\nList successfully cleared\n";**

**break;**

**case 9:**

**List.Clear\_List();**

**cout << "\nShutting down...\n";**

**return 0;**

**default:**

**cout << "\nERROR: You entered an incorrect number\n";**

**Clear\_cin();**

**break;**

**}**

**}**

**}**

**Додаток В. Реалізація методу For\_List()**

**int Menu::For\_List() const{**

**CList List;**

**Bird bird1 {false, "Cockoo", 123, {444, 9, 2, true}, Female};**

**Bird bird2 {true, "Sparrow", 37, {800, 15, 0, false}, Male};**

**Bird bird3 {true, "Pigeon", 46, {763, 81, 3, true}, Male};**

**Bird bird4 {false, "Eagle", 890, {2780, 10, 5, true}, Male};**

**while (true) {**

**int action;**

**cout << "\n|--------------------------------------------------------------------|\n";**

**cout << " Choose an action: \n";**

**cout << "\t1. Start filling\n";**

**cout << "\t2. Print all\n";**

**cout << "\t3. Get element by index\n";**

**cout << "\t4. Get amount of elements in list\n";**

**cout << "\t5. Add element\n";**

**cout << "\t6. Delete element\n";**

**cout << "\t7. Clear list\n";**

**cout << "\t8. Exit\n";**

**cout << "|--------------------------------------------------------------------|\n\n";**

**cout << " --> ";**

**cin >> action;**

**switch (action) {**

**case 1:**

**List.Add\_Bird(&bird1);**

**List.Add\_Bird(&bird2);**

**List.Add\_Bird(&bird3);**

**cout << "\nList successfully filled\n";**

**break;**

**case 2:**

**cout << "\nList:\n";**

**List.Print\_All();**

**break;**

**case 3:**

**cout << "Enter element index (start from 0)\n";**

**cout << " --> ";**

**int index;**

**cin >> index;**

**Clear\_cin();**

**if (!Index\_Check(index, List.Get\_N(), 0)) continue;**

**cout << "\nAsked element:\n";**

**List.Get\_Bird\_by\_index(index)->Print();**

**break;**

**case 4:**

**cout << "Amount of elements in list: " << List.Get\_N() + 1 << endl;**

**break;**

**case 5:**

**cout << "Element, that will be added:\n";**

**bird4.Print();**

**cout << "\nEnter position where you want to insert\n";**

**cout << " --> ";**

**int index1;**

**cin >> index1;**

**Clear\_cin();**

**if (!Index\_Check(index1, List.Get\_N(), 1)) continue;**

**List.Add\_Bird\_by\_index(&bird4, index1);**

**cout << "\nElement successfully added\n";**

**break;**

**case 6:**

**cout << "\nEnter position where you want to delete\n";**

**cout << " --> ";**

**int index2;**

**cin >> index2;**

**Clear\_cin();**

**if (!Index\_Check(index2, List.Get\_N(), 0)) continue;**

**List.Delete\_Bird(index2);**

**cout << "\nElement successfully deleted\n";**

**break;**

**case 7:**

**List.Clear\_List();**

**cout << "\nList successfully cleared\n";**

**break;**

**case 8:**

**List.Clear\_List();**

**cout << "\nShutting down...\n";**

**return 0;**

**default:**

**cout << "\nERROR: You entered an incorrect number\n";**

**Clear\_cin();**

**break;**

**}**

**}**

**}**