Лабораторна робота №23.
Вступ до ООП.

- **Вимоги**
- **Розробник**
  - Гладков Костянтинт Сергіович
  - Студент групи КІТ-320;
  - 01-march-2021.
- **Основне завдання**

Загальне завдання.

Поширити попередню лабораторну роботу таким чином:

- використання функцій printf/scanf замінити на використання cin/cout;
- усі конкатенації рядків замінити на використання stringstream;
- замінити метод виводу інформації про об'єкт на метод, що повертає рядок-інформацію про об'єкт, який далі можна виводити на екран;

```
std::string MyObject::toString();
```

- замінити метод вводу інформації про об'єкт на метод, що приймає рядок з інформацією про об'єкт, обробляє його та створює об'єкт на базі цієї інформації;
- поширити клас-список, шляхом реалізації методів роботи з файлами за допомогою файлових потоків (fstream) (якщо використовувалися функції fprintf/fscanf – замінити їх на класи ifsteam/ofstream), при цьому сигнатури методів повинні виглядати таким чином:
  - читання (List – клас-список об'єктів, при цьому слід пам'ятати, що при повторному читанні з файлу, попередні дані списку повинні бути очищені):

```
void List::readFromFile(string fileName);
```

  - запис:

```
void List::writeToFile(string fileName);
```

- продемонструвати відсутність витоків пам'яті;
- продемонструвати роботу розроблених методів за допомогою модульних тестів;
- не використовувати конструкцію "using namespace std;", замість цього слід робити "using" кожного необхідного класу: `using std::string`, `using std::cout`;

- **Опис роботи**

Програма працює за тим же принципом, що і колишні, однак використовує ООП, полегшуючи роботу. Також там присутній читання з файлу і запис в нього (ну і інші слабкі зміни)

- **Функціональне призначення**

Дана програма може бути використана для маніпуляцій з нашим динамічним масивом, який має в собі класи. Таким чином ми отримуємо великий функціонал на увазі більше кількості можливих методів для роботи з нашим масивом. Також там присутній читання з файлу і запис в нього (ну і інші слабкі зміни)

- **Опис логічної структури**

Данна програма працює за принципом створення класу, в якому зберігається динамічний масив іншого класу, а також за допомогою методів, які контролюється цей масив. Також там присутній читання з файлу і запис в нього (ну і інші слабкі зміни)

Функція main містить в собі виклик функцій, а також корисних властивостей для функцій

class List - клас, який зберігає в собі динамічний масив іншого класу, а також які всередині себе має всі методи роботи з даним класом.

class Agency - базовий клас.

void List::expand() - метод, який використовується для розширення обсягу нашого динамічного масиву

void List::add_ag - метод, який використовується для розширення нашого масиву, а точніше додавання в даній оновлений масив нового елемента за рахунок спеціального коструктора

Agency& List::get_ag - метод, який дозволяє отримати елемент масиву за індексом.

void List::remove_last_ag() - метод, який прибирає останнім агенство з масиву.

int List::get_oldest_ag() - метод, який дозволяє знайти самий старший елемент в масиві (індекс).

void List::read_from_file - метод для читання з файлу

void List::write_to_file - метод для запис у файл

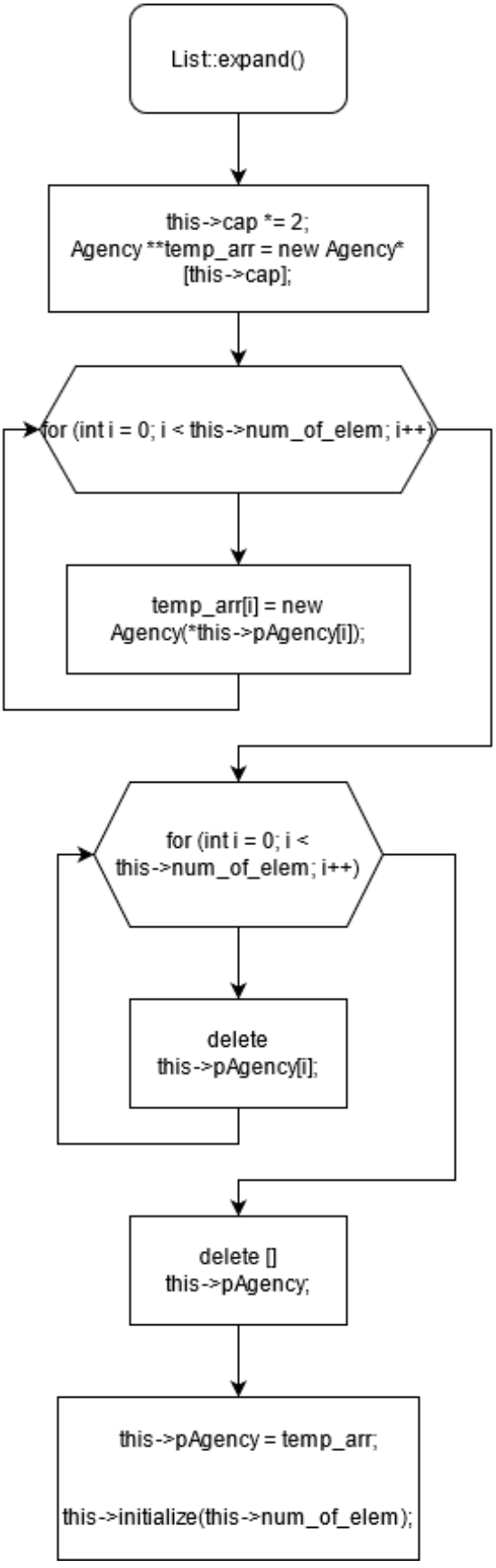

рис. 1 - void List::expand()

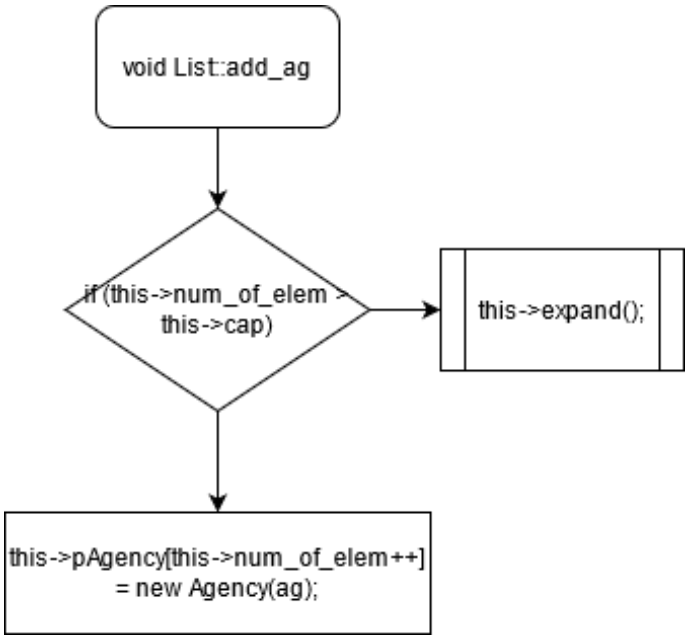

рис. 2 - void List::add_ag

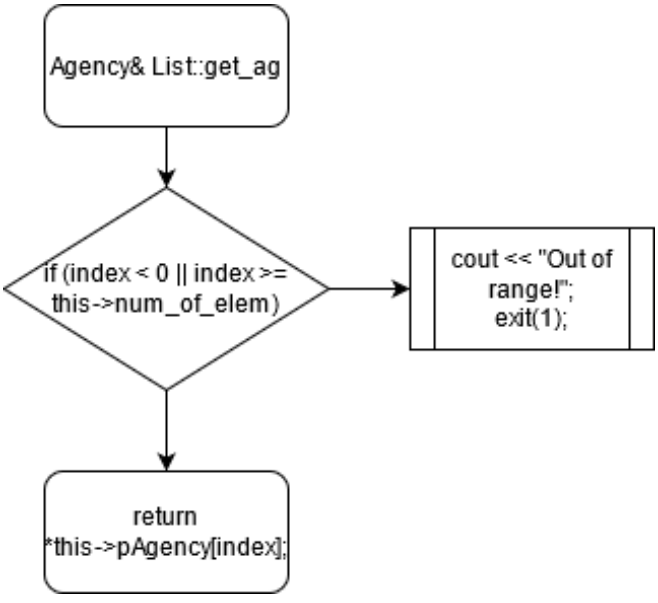

рис. 3 - Agency& List::get_ag

рис. 4 - void List::remove_last_ag()

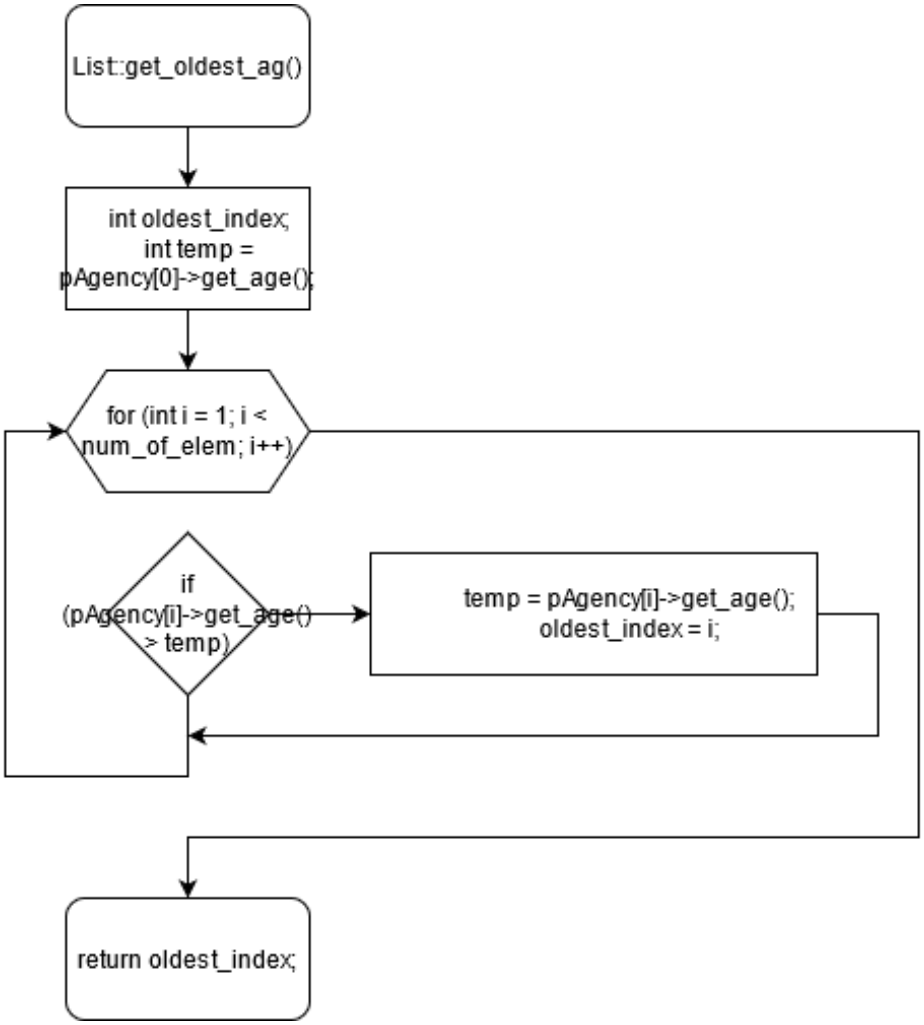

рис. 5 - int List::get_oldest_ag()

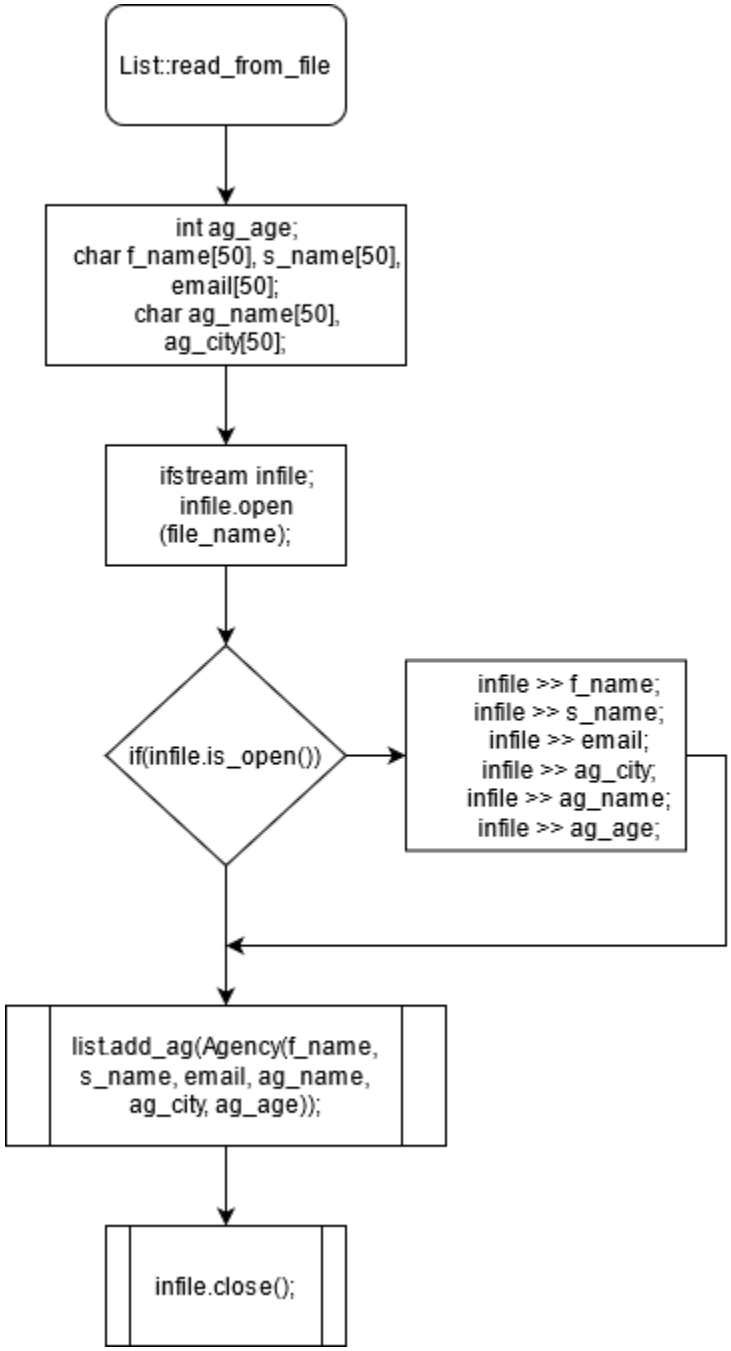

рис. 6 - List::read_from_file

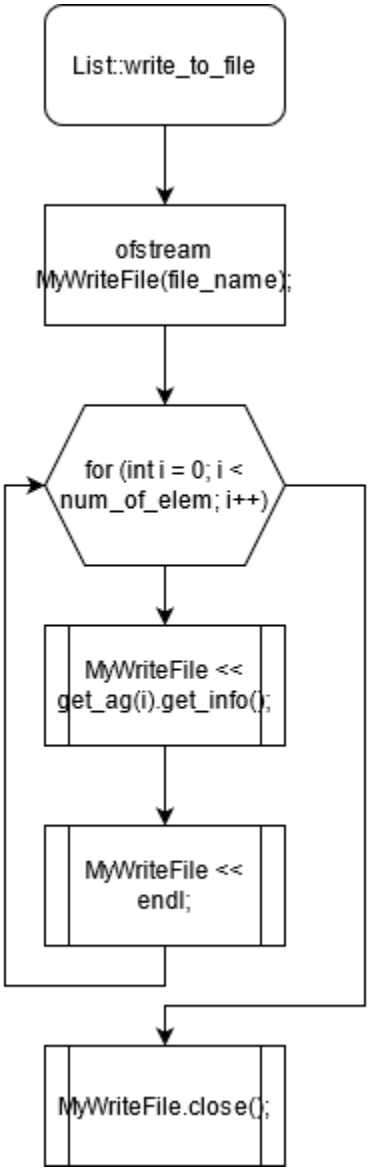

рис. 7 - List::write_to_file

- **Важливі елементи програми**

```
List my_list;

my_list.add_ag(Agency("Kostya", "Gladkov", "Email", "help", "London", 10));
my_list.add_ag(Agency("Someone", "Noone", "Email", "nope", "London", 15));

for (int i = 0; i < my_list.get_num(); i++){
    cout << my_list.get_ag(i).get_info() << "\n";
}
```

```
this->cap *= 2;

Agency **temp_arr = new Agency*[this->cap];

for (int i = 0; i < this->num_of_elem; i++)
{
    temp_arr[i] = new Agency(*this->pAgency[i]);
}

for (int i = 0; i < this->num_of_elem; i++)
{
    delete this->pAgency[i];
}
delete [] this->pAgency;

this->pAgency = temp_arr;

this->initialize(this->num_of_elem);
```

- **Варіанти використання**

Дана програма може бути використана для роботи з базовим класом, а також класом-списком, який буде зберігати в собі динамічний масив базового класу. Також дані можуть братися з файлу і в кінці туди записуватися.

Висновки

У даній лабораторній роботі були придбані знання роботи з ООП і отримав досвід роботи з файлами в с ++