Лабораторна робота №26.
Спадкування.

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

Лабораторна робота №26. ООП. Спадкування

Загальне завдання

Модернізувати попередню лабораторну роботу шляхом:

- додавання класів-спадкоємцев з розділу "Розрахункове завдання / Індивідуальні завдання", котрі будуть поширювати функціонал "базового класу" відповідно до індивідуального завдання;
- додавання ще класу-списку для кожного класу-спадкоємцю, що буде керувати лише елементами стосовного класу-спадкоємця;

Додаткові умови виконання завдання:

- 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;
- у проекті не повинні використовуватися бібліотеки введення / виведення мови С, а також не повинні використовуватися рядки типу `char*`.

- **Опис роботи**

Програма працює за тим же принципом, що і колишні, однак використовує ООП, полегшуючи роботу. А за рахунок успадкування створюються додаткові класи і класи списки до них.

- **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. А за рахунок успадкування створюються додаткові класи і класи списки до них.

- **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.

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.

class Law_agency - класи спадкоємці agency

class Mar_agency - класи спадкоємці agency

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

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

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 defence_in_court() const - метод, який знаходить агенства, який займаються захистом в суді

void kharkov_agencies() const - метод, який знаходить агенства, які знаходяться в харкові

int most_won_cases() const - метод, який знаходить агенство з найбільшою кількістю перемог

void no_weekends_mar_agencies() const - метод, який знаходить агенства, які працюють без вихідних

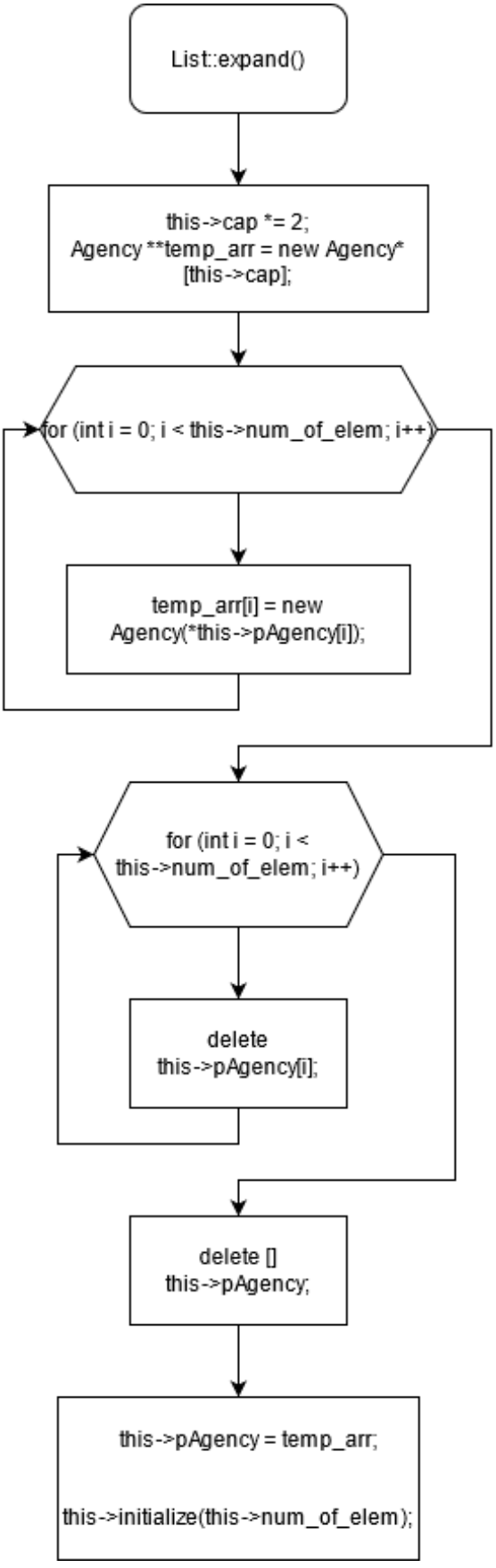

рис. 1 - void List::expand()

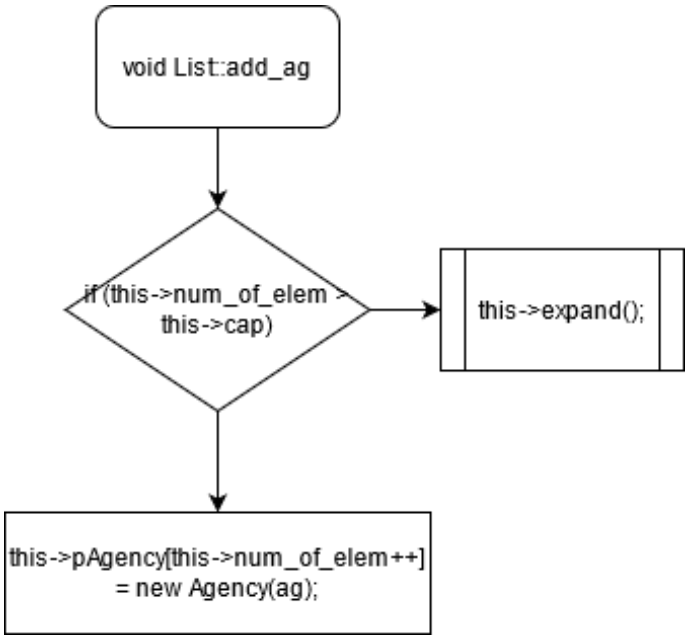

рис. 2 - void List::add_ag

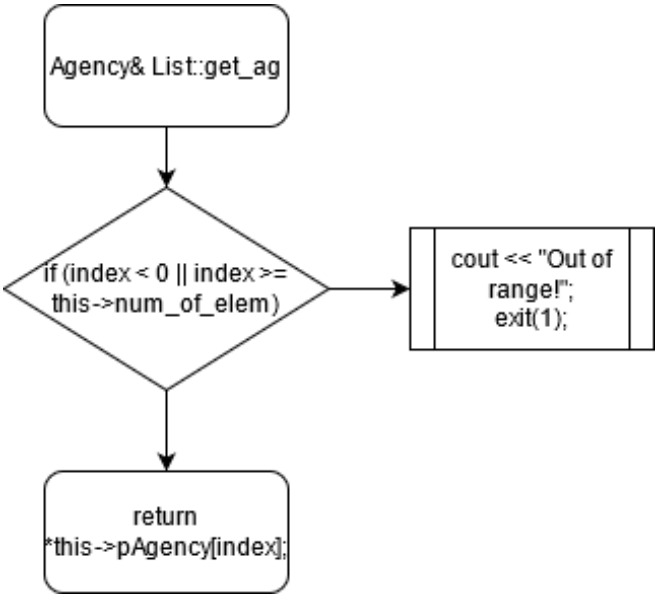

рис. 3 - Agency& List::get_ag

рис. 4 - void List::remove_last_ag()

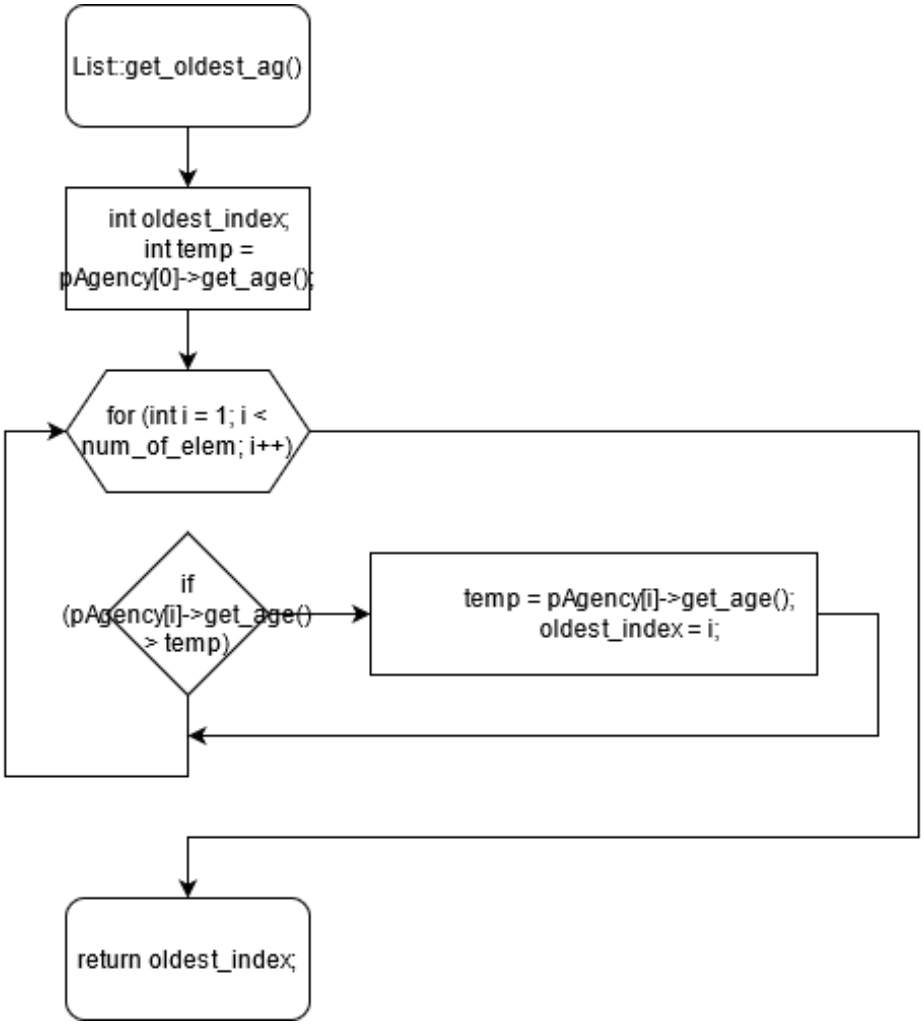

рис. 5 - int List::get_oldest_ag() const

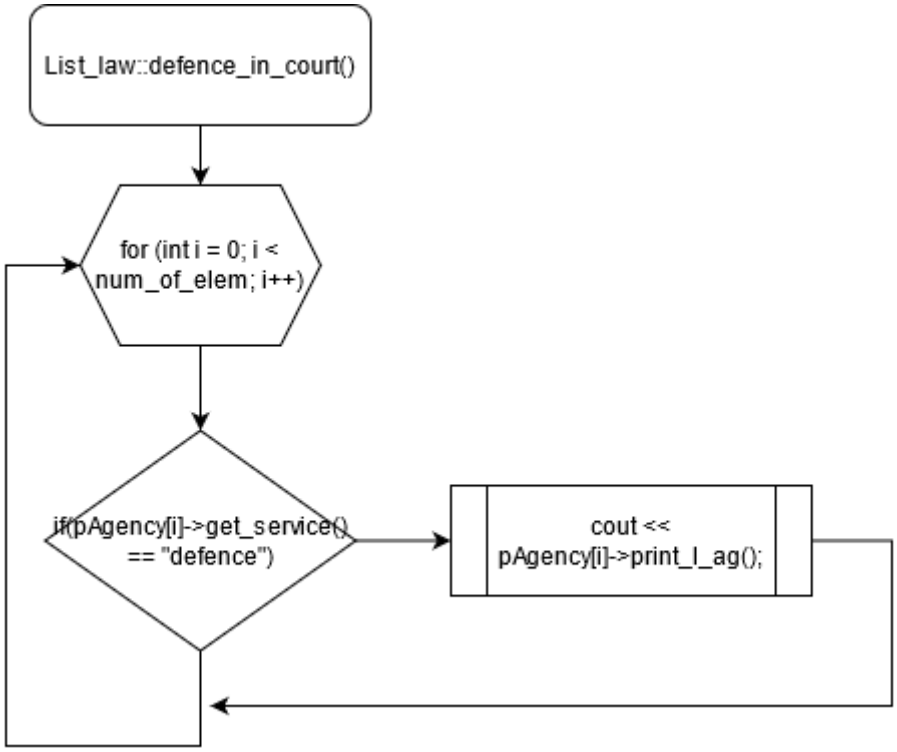

рис. 6 - void defence_in_court() const

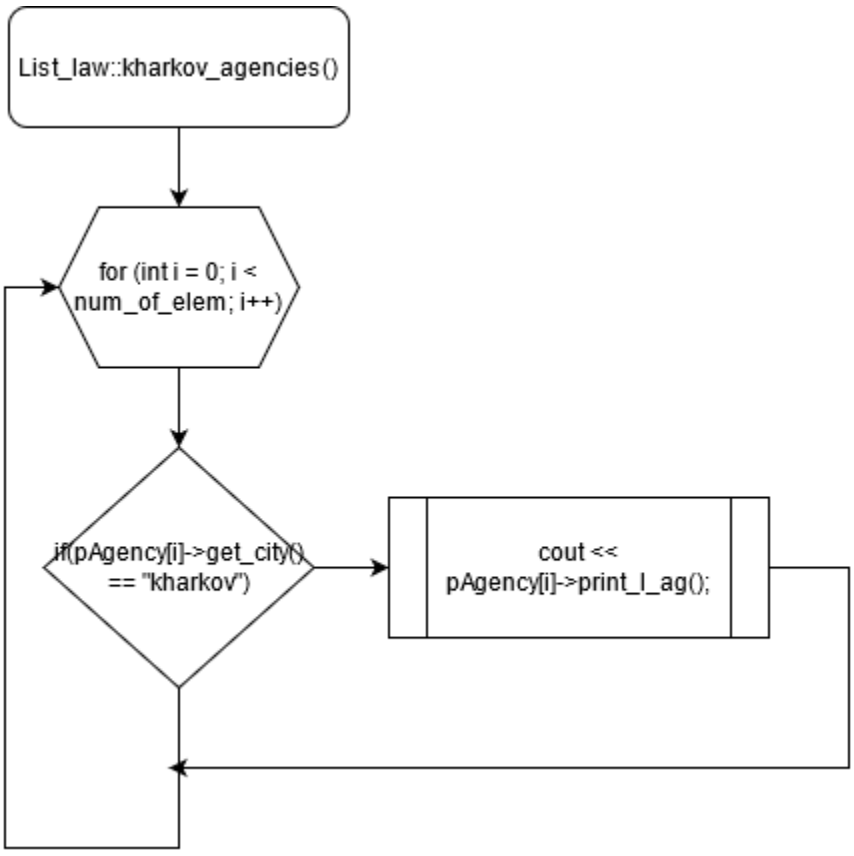

рис. 7 - void kharkov_agencies() const

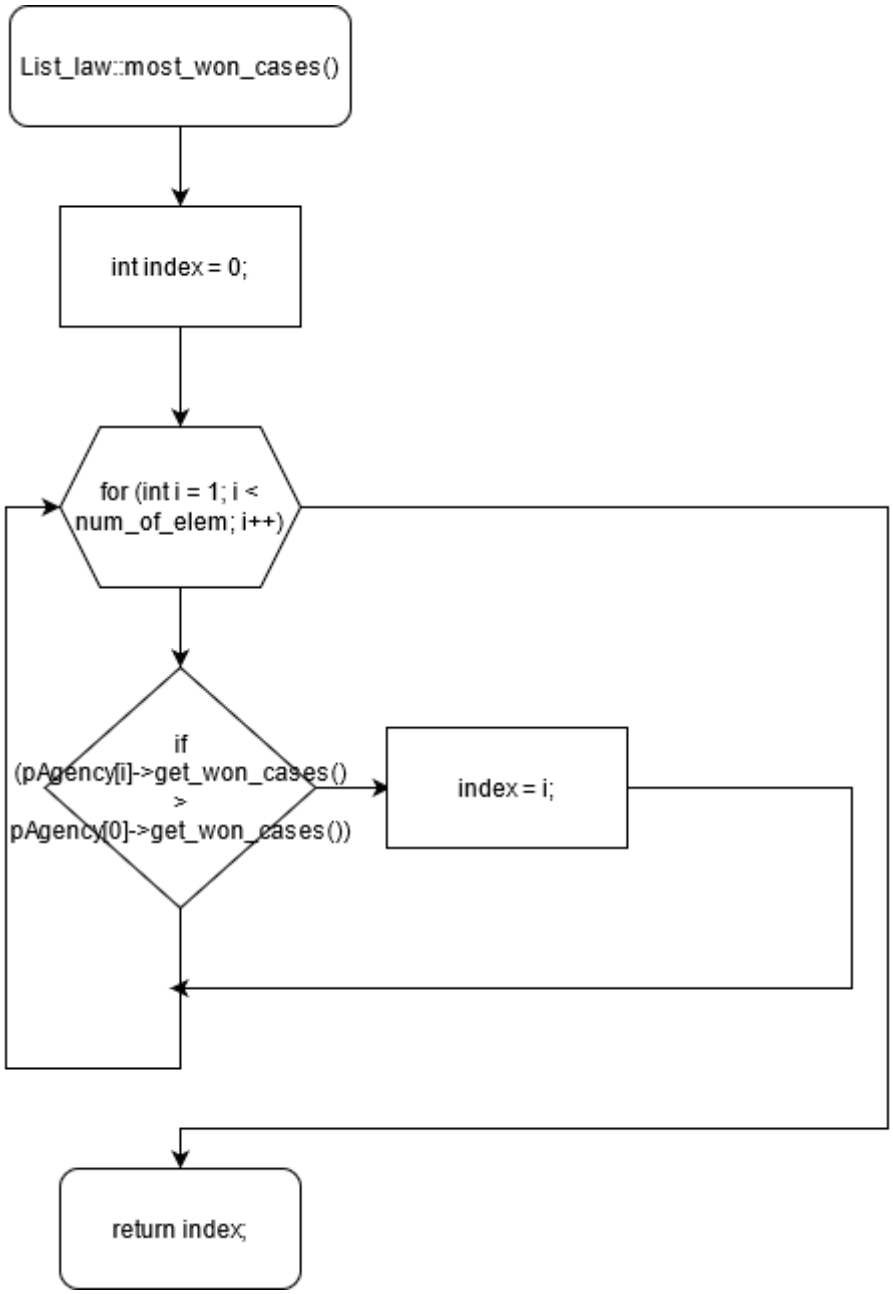

рис. 8 - int most_won_cases() const

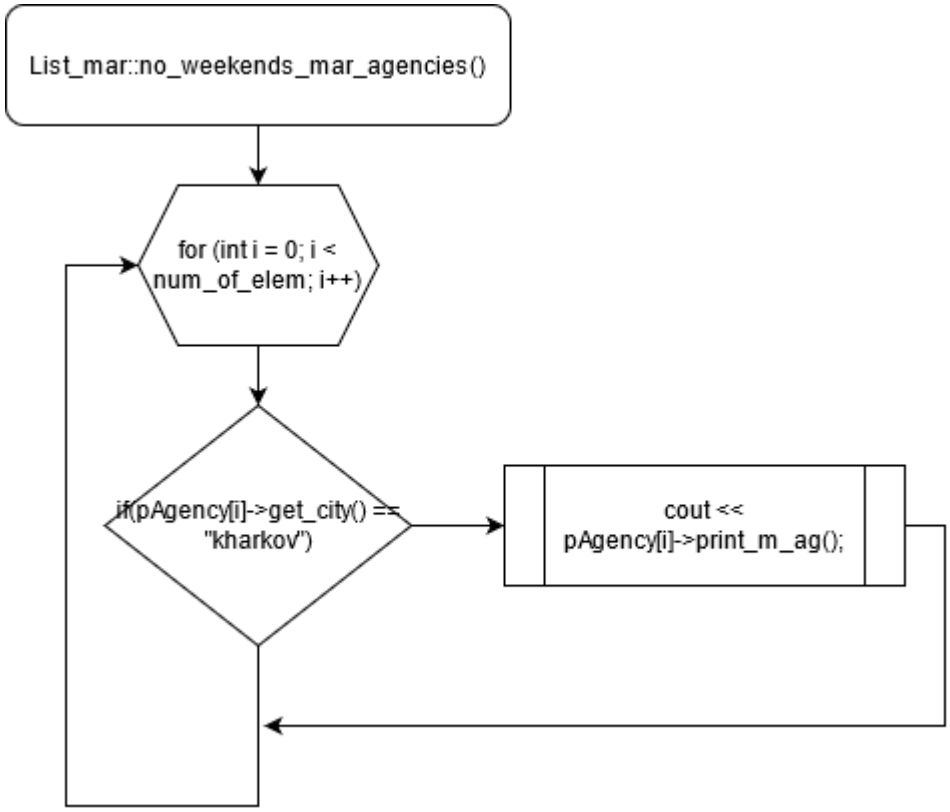

рис. 9 - void no_weekends_mar_agencies() const

- **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, його спадкоемцями, а також класами-списками, які будуть зберігати в собі динамічний масив класу.

Висновки

У даній лабораторній роботі були придбані знання роботи з спадкуванням ООП.