Лабораторна робота №23.
STL 2.

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

Загальне завдання

Розшири попередню роботу, зробивши клас-список STL ітеративним. Для демонстрації ітеративності, наступний код повинен працювати (Object - тип поточного базового класу):

```
MyList<Object> list;
for (Object &o : list) {
    // actions with object
}
```

- **Опис роботи**

Програма працює за принципом успадкування та поліморфізму об'єкта. Таким чином маємо великий функціонал з маленькою навантаженням.

- **Функціональне призначення**

Програма може бути використана для створення сховища з об'єктом, який потрібен вам, а також його подальшим користуванням.

- **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.

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

void kharkov_agencies() - Функція знаходить все агенції в харкові

int most_wins_law() - метод знаходить агенство з Найбільший числом перемог

void defence_in_court() - метод знаходить все агенції із захистом в суді

void no_weekends_mar_agencies() - метод знаходжу агенства, які працюють без вихідних

void List::iterate() - итерирование.

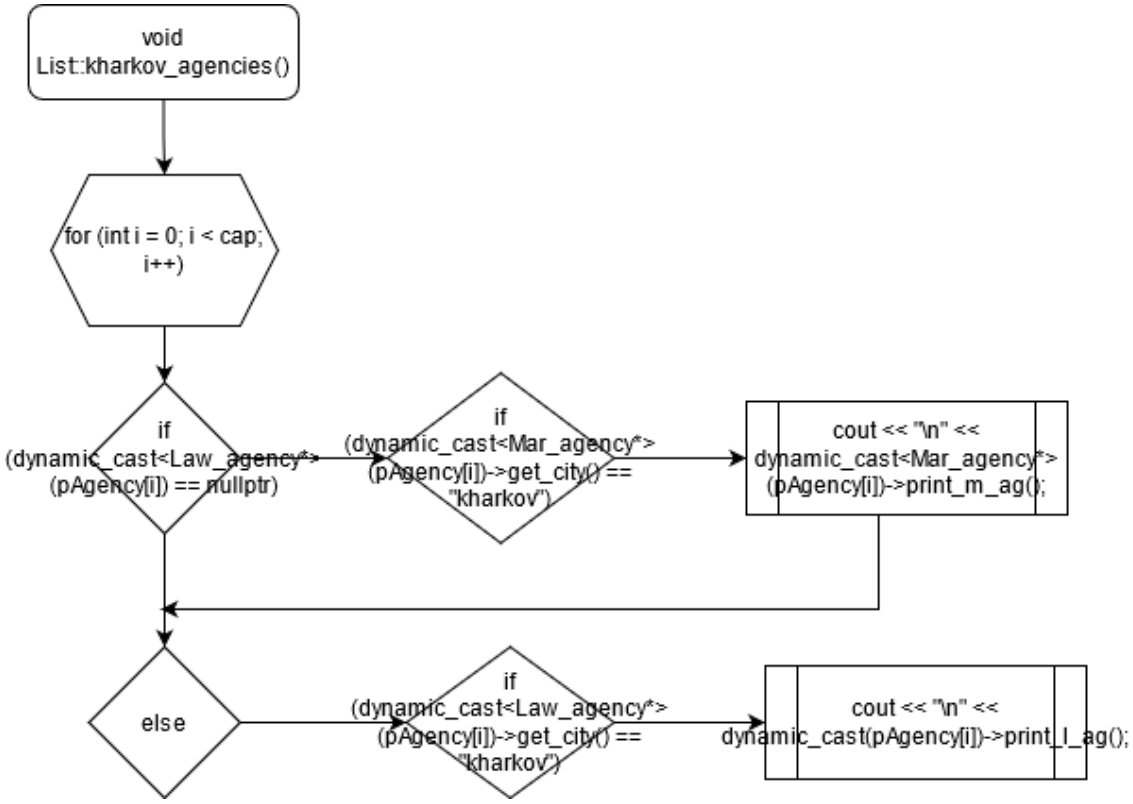

рис. 1 - kharkov_agencies

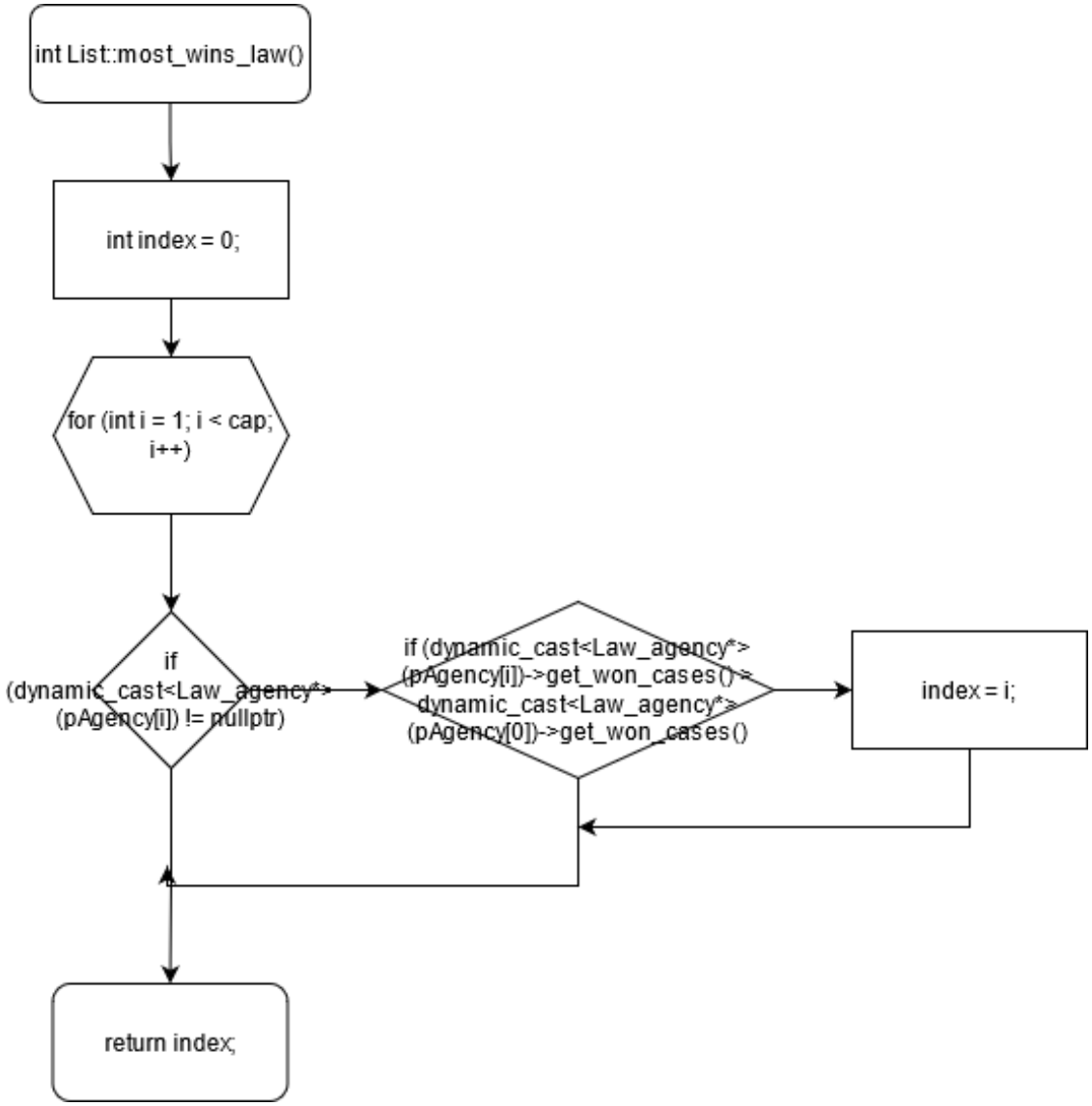

рис. 2 - most_wins_law

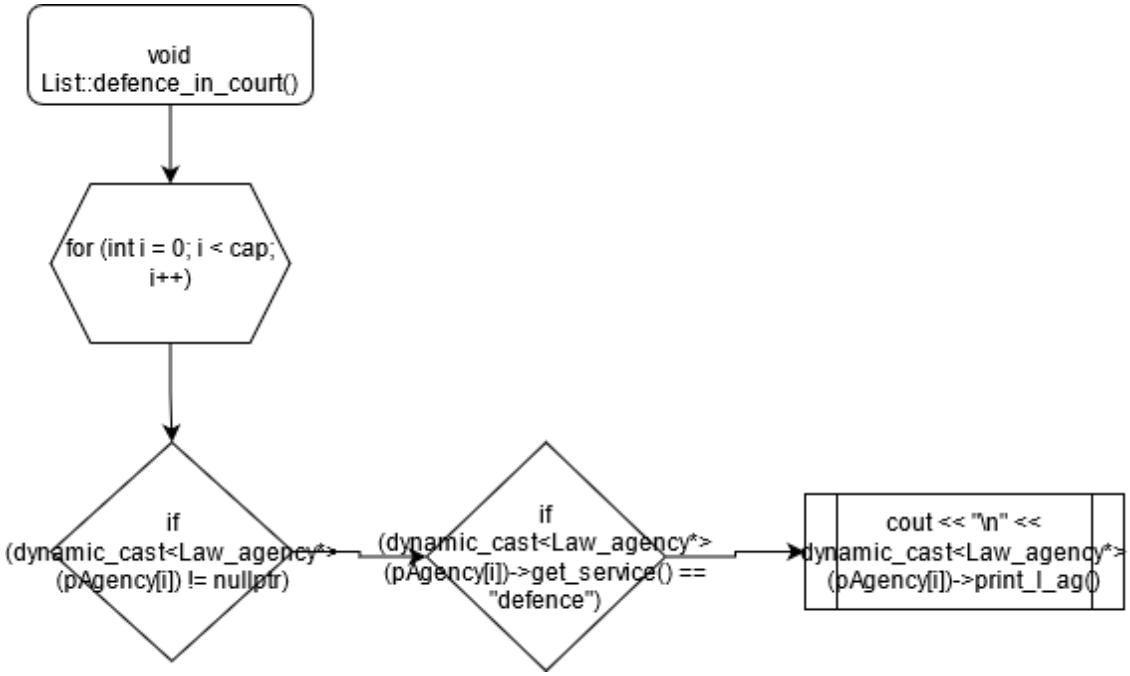

рис. 3 - defence_in_court()

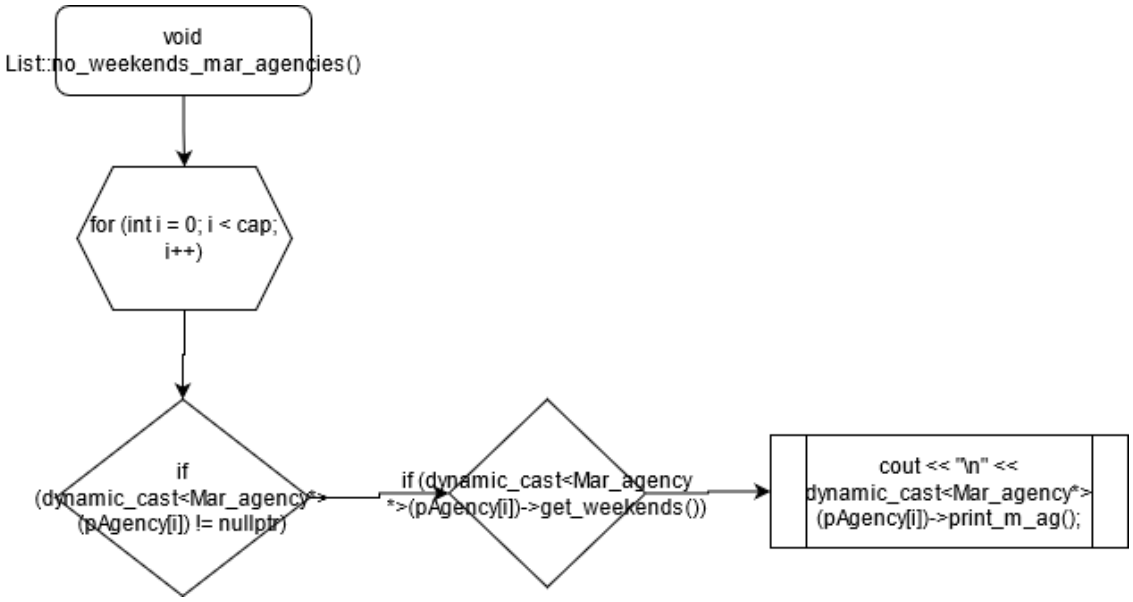

рис. 4 - no_weekends_mar_agencies()

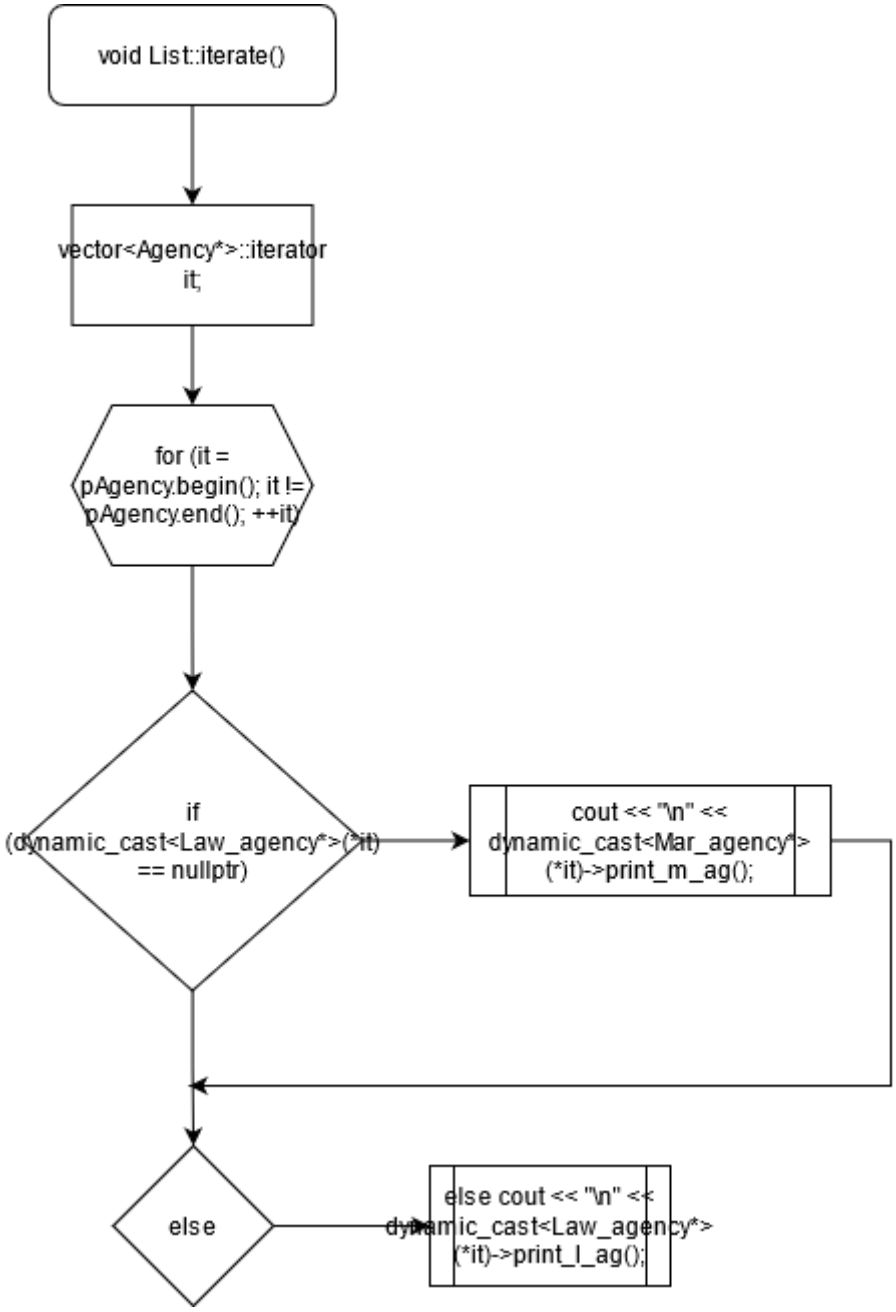

рис. 5 - void List::iterate()

- **Важливі елементи програми**

```
void List::add(Agency *ag){
```

```
    pAgency.push_back(ag);
}



void List::iterate() {

    vector<Agency*>::iterator it;
    for (it = pAgency.begin(); it != pAgency.end(); ++it){
        if (dynamic_cast<Law_agency*>(*it) == nullptr){
            cout << "\n" << dynamic_cast<Mar_agency*>(*it)->print_m_ag();
        }
        else cout << "\n" << dynamic_cast<Law_agency*>(*it)->print_l_ag();
    }
}







    for (int i = 0; i < pAgency.size(); i++){
        if (dynamic_cast<Law_agency*>(pAgency[i]) == nullptr){
            if (dynamic_cast<Mar_agency*>(pAgency[i])->get_city() == "kharkov"){
                cout << "\n" << dynamic_cast<Mar_agency*>(pAgency[i])->print_m_ag();
            }
        }
        else {
            if (dynamic_cast<Law_agency*>(pAgency[i])->get_city() == "kharkov"){
                cout << "\n" << dynamic_cast<Law_agency*>(pAgency[i])->print_l_ag();
            }
        }
    }



    for (int i = 0; i < pAgency.size(); i++){
        if (dynamic_cast<Law_agency*>(pAgency[i]) != nullptr){
            if (dynamic_cast<Law_agency*>(pAgency[i])->get_service() == "defence"){
                cout << "\n" << dynamic_cast<Law_agency*>(pAgency[i])->print_l_ag();
            }
        }
        else {
        }
    }



class Functor{
public:
    bool operator()(int a, int b){
        return (a < b);
    }
```

```
};

class List{
private:
    vector<Agency*>pAgency;
```

- **Варіанти використання**

  Дана програма може бути використана для зберігання класів і роботи з ними.

Висновки

У даній лабораторній роботі були придбані знання роботи з STL (итератори);