**Лабораторна робота №23.**

**Розумнi показчики.**

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



* **Опис роботи**

Програма працює за тим же принципом, що і колишні, однак використовує ООП, полегшуючи роботу. Також використання розумних покажчиків допомагає змінювати дані

* **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.

* **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::change\_vals() - змінює дані за допомогою розумних показників

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 - change\_vals

* **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);