**Лабораторна робота №26.**

**Спадкування.**

* **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. 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 (iндекс).

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, яки будуть зберігати в собі динамічний масив класу.

**Висновки**

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