# ЛАБОРАТОРНА РОБОТА №8. ПЕРЕВАНТАЖЕННЯ ОПЕРАТОРІВ

**Тема.** Перевантаження операторів. Серіалізація.

**Мета.** Отримати знання про призначення операторів, визначити їх ролі в житті об’єкта та можливість перевизначення.

## 1 ВИМОГИ

### 1.1 Розробник

- Котенко Сергій Миколайович;

- Студент групи КІТ 102.8(а);

- 28-05-2019р..

### 1.2 Загальне завдання

*Поширити попередню лабораторну роботу наступним чином:*

- в базовому класі, та класі/класах-спадкоємцях перевантажити:

- оператор присвоювання;

- оператор порівняння (на вибір: == , < , > , >= , <= , != );

- оператор введення/виведення;

- в класі-списку перевантажити:

- оператор індексування ( [ ] );

- оператор введення/виведення з акцентом роботи в тому числі і з файлами. При цьому продовжувати використовувати регулярні вирази для валідації введених даних.

*Додаткові умови виконання завдання:*

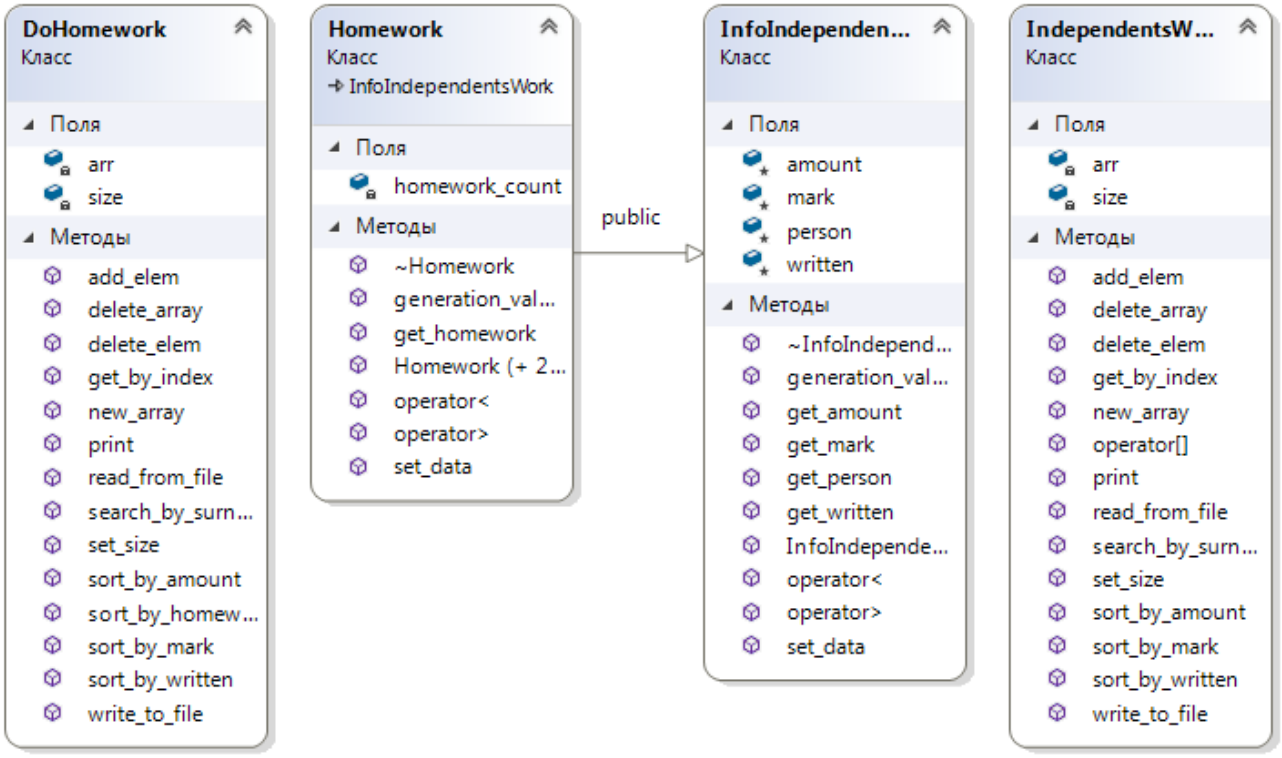
- продемонструвати відсутність витоків пам’яті;

- продемонструвати роботу розроблених методів за допомогою модульних тестів;

- не використовувати конструкцію «using namespace std;» , замість цього слід роботи «using» кожного необхідного класу:using std::string,using std::cout.

# 2 ОПИС ПРОГРАМИ

## 2.1 Опис логічної структури




*Діаграма класу InfoIndependentsWork*:

- ✓ ~InfoIndependentsWork - Деструктор класу;
- ✓ generation_values – Генерація випадкових значень;
- ✓ get_amount , get_mark , get_surname , get_written - Отримання даних;
- ✓ InfoIndependentsWork - Конструктор класу;
- ✓ set_data - Встановлення значень ;
- ✓ operator<> - Перевантаження операторів порівняння.

*Діаграма класу IndependentsWork :*

- ✓ add_elem - Додавання нового елементу;
- ✓ delete_array - Видалення масиву;
- ✓ delete_elem - Видалення елементу;
- ✓ get_by_index - Отримання даних за індексом;
- ✓ new_array - Створення масиву;
- ✓ print - Вивід даних на екран;
- ✓ read_from_file – Читання даних з файлу;
- ✓ search_by_surname – Пошук за прізвищем студента;
- ✓ set_size - Отримання розміру для створення масиву;
- ✓ sort_by_amount, sort_by_mark, sort_by_written – Сортування даних за певним критеріем;
- ✓ write_to_file – Запис результату у файл.

*Діаграма класу Homework* :

- ✓ ~ Homework - Деструктор класу;
- ✓ generation_values – Генерація випадкових значень;
- ✓ get_homework - Отримання даних;
- ✓ Homework - Конструктор класу;
- ✓ set_data - Встановлення значень .
- ✓ operator<> - Перевантаження операторів порівняння.

*Діаграма класу DoHomework :*

- ✓ add_elem - Додавання нового елементу;
- ✓ delete_array - Видалення масиву;
- ✓ delete_elem - Видалення елементу;
- ✓ get_by_index - Отримання даних за індексом;
- ✓ new_array - Створення масиву;
- ✓ print - Вивід даних на екран;
- ✓ read_from_file – Читання даних з файлу;
- ✓ search_by_surname – Пошук за прізвищем студента;
- ✓ set_size - Отримання розміру для створення масиву;
- ✓ sort_by_amount, sort_by_mark, sort_by_written, sort_by_homework – Сортування даних за певним критерієм;
- ✓ write_to_file – Запис результату у файл.

## 2.2 Фрагменти коду

```
friend std::ostream& operator<< (std::ostream &out, const InfoIndependentsWork &obj) {
    out << obj.person << ": " << obj.mark << std::endl;
    return out;
}
friend std::istream& operator>> (std::istream &in, InfoIndependentsWork &obj) {
    in >> obj.amount;
    in >> obj.written;
    in >> obj.mark;
    in >> obj.person;

    return in;
}
```

Рисунок 2.1 – Перевантаження операторів введення/виведення

```
bool operator< (const InfoIndependentsWork obj) {
    return (amount < obj.amount && written < obj.written && mark < obj.mark);
}
bool operator> (const InfoIndependentsWork obj) {
    return (amount > obj.amount && written > obj.written && mark > obj.mark);
}
```

Рисунок 2.2 – Перевантаження операторів порівняння

```
InfoIndependentsWork& operator= (const InfoIndependentsWork &obj) {
    amount = obj.amount;
    written = obj.written;
    mark = obj.mark;
    return *this;
}
```

Рисунок 2.3 – Перевантаження оператора присвоювання

# 3 ВАРІАНТИ ВИКОРИСТАННЯ

## 3.1 Опис поведінки програми

Програма працює наступнім чином:

1) Введення користувачем кількості вивідних даних, створення масиву даних та виведення на екран

2) Вивід на екран можливих опцій програми , обирання користувачем опції:

2.0) Вихід з програми

2.1) Додавання нового елементу

2.2) Видалення певного елементу

2.3) Пошук за індексом

2.4) Пошук за прізвищем

2.5) Сортування за певним критерієм та у певному порядку

3) Перевірка на витоки пам’яті

# ВИСНОВОК

В інтегрованому середовищі *Visual Studio* розроблена програма мовою C++. Виконання програми дозволяє продемонструвати коректність роботи програм для створення перевантажених операторів та їх використання.