ЛАБОРАТОРНА РОБОТА №5. АГРЕГАЦІЯ ТА КОМПОЗИЦІЯ

Тема. Класи. Агрегація. Композиція. Ключові слова typedef та auto.

Мета. Порівняти поняття агрегація та композиція. Отримати знання про призначення ключових слів typedef та auto.

1 ВИМОГИ

1.1 Розробник

- Котенко Сергій Миколайович;

- Студент групи КІТ 102.8(а);

- 19-05-2019р..

1.2 Загальне завдання

Дослідити заздалегідь визначені типи даних з бібліотеки <cstddef> / <stddef.h>. Модернізувати розроблені у попередній роботі класи наступним чином:

- замінити типи даних, що використовуються при індексуванні на типи з вказаної бібліотеки;

- створити власний синонім типу, визначивши його необхідність;

- створити/оновити функцію сортування масиву, де крім поля, по якому виконується сортування, передається і вказівник на функцію, яка визначає напрям сортування;

- в базовий клас додати два поля, що мають кастомний тип даних (тип даних користувача) та які будуть відображати відношення «агрегація» та «композиція», при цьому оновити методи читання та запису об’єкта;

- ввести використання ключового слова auto як специфікатор зберігання типу змінної. Визначити плюси та мінуси цього використання.

Додаткові умови виконання завдання:

- продемонструвати відсутність витоків пам’яті;

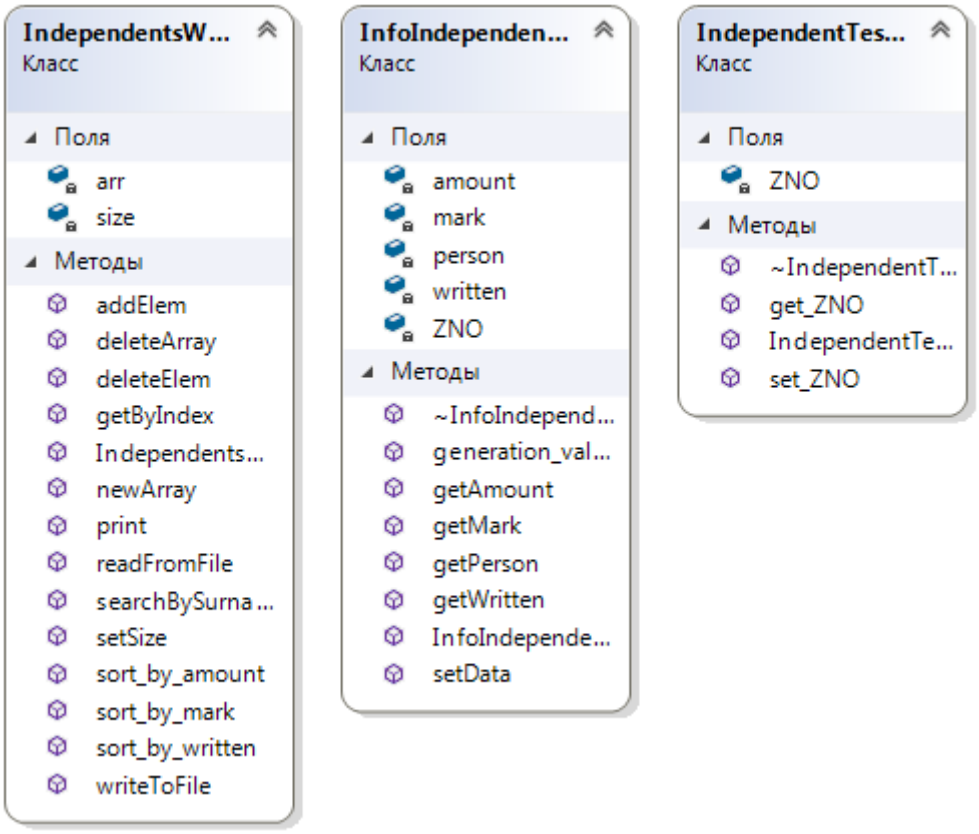
- продемонструвати роботу розроблений методів за допомогою модульних тестів;

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

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

2 ОПИС ПРОГРАМИ

2.1 Опис логічної структури



Діаграма класу InfoIndependentsWork:

- ✓ ~InfoIndependentsWork - Деструктор класу;
- ✓ generation_values – Генерація випадкових значень;
- ✓ getAmount , getMark , getSurname , getWritten - Отримання даних;
- ✓ InfoIndependentsWork - Конструктор класу;
- ✓ setData - Встановлення значень .

Діаграма класу IndependentsWork :

- ✓ addElem - Додавання нового елементу;
- ✓ deleteArray - Видалення масиву;
- ✓ deleteElem - Видалення елементу;
- ✓ getByIndex - Отримання даних за індексом;
- ✓ newArray - Створення масиву;
- ✓ print - Вивід даних на екран;
- ✓ readFromFile – Читання даних з файлу;
- ✓ searchBySurname – Пошук за прізвищем студента;
- ✓ setSize - Отримання розміру для створення масиву;
- ✓ sort_by_amount, sort_by_mark, sort_by_written – Сортування даних за певним критерієм;
- ✓ writeToFile – Запис результату у файл.

2.2 Фрагменти коду

```
auto i = 0;
std::cout << "Enter size : ";
std::cin >> i;
InfoIndependentsWork * arr = new InfoIndependentsWork[i];

{

    IndependentsWork Work(arr, i);

    std::regex regex_spaces("[\\s]{2,}");
    std::regex regex_firstSymbol("^[A-Z]");

    Work.setSize(i);
    system("cls");
    std::string *person = new std::string[i];
    Work.readFromFile(person);

    Work.newArray(person);
    delete[] person;
    system("cls");
    Work.print();
```

Рисунок 2.1 – Агрегація (Зв'язок об’єктів між собою , можливість використання масиву після завершення роботи основної частини)

```
void IndependentsWork::sort_by_mark(bool(*sort)(int a, int b)) {
    InfoIndependentsWork temp;
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            if (sort(arr[i].getMark(), arr[j].getMark())) {
                temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
}
```

Рисунок 2.2 – Сортування даних за оцінкою

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

3) Перевірка на витоки пам’яті

3.2 Ілюстрація роботи програми

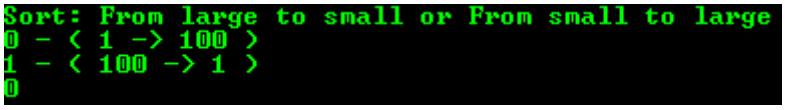


Рисунок 3.1 – Вибір напрямку сортування

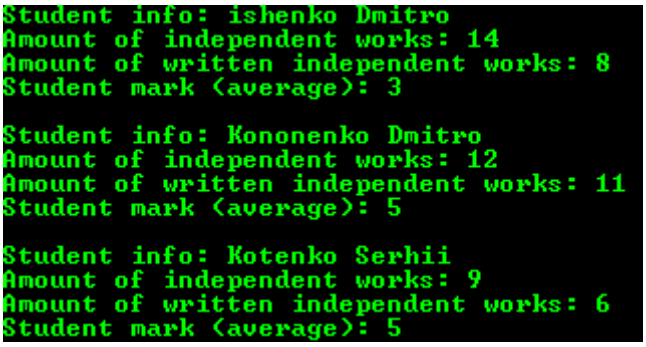


Рисунок 3.2 – Результат сортування даних (оцінок) від меншої до більшої

ВИСНОВОК

В інтегрованому середовищі *Visual Studio* розроблена програма мовою С++. Виконання програми дозволяє продемонструвати коректність роботи програм для створення агрегації , композиції , сортування та результати їх використання.