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

Інформація про розробника:

- Кононенко Дмитро Олексійович
- НТУ “ХПІ”,
- КІТ 102.8а

1.2 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.

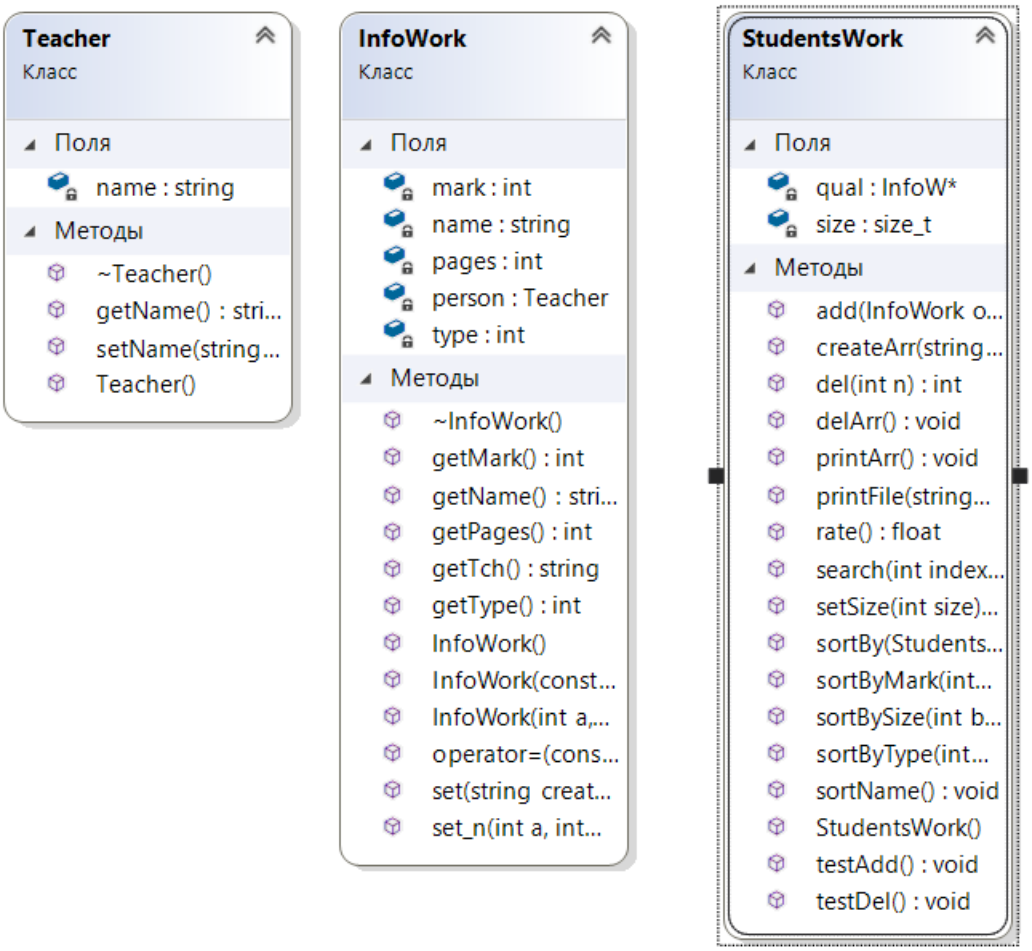
2. Опис програми

2.1 Призначення

Программа створює динамічний масив об'єктів класу за допомогою іншого класу

2.2 Опис логічної структури

Рисунок 1 — діаграма класів




Class InfoWork містить такі поля як:
1. int mark – оцінка за кваліфікаційну роботу
2.string name – ім'я студента
3.int pages – кількість сторінок в роботі
4.int type – тип роботи(магістр\бакалавр)

Class Teacher містить поле “string name – ім'я вчителя”, а також має методи обробки цього поля:

1. void setName(string tmp) — встановлює ім'я.
2. string getName() - повертає поле name.

Class StudentsWork створює масив об'єктів класу InfoWork, а також має методи для обробки цього масиву, а саме:

1. void add(int n) — метод приймає індекс(місце розташування) нового елементу та вставляє його в це місце.
2. void del(int n) — метод приймає індекс(місце розташування) елементу та видаляє його.
3. float rate() - метод який знаходить відсоток магістрів у масиві.
4. void createArr() - метод який створює масив об'єктів класу InfoWork.
5. void printArr() - виводить на екран масив об'єктів класу InfoWork
6. void search(int index) – виводить об'єкт з індексом який приймає функція
7. void setSize(int size) – встановлює розмір масиву об'єктів InfoWork
8. void printFile(string file) – метод виводить в файл масив об'єктів класу InfoWork
9. void sortName() - метод, що виводить на екран список усіх об’єктів, які мають одне або більше полів з щонайменше двома словами
10. void sortByMark(`bool(*)(int a,int b)`) – метод, який сортує массив за полем int mark.
11. void sortBySize(`bool(*)(int a,int b)`) – метод, який сортує массив за полем int size.
12. void sortByType(`bool(*)(int a,int b)`) – метод, який сортує массив за полем int type (бакалавр\магістр).

*інші методи не входять у завдання лабораторної роботи(за значенням функції звертатися до розробника)

**3**.Варіанти використання
Програма складається з 5 файлів:
1.StudentsWork.cpp
2.StudentsWork.h
3.InfoWork.h
4.InfoWork.cpp
5.Laba1.cpp
6.Teacher.cpp
7.Teacher.h

Рисунок 1 — Композиція між класом InfoWork та Teacher.

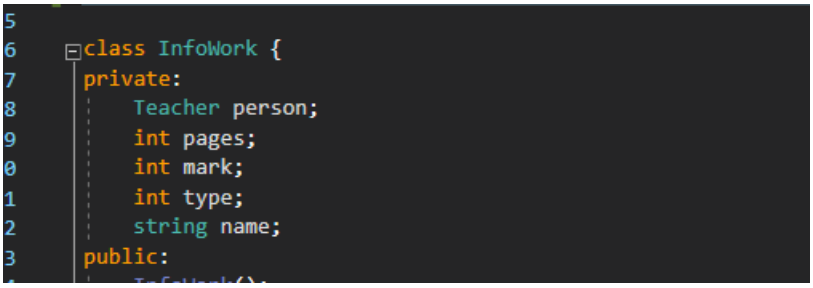



Рисунок 2 — агрегація між класом StudentsWork та  InfoWork

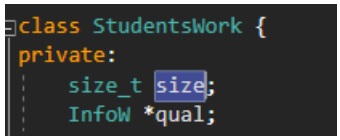



Рисунок 3 — методи сортування

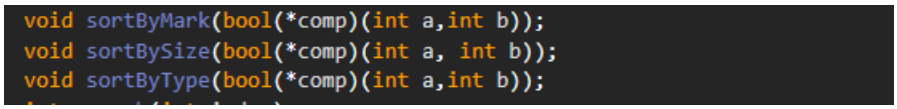
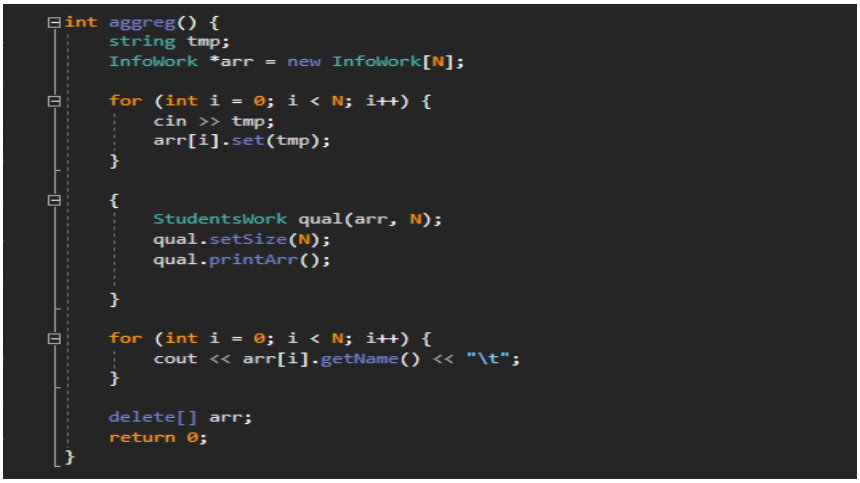


Рисунок 4 — агргеція між класом InfoWork та StudentsWork



```
int aggreg() {
    string tmp;
    InfoWork *arr = new InfoWork[N];

    for (int i = 0; i < N; i++) {
        cin >> tmp;
        arr[i].set(tmp);
    }

    {
        StudentsWork qual(arr, N);
        qual.setSize(N);
        qual.printArr();

    }

    for (int i = 0; i < N; i++) {
        cout << arr[i].getName() << "\t";
    }

    delete[] arr;
    return 0;
}
```

Висновок: зустрів труднощі з агрегацією, вважаю, що агрегація є гіршим видом відношень між класами за композиція, тому-що:

1) Складне в реалізовані за композицію;
2) Менш захищена ніж композиція(використовуються покажчики / посилання, які вказують / посилаються на частини поза класом);
3) Клас не має відповідальності за створення/видалення своїх частин