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

Інформація про розробника:

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

1.2 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.

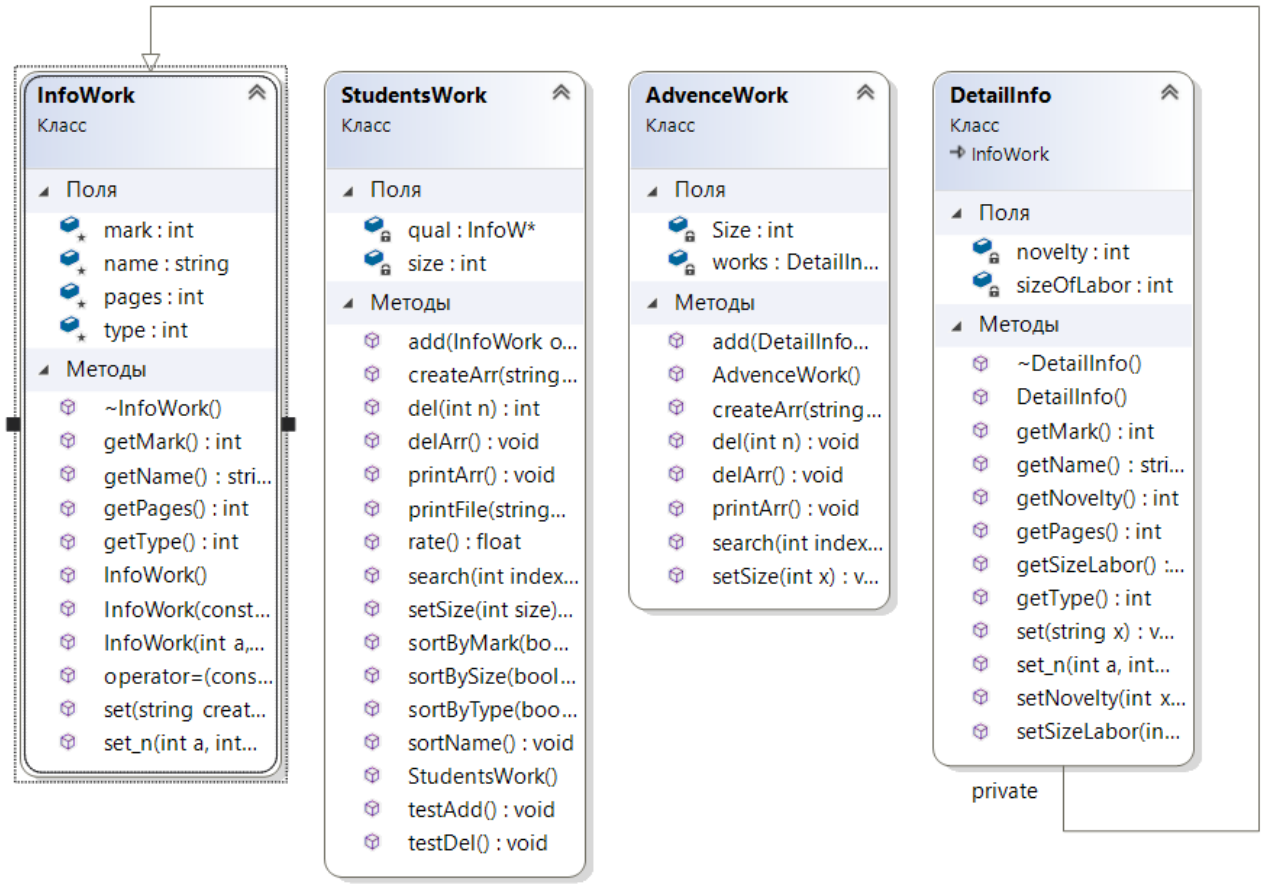
2. Опис програми

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 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)

Class DetailWork успадковує поля класса InfoWork, а також має свої поля:

1. int novelty — поле яке відображає новизну.
2. int sizeOfLabor – поле яке відображає об’єм розділу з охороні праці.

Class AdvenceWork створює масив об'єктів класу DetailWork, а також має методи для обробки цього масиву, а саме:
1. void printArr() - виводить на екран масив об'єктів класу DetailWork.
2. void add(int n) — метод приймає індекс(місце розташування) нового елементу та вставляє його в це місце.
3. void del(int n) — метод приймає індекс(місце розташування) елементу та видаляє його.
4. void search(int index) – виводить об'єкт з індексом який приймає функція

3.Варіанти використання

Програма складається з 5 файлів:

- StudentsWork.cpp
- StudentsWork.h
- InfoWork.h
- InfoWork.cpp
- Laba1.cpp
- AdvenceWork.cpp
- AdvenceWork.h
- DetailInfo.h
- DetailInfo.cpp

3.1 Перевантажені оператори:

```
27    InfoWork& operator= (const InfoWork obj) {
28        pages = obj.pages;
29        mark = obj.mark;
30        type = obj.type;
31        name = obj.name;
32        return *this;
33    }
34    bool operator== (const InfoWork obj) {
35        return (pages == obj.pages && type == obj.type && mark == obj.mark && name == obj.name);
36    }
37    bool operator!= (const InfoWork obj) {
38        return (pages != obj.pages && type != obj.type && mark != obj.mark && name != obj.name);
39    }
40    friend std::ostream& operator<< (std::ostream &out, const InfoWork &obj){
41        out << obj.name << ": " << obj.mark << endl;
42        return out;
43    }
44    friend std::istream& operator>> (std::istream &in, InfoWork &obj) {
45        in >> obj.pages;
46        in >> obj.mark;
47        in >> obj.type;
48        in >> obj.name;
49
50        return in;
51    }
```

Рисунок 3.1 — перевантажені оператори порівняння, присвоювання , оператор введення/виведення.

```
InfoWork &operator[] (const int index)
{
    cout << "Operator []" << endl;
    return qual[index];
}
friend std::ostream& operator<< (std::ostream &out, const StudentsWork &obj) {
    out << "Size: " <<obj.size <<endl;
    for (int i = 0; i < obj.size; i++) {
        out << obj.qual[i];
    }
    return out;
}
friend std::istream& operator>> (std::istream &in, StudentsWork &obj) {
    in >> obj.size;
    for (int i = 0; i < obj.size; i++) in >> obj.qual[i];
    return in;
}
```

Рисунок 3.2 — перевантажені оператори оператор індексування ([ ]), оператор введення/виведення

```
for (int i = 0; i < qualWork.getSize(); i++)
    cout << qualWork[i];

cout << qualWork;
```

Рисунок 3.3 — використання оператору індексування, та оператору виведення

```
251  bool StudentsWork::check_exist(InfoWork tmp) {
252      for (int i = 0; i < size; i++) {
253          if (qual[i] == tmp) {
254              return true;
255          }
256      }
257      return false;
258  }
259
```

Рисунок 3.4 — використання оператору “==”

Висновок: навчився перевантажувати оператори