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

Розробник : Кабак О.Р. ,НТУ “ХПІ” ,КІТ102.8а

1.1 Основне завдання

Поширити попередню лабораторну роботу наступним чином:

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

-оператор присвоювання;

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

-оператор введення/виведення;

2)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.

1.2 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;

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

2.Опис програми

2.1. Функціональне призначення

Програма створена для генерування динамічного масиву самостійних робіт студента з сутністю спадкоємців “базового класу”.

```
ostream& operator << (ostream& os, const TestsInfo& o) {
    os << o.num << " " << o.surname << " " << o.works << " " << o.pages << " " << o.mark << " ";
    return os;
}

istream& operator >> (istream& is, TestsInfo& o) {
    cout << "Enter the data in this order:ID->surname->number of works->number of pages in work->Mark\n";
    is >> o.num;
    is >> o.surname;
    is >> o.works;
    is >> o.pages;
    is >> o.mark;

    return is;
}
```

Рис 3.1 Приклад перевантаження операторів << та >> у базовому класі (аналогічно у класах спадкоємцях)

```
TestsInfo& TestsInfo::operator = (const TestsInfo& o){
    this->num = o.num;
    this->works = o.works;
    this->pages = o.pages;
    this->mark = o.mark;
    this->surname = o.surname;

    cout << "Operator = worked here.\n";
    return *this;
}
bool TestsInfo::operator == (const TestsInfo& o) {
    return this->works == o.works && this->pages == o.pages;
}
```

Рис 3.2 Приклад перевантаження операторів = та == у базовому класі(аналогічно у классах спадкоємцях)

```
TestsInfo& StudentsWorks::operator[](int index){
    return *list[index];
}
ostream& operator<< (ostream& os, const StudentsWorks& o) {
    os << o.SIZE << endl;
    for (int i = 0; i < o.SIZE; i++) {
        os << o.list[i];
    }
    return os;
}
istream& operator>> (istream& is, StudentsWorks& o) {
    is >> o.SIZE;
    for (int i = 0; i < o.SIZE; i++) {
        is >> *o.list[i];
    }
    return is;
}
```

Рис 3.3 Приклад перевантаження операторів [] , >> та << у класі-списку

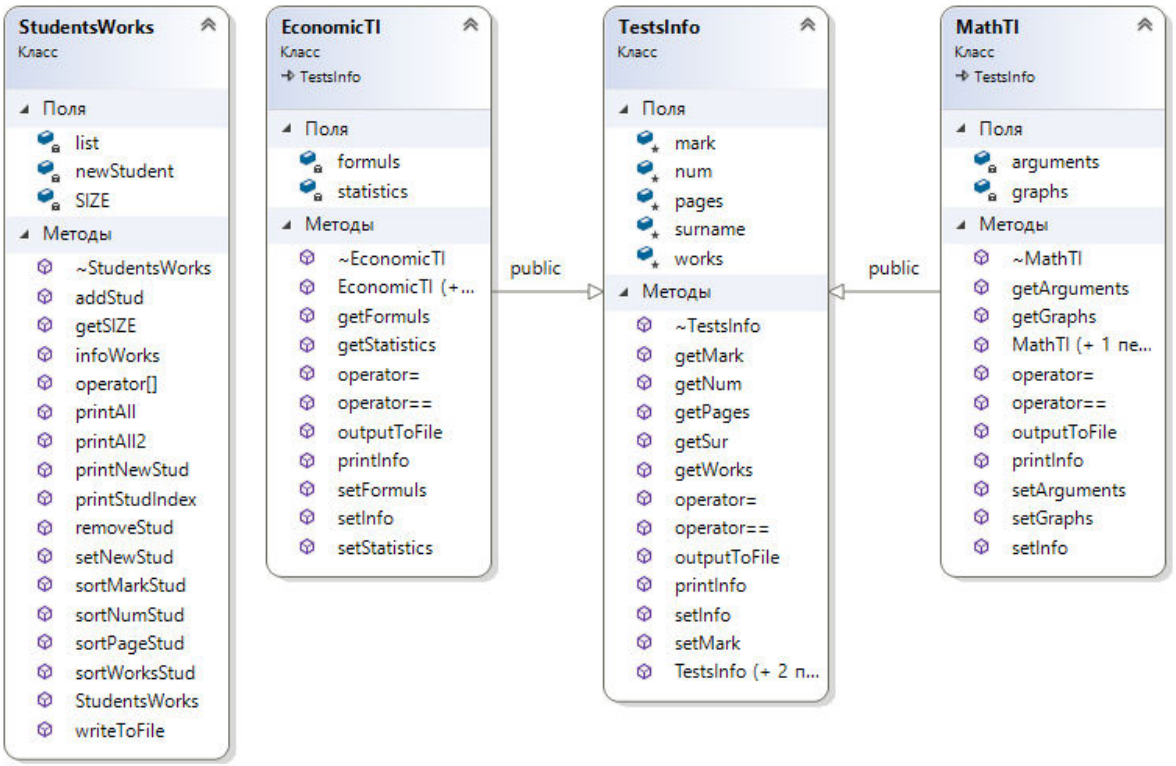

Рис 3.4 Діаграми в яких показані перевантажені методи у классах

Висновки

Отримано знання про призначення операторів,визначена роль у існуванні в об’єкті і отримані навички для перевизначення даних операторів