Лабораторна робота №25

Тема: Перевантаження операторів.

Мета: Навчитися використовувати перевантаження для різних методів.

Індивідуальне завдання

Загальне завдання.

Поширити попередню лабораторну роботу таким чином:

- у базовому класі, та класі/класах-спадкоємцях перевантажити:
  - оператор присвоювання;
  - оператор порівняння (на вибір: == , < , > , >= , <= , !=);
  - оператор введення / виведення;
- у класі-списку перевантажити:
  - оператор індексування ([ ]);
  - введення / виведення з акцентом роботи, у тому числі і з файлами. При цьому продовжувати використовувати регулярні вирази для валідації введених даних.

Хід роботи

1. Доповнення коду для базового класу, класу списку та класу помічника.

```
Pupils& Pupils::operator= (const Pupils& obj) {
    characteristic = obj.characteristic;
    grade = obj.grade;
    insurance = obj.insurance;
    age = obj.age;
    numberSchool = obj.numberSchool;
    fullname = obj.fullname;

    return *this;
}
```

Рисунок 1.1 - Перевантажений метод присвоювання.

```
friend bool operator==(const Pupils& obj1, const Pupils& obj2) {
    return (obj1.grade == obj2.grade);
}
friend bool operator!=(const Pupils& obj1, const Pupils& obj2) {
    return !(obj1 == obj2);
}
```

Рисунок 1.2 - Перевантажені методи порівняння з використанням конструкції friend.

```
friend std::ostream& operator<< (std::ostream& out, const Pupils& obj) {
    out << "\nNumberSchool:" << obj.numberSchool;
    out << "\nFullname:" << obj.fullname;
    out << "\nAge:" << obj.age;
    out << "\nCharacteristic:" << obj.characteristic;
    out << "\nGrade" << obj.grade;
    out << "\nInsurance:" << obj.insurance ? "Yes" : "No";
    return out;
}
friend std::istream& operator>> (std::istream& in, Pupils& obj) {
    in >> obj.numberSchool;
    in >> obj.fullname;
    in >> obj.age;
    in >> obj.characteristic;
    in >> obj.grade;
    in >> obj.insurance;

    return in;
}
```

Рисунок 1.3 - Перевантажені оператори вводу/виводу.

```
Pupils& List::getPupils(const int index) {
    return list[index];
}
```

Рисунок 1.4 - Перевантажений оператор індексування.

```
friend std::ostream& operator << (std::ostream& out, const List& obj) {
    for (int i = 0; i < obj.list.size(); i++)
        out << obj.list[i];

    return out;
}
```

Рисунок 1.5 - Перевантажений оператор виводу для списку.

```
void List::addObjects() {
    bool choice = true;
    Pupils tmp;
    int digit;

    while (choice) {
        getchar();
        cout << "\n\nCreate a new pupils now.\n Enter full name : ";
        tmp.setFullname(check(regName));
        cout << "    Enter age : ";
        do {
            cin >> digit;
            if (digit >= 5 && digit <= 20)
            {
                tmp.setAge(digit);
                break;
            }
            else
                cout << "\nTry again : ";
        } while (true);
        cout << "    Enter number school : ";
        do {
            cin >> digit;
            if (digit >> 0 && digit <= 999)
            {
                tmp.setNumberSchool(digit);
                break;
            }
            else
                cout << "\nTry again : ";
        } while (true);
        cout << "    Describe the positive qualities in 3 words : ";
        tmp.setCharac(check(regName));
        cout << "    Enter grade : ";
        do {
            cin >> digit;
            if (digit >= 1 && digit <= 12)
            {
                tmp.setGrade(digit);
                break;
            }
            else
                cout << "\nTry again : ";
        } while (true);
```

Рисунок 1.6.1 - Метод для зчитування, валідації та додавання елементів у список.

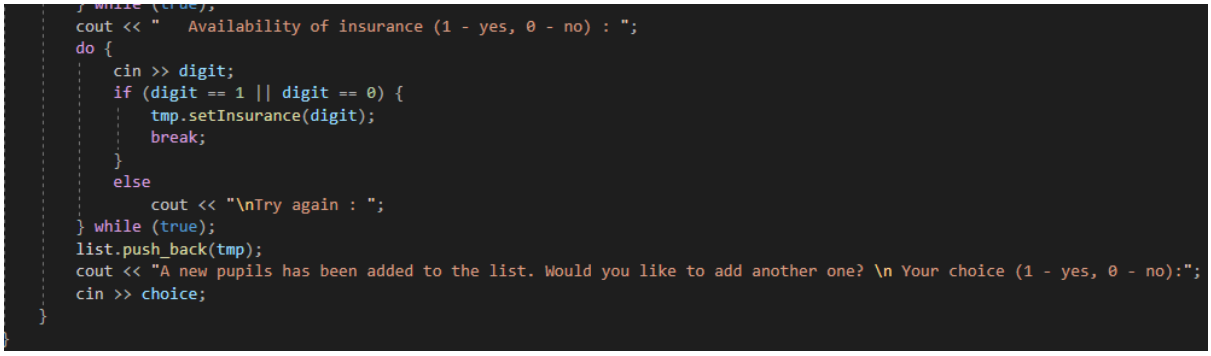

Рисунок 1.6.2 - Метод для зчитування, валідації та додавання елементів у список.

Висновок: Навчився використовувати перевантаження для різних методів.