

Лабораторна робота №25

Тема: Перевантаження операторів.

Мета: Навчитися використовувати модульні тести.

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.

```
Employee& Employee::operator= (const Employee& obj) {  
    characteristic = obj.characteristic;  
    workExperience = obj.workExperience;  
    insurance = obj.insurance;  
    fullName = obj.fullName;  
    company = obj.company;  
    mail = obj.mail;  
  
    return *this;  
}
```

Рисунок 1.1 - Перевантажений метод присвоювання.

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

Рисунок 1.2 - Перевантажені методи порівняння з використанням конструкції friend.

```
friend std::ostream& operator<< (std::ostream& out, const Employee& obj) {
    out << "\nCompany:" << obj.company;
    out << "\nMail:" << obj.mail;
    out << "\nFullname:" << obj.fullName;
    out << "\nCharacteristic:" << obj.characteristic;
    out << "\nWork experience in years:" << obj.workExperience;
    out << "\nInsurance:" << obj.insurance ? "Yes" : "No";
    return out;
}
friend std::istream& operator>> (std::istream& in, Employee& obj) {
    in >> obj.company;
    in >> obj.mail;
    in >> obj.fullName;
    in >> obj.characteristic;
    in >> obj.workExperience;
    in >> obj.insurance;

    return in;
}
```

Рисунок 1.3 - Перевантажені оператори вводу/виводу.

```
Employee& List::getEmployee(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;
    Employee tmp;
    int digit;

    while (choice) {
        getchar();
        cout << "\n\nCreate a new employee now.\n  Enter full name : ";
        tmp.setFullName(check(regName));
        cout << "    Enter mail : ";
        tmp.setMail(check(regMail));
        cout << "    Enter company name : ";
        tmp.setCompany(check(regName));
        cout << "    Describe the positive qualities in 3 words : ";
        tmp.setCharac(check(regName));
        cout << "    Enter work experience in years : ";
        do {
            cin >> digit;
            if (digit >= 0 && digit <= 480) {
                tmp.setWorkExp(digit);
                break;
            }
            else
                cout << "\nTry again : ";
        } while (true);
        cout << "    Availability of insurance (1 - yes, 0 - no) : ";
        do {
            cin >> digit;
            if (digit == 1 || digit == 0) {
                tmp.setInsurance(digit);
                break;
            }
            else
                cout << "\nTry again : ";
        } while (true);
        list.push_back(tmp);
        cout << "A new employee has been added to the list. Would you like to add another one? \n Your choice (1 - yes, 0 - no):";
        cin >> choice;
    }
}

```

Рисунок 1.6 - Метод для зчитування, валідації та додавання елементів у список.

Висновок: Навчився використовувати перевантаження для різних методів.