МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім.І. Сікорського

Кафедра автоматизованих систем обробки інформації і управління

КУРСОВА РОБОТА

з дисципліни

“Програмування 2. Структури даних та алгоритми”

на тему

“Об'єктно-орієнтоване програмування”

Студента 2 курсу ЗПІ-зп21 групи

Спеціальності 121 “Інженерія програмного забезпечення”

Керівник Проскура Світлана Леонідівна

Кількість балів: _____

Національна оцінка _____

Члени комісії _____ _____

_____ _____

Київ – 2023 рік

ЗМІСТ

ВСТУП

Курсова робота з програмування є важливим етапом навчання, спрямованим на закріплення та практичне використання основних концепцій об'єктно-орієнтованого підходу (ООП) та розвиток навичок у створенні сучасних програмних продуктів. У рамках даної курсової роботи, метою є не лише оволодіння теоретичними знаннями з ООП, але й їх практичне застосування для розробки програмного забезпечення на базі мови програмування C++.

Цілью даної курсової роботи є вивчення та застосування основних принципів об'єктно-орієнтованого програмування для створення сучасного програмного продукту. Розробка програмного забезпечення з використанням ООП дозволяє створити ефективніші, модульні та розширювані системи, що забезпечують високу зручність їх використання та підтримку.

Вирішувана задача включає розробку нового або вдосконалення існуючого програмного продукту з використанням об'єктно-орієнтованого підходу мовою програмування C++. Об'єктно-орієнтоване програмування дозволяє організувати код у вигляді класів та об'єктів, що взаємодіють між собою. Це забезпечує більшу модульність, підтримку поліморфізму, успадкування та інші переваги, що дозволяють покращити структуру та функціональність програмного продукту.

Однією з головних складнощів розробки програмного забезпечення є його підтримка та розширення. Об'єктно-орієнтований підхід дозволяє розбити систему на незалежні компоненти (класи), які можна модифікувати окремо без впливу на решту системи. Це забезпечує високу розширюваність та зручність у підтримці програмного продукту.

Крім того, розробка програмного забезпечення з використанням ООП дозволяє підвищити повторне використання коду. Використання класів та об'єктів дозволяє створювати бібліотеки, які можна використовувати в інших проектах. Це сприяє прискоренню процесу розробки та покращенню якості програмного продукту.

Отже, дана курсова робота має на меті закріплення на практиці вміння використовувати основні концепції об'єктно-орієнтованого підходу, створювати сучасне програмне забезпечення з використанням мови C++ та впроваджувати його в реальні інформаційні та комп'ютерні системи. Проведення цієї роботи допоможе набути необхідні навички та знання, які є важливим етапом у подальшому розвитку як професійного програміста.

Виконання курсової роботи сприятиме поглибленню розуміння об'єктно-орієнтованого підходу, розвитку практичних навичок у розробці програмного забезпечення та формуванню вмінь у проектуванні, розробці, налагодженні та супроводженні сучасних програмних продуктів. Крім того, результати цієї роботи можуть бути використані в якості основи для подальшого дослідження та розробки комп'ютерних систем з використанням об'єктно-орієнтованого підходу.

У процесі виконання курсової роботи було проведено дослідження літературних джерел, аналіз існуючих програмних продуктів, розроблено план роботи та оцінено обсяг необхідної роботи. Отримані результати і висновки представлені у подальших розділах пояснювальної записки.

ОСНОВНА ЧАСТИНА

1. Аналіз предментної області.

Номер варіанта	Предметна галузь	Опис предметної галузі
13	Факультет. Облік студентів, та планування заннять.	На факультеті навчаються студенти певних груп та спеціальностей. Для студентів організовано робочий процес за їх навчальним планом. Факультет має фонд аудиторій. Викладачі ведуть дисципліни згідно їх навантаження. Заняття для студентів факультету плануються у загальний розклад, при чому лабораторні роботи проводяться для однієї групи в певній аудиторії, лекції – на цілий потік.

На факультеті існує потреба в системі управління робочим процесом для студентів, викладачів та аудиторій. Система повинна забезпечувати ефективну організацію навчального процесу на основі навчальних планів, навантаження викладачів та наявного фонду аудиторій.

Одним із головних функціональних вимог є можливість планування занять для студентів з урахуванням їх навчальних планів. Навчальні плани визначають групи та спеціальності студентів. Система повинна дозволяти призначати предмети для кожної групи згідно їх навчального плану.

Крім того, викладачі мають вести дисципліни відповідно до свого навантаження. Система повинна забезпечувати можливість визначення навантаження викладачів та призначення їм дисциплін для проведення.

Окремо слід врахувати, що лабораторні роботи проводяться для конкретної групи студентів у певній аудиторії, тоді як лекції проводяться для всього потоку студентів. Система повинна забезпечувати можливість призначення аудиторій для лабораторних робіт та відображення розкладу лекцій для потоку.

Таким чином, це завдання вимагає розробки системи, яка буде включати модулі для планування занять для студентів з урахуванням навчальних планів, призначення дисциплін викладачам, а також призначення аудиторій для лабораторних робіт та відображення розкладу лекцій для потоку студентів.

2. Кодування (програмування).

```
#include <iostream>
#include <string>
#include "algorithm"
#include <vector>
#include <set>

class Lesson
{
private:
    std::string subject;
    std::string auditorium;
    std::string time;
    std::string day;

public:
    Lesson(
            const std::string& subject,
            const std::string& auditorium,
            const std::string& time,
            const std::string& day
    ) : subject(subject), auditorium(auditorium), time(time), day(day) {}

    std::string getSubject() const {
        return subject;
    }

    std::string getAuditorium() const {
        return auditorium;
    }

    std::string getTime() const {
        return time;
    }

    std::string getDay() const {
        return day;
    }
};

class Student {
private:
    std::string name;
    std::string group;
    std::string specialty;

public:
    Student(const std::string& name, const std::string& group, const std::string& specialty)
            : name(name), group(group), specialty(specialty) {}

    std::string getName() const {
        return name;
    }

    std::string getGroup() const {
        return group;
    }

    std::string getSpecialty() const {
        return specialty;
```

```
    }
};

class Lecture : public Lesson {
private:
    std::string specialty;

public:
    Lecture(
            const std::string& subject,
            const std::string& auditorium,
            const std::string& _specialty,
            const std::string& time,
            const std::string& day
    ) : Lesson(subject, auditorium, time, day)
    {
        specialty = _specialty;
    }

    std::string getSpecialty() const {
        return specialty;
    }
};

class Practice : public Lesson {
private:
    std::string specialty;
    std::string group;

public:
    Practice(
            const std::string& subject,
            const std::string& auditorium,
            const std::string& _specialty,
            const std::string& _group,
            const std::string& time,
            const std::string& day
    ) : Lesson(subject, auditorium, time, day)
    {
        specialty = _specialty;
        group = _group;
    }

    std::string getSpecialty() const {
        return specialty;
    }

    std::string getGroup() const {
        return group;
    }
};

class Faculty {
private:
    std::vector<Student> students;
    std::vector<Lecture> lectures;
    std::vector<Practice> practices;

public:
    std::vector<Student> getStudents() {
        return students;
    }
```

```
std::vector<Lecture> getLectures() {
    return lectures;
}

std::vector<Practice> getPractices() {
    return practices;
}

void addStudent(const std::string& name, const std::string& group, const std::string& specialty) {
    Student student(name, group, specialty);

    addStudent(student);
}

void addStudent(const Student& student) {
    students.push_back(student);

    std::cout << "Студента успішно додано!" << std::endl;
}

void addLecture(
        const std::string& subject,
        const std::string& auditorium,
        const std::string& specialty,
        const std::string& time,
        const std::string& day
) {
    Lecture lecture(subject, auditorium, specialty, time, day);

    addLecture(lecture);
}

void addLecture(const Lecture& lecture) {
    lectures.push_back(lecture);

    std::cout << "Лекцію успішно додано!" << std::endl;
}

void addPractice(
        const std::string& subject,
        const std::string& auditorium,
        const std::string& specialty,
        const std::string& group,
        const std::string& time,
        const std::string& day
) {
    Practice practice(subject, auditorium, specialty, group, time, day);

    addPractice(practice);
}

void addPractice(const Practice& practice) {
    practices.push_back(practice);

    std::cout << "Практичне заняття успішно додано!" << std::endl;
}

void displayStudents() const {
    std::cout << "Список студентів:" << std::endl;
    for (const auto& student : students) {
        std::cout << "Ім'я: " << student.getName() << " | Спеціальність: " << student.getSpecialty()
```

```
                << " | Група: " << student.getGroup() << std::endl;
        }
    }

    void displayLectures() const {
        std::cout << "Список лекцій:" << std::endl;
        for (const auto& lecture : lectures) {
            std::cout << "Дисципліна: " << lecture.getSubject() << " | Аудиторія: " << lecture.getAuditorium()
                << " | Спеціальність: " << lecture.getSpecialty() << " | Час проведення: " <<
lecture.getTime()
                << " | День проведення: " << lecture.getDay() << std::endl;
        }
    }

    void displayPractices() const {
        std::cout << "Список практичних занять:" << std::endl;
        for (const auto& practice : practices) {
            std::cout << "Дисципліна: " << practice.getSubject() << " | Аудиторія: " <<
practice.getAuditorium()
                << " | Спеціальність: " << practice.getSpecialty() << " | Група: " << practice.getGroup()
                << " | Час проведення: " << practice.getTime() << " | День проведення: " <<
practice.getDay()
                << std::endl;
        }
    }

    void displayGroups() const {
        std::set<std::string> groups;
        for (const auto& student : students) {
            groups.insert(student.getGroup());
        }

        for (const auto& practice : practices) {
            groups.insert(practice.getGroup());
        }

        std::cout << "Список груп студентів:" << std::endl;
        for (const auto& group : groups) {
            std::cout << group << std::endl;
        }
    }

    void displayStudentsByGroup(const std::string& group) const {
        std::cout << "Студенти з групи " << group << ":" << std::endl;
        for (const auto& student : students) {
            if (student.getGroup() == group) {
                std::cout << "Ім'я: " << student.getName() << " | Спеціальність: " << student.getSpecialty()
                    << std::endl;
            }
        }
    }

    void displaySpecialties() const {
        std::set<std::string> specialties;
        for (const auto& student : students) {
            specialties.insert(student.getSpecialty());
        }

        for (const auto& lecture : lectures) {
            specialties.insert(lecture.getSpecialty());
        }
```

```
        for (const auto& practice : practices) {
            specialties.insert(practice.getSpecialty());
        }

        std::cout << "Список спеціальностей:" << std::endl;
        for (const auto& specialty : specialties) {
            std::cout << specialty << std::endl;
        }
    }

    void displayGroupsBySpecialty(const std::string& specialty) const {
        std::set<std::string> groups;
        for (const auto& student : students) {
            if (student.getSpecialty() == specialty) {
                groups.insert(student.getGroup());
            }
        }

        for (const auto& practice : practices) {
            if (practice.getSpecialty() == specialty) {
                groups.insert(practice.getGroup());
            }
        }

        std::cout << "Групи для спеціальності " << specialty << ":" << std::endl;
        for (const auto& group : groups) {
            std::cout << group << std::endl;
        }
    }

    void displayLecturesAndPracticesByDay(const std::string& day) const {
        std::vector<Lesson> lessons;

        for (const auto& lecture : lectures) {
            if (lecture.getDay() == day) {
                lessons.push_back(lecture);
            }
        }

        for (const auto& practice : practices) {
            if (practice.getDay() == day) {
                lessons.push_back(practice);
            }
        }

        std::sort( lessons.begin(), lessons.end(), [](const Lesson &a, const Lesson &b){
            return (a.getTime() < b.getTime());
        });

        std::cout << "Список занять на " << day << ": " << std::endl;

        for (Lesson lesson : lessons)
        {
            std::cout << "Дисципліна: " << lesson.getSubject() << " | Аудиторія: " << lesson.getAuditorium()
                      << " | Час проведення: " << lesson.getTime() << std::endl;
        }

        lessons.clear();
    }
};

int main() {
```

```
Faculty faculty;
int choice;

while (true) {
    std::cout << "Оберіть операцію:" << std::endl;
    std::cout << "1. Додати студента" << std::endl;
    std::cout << "2. Додати лекцію" << std::endl;
    std::cout << "3. Додати практичне заняття" << std::endl;
    std::cout << "4. Переглянути список студентів" << std::endl;
    std::cout << "5. Переглянути список лекцій" << std::endl;
    std::cout << "6. Переглянути список практичних занять" << std::endl;
    std::cout << "7. Переглянути список груп студентів" << std::endl;
    std::cout << "8. Переглянути студентів з певної групи" << std::endl;
    std::cout << "9. Переглянути список спеціальностей студентів" << std::endl;
    std::cout << "10. Переглянути групи для певної спеціальності" << std::endl;
    std::cout << "11. Переглянути графік занять на день" << std::endl;
    std::cout << "0. Вийти" << std::endl;

    std::cout << "Оберіть операцію: ";
    std::cin >> choice;

    if (choice == 1) {
        std::string name, group, specialty;
        std::cout << "Введіть ім'я студента: ";
        std::cin >> name;
        std::cout << "Введіть спеціальність студента: ";
        std::cin >> specialty;
        std::cout << "Введіть групу студента: ";
        std::cin >> group;
        faculty.addStudent(name, group, specialty);
    } else if (choice == 2) {
        std::string subject, auditorium, specialty, time, day;
        std::cout << "Введіть назву дисципліни лекції: ";
        std::cin >> subject;
        std::cout << "Введіть спеціальність: ";
        std::cin >> specialty;
        std::cout << "Введіть аудиторію: ";
        std::cin >> auditorium;
        std::cout << "Введіть день проведення лекції: ";
        std::cin >> day;
        std::cout << "Введіть чвс проведення лекції: ";
        std::cin >> time;
        faculty.addLecture(subject, auditorium, specialty, time, day);
    } else if (choice == 3) {
        std::string subject, auditorium, specialty, group, time, day;
        std::cout << "Введіть назву дисципліни практичного заняття: ";
        std::cin >> subject;
        std::cout << "Введіть спеціальність: ";
        std::cin >> specialty;
        std::cout << "Введіть групу: ";
        std::cin >> group;
        std::cout << "Введіть аудиторію: ";
        std::cin >> auditorium;
        std::cout << "Введіть день проведення практичного заняття: ";
        std::cin >> day;
        std::cout << "Введіть чвс проведення практичного заняття: ";
        std::cin >> time;
        faculty.addPractice(subject, auditorium, specialty, group, time, day);
    } else if (choice == 4) {
        faculty.displayStudents();
    } else if (choice == 5) {
        faculty.displayLectures();
```

```
        } else if (choice == 6) {
            faculty.displayPractices();
        } else if (choice == 7) {
            faculty.displayGroups();
        } else if (choice == 8) {
            std::string group;
            std::cout << "Введіть назву групи: ";
            std::cin >> group;
            faculty.displayStudentsByGroup(group);
        } else if (choice == 9) {
            faculty.displaySpecialties();
        } else if (choice == 10) {
            std::string specialty;
            std::cout << "Введіть назву спеціальності: ";
            std::cin >> specialty;
            faculty.displayGroupsBySpecialty(specialty);
        } else if (choice == 11) {
            std::string day;
            std::cout << "Введіть день проведення занять: ";
            std::cin >> day;
            faculty.displayLecturesAndPracticesByDay(day);
        } else if (choice == 0) {
            break;
        } else {
            std::cout << "Невірний вибір. Спробуйте ще раз." << std::endl;
        }

        std::cout << std::endl;
    }

    return 0;
}
```

3. Опис програмного забезпечення.

У програмному коді програми використовуються наступні класи:

Lesson - цей клас представляє заняття (урок) і має такі приватні поля:

subject (типу *std::string*) - назва дисципліни;
auditorium (типу *std::string*) - номер аудиторії;
time (типу *std::string*) - час проведення заняття;
day (типу *std::string*) - день проведення заняття.

Клас має конструктор, який приймає значення для кожного поля, а також публічні методи доступу (get) до кожного з полів.

Student - цей клас представляє студента і має такі приватні поля:

name (типу *std::string*) - ім'я студента;
group (типу *std::string*) - назва групи, в якій навчається студент;
specialty (типу *std::string*) - назва спеціальності, на яку навчається студент.

Клас має конструктор, який приймає значення для кожного поля, а також публічні методи доступу (get) до кожного з полів.

Lecture - цей клас успадковує клас Lesson і додає ще одне приватне поле:

specialty (типу *std::string*) - спеціальність, на яку відповідає лекція.

Клас має конструктор, який приймає значення для кожного поля, а також публічний метод доступу (get) до поля specialty.

Practice - цей клас успадковує клас Lesson і додає два приватних поля:

specialty (типу *std::string*) - спеціальність, на яку відповідає практичне заняття;
group (типу *std::string*) - група, в якій проводиться практичне заняття.

Клас має конструктор, який приймає значення для кожного поля, а також публічні методи доступу (get) до полів specialty і group.

Faculty - цей клас представляє факультет і має такі приватні поля:

students (типу *std::vector<Student>*) - вектор студентів факультету;
lectures (типу *std::vector<Lecture>*) - вектор лекцій факультету;
practices (типу *std::vector<Practice>*) - вектор практичних занять факультету.

Клас має публічні методи для додавання студентів, лекцій та практичних занять до відповідних векторів, а також методи для отримання списків студентів, лекцій та практичних занять.

Крім власних класів, програма використовує також стандартні класи з бібліотеки C++, такі як *std::string*, *std::set* і *std::vector*.

Інтерфейс користувача програми:

```
[redacted]@debian:~/Documents/kpi/basics_of_programming_2/coursework$ g++ coursework.cpp -o coursework
[redacted]@debian:~/Documents/kpi/basics_of_programming_2/coursework$ ./coursework
Оберіть операцію:
1. Додати студента
2. Додати лекцію
3. Додати практичне заняття
4. Переглянути список студентів
5. Переглянути список лекцій
6. Переглянути список практичних занять
7. Переглянути список груп студентів
8. Переглянути студентів з певної групи
9. Переглянути список спеціальностей студентів
10. Переглянути групи для певної спеціальності
11. Переглянути графік занять на день
0. Вийти
Оберіть операцію:
```

4. Результати тестування.

Для тестування програмного коду програми, я створюю альтернативну функцію *main*, у якій, спочатку створюю об'єкти з попередніми даними для студентів, лекцій та практичних занять. Потім створюю об'єкт факультету і додаю ці дані до факультету за допомогою відповідних методів. Надалі перевіряю правильність виконання кожної із операцій мого програмного коду.

```
int main() {
    Student student1("Сікорський Ігор Іванович", "Група А", "Комп'ютерні науки");
    Student student2("Нечай Михайло Потапович", "Група Б", "Електротехніка");

    Lecture lecture1(
            "Лекція з математики",
            "101",
            "Комп'ютерні науки",
            "10:00",
            "Понеділок"
    );
    Lecture lecture2(
            "Лекція з фізики",
            "201",
            "Електротехніка",
            "02:00",
            "Середа"
    );

    Practice practice1(
            "Практична з комп'ютерних наук",
            "303",
            "Комп'ютерні науки",
            "Група А",
            "09:00",
            "Вівторок"
    );
    Practice practice2(
            "Практична з електротехніки",
            "405",
            "Електротехніка",
            "Група Б",
            "11:00",
            "Четвер"
    );

    Faculty faculty;

    std::cout << "Перевірка операції №1. Додання студента." << std::endl;

    faculty.addStudent(student1);
    faculty.addStudent(student2);

    std::cout << std::endl;

    std::cout << "Перевірка операції №2. Додання лекції." << std::endl;

    faculty.addLecture(lecture1);
    faculty.addLecture(lecture2);

    std::cout << std::endl;
```

```
    std::cout << "Перевірка операції №3. Додання практичного заняття." << std::endl;

    faculty.addPractice(practice1);
    faculty.addPractice(practice2);

    std::cout << std::endl;

    std::cout << "Перевірка операції №4. Перегляд списку студентів." << std::endl;

    faculty.displayStudents();

    std::cout << std::endl;

    std::cout << "Перевірка операції №5. Перегляд списку лекцій." << std::endl;

    faculty.displayLectures();

    std::cout << std::endl;

    std::cout << "Перевірка операції №6. Перегляд списку практичних занять." << std::endl;

    faculty.displayPractices();

    std::cout << std::endl;

    std::cout << "Перевірка операції №7. Перегляд списку груп студентів." << std::endl;

    faculty.displayGroups();

    std::cout << std::endl;

    std::cout << "Перевірка операції №8. Перегляд списку студентів у заданій групі (виконую пошук за
групою \""
              << student2.getGroup() << "\")" << std::endl;

    faculty.displayStudentsByGroup(student2.getGroup());

    std::cout << std::endl;

    std::cout << "Перевірка операції №9. Перегляд списку спеціальностей." << std::endl;

    faculty.displaySpecialties();

    std::cout << std::endl;

    std::cout << "Перевірка операції №10. Перегляд списку груп студентів певної спеціальності "
              << "(виконую пошук за спеціальністю \"" << student1.getSpecialty() << "\")" << std::endl;

    faculty.displayGroupsBySpecialty(student1.getSpecialty());

    std::cout << std::endl;

    std::cout << "Перевірка операції №11. Перегляд графіку заннять на день "
              << "(виконую пошук за днем \"" << lecture2.getDay() << "\")" << std::endl;

    faculty.displayLecturesAndPracticesByDay(lecture2.getDay());

    std::cout << std::endl;

    return 0;
}
```

Результат виконання тесту:

```
[redacted]@debian:~/Documents/kpi/basics_of_programming_2/coursework$ g++ coursework.cpp -o coursework
[redacted]@debian:~/Documents/kpi/basics_of_programming_2/coursework$ ./coursework
Перевірка операції №1. Додання студента.
Студента успішно додано!
Студента успішно додано!

Перевірка операції №2. Додання лекції.
Лекцію успішно додано!
Лекцію успішно додано!

Перевірка операції №3. Додання практичного заняття.
Практичне заняття успішно додано!
Практичне заняття успішно додано!

Перевірка операції №4. Перегляд списку студентів.
Список студентів:
Ім'я: Сікорський Ігор Іванович | Спеціальність: Комп'ютерні науки | Група: Група А
Ім'я: Нечай Михайло Потапович | Спеціальність: Електротехніка | Група: Група Б

Перевірка операції №5. Перегляд списку лекцій.
Список лекцій:
Дисципліна: Лекція з математики | Аудиторія: 101 | Спеціальність: Комп'ютерні науки | Час проведення: 10:00 | День проведення: Понеділок
Дисципліна: Лекція з фізики | Аудиторія: 201 | Спеціальність: Електротехніка | Час проведення: 02:00 | День проведення: Середа

Перевірка операції №6. Перегляд списку практичних занять.
Список практичних занять:
Дисципліна: Практична з комп'ютерних наук | Аудиторія: 303 | Спеціальність: Комп'ютерні науки | Група: Група А | Час проведення: 09:00 | День проведення: Вівторок
Дисципліна: Практична з електротехніки | Аудиторія: 405 | Спеціальність: Електротехніка | Група: Група Б | Час проведення: 11:00 | День проведення: Четвер

Перевірка операції №7. Перегляд списку груп студентів.
Список груп студентів:
Група А
Група Б

Перевірка операції №8. Перегляд списку студентів у заданій групі (виконую пошук за групою "Група Б")
Студенти з групи Група Б:
Ім'я: Нечай Михайло Потапович | Спеціальність: Електротехніка

Перевірка операції №9. Перегляд списку спеціальностей.
Список спеціальностей:
Електротехніка
Комп'ютерні науки

Перевірка операції №10. Перегляд списку груп студентів певної спеціальності (виконую пошук за спеціальністю "Комп'ютерні науки")
Групи для спеціальності Комп'ютерні науки:
Група А

Перевірка операції №11. Перегляд графіку заннять на день (виконую пошук за днем "Середа")
Список занять на Середа:
Дисципліна: Лекція з фізики | Аудиторія: 201 | Час проведення: 02:00
```

5. Висновок.

Написаний код, представляє систему для керування розкладом занять в університеті. Він включає класи *Lesson*, *Student*, *Lecture*, *Practice* та *Faculty*.

Клас *Lesson* представляє окремий урок і містить інформацію про предмет, аудиторію, час та день проведення.

Клас *Student* представляє студента і містить інформацію про його ім'я, групу та спеціальність.

Клас *Lecture* успадковує клас *Lesson* і додає інформацію про спеціальність.

Клас *Practice* також успадковує клас *Lesson* і додає інформацію про спеціальність та групу.

Клас *Faculty* містить вектори студентів, лекцій та практичних занять. Він надає різні методи для додавання студентів, лекцій і практичних занять, а також для відображення списків студентів, лекцій, практичних занять, груп студентів, студентів за групою, спеціальностями студентів, групами за спеціальністю та розкладом занять на певний день.

У функції *main* використовується об'єкт *Faculty*, який надає можливість взаємодії з системою через текстове консольне меню. Користувач може виконувати різні операції, такі як додавання студентів, лекцій і практичних занять, а також відображення різних списків та розкладу занять.

Цей код можна використовувати як основу для подальшого розширення і розробки системи керування розкладом занять в університеті.