

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Інститут прикладного системного аналізу

(повна назва інституту/факультету)

Кафедра системного проектування

(повна назва кафедри)

«На правах рукопису»

УДК 004.852

«До захисту допущено»

Завідувач кафедри

_____ А.І.

Петренко

(підпис)

(ініціали,

прізвище)

“ _____ ” _____ 2018

р.

Магістерська дисертація

зі спеціальності
проектування

Інформаційні системи та технології

(код і назва спеціальності)

на тему: Вирішення задачі автоматизованого формування розкладу навчального закладу за допомогою генетичних алгоритмів

Виконав: студент 6 курсу, групи ДА-61м

(шифр групи)

Мулява Ігор Ярославович

(прізвище, ім'я, по батькові)

(підпис)

Науковий керівник доцент, к.т.н. Безносик О.Ю.

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Консультант
О.Ю.

Розроблення стартап-проекту доцент, к.т.н. Безносик

(назва розділу)

(науковий ступінь, вчене звання, прізвище, ініціали)

(підпис)

Рецензент _____

(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали)

(підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

(підпис)

Київ – 2018

**Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»**

Інститут/факультет Інститут прикладного системного аналізу
(повна назва)

Кафедра _____ Системного проектування _____
(повна назва)

Рівень вищої освіти _____ Другий (Магістерський) _____

Спеціальність _____ Інформаційні системи та технології проектування _____
(код і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ А.І. Петренко
(підпис) (ініціали, прізвище)

« ____ » _____ 2018 р.

**ЗАВДАННЯ
на магістерську дисертацію студенту**

Мулява Ігорю Ярославовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації «Вирішення задачі автоматизованого формування розкладу навчального закладу за допомогою генетичних алгоритмів»
науковий керівник дисертації Безносик Олександр Юрійович, к.т.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « ____ » _____ 20__ р. № _____

2. Строк подання студентом дисертації 01.05.2018

3. Об'єкт дослідження Розклад навчального закладу

4. Предмет дослідження Можливість автоматизованого формування розкладу за допомогою генетичних алгоритмів

5. Перелік завдань, які потрібно розробити Провести огляд існуючих методів та технологій побудови розкладів, розробити програму з реалізацією за допомогою генетичного алгоритму та провести їх глибокий аналіз та порівняння, оформити роботу на основі отриманих результатів

6. Орієнтовний перелік публікацій Мулява І. Я. Вирішення задачі автоматизованого формування розкладу навчального закладу за допомогою генетичних алгоритмів // Міжнародний науковий журнал "Інтернаука". — 2018. — №9; Мулява І.Я. Програмна модель формування розкладу навчальних занять//Наука онлайн: Міжнародний електронний науковий журнал — 2018. — №5

7. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Реалізація стартап-проекту	Безносик О. Ю., доцент		

8. Дата видачі завдання 01.02.2018

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	01.02.2018	
2	Збір інформації та аналіз літератури	15.02.2018	
3	Проведення огляду існуючих методів колаборативної фільтрації	28.02.2018	
4	Реалізація алгоритмів колаборативної фільтрації та їх тестування.	13.04.2018	
5	Аналіз та порівняння результатів моделювання	25.04.2018	
6	Оформлення дипломної роботи	30.04.2018	
7	Отримання допуску до захисту та подача роботи в ДЕК	10.05.2018	

Студент

(підпис)

Мулява І.Я.

(ініціали, прізвище)

Науковий керівник дисертації

(підпис)

Безносик О.Ю.

(ініціали, прізвище)

РЕФЕРАТ НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ

виконану на тему: Вирішення задачі автоматизованого формування розкладу навчального закладу за допомогою генетичних алгоритмів студентом Мулявою Ігорем Ярославовичем

Робота виконана на 91 сторінці, включає в собі 35 рисунків, 23 таблиці, 15 посилань, 1 додаток на 15 сторінок.

Актуальність теми

Самосвіта з кожним роком перетворюється у необхідність завдяки розвитку інтернету та вільному доступу до інформації. Всюди з'являються онлайн курси, онлайн університети та школи при компаніях, які виховують собі співробітників. Щоб організувати процес треба мати план, який допоможе ефективно оптимізувати час та можливості всіх учасників процесу.

Дана робота присвячена автоматизованому знаходженню розкладу навчального закладу, яка допоможе розумним чином розрахувати навчальний процес.

Мета та задачі дослідження

Метою роботи є розробка алгоритму, який буде знаходити та оптимізувати розклад. Розглянути існуючі рішення та підібрати метод оптимізації.

Рішення поставлених завдань та досягнуті результати

В ході дослідження було вибрано генетичний алгоритм, як метод формування та оптимізації цільової функції розкладу. Запропоновано використанні фіт-функції на основі наперед заданих вимог до розкладу. Розроблено метод генерації потенційних розкладів на основі використання генетичного алгоритму. Визначено основні особливості та метод задання фіт-функції.

Об'єкт досліджень

Розклад навчального процесу

Предмет досліджень

Алгоритми оптимізації, які можуть бути застосовані для вирішення задачі формування розкладу.

Методи досліджень

Для вирішення проблеми в даній роботі використовуються методи аналізу і синтезу, системного аналізу, порівняння, логічного узагальнення результатів.

Наукова новизна

Наукова новизна полягає у суміщенні нечіткої логіки у вигляді цільової функції на основі суб'єктивних переваг разом з генетичним алгоритмом.

Практичне значення одержаних результатів

Отримані результати можуть використовуватись у майбутніх дослідженнях за напрямком дослідження формування розкладу. Реалізацію даного алгоритму можна використовувати при формуванні розкладу навчального процесу.

Публікації

1. Мулява І. Я. Вирішення задачі автоматизованого формування розкладу навчального закладу за допомогою генетичних алгоритмів // Міжнародний науковий журнал "Інтернаука". — 2018. — №9.
2. Мулява І.Я. Програмна модель формування розкладу навчальних занять // Наука онлайн: Міжнародний електронний науковий журнал — 2018. — №5.

Ключові слова

Розклад навчального процесу, методи оптимізації, генетичний алгоритм, урахування жорстких вимог, урахування нежорстких умов.

РЕФЕРАТ НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ

выполненную на тему: Решение задачи автоматизированного формирования расписания учебного заведения с помощью генетических алгоритмов студентом Мулявой Игорем Ярославичем

Работа выполнена на 91 странице, содержит 35 рисунков, 23 таблицы, 15 ссылок, 1 приложение на 15 страниц.

Актуальность темы

Самообразование с каждым годом превращается в необходимость благодаря развитию интернета и свободному доступу к информации. Повсюду появляются онлайн курсы, онлайн университеты и школы при компаниях, которые воспитывают себе сотрудников. Чтобы организовать процесс надо иметь план, который поможет эффективно оптимизировать время и возможности всех участников процесса.

Данная работа посвящена автоматизированному нахождению расписания учебного заведения, которая поможет разумным образом рассчитать учебный процесс.

Цель и задачи исследования

Целью работы является разработка алгоритма, который будет находить и оптимизировать расписание. Рассмотреть существующие решения и подобрать метод оптимизации.

Решение поставленных задач и достигнутые результаты

В ходе исследования был выбран генетический алгоритм, как метод формирования и оптимизации целевой функции расписания. Предложено использование фит-функции на основе заранее заданных требований к расписанию. Разработан метод генерации потенциальных расписаний на основе использования генетического алгоритма. Определены основные особенности и метод задания фит-функции.

Объект исследований

Расписание учебного процесса

Предмет исследований

Алгоритмы оптимизации, которые могут быть применены для решения задачи формирования расписания.

Методы исследований

Для решения проблемы в данной работе используются методы анализа и синтеза, системного анализа, сравнения, логического обобщения результатов.

Научная новизна

Научная новизна заключается в совмещении нечеткой логики в виде целевой функции на основании субъективных предпочтений вместе с генетическим алгоритмом.

Практическое значение полученных результатов

Полученные результаты могут использоваться в будущих исследованиях по направлению исследования формирования расписания. Реализацию данного алгоритма можно использовать при формировании расписания учебного заведения.

Публикации

1. Мулява И. Я. Решение задачи автоматизированного формирования расписания учебного заведения с помощью генетических алгоритмов // Международный научный журнал "Интернаука". — 2018. — №9. 2. Мулява И.Я. Программная модель формирования расписания учебных занятий // Наука онлайн: Международный электронный научный журнал — 2018. — №5.

Ключевые слова

Расписание учебного заведения, методы оптимизации, эволюционный алгоритм, учета жестких требований, учета нежестких условий.

ABSTRACT ON THE MASTER THESIS

done on the topic: Solving the problem of automated formation of a school schedule
with the help of genetic algorithms
by a student Igor Mulyava

The work is done on 91 pages, contains 35 figures, 23 tables, 9 links, 1 attachment on 15 pages.

Actuality of theme

In our time, self-education has evolved from superiority over others, to the vital necessity that is only gaining momentum. Everywhere there are online courses, online universities and schools with companies that train their employees. To organize the process, we need to have a plan that can effectively optimize the time and opportunities of all participants in the process.

This thesis is devoted to the automated finding of the schedule of an educational institution, which will help to reasonably calculate the educational process.

Purpose and tasks of the research

The purpose of the work is to develop an algorithm that will find optimization of the schedule. Consider the existing solutions and pick up the optimization method.

Solution of the tasks and achieved results

During investigation, a genetic algorithm has been chosen, as a method for the formation and optimization of the target function of the decomposition. The use of fitness functions is proposed on the basis of predetermined scheduling requirements. The method of generating potential schedules based on the use of the genetic algorithm is developed. The basic features and method of setting the fitness function are determined.

Object of research

Schedule of educational institutions

Subject of research

Optimization algorithms that can be applied to solving the problem of scheduling.

Research methods

To solve the problem in this paper we use methods of analysis and synthesis, system analysis, comparison, logical generalization of the results.

Scientific novelty

Scientific novelty consists in combining fuzzy logic in the form of a target function and the basis of subjective advantages along with the genetic algorithm.

The practical value of the results

The results obtained can be used in future research in the direction of the study of the formation of a schedule. Implementation of this algorithm can be used in the formation of a school schedule.

Publications

1. I. Mulyava Solving the task of automated formation of an educational institution's schedule with the help of genetic algorithms // International scientific journal "Internet science". — 2018. — №9. 2. I. Mulyava The program model for forming the schedule of training sessions // Science Online: International Electronic Journal of Journalism — 2018. — №5.

Keywords

Schedule of an educational institution, methods of optimization, evolutionary algorithm, taking into account strict requirements, taking into account non-rigid conditions.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	16
ВСТУП.....	17
1 ВІДОМІ МЕТОДИ І РІШЕННЯ ЗАДАЧІ ФОРМУВАННЯ РОЗКЛАДІВ	19
1.1 Що таке розклад.....	19
1.1.1 Предмети	19
1.1.2 Викладачі	20
1.1.3 Студенти	20
1.1.4 Аудиторії	21
1.2 Розгляд можливих варіантів реалізації генерації розкладу і вибір варіанту для розробки.....	21
1.3 Висновки	24
2 МЕТОДИ СТВОРЕННЯ ТА ОПТИМІЗАЦІЇ РОЗКЛАДУ	25
2.1 Загальний опис алгоритму.....	25
2.2 Розробка цільової функції	27
2.3 Модель розкладу	29
2.4 Розробка алгоритму роботи програми генерації розкладу та його оцінки на основі переваг	32
2.4.1 Перша генерація розкладу.....	34
2.4.2 Змішування та створення нової генерації розкладів	35
2.4.3 Селекція та струс розкладів	38
2.4.4 Вихід із циклу формування розкладів.....	38
2.5 Висновки	38
3 РОЗРОБКА ПРОГРАМНОЇ МОДЕЛІ ТА ЇЇ ТЕСТУВАННЯ.....	39
3.1 Вибір середовища програмування.....	39

	15
3.2 Вхідні файли	41
3.3 Розробка опису програми на мові Python	44
3.3.1 Клас Dictionary	44
3.3.2 Клас Rozklad	47
3.3.3 Клас Generation	47
3.4 Розробка плану тестування розкладу	51
3.5 Розробка тестового додатку	51
3.6 Виконання тестування додатку на реальних даних	54
3.7 Висновки	59
4 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ “ГЕНЕРАТОР РОЗКЛАДУ”	61
4.1 Опис ідеї проекту	61
4.2 Технологічний аудит ідеї проекту	63
4.3. Аналіз ринкових можливостей запуску стартап-проекту	64
4.4 Розробка ринкової стратегії проекту	72
4.5 Розробка маркетингової програми	76
4.6 Висновки	80
ВИСНОВКИ	81
ПЕРЕЛІК ПОСИЛАНЬ	83
ДОДАТОК	85
ЛІСТИНГ ПРОГРАМИ	85

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Тех. підтримка – технічна підтримка, яка полягає у зворотному зв'язку між розробником та користувачем, при якому перший може допомагати іншому у вирішенні проблем зв'язаних з продуктом

Debug - процес пошуку та видалення несправної дії у програмі

Python – інтерпретована мова програмування, описана на різних мовах, яка знаходить у відкритому доступі.

Numpy – відкрита бібліотека для мови Python для математичної обробки чисел.

Pandas – відкрита бібліотека для мови Python для обробки даних на основі Numpy, яка формує дані у вигляді таблиць.

IDE – Integrated Development Environment – середовище розробки.

XAML - Extensible Markup Language – це мова розмітки використана в WPF.

C# - мова програмування від компанії Microsoft на основі CLR.

IPython Notebook – файл, створений додатком Jupyter Notebook, який містить собі і код, і вивід у форматі веб-сторінки.

Jupyter Notebook – IDE для створення ноутбуків.

ВСТУП

Задача складання розкладу розв'язується у багатьох галузях. Розклад є важливим завданням для навчального закладу. Насамперед, при плануванні виробництва, організації товарних перевезень, проектуванні та проведенні навчальних занять у середній, професійно-технічній та вищій школі. В її основу покладено необхідність забезпечення оптимального розподілу робіт серед виконавців, враховуючи просторові та часові обмеження.

Дану роботу виконує оператор, для якого ця робота є рутинною та ітеративною. Комп'ютер з цим може впоратись ефективніше. Необхідно зібрати велику кількість інформації. Враховуючи велику кількість аудиторій, предметів, викладачів, це завдання стає складнішим з розширенням навчального закладу.

Розклад залежить від багатьох факторів. Їх можна розділити на жорсткі та нежорсткі параметри. Об'єктивні – це інформація про університет, в якій зберігається інформація про аудиторії та предмети. Суб'єктивні – це побажання студентів та викладачів.

Це завдання є складним для людини, бо вимагає видавання результату у багатовимірному просторі та оптимізувати його. Треба тримати в голові всі аудиторії, при тому, що є різні види аудиторій, всі побажання студентів та викладачів, що означає необхідність цю інформацію в особистій формі зібрати, та необхідність з'єднати всі ці дані в розклад так, щоб всі були задоволені.

З іншого боку комп'ютер з цим завданням може легко впоратись. Зберігати інформацію про аудиторії в базі даних, вивантажити інформацію, сформулювати розклад та оптимізувати його – це завдання обчислювальної системи.

Оптимальність у даному контексті визначає ефективність робочих процесів. Якщо для промислового виробництва – це максимальна кількість виробленої продукції за одиницю часу, економія матеріалів та енергетичних ресурсів, зменшення собівартості продукції, підвищення продуктивності праці,

то для процесів організації навчання – це деякий розподіл занять по аудиторіях з урахуванням обмежень на спеціалізацію лабораторій та час роботи викладачів і студентів [10].

І, якщо для промислових задач забезпечення оптимальності розкладу робіт відіграло визначальну роль, то при складанні розкладу занять поняття оптимальності втрачало сенс, оскільки відповідна задача є NP-повною, а відповідальні особи при його складанні виходять лише із навчальних планів та об'єктивних обмежень і працюють в режимі «реального часу», змінюючи розклад на вимогу чи прохання того або іншого викладача. Очевидно, що одержаний таким чином розклад є далеким від оптимального і викликає нарікання як студентів, так і викладачів. Зрозуміло також, що оптимальність розкладу є поняттям неформалізованим, залежним від того, який критерій буде покладений в його основу [10].

Тому метою даної дипломної роботи є вивчення і реалізація цього процесу.

1 ВІДОМІ МЕТОДИ І РІШЕННЯ ЗАДАЧІ ФОРМУВАННЯ РОЗКЛАДІВ

1.1 Що таке розклад

Розклад – це чіткий план взаємодії людей у робочому процесі, розписаний у часі і просторі. За класичним означенням, розклад – це документ підприємства, який регламентує робочий ритм, визначає часові обмеження всіх робочих процесів і формує оптимальне розділення такого важливого ресурсу як час[7].

Формування розкладу вимагає врахування багатьох факторів: жорстких вимог та нежорстких. Жорсткими, або об'єктивними, умовами є:

- навчальний план,
- викладачі,
- студенти,
- аудиторії,
- час.

До нежорстких умов треба віднести:

- вимоги і побажання викладачів,
- вимоги і побажання студентів.

Жорсткі умови повинні виконуватись завжди, бо інакше розклад є хибним і збитковим. Нежорсткі умови можуть і не виконуватись, але їх виконання напряму впливає на ефективність розкладу з психологічної точки зору.

Розглянемо сам навчальний процес.

1.1.1 Предмети

Предмет – це певна наукова дисципліна, яку проводить певний викладач певній групі студентів. Предмет може вести викладач і в одній групі, і в

декількох групах одночасно, також декілька викладачів можуть вести у одній групі.

Предмети мають форми занять: лекції, практики, лабораторні. До класичних треба віднести ще семінари, самостійні заняття, екзамени, факультативи, проте їх дана задача не розглядає.

Кожна форма заняття повинна у часі займати дві академічні години, або пару. Ми опустимо варіанти, коли можу бути пів-пара, або більше.

1.1.2 Викладачі

Викладачі – це працівники університету, які проводять заняття. Викладач має посаду, наукове звання та ступінь, які відповідають їх досягненням на кафедрі. Чим вища посада, тим більш пріоритетні є вимоги даного викладача.

Є фізичні обмеження: викладач не може бути в двох аудиторіях одночасно. Нежорсткі: зручні для нього години, аудиторії, які зручно розташовані.

У кожного викладача є свої вимоги до розкладу і пріоритети цих вимог, тому їх всі треба враховувати. А також кожен викладач має свою важливість, яка вираховується через його посаду, науковий ступінь та звання, для спрощення обрахунків.

Певні викладачі можуть вести певні форми навчання, проте ці зв'язки уже відомі з розподілу навантаження.

1.1.3 Студенти

Студенти – такі самі важливі учасники процесу, як і викладачі, проте їх набагато більше, тому у розкладі будемо враховувати лише групи студентів, а не кожного окремо. Студенти діляться на потоки за терміном навчання – курси, та на групи по 25 – 35 людей. Кожна група має свій код, який складається з назви, номеру курсу та номеру групи.

Жорсткі вимоги є такі самі як і для викладачів, з додатковою вимогою щодо розміру аудиторій – якщо аудиторія мала, то вся група в неї не поміститься.

Щодо формування вимог все простіше, ніж з викладачами. Вимоги кожної групи будуть враховуватись залежно від голосування студентів. Тому кожній групі буде відповідати один список вимог і один список пріоритет. Формуватися голосування буде таким чином: після збору голосів та всіх вимог кожного студента, всі вимоги збираються в один список. Суперечні вимоги повністю викреслюються. Наприклад, один студент хоче, щоб у вівторок не було пар, а інших хоче навпаки. Якщо два студента не можуть дійти консенсусу, то вимоги обох відкидається [10].

Після цього формується матриця порівнянь остаточних вимог (матриця Сааті) та знаходиться власний вектор найбільшого власного числа. Цей вектор і буде відображати пріоритети та бажання всієї групи і буде використаний в цільовій функції, про яку ми поговоримо в наступних розділах [7].

1.1.4 Аудиторії

Кожна аудиторія – це кімната в приміщенні навчального закладу, обладнана спеціальним чином для проведення занять.

Аудиторія містить певну кількість місць, на яку вона розрахована. Проте деякі аудиторії можуть проводити декілька типів занять. Але такі аудиторії будуть використовуватись тільки тоді, коли відповідні будуть зайняті.

1.2 Розгляд можливих варіантів реалізації генерації розкладу і вибір варіанту для розробки

Дана задача є критичною для навчальних закладів. І всі вони намагались вирішити це завдання створення розкладу автоматично, або хоча б автоматизовано [7].

Одною з таких систем є система, розроблена Вінницьким університетом для формування розкладу магістрів та сесії, що описана в [3-4]. Схема будується на основі трьох важливих таблиць: магістранти, викладачі та дисципліни. Також ця схема враховує всі вимоги Болонського процесу та регулює їх виконання [7].

Дисципліни в розкладі так само діляться на лекційні, практичні, лабораторні та семінари. Кожне заняття потребує певну кількість часу і може викладатися викладачем з множини викладачів магістрантам з множини студентів. Тому формування розкладу полягає в рівномірному та оптимальному розміщенні предметів між викладачами та студентами. Також в даному способі існує четвертий потік даних – потік індивідуального плану магістрів, куди входять загальний потік, потоки для практичних та семінарських занять, фахові потоки та об'єднані потоки магістрантів [7].

Цільова функція, яка оцінює розклад в даній моделі, має такий вигляд:

$$\delta = \max \left(\sum_{k=1}^{m_t} w_k \sum_{j=1}^{m_x} \frac{v_{kj} \pi_{kj}}{\sum_{i=1}^{m_x} \pi_{ki}} + \sum_{j=1}^{m_h} n_k \cdot h_k \right) \quad (1)$$

де v_{kj} – індикаторна функція, яка визначає чи врахована вимога викладача для певної дисципліни, π_{ki} – пріоритет цієї вимоги, w_k – ваговий коефіцієнт рейтингу викладача, π_{kj} – пріоритети комплексу умов викладача [10].

Сам алгоритм розроблений був за таким принципом:

1. Набір інформації про дисципліни.
2. Набір та встановлення взаємозв'язків між викладачами і заняттями.
3. Набір та встановлення взаємозв'язків між групами студентів і заняттями.
4. Введення обмежень та вимог.
5. Формування розкладу.
6. Якщо розклад пройшов перевірку – передати на оптимізацію, якщо ні – переробити.

7. Оптимізація розкладу.
8. Вивід на екран або у XLS файл.

Повний алгоритм можна побачити на рисунку 1.

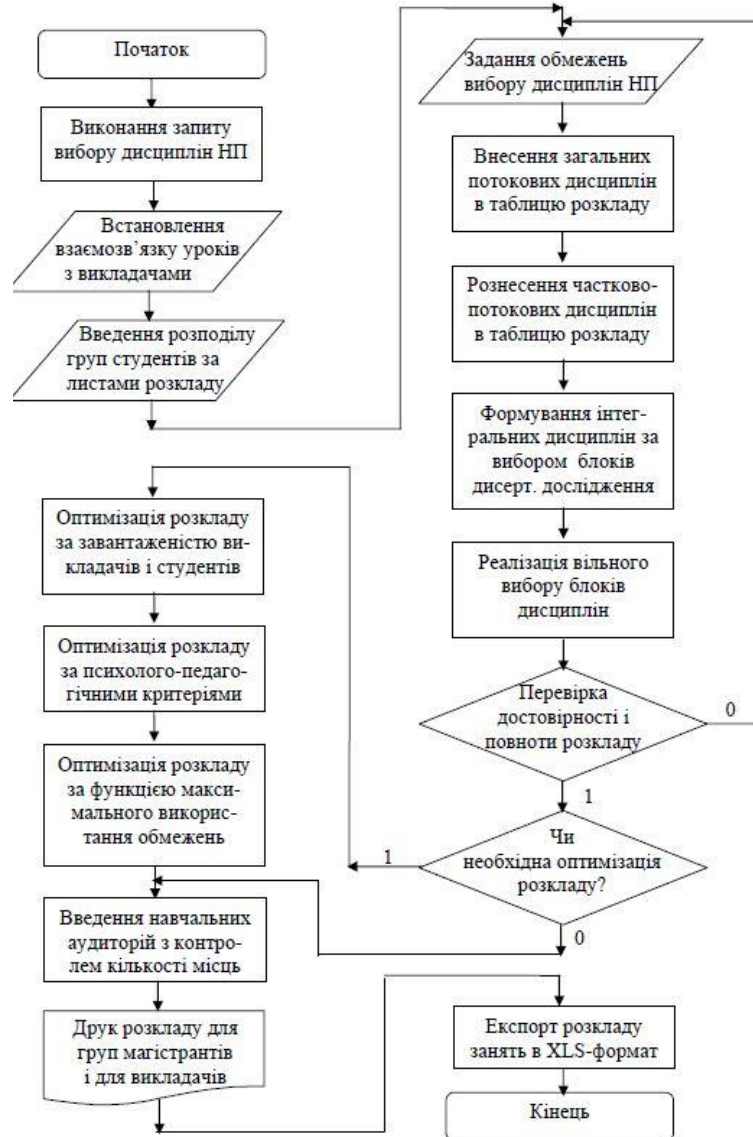


Рисунок 1.1 - Алгоритм реалізації методу формування розкладу магістратури [3]

Даний спосіб реалізації є схожий на той, що буде використаний в даній роботі, проте оптимізація та формування буде відрізнитись. Також алгоритм, описаний в даній роботі, буде враховувати не тільки вимоги викладачів, а і студентів.

1.3 Висновки

Після оцінки вищенаведеної інформації можна зробити такі висновки:

- Розклад повинен відображати навчальний процес і бути адекватним.
- Адекватність гарантується виконанням жорстких умов.
- Розклад повинен відображати побажання студентів та викладачів.
- Повинен бути створений зручний інтерфейс для користування.

Тому можна зробити висновок, що формуватися розклад буде відповідно плану, з'єднуючи дисципліни, групи, викладачів і аудиторії. Після цього розклад буде оцінюватися цільовою функцією, щоб визначити, наскільки вигідним він є. Його адекватність буде гарантуватись перевітками та самим алгоритмом. Після того як розклад буде сформований та оцінений, його треба оптимізувати певним методом та оцінити знову [7]. Оптимізувати та оцінювати треба до тих пір, поки не буде досягнуто потрібної точності.

2 МЕТОДИ СТВОРЕННЯ ТА ОПТИМІЗАЦІЇ РОЗКЛАДУ

2.1 Загальний опис алгоритму

Основною метою даної роботи є побудова зручного та оптимального розпорядку навчання, який буде максимально відповідати побажанням людей та полегшить завдання відповідальній за розклад людині. Це буде реалізовано завдяки генетичному алгоритму та цільовій функції, яка відповідатиме побажанням. Сама цільова функція буде складатися з двох частин: переваги голосування студентів та викладачів. Записано це буде в формі матриці переваг, яка буде побудована через голосування. За допомогою даної функції ми зможемо оцінити розклад [7, 10].

Сам розклад буде генеруватися за допомогою певного алгоритму, після чого буде оцінений описаною вище функцією та, у разі необхідності, буде удосконалений в наступній генерації та оцінений знову. І так до тих пір поки не буде знайдений оптимальний варіант.

Всі необхідні дані будуть зберігатись у базі даних, з якої оператор буде отримувати дані про жорсткі умови та зможе добавляти нову інформацію про навчальний заклад. Вивід розкладу буде виконуватись в XLS файл для зручного використання [7]. Повний опис алгоритму можна побачити на рис. 2.1.

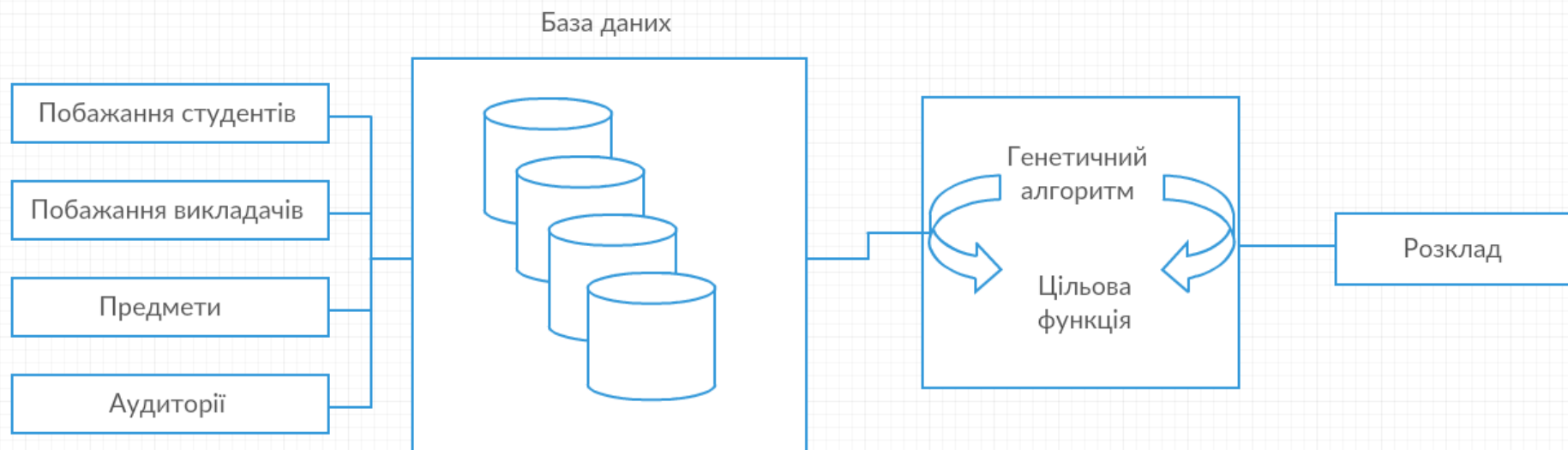


Рисунок 2.1 – Архітектура програми [7]

2.2 Розробка цільової функції

Формуючи цільову функцію, треба врахувати багато факторів, які визначають та оцінюють розклад як ефективний, коректний та оптимальний з точки зору навчального процесу. Давайте визначимо основні вимоги до функції:

- Функція повинна бути відображенням виконання вимог навчального процесу.
- Функція повинна давати більше значення тоді, коли її аргумент є вигіднішим і кращим за гірші аргументи як з об'єктивної так і з суб'єктивної точки зору.
- Функція може мати однакові значення для різних розкладів, з цього випливає, що класичні методи оптимізації до неї застосувати не можна.
- Значення, які вона повертає, не можуть бути від'ємні.
- Функція повинна мати потенціал до розширення на випадок, якщо кількість викладачів чи груп зростає.

З огляду на ці вимоги можна сформулювати формулу (2), яка буде діяти на множині (3). Ця функція є дискретною з великою кількістю розривів. Залежить вона напряму від виконання вимог розкладом. Гарантуються це завдяки двом індикаторним функціям [10].

$$F(r) = a_s \sum_{j=1}^l x_j \chi\{Z_j^v\} + a_L \sum_{j=1}^K y_j \sum_{i=1}^M \chi\{L_i \in T_i\} \sum_{j=1}^K d_{ij}^j \chi\{Z_{ij}^{Tj}\} \rightarrow \max \quad (2)$$

$$r \in \Omega(P, S, L, A) \quad (3)$$

Тут r – розклад, a_s, a_L – вагові коефіцієнти, що вказують на пріоритети викладачів і студентів як суб'єктів навчального процесу, x_j, y_j – пріоритети вимог студентів і викладачів, Z_j^v – вимоги груп студентів, L_i – викладачі, T_j – групи викладачів, Z_{ij}^{Tj} – переваги викладачів, d_{ij}^j – пріоритети таких побажань, l – кількість вимог студентів, K – кількість груп викладачів, які розподілені

посадами, науковими ступенями та вченими званнями, M – кількість викладачів, n – кількість викладачів в i -й групі, Ω – область обмежень, P, S, L, A – множини навчальних дисциплін, студентів, викладачів і аудиторій відповідно.

Пріоритетні вектори будуть формуватися вручну оператором через обмеження в ресурсах і часі під час виконання цієї роботи, але в майбутньому залишає простір до розширення [7]. Кожен вектор буде множитись на вектор-індикатор, який у відповідних позиціях буде мати «1», якщо вимога виконується, та «0», якщо – ні, як у формулі (4). Це забезпечить простоту реалізації, що дозволить програмі просто перевіряти певні умови і передавати обрахунок якомусь іншому методу [10].

$$\chi\{Z_j^v\} = \begin{cases} 1, \text{ якщо } Z_j^v \text{ виконується} \\ 0, \text{ в іншому випадку} \end{cases} \quad (4)$$

Цільова функція є досить складною, тому давайте приведемо спрощений приклад, щоб було ясно зрозуміло як вона працює. Вихідні дані для неї є вектор вимог студентських груп, вектори вимог кожного викладача та вектор пріоритетів самих викладачів, який у відповідних позиціях має коефіцієнт пріоритету кожного викладача. Спочатку іде перевірка виконання вимог, під час якої ці вектори множаться на відповідні індикаторні функції [10].

Після цього отримані значення для викладачів множаться на вектор пріоритетів і отримується остаточне значення функції для викладачів. В кінці ми повинні визначити які побажання важливіші: викладачів чи студентів, і тому помножимо значення функції для студентів на коефіцієнт студентів, а викладачів – на коефіцієнт викладачів. І результуючі значення сумуємо. Зрозуміло, що керуючи кожним вектором, кожним множником, ми можемо оцінювати розклад з багатьох боків, формуючи наші вимоги. Це дозволить нам мати простий та прямий контроль над оптимізаційним процесом.

Оскільки дана функція має не неперервний характер, то застосовувати до неї методи класичного математичного аналізу не можна. Тому було

запропоновано використовувати генетичний алгоритм. Але перед тим треба сформулювати як буде побудований розклад у пам'яті.

2.3 Модель розкладу

З огляду на інформацію, описану в першому розділі, ми можемо створити математичну модель розкладу. Напишемо розклад у такій формі, як показано у табл. 1.

Таблиця 1. - Початкова модель розкладу

День	Пара	Курс	Група	Предмет	Викладач	Тип	Аудиторія
...

Досить неочевидним фактом є те, що саме є первинним ключем у даній таблиці, бо це залежить на пряму від того, з якої сторони підійти до задачі. У даній роботі первинним ключем було вибрано комплект з Предмета, Типу заняття та Групи, у якій це заняття буде проводитись, бо саме ця інформація однозначно визначає процес навчання з точки зору студента. Тут можна зауважити, що Викладач теж є важливим полем, але один предмет можуть вести декілька викладачів, що робить його неунікальним ключем [10]. Також можна зауважити, що один предмет може викладатися у декількох групах [7].

Дана модель, описана у табл. 1, потребує спрощення, оптимізації та вдосконалення. Всю інформацію про розклад ми будемо розмішувати в паралелепіпед. Спочатку ми розділимо її на 4 групи, в які ми об'єднаємо певні поля як показано у формулах 5, 6, 7, 8:

$$X_1 = \langle \text{День} \rangle \quad (5)$$

$$X_2 = \langle \text{Пара} \rangle \quad (6)$$

$$X_3 = \langle \text{Аудиторія} \rangle \quad (7)$$

$$Z = \langle \text{Викладач} - (\text{Предмет} - \text{Тип}) - (\text{Курс} - \text{Група}) \rangle \quad (8)$$

Тоді X_1, X_2, X_3 – координати вузла у паралелепіпеді, а Z – значення вузла. Поля Група і Курс (далі просто Група) можна об'єднати в одне поле, бо вони однозначно відрізняють групу на факультеті. Предмет і Тип (далі просто Предмет) об'єднуються, бо їх суміщення однозначно визначаються одиницю навчального процесу з точки зору дисципліни. День, Пара і Аудиторія не розділяються, бо вони відображають фізичні жорсткі умови, які не можна порушити, і це буде гарантувати нам те, що в одній аудиторії в той самий час не буде проходити два заняття [7].

Поля Викладач – Предмет – Група будуть міститись у вузлах паралелепіпеда, що виходить з точки зору звичайної логіки. Тому запис у списку розкладу у нашій моделі буде виглядати так:

$$r_n = (D_i, T_j, R_k, N_l) \quad (9)$$

Тут D_i – день тижня, T_j – номер пари, R_k – аудиторія, N_l – ланка, яка з'єднує в собі ключ $\langle \text{Викладач} - \text{Предмет} - \text{Група} \rangle$, r_n – Розклад. Звідси можна зробити висновок про розмірність паралелепіпеду, який буде мати 3 виміри з четвертим у вузлах. Розміри паралелепіпеду будуть статичні, оскільки в тижні всього 6 робочих днів, та в день може бути лише 5 пар, а кількість аудиторій буде братись з даних про кафедру [7]. Графічно можна зобразити розклад як показано на рис. 2.2.

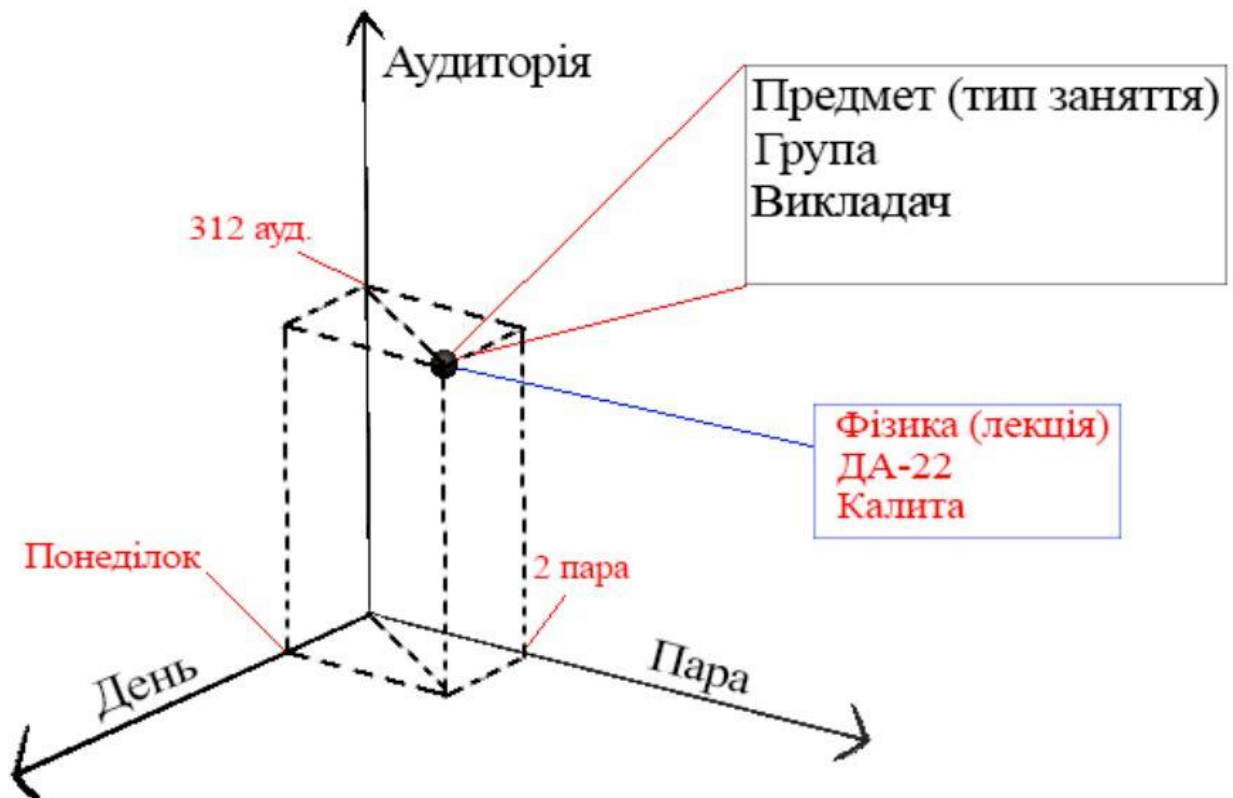


Рисунок 2.2. – Графічне уявлення розкладу

Для кращого розуміння рис. 2.2 давайте розглянемо приклад. Нехай ключ складається з лекції з дисципліни «Фізика», яка буде проводитись у групі ДА-22 викладачем Віктором Михайловичем Калитою. Пара буде проводитись в 312 аудиторії в понеділок на другій парі. Тому, як показано на рис. 2.2, у паралелепіпеді за координатами (1, 2, 312), що відповідає першому дню тижня Понеділку, другій парі та відповідній аудиторії, буде знаходитись ланка <Фізика(лекція) – ДА-22 – Калита>. Очевидно, що в цей час і в цій аудиторії інше заняття проводитись не може [7].

Формуючи розклад таким чином, ми зможемо забезпечувати жорсткі умови та мати зручний доступ до будь-якого значення чи предмету в розкладі. Треба зауважити, що обмеження, яке полягає в тому, що один викладач та одна група можуть бути лише в одній ланці одночасно, гарантуватись даною схемою не може, тому це буду гарантувати алгоритм генерації розкладу [7].

2.4 Розробка алгоритму роботи програми генерації розкладу та його оцінки на основі переваг

Після вибору моделі розкладу та цільової функції ми повинні поговорити про метод оптимізації цього розкладу. Це буде спосіб, який допоможе там покращити розклад та зробити так, щоб він відповідав вимогам, які були визначені у функції [10].

Для виконання даної задачі було обрано генетичний алгоритм, який полягає в тому, що кожна наступна генерація розкладів буде формуватися з минулих ітерацій. Після чого буде йти відбір, під час якого всі слабші розклади будуть відкидатись. Таким чином, цільова функція буде рости без жодного втручання з сторони користувача, оскільки ця модель є закритою. Проте якість результатів, які вона видає, буде рости [7].

Загальні кроки, які треба буде реалізувати алгоритмом, такі:

1. Створення першої генерації розкладів випадковим чином.
2. Оцінка цих розкладів цільовою функцією.
3. Генерація нового “потомства” з минулої ітерації.
4. Відкидання гірших розкладів.
5. Повторювати кроки 2-4 до тих пір, поки не буде отримана задана точність, або кількість кроків перебільшить допустиму.

Для зручності алгоритм також показаний на рис. 2.3.

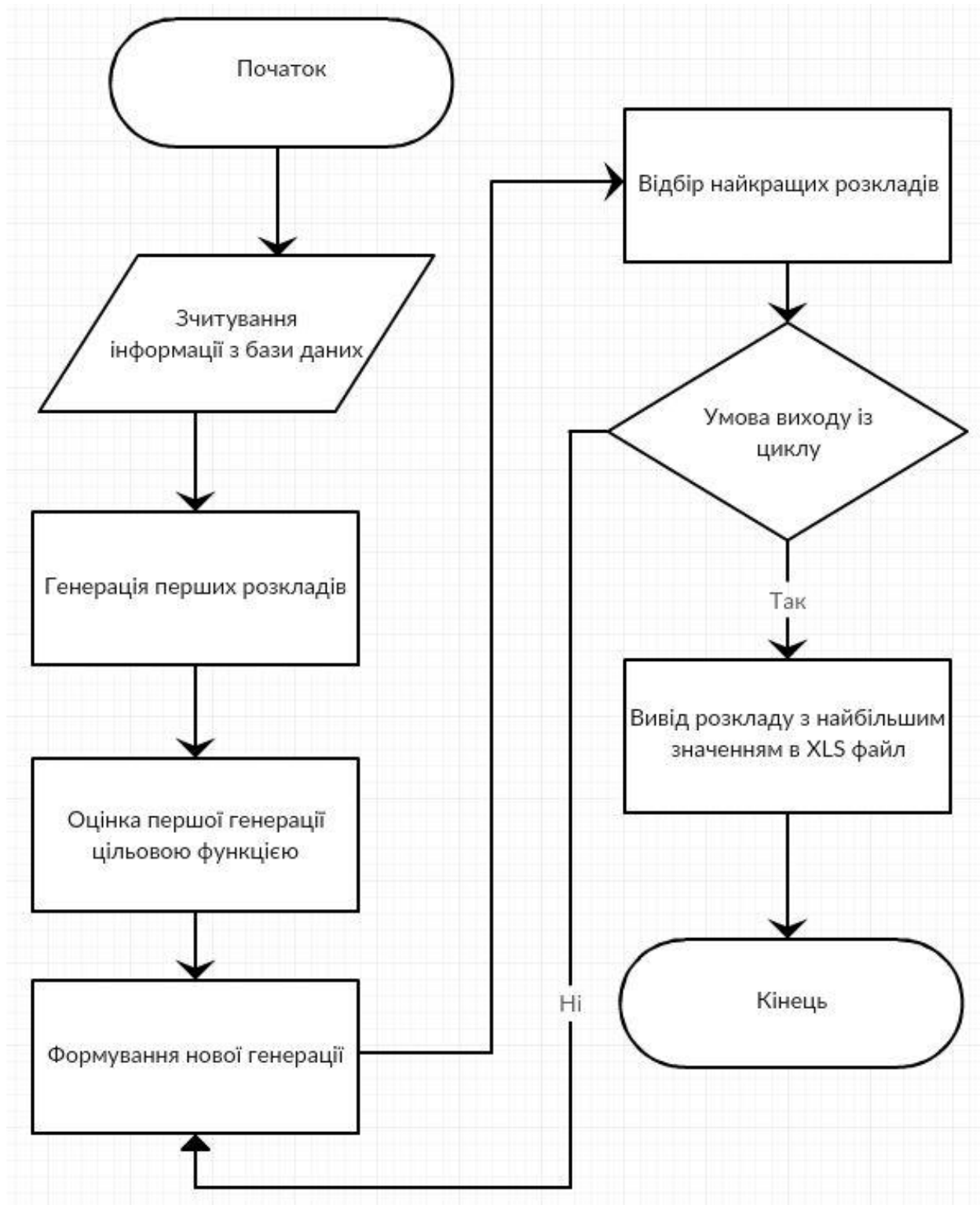


Рисунок 2.3 – Блок-схема алгоритму формування розкладу

Звісно ж, це більш загальний і не конкретний опис алгоритму, але він показує основну ідею, яка полягає в тому, що розклади будуть генеруватись випадково, змішуватись випадково, проте невипадково відбиратись. Це означає,

що шанс відібрати хороший розклад буде зростати. І завдяки сучасним технологіям такі обрахунки будуть відбуватись досить швидко.

2.4.1 Перша генерація розкладу

Перший список розкладів буде формуватися випадково. Спочатку алгоритмом вибирається випадковий день тижня та час, вибирається аудиторія, щоб було достатньо місць на групу. Потім іде перевірка, чи нема в цей час в цій аудиторії якогось заняття, якщо нема, то назначаємо заняття, яке вибрали раніше зі списку занять, якщо є, то шукаємо інші координати [7].

І так повторюємо для кожного заняття. Коли розклад готовий, перевіряємо його на валідність та записуємо його у список розкладів. Коли список розкладів буде заповнений, виходимо з циклу. Блок-схему цього кроку можна побачити на рис. 2.4.

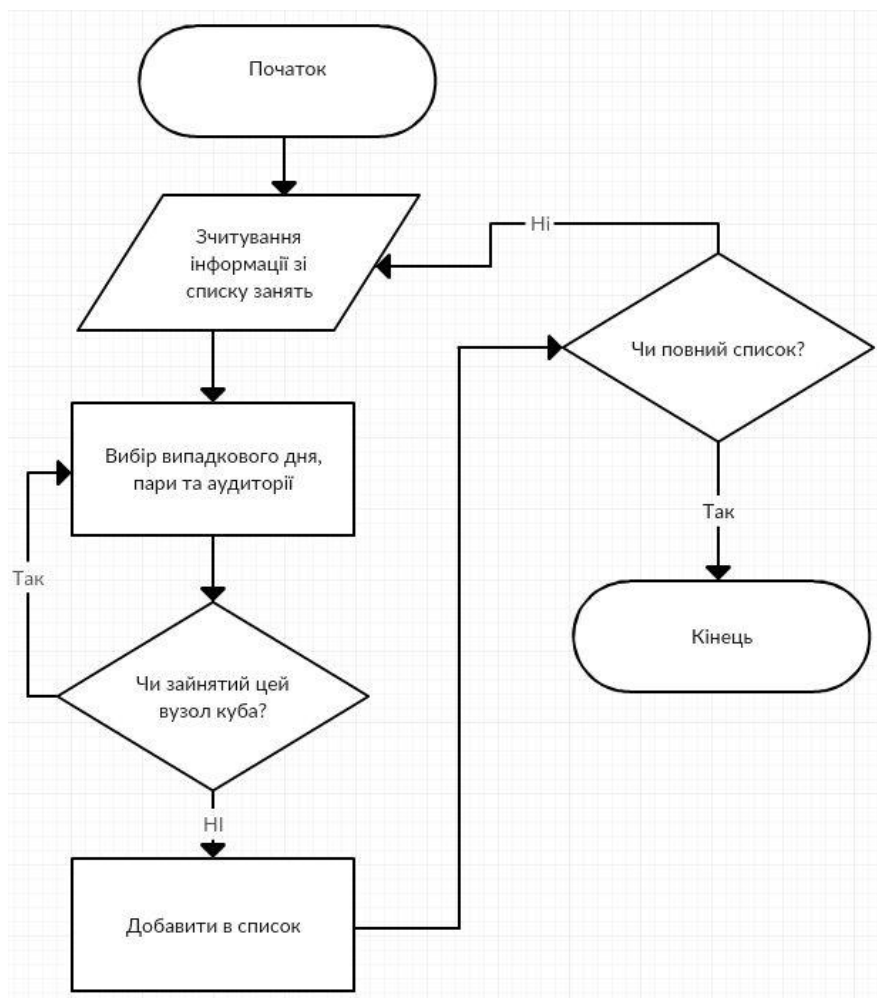


Рисунок 2.4 – Блок-схема формування першої генерації

2.4.2 Змішування та створення нової генерації розкладів

Першим кроком у створенні нового покоління є змішування двох розкладів так, щоб їхні нащадки мали риси обох з батьків. Це буде реалізовано в програмі наступним чином. Спочатку формується відрізок, який ділиться на частини, розміри яких дорівнюють відсотковому відношенню цільової функції кожного розкладу до загальної суми вартостей всієї популяції як показано на рис. 2.5. Це робиться для того, щоб розклад, який має більше значення функції, мав більше шансів на «розмноження», або мав більше шансів на те, що алгоритм його вибере для змішування [7].



Рисунок 2.5 – Відрізок вибору «батьків»

Після цього кроку, випадково вибираються дві координати на відрізку. Якщо координата попаде та той інтервал, який відповідає певному розкладу, то цей розклад і буде вибраний для змішування [7].

Змішування двох розкладів відбувається так: спочатку випадково вибираємо координати паралелепіпеду (X_1, Y_1, Z_1) . Після чого перевіряємо, чи нема в цій позиції якоїсь дисципліни, якщо нема – шукаємо далі. Після того як ми знайшли дисципліну, ми шукаємо відповідну дисципліну у другому «батьківському» розкладі з випадковими координатами (X_2, Y_2, Z_2) . Цей процес можна побачити на рис. 2.6.

Після цього відбувається обмін, але дисциплінами, а не координатами. Тобто дисципліну 1 з координатами (X_1, Y_1, Z_1) , яку ми вибрали на першому кроці, ми переміщуємо на позицію (X_2, Y_2, Z_2) , а дисципліну 2 з координатами

(X_2, Y_2, Z_2) ми переміщуємо за координатами (X_1, Y_1, Z_1). Кінцевий результат змішування можна побачити на рис. 2.7.

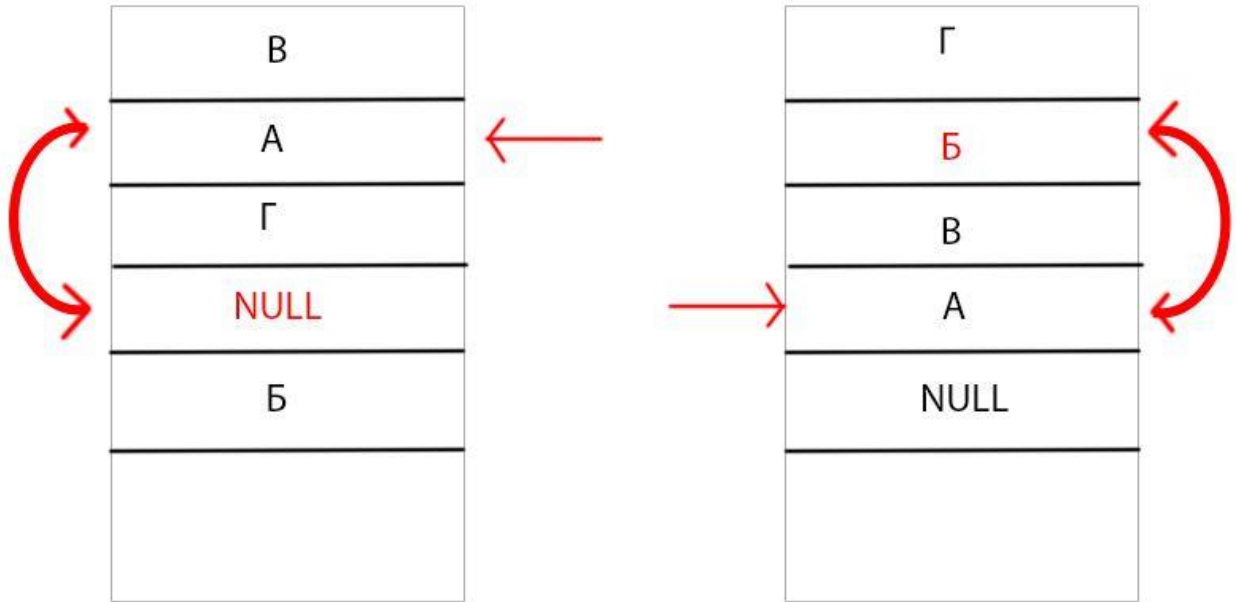


Рисунок 2.6 – Перший крок змішування

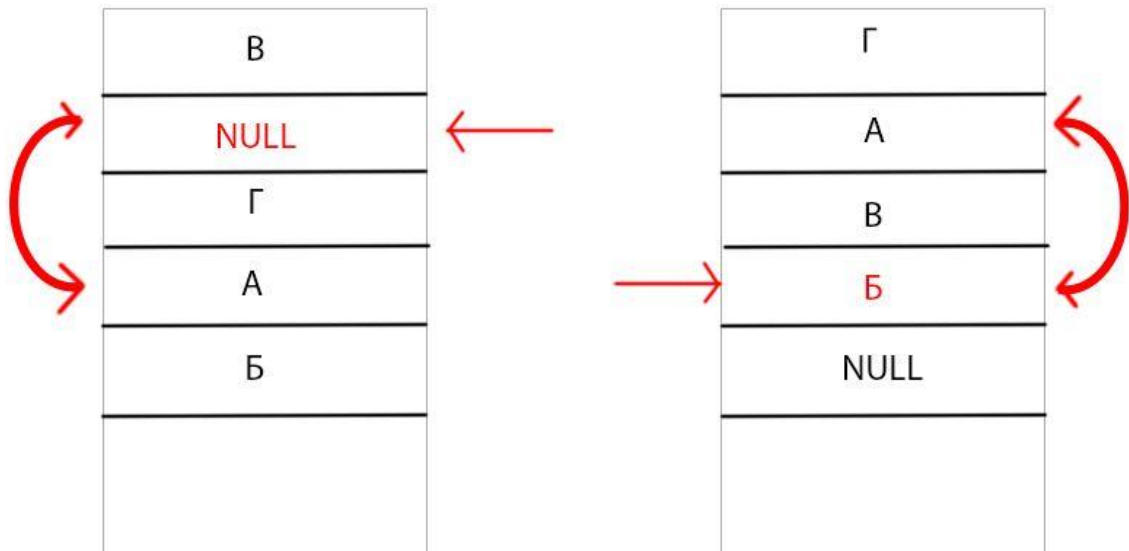


Рисунок 2.7 – Після змішування

Як видно на рисунку, множина дисциплін не змінилася. Це дуже важливий момент. Справа в тому, що якщо ми робимо зміни в самому розкладі, не беручи

нічого ззовні, то у нас збережеться цілісність навчального плану та будуть гарантуватись жорсткі вимоги, які реалізовані через пералелепід.

Після цього йде перевірка валідності розкладу, яка проходить таким чином. Спочатку робимо зріз розкладу для кожного викладача та групи студентів. Після цього ми перевіряємо цей зріз на наявність дублікатів, тобто шукаємо, чи нема такого, що один викладач знаходиться в різних аудиторіях одночасно, або одна група має дві пари одночасно. Якщо обидва «дочірні» розклади пройшли перевірку, то записуємо їх в загальний список. Так ми робимо до тих пір, поки кількість нових розкладів не буде дорівнювати кількості батьків [7].

Загальну схему формування нових потомків можна побачити на рис 2.8.

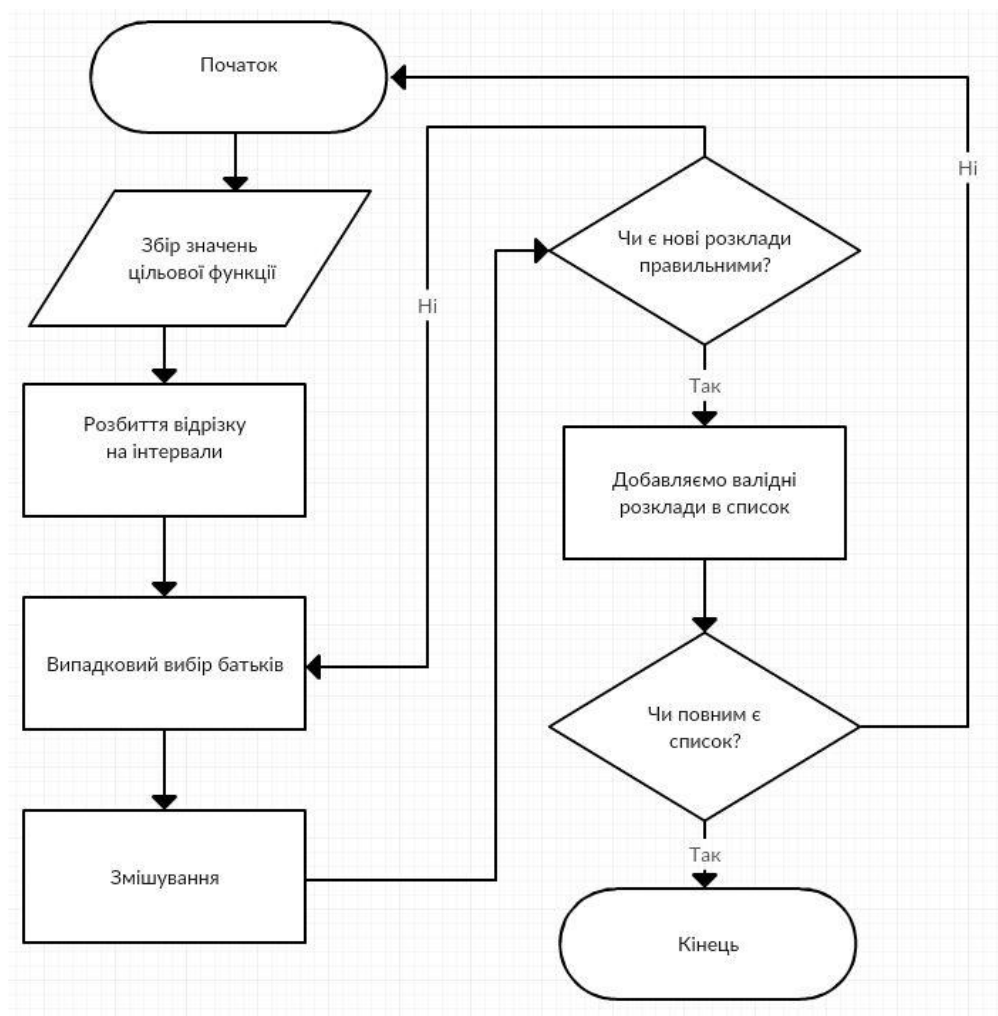


Рисунок 2.8 – Блок-схема формування потомства

2.4.3 Селекція та струс розкладів

Селекція – це процес, при якому погані розклади відкидаються, а хороші залишаються до наступного кроку алгоритму. Спочатку ми беремо всі розклади і сортуємо їх за значенням їх функцій. Після чого ми відкидаємо рівно половину розкладів.

Струс – це процес штучного введення нових нащадків з метою збагачення генофонду. Цей засіб алгоритм повинен використовувати тоді, коли функція не буде міняти значень після великої кількості кроків. Відбувається він аналогічно до першої генерації, тільки нові розклади записуються в список після старих, після чого іде змішування і селекція.

2.4.4 Вихід із циклу формування розкладів

Після закінчення кожної ітерації буде проводитись обрахунок різниці між найбільшим та найменшим значеннями в списку генерації:

$$\max_i \left| \max_j F_{ij} - \min_j F_{ij} \right| < \varepsilon \quad (10)$$

Тут F_{ij} – j -тий розклад на i -тій ітерації, ε – задана точність. Точність алгоритму ми задаємо виходячи з експериментальних досліджень та потужності системи, на якій програма виконується, бо чим більша точність, тим довше буде відбуватися пошук рішення.

2.5 Висновки

В даному розділі було розглянуто основні алгоритми та моделі, які будуть використані для виконання даної задачі. Вибрано багатовимірний паралелепіпед у якості моделі розкладу. Генетичний алгоритм було вибрано як методі оптимізації. Також було розглянуто цільову функцію, її основні складові, які будуть враховувати суб'єктивні вимоги. Розглянуто метод селекції та змішування нащадків.

3 РОЗРОБКА ПРОГРАМНОЇ МОДЕЛІ ТА ЇЇ ТЕСТУВАННЯ

3.1 Вибір середовища програмування

Для виконання даної задачі нам потрібно вибрати середовище програмування, яке буде виконувати наші вимоги, і також мати невисоку ціну. Більшість мов програмування є безкоштовними, тому необхідно виконати лише вимоги [8, 11].

Основними вимоги до середовища програмування для виконання даного проекту будуть:

- Легкість написання коду,
- Потужність мови програмування,
- Швидкість компіляції,
- Швидкість та ефективність роботи,
- Наявність технічної підтримки,
- Наявність бібліотек та інструментів, що можуть полегшити розробку,
- Зручний інтерфейс та інструментарій для оптимізації на відладки коду,
- Легке підключення до бази даних,
- Легкий функціонал для адміністрування бази даних.

Виходячи з цих вимог, вибір було зупинено на мові Python з IDE Jupyter Notebook та пакет Pandas для роботи з об'єктами. Причини цього вибору:

- Безкоштовна платформа,
- Дуже потужний інструментарій завдяки бібліотекам та вбудованим класам,

- Зручний редактор коду з виділенням синтаксису, перевіркою синтаксису в реальному часі та підказками, які допомагають виправляти помилки,
- Інтерпретованість мови,
- Об'єктно-орієнтованість мови програмування, що допоможе реалізувати велику кількість проблем,
- Автоматична система стирання зайвих даних допоможе рівномірно розділяти ресурси системи,
- Легкість написання коду на мові Python,
- Можливість компіляції для мультиплатформного користування.

Опис програмної реалізації та архітектура бази даних буде описана нижче. Вся реалізація була написана використовуючи лише інструменти Jupyter Notebook (рис. 3.1) [8].



Рисунок 3.1 – Jupyter Notebook [5]

3.2 Вхідні файли

Для того, щоб реалізувати схему, описану в розділі 2, потрібно сформувати систему вхідних файлів. Виходячи з опису в табл. 1, ми повинні створити такі файли:

1. Файл аудиторій,
2. Файл викладачів,
3. Файл груп студентів,
4. Навчальний план.

Кожна з цих таблиць повинна містити інформацію про заклад, а також реалізувати певні зв'язки одна з одною. Таблиця аудиторій повинна відображати номер аудиторії, її тип та кількість місць. Таблиця викладачів несе в собі інформацію про ім'я викладача, його науковий ступінь, яку дисципліну він веде, а також які саме форми навчання він проводить [8].

Таблиця студентів зберігає основні відомості про групи, а саме номер групи, курс, кількість студентів і код групи. Таблиця предметів зберігає назву предмету, групу, у якої дана дисципліна ведеться, та кількість лекцій, практик та лабораторних занять з цієї дисципліни, які повинні бути проведені протягом тижня. Виходячи з даного опису, ми створимо базу даних, побудовану на таких таблицях: Discipline – таблиця дисциплін, Teacher – таблиця викладачів, Rooms – таблиця аудиторій і Groups – таблиця студентів [11].

— Discipline

- Id – ID дисципліни
- disc_name – назва дисципліни
- teachers – викладачі, які ведуть курс
- group_no – групи, в якій проводиться
- type – тип заняття

— Teacher

- Id– ID викладача

- name – Ім'я викладача
- rank – наукова ступінь, звання та посада.

— Groups

- Id– ID групи
- name – номер групи
- Course – курс
- size – кількість студентів в групі.

— Rooms

- Id – номер аудиторії
- Places – кількість місць.

Повний опис зв'язків цих таблиць можна побачити на рис. 3.2. Зв'язки між таблицями будуть такі: один до багатьох від викладача до предметів, бо один викладач може вести багато дисциплін, а також один до багатьох від груп до предметів, бо одна група має багато предметів [11].

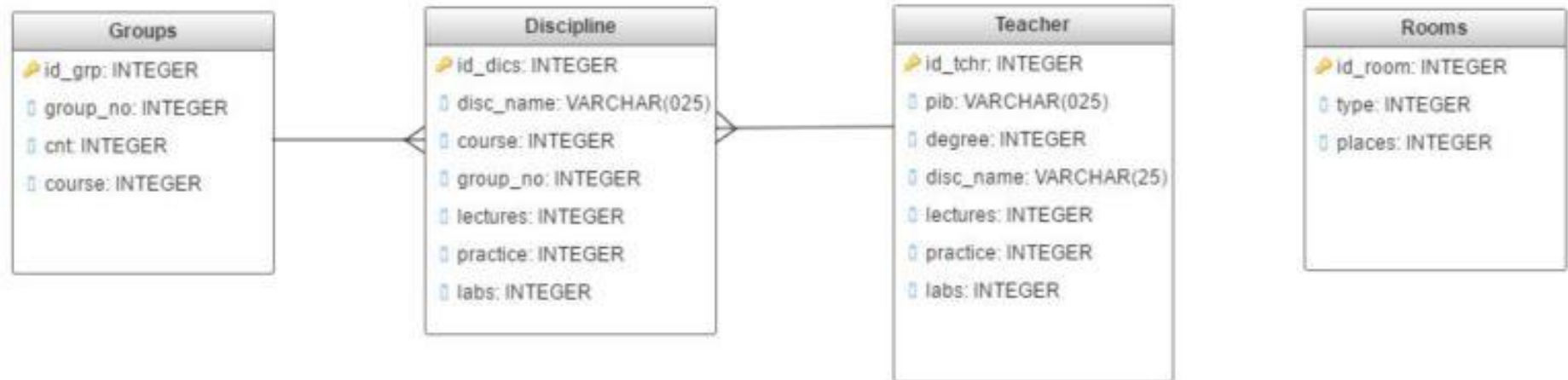


Рисунок 3.2 – Архітектура бази даних [8]

3.3 Розробка опису програми на мові Python

Щоб реалізувати ці задачі, ми створимо певний набір класів (рис. 3.3).

3.3.1 Клас Dictionary

Кожен з цих класів буде мати підкласи, які будуть реалізовувати свої завдання. Dictionary буде зберігати інформацію, яка буде завантажена з бази даних. Тому кожне його поле містить списки об'єктів, які зберігають інформацію про навчальний процес, та методи, які ці дані завантажують з бази даних. Опис класу можна побачити на рис. 3.4. Діаграму залежностей можна розглянути на рис. 3.5. У полях атрибутів цього класу можна побачити списки для кожної бази даних [8, 11].

Треба звернути уваги список предметів `subj_list`. Предмети в ньому зберігаються не так як в базі даних. Вони формуються в класі `Subj`, який зберігає кожну форму навчання з кожного предмету окремо та показаний на рис. 3.5. Наприклад, нехай у нас предмет Математика, який повинен проводитись в деякій групі. Повинно бути 2 лекції на тиждень та 1 практика, тому, виходячи з цього, у нас буде 3 записи в списку занять. Це полегшить нам подальшу генерацію розкладу [11].

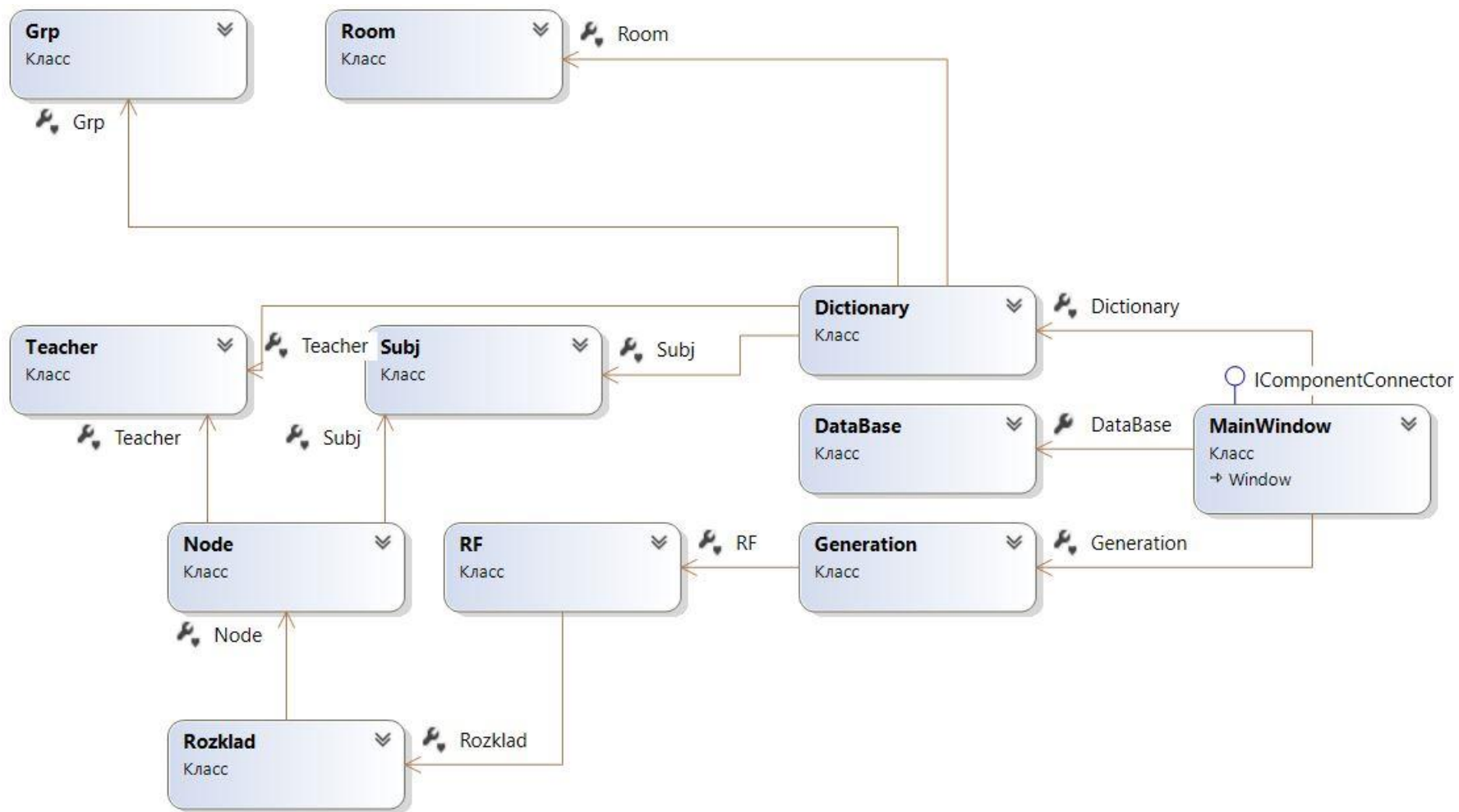


Рисунок 3.3 – Діаграма класів проекту

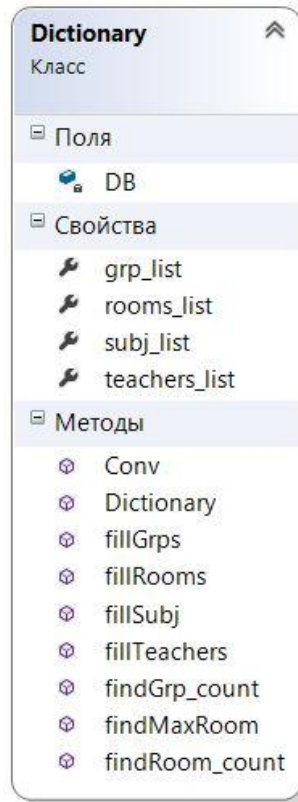


Рисунок 3.4 – Клас Dictionary

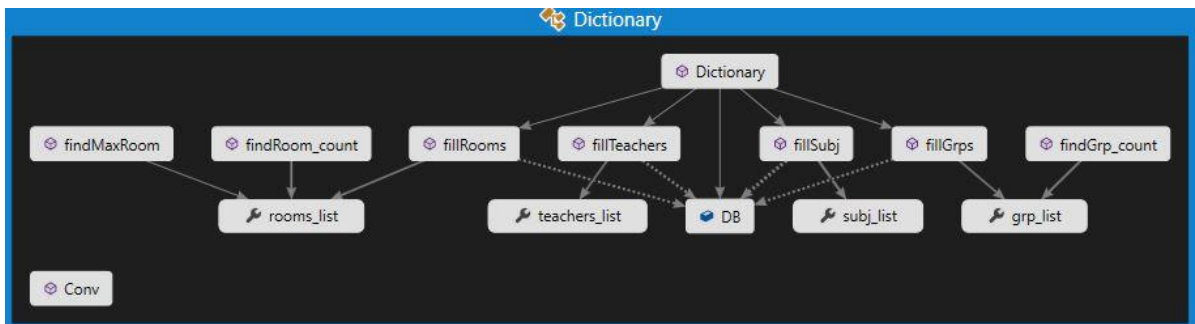


Рисунок 3.5 – Діаграма залежності класу Dictionary

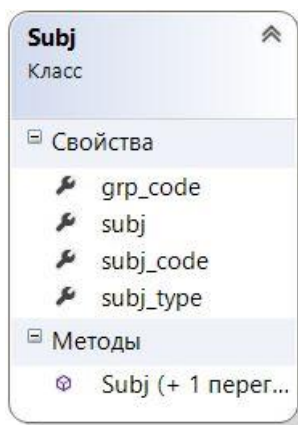


Рисунок 3.6 – Клас Subj

3.3.2 Клас Rozklad

Наступним класом, який ми розглянемо, буде Rozklad. Цей клас зберігає описаний в розділі 2 паралелепіпед, який має розмірність 6 на 6 на кількість аудиторій [8]. Діаграма залежностей показана на рис 3.7.

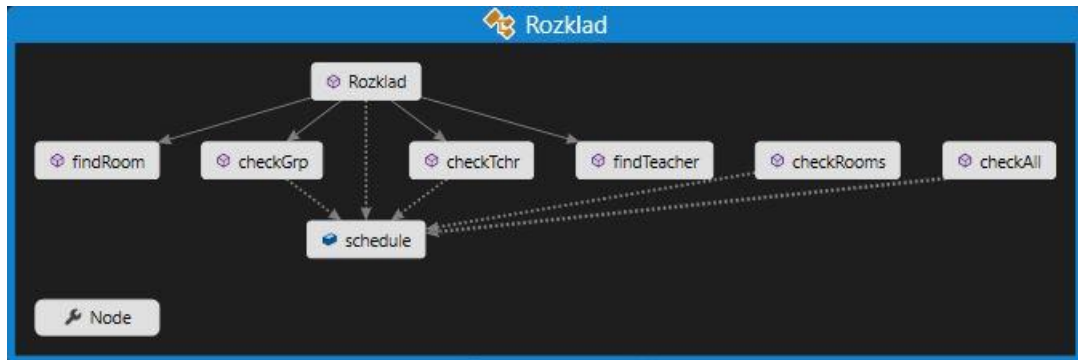


Рисунок 3.7 – Діаграма залежності класу Rozklad

Всі методи з приставкою check на початку реалізують перевірку адекватності розкладу по кожному критерію окремо, а потім викликаються з одного місця в методі checkAll. Тому, роблячи висновок, можна сказати, що цей клас точно відображає суть поставленого завдання [8].

3.3.3 Клас Generation

Даний клас зберігає в собі всі списки генерацій, а також загальний список, який буде сортуватись. Він рахує цільову функцію, а також реалізує генерацію, селекцію, еволюцію та струс популяції. Це основний клас, який реалізує поставлені нами задачі [8]. Його опис можна побачити на рис 3.8.

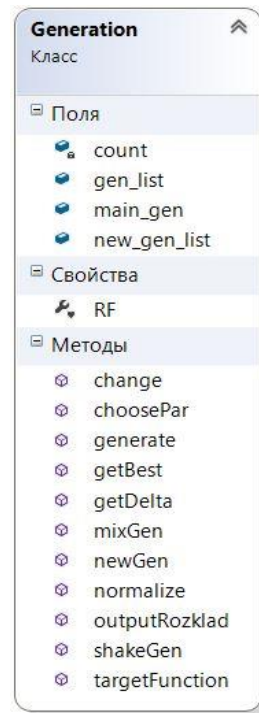


Рисунок 3.8 – Клас Rozklad

Опис його зв'язків з іншими класами можна побачити на рис 3.9.

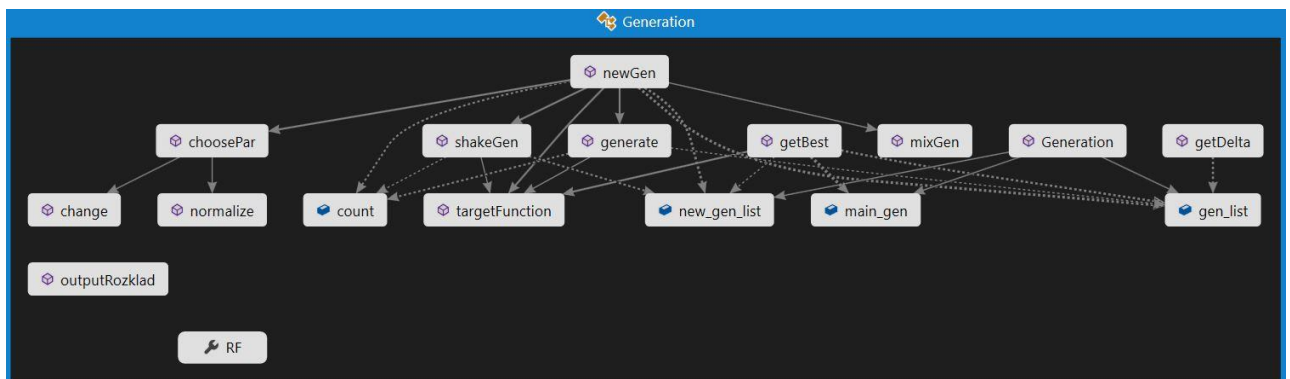


Рисунок 3.9 – Діаграма залежності класу Generation

На діаграмі чітко видно методи, необхідні для реалізації алгоритму. Поля `gen_list`, `new_gen_list` та `main_gen` відповідають списку основної генерації, новій генерації та списку, який їх об'єднує для реалізації сортування та селекції.

Методи `generate`, `mixGen`, `getBest`, `shakeGen` здійснюють генерацію, селекцію, еволюцію та струс популяції відповідно. Метод `targetFunction` рахує цільову функцію. Методи `change` і `normalize` формують вектор цільової функції, необхідний для змішування [8].

Також є службовий клас DataBase, який містить в собі методи, які з'єднують базу даних з програмним кодом. Він містить код для створення запитів до бази. Кожен запит є прописаний в окремому файлі та оптимізований перед його запуском в проекті [8].

Також вартує звернути увагу на клас RF. Його суть полягає в тому, щоб створити однозначну структуру, яка буде зберігати значення цільовою функції розкладу разом з самим розкладом і буде перераховувати це значення кожен раз, коли розклад буде мінятиь. Загальна діаграма зв'язків між класами зображена на рис 3.10.

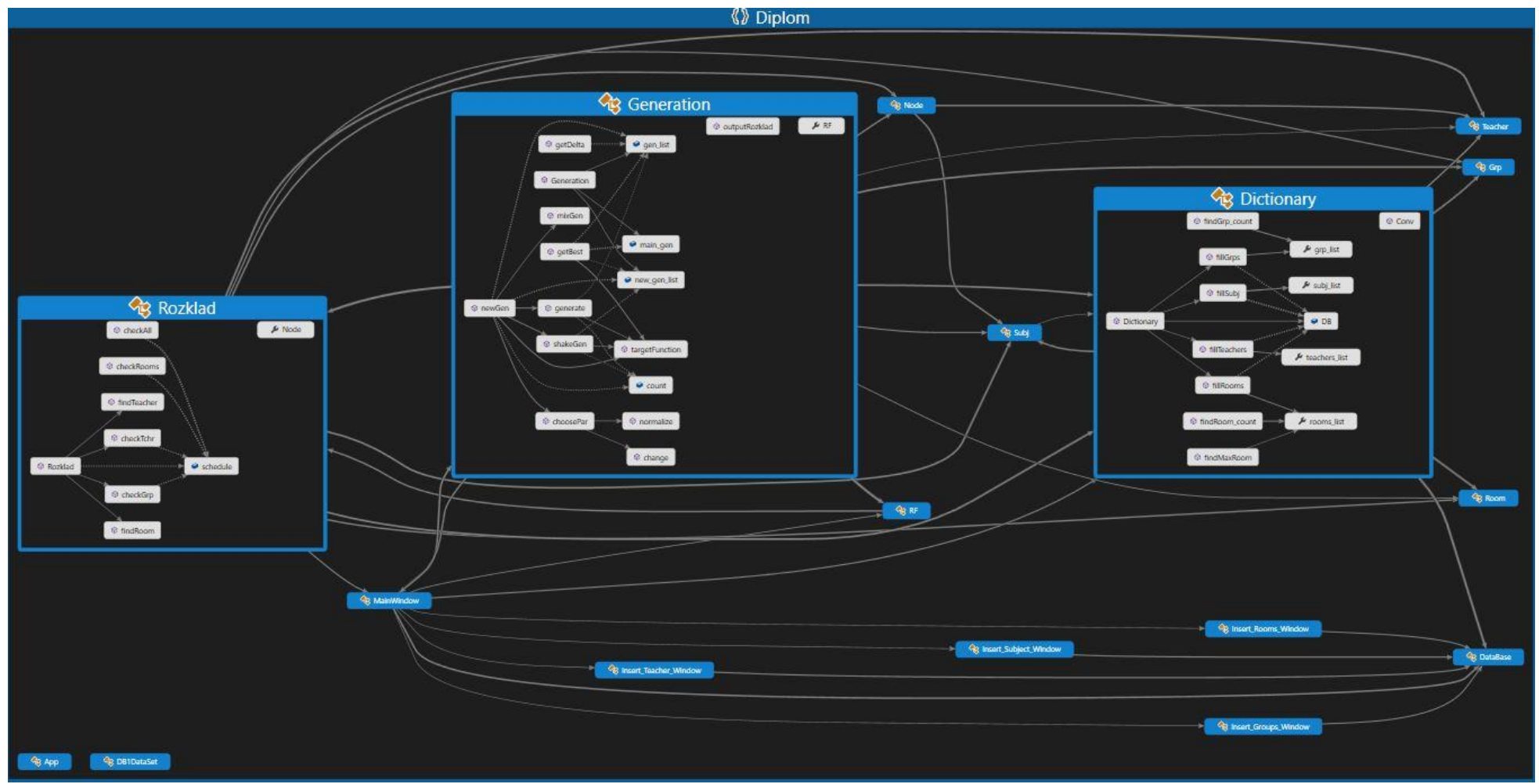


Рисунок 3.10 – Загальна діаграма залежності класів

3.4 Розробка плану тестування розкладу

Для того, щоб перевірити роботу програми, треба проаналізувати цілі, яких ми хотіли досягти, будуючи розклад. Виходячи з вимог, описаних раніше, треба сформулювати такі вимоги до розкладу:

- Зберігає всю необхідну інформацію,
- Має методи введення інформації та виведення результату обрахунків,
- Проводить всі необхідні обрахунки,
- Результатом роботи є готовий розклад занять для навчального закладу.

Проведення тестування по кожному пункту гарантує адекватність результату, його правильність та впевненість у правильній роботі алгоритму.

3.5 Розробка тестового додатку

Інтерфейс програми складається з основного вікна, яке показано на рисунку 3.11. Це вікно реалізує весь основний інструментарій, який необхідний для формування розкладу. На цьому вікні містяться кнопки адміністрування базою даних, які дозволяють добавляти нові дисципліни, викладачів, студентів та аудиторії, а також містить таблицю, де всі ці дані можна переглянути перед генерацією [11].

При натисканні на кнопки будуть виводитися окремі діалогові вікна, які показані на рис 3.12, де зручно пояснено метод користування цим інтерфейсом. Головна кнопка “Генерувати” при натисканні запускає всі необхідні для обрахунку алгоритми, які створюють першу генерацію, реалізують генетичний алгоритм та відбирають найкращі розклади [8].

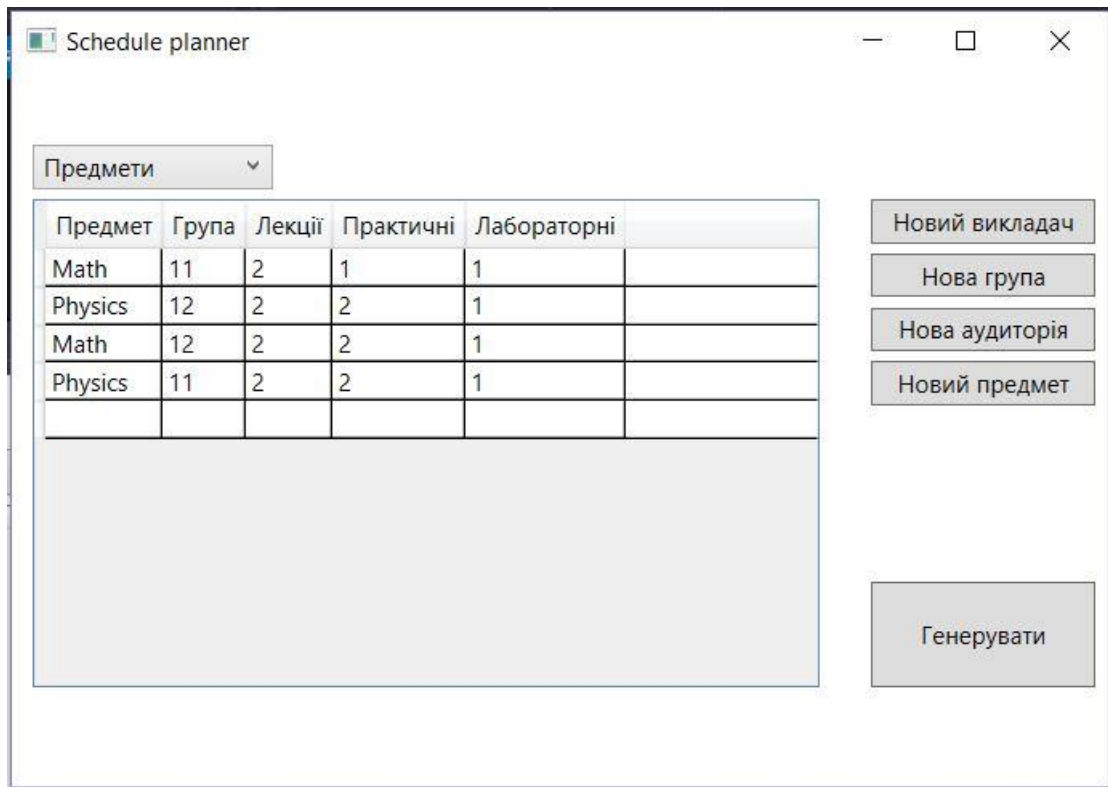


Рисунок 3.11 – Головне вікно

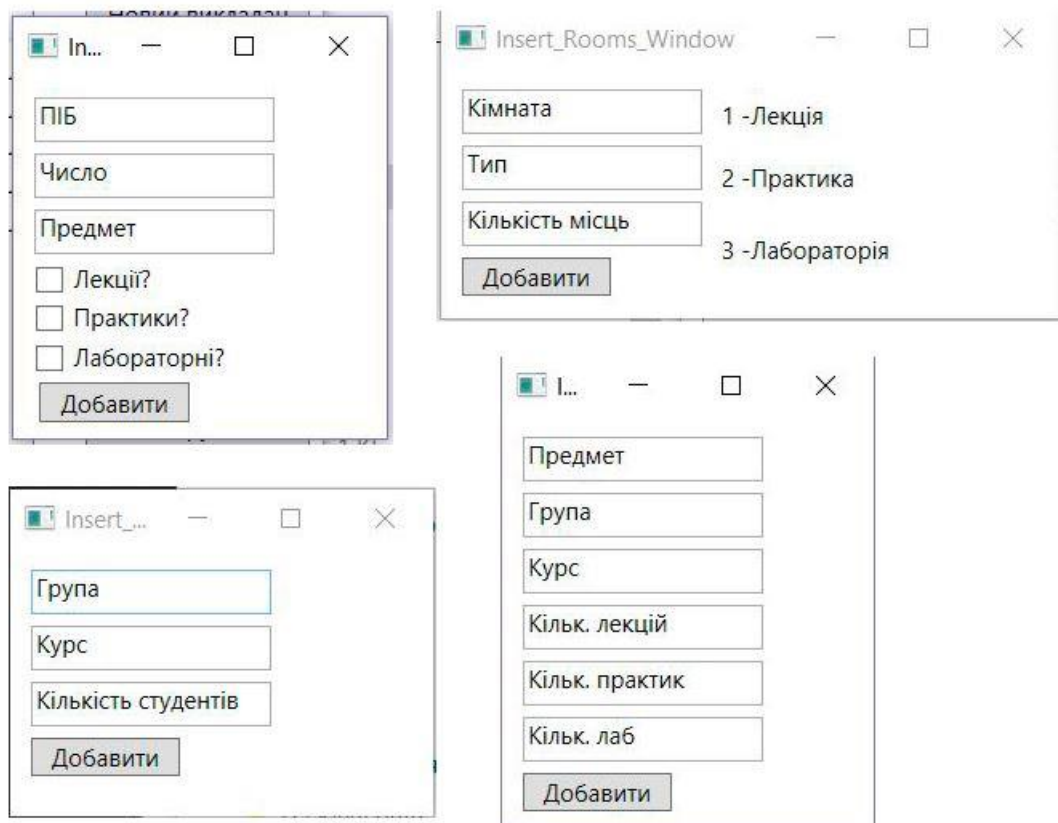


Рисунок 3.12 – Діалогові вікна введення даних

Після успішної генерації розкладу, він буде завантажений у XLSX файл на диску в папку з виконуваним файлом. Це зроблено для зручності у його публікації, та, якщо необхідно, редагуванні.

Сама тека, де будуть знаходитись всі необхідні для роботи файли, буде виглядати так, як показано на рис 3.13. Тут видно і файли запитів, і файли бази даних, і виконувані файли, і файл виводу розкладу.

Имя	Дата изменения	Тип	Размер
DB1.mdf	13.06.2016 0:20	Файл "MDF"	3 200 КБ
DB1_log.ldf	13.06.2016 0:20	Файл "LDF"	832 КБ
Diplom.exe	13.06.2016 0:13	Приложение	130 КБ
Diplom.exe.config	12.05.2016 18:05	XML Configuration...	1 КБ
Diplom.pdb	13.06.2016 0:13	Program Debug D...	252 КБ
Diplom.vshost.exe	13.06.2016 0:14	Приложение	23 КБ
Diplom.vshost.exe.config	12.05.2016 18:05	XML Configuration...	1 КБ
Rozklad.xlsx	13.06.2016 0:19	Лист Microsoft Ex...	9 КБ
SQL_groups.sql	25.05.2016 18:53	Файл "SQL"	1 КБ
SQL_rooms.sql	25.05.2016 21:18	Файл "SQL"	1 КБ
SQL_subjects.sql	25.05.2016 21:42	Файл "SQL"	1 КБ
SQL_teachers.sql	25.05.2016 18:49	Файл "SQL"	1 КБ

Рисунок 3.13 – Тека з файлами проекту

Вивід в XLSX файл здійснюється у такому форматі: колонки розділяють розклад на групи, так, щоб кожен стовпчик відповідав певній групі, рядки розділені на групи по 6, відповідно до кожного дня тижня, і при цьому кожен день тижня розділений на 6 пар. Таким чином, єдиний вивід для кожної групи формує окремий розклад. Також є можливість виводити розклад для кожного викладача, щоб полегшити поширення інформації та користування даним додатком. Тоді замість групи студентів буде виводитись в колонках прізвища викладачів. Вивід розкладу, побудованому на тестових даних, показано на рис 3.14.

	A	B	C
1		Група 11	Група 12
17	Середа 4 пара		Math практика Bohanov 102
18	Середа 5 пара		
19	Середа 6 пара		
20	Четвер 1 пара	Math лабораторна Statkevics 203	Physics практика Kalita 102
21	Четвер 2 пара		
22	Четвер 3 пара		
23	Четвер 4 пара	Math лекція Bohanov 101	
24	Четвер 5 пара		Math лекція Bohanov 101
25	Четвер 6 пара		Physics лекція Kalita 101
		Math практика	

Лист1 (+)

ГОТОВО

Рисунок 3.14 – Excel файл з розкладом

3.6 Виконання тестування додатку на реальних даних

Для перевірки роботи програми було введено дані про кафедру СП в базу даних додатку. Були введені прізвища викладачів, аудиторії та групи студентів. Було розглянуто 2 семестр 4 курсу потоку ДА-2Х. Переглянути вихідні дані можна через інтерфейс на рис. 3.15-3.18.

Schedule planner

Викладачі

Ім'я	Предмет	Лектор?	Практик?	Приймає лабор.
Golubova	CPP	Not a lecturer	Not a practitian	Labratorian
Grechko	TZI	Not a lecturer	Not a practitian	Labratorian
Kapshuk	TZI	Lector	Not a practitian	Dont even try
Kharchenko	CPP	Lector	Not a practitian	Labratorian
Kiseliiov	SA	Lector	Not a practitian	Dont even try
Roshchyna	Economics	Lector	Practitian	Dont even try

Новий викладач

Нова група

Нова аудиторія

Новий предмет

Генерувати

Рисунок 3.15 – Вихідні дані про викладачів

Schedule planner

Студенти

Курс	Група	Кількість	Код групи
4	1	29	41
4	2	30	42
0	2	58	2

Новий викладач

Нова група

Нова аудиторія

Новий предмет

Генерувати

Рисунок 3.16 – Вихідні дані про студентів

Schedule planner

Аудиторії

Аудиторія	Тип	Кільсть місць
101	Lab	35
102	Lab	35
103	Practice	35
203	Lab	35
206	Lab	40
303	Lecture	60
304	Lecture	60
307	Lecture	60
309	Practice	60
310	Practice	60
312	Lecture	60

Новий викладач

Нова група

Нова аудиторія

Новий предмет

Генерувати

Рисунок 3.17 – Вихідні дані про аудиторії

Schedule planner

Предмети

Предмет	Група	Лекції	Практичні	Лабораторні
TZI	41	0	0	1
TZI	42	0	0	1
TZI	2	2	0	0
CPP	41	0	0	1
CPP	42	0	0	1
CPP	2	2	0	0
SA	41	0	0	1
SA	42	0	0	1
SA	2	2	0	0
Economics	41	0	1	0
Economics	42	0	1	0

Новий викладач

Нова група

Нова аудиторія

Новий предмет

Генерувати

Рисунок 3.18 – Вихідні дані про навчальний план на тиждень

Сформований таким чином розклад можна побачити на рис. 3.19.

	Понеділок						Вівторок						
	1 пара	2 пара	3 пара	4 пара	5 пара	6 пара	1 пара	2 пара	3 пара	4 пара	5 пара	6 пара	
Група 21	SA лекція Kiseliov 304	Economics практика Roshchyna 103	TZI лабораторна Grechko 101	TZI лекція Karshuk 307	CPP лабораторна Kharchenko 206			SA лабораторна Chekalyuk 206	CPP лекція Kharchenko 303	SA лекція Kiseliov 304	Economics практика Roshchyna 310		
Група 22													
	Середа						Четвер						
	1 пара	2 пара	3 пара	4 пара	5 пара	6 пара	1 пара	2 пара	3 пара	4 пара	5 пара	6 пара	
Група 21	Воєнна підготовка								TZI лабораторна Grechko 102				
Група 22													
	П'ятниця						Субота						
	1 пара	2 пара	3 пара	4 пара	5 пара	6 пара	1 пара	2 пара	3 пара	4 пара	5 пара	6 пара	
Група 21		CPP лекція Kharchenko 307						SA лабораторна Chekalyuk 101	TZI лекція Karshuk 307				
Група 22													

Рисунок 3.19 –Згенерований програмою розклад

Даним розкладом виконуються всі жорсткі вимоги: немає жодних дублікатів, жоден викладач чи група студентів не знаходяться у двох місцях одночасно та вибрані відповідні аудиторії залежно від кількості студентів та типу заняття. Час обрахунку склав 2,35 секунди за 7 кроків [8].

З точки зору нежорстких вимог, розклад також реалізує ці побажання, а саме мала кількість перших пар, відсутність вікон у викладачів та студентів та мінімум пар у суботу [11].

Виконання цих вимог показує, що генерація розкладу пройшла успішно. На рис. 3.20 наведено графік залежності цільової функції від кількості кроків.

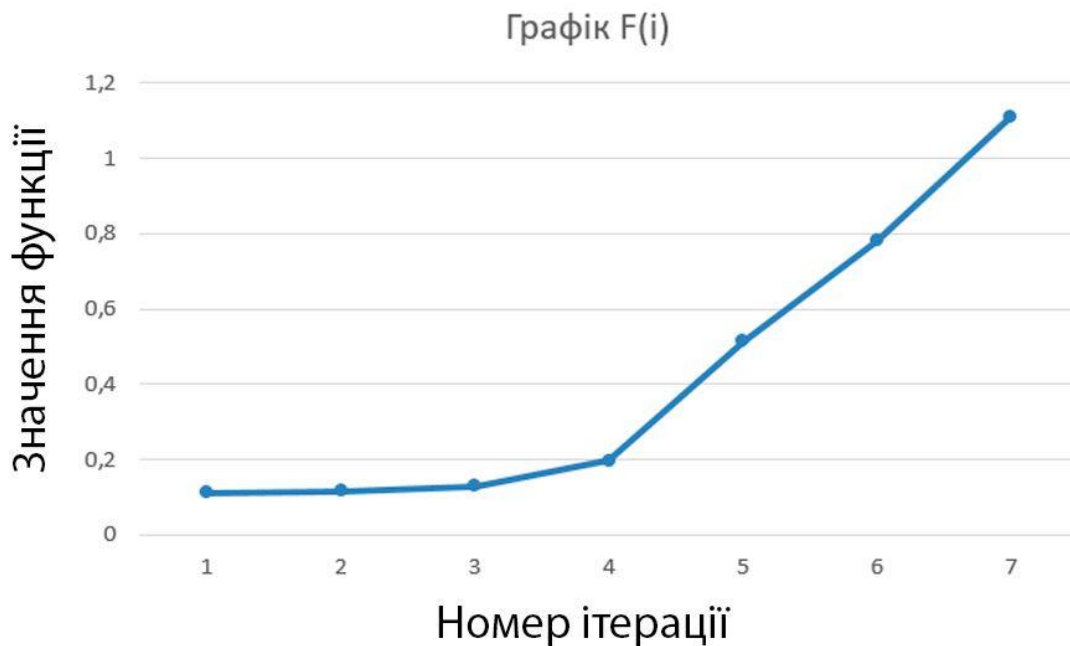


Рисунок 3.20 –Графік залежності середнього значення функції по всій генерації від ітерації еволюційного циклу

На рис. 3.21 наведено графік, який показує залежність різниці між мінімумом та максимумом цільової функції від ітерації [8].

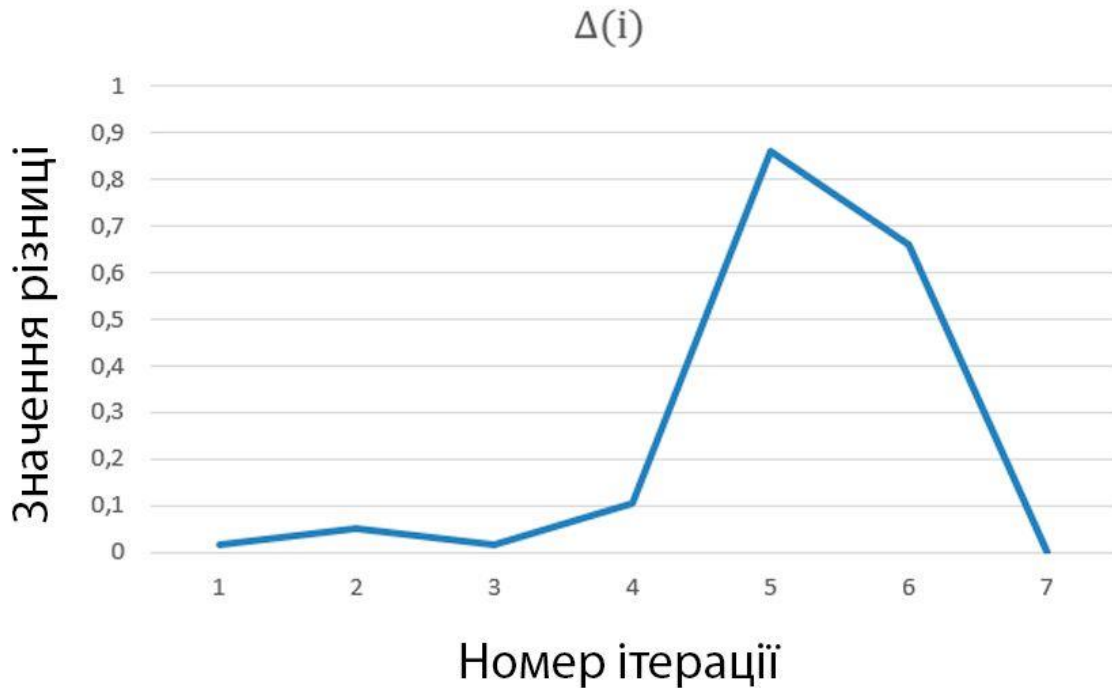


Рисунок 3.21 –Графік залежності залежність різниці між мінімумом та максимумом цільової функції від ітерації

3.7 Висновки

З огляду на отримані результати можна зробити наступні висновки. Програма формує розклад, який гарантовано відповідає жорстким вимогам. Отриманий розклад відповідає нежорстким вимогам викладачів та студентів лише частково, але достатньо, щоб рівень виконання був задовільний. Це важливий факт, бо задовольнити всіх одночасно неможливо. Вимоги викладачів були виконані в пріоритетному режимі, а вимоги викладачів, які є вищими за званнями, були виконані навіть краще [11].

Щодо часу, то виконання програми зайняло дуже мало часу, щоб згенерувати готовий робочий розклад. Цей час компенсує дні, а може навіть тижні роботи оператора факультету, якщо не враховувати інформацію, яку вводить оператор лише один раз. Такі дані, як інформація про аудиторії, викладачі, групи не часто потребує змін. А навчальні плани змінюються періодично, але не дуже сильно [8].

Щодо залежності цільової функції від ітерації, видно, що середнє її значення росте з кожною ітерацією, що є чудовим доказом коректної роботи алгоритму. Але значення різниці містить стрибок. Він в першу чергу зумовлений тим, що у певний момент часу програма випадково сформувала розклад, який мав дуже велике значення цільової функції, і це значення було набагато більше, ніж інші в його генерації. Проте після пізніших ітерацій, ця різниця спала завдяки змішуванню цього «альфа» розкладу з іншими в ітерації [8].

Результуючи все сказане вище, можна зробити висновок, що генерація пройшла успішно і розклад пройшов тестування.

4 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ “ГЕНЕРАТОР РОЗКЛАДУ”

У даному розділі проводиться опис програмного продукту, призначеного для створення розкладу занять кафедри використовуючи базу даних з інформацією про навчальний процес як стартап-проект.

За мету цього розділу взято необхідність переведення наукової ідеї у реальний продукт, який необхідно оцінювати з точки зору стану ринку та попиту; рекламувати та придумувати для нього стратегію маркетингу; та реалізовувати у сучасних реаліях глобалізованої та перенасиченої економіки.

4.1 Опис ідеї проекту

Опис стартап-проекту “Генератор розкладу” наведено у табл. 4.1.

Таблиця 4.1 – Опис ідеї стартап-проекту

<i>Зміст ідеї</i>	<i>Напрямки застосування</i>	<i>Вигоди для користувача</i>
Система, яка при завантаженні в неї інформацію про навчальний план, викладачів та студентів, а також про аудиторії, формує оптимальний розклад.	<ol style="list-style-type: none"> 1. Формування розкладу “з нуля” 2. Оптимізація існуючого розкладу 3. Налаштування розкладу під існуючий недозаповнений розклад 	<ol style="list-style-type: none"> 1. Скорочення процесу формування розкладу 2. Автоматична оптимізація 3. Уникнення ітеративного процесу

У таблиці описано основні напрямки застосування продукту, а також вигоди для користувача. Сама розробка формування розкладу автоматично є необхідною у наш час, коли кількість онлайн навчальних закладів зростає, а потреба у постійній освіті та збільшенні кваліфікації є уже необхідністю. Сам процес є ітеративним, бо вимагає узгодження багатьох питань з усіма учасниками.

Для реалізації проекту спочатку треба визначити його сильні сторони та конкурентну здатність з існуючими рішеннями.

Таблиця 4.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ n/ n	Техніко- економічні характерис- тики ідеї	<i>(потенційні) товари/концепції конкурентів</i>				W (слаб- ка сторо- на)	N (нейт- ральна сторона)	S (силь- на сторо- на)
		Проект	Конку- рент 1	Конку- рент 2	Конку- рент 3			
1.	Форма виконання	Програма	Програма	Програма	Програма	+		
2.	Собівартість	Низька	Висока	Низька	Висока			+
3.	Кросплатформність	Так	Ні	Ні	Ні		+	
4.	Наявність можливості налаштувати переваги алгоритму	Так	Ні	Ні	Ні			+
5.	Наявність можливості доповнювати існуючі рішення	Так	Ні	Ні	Так			+
6.	Застосування логічного виведення	Так	Ні	Так	Так			+
7.	Горизонтальне масштабування	Ні	Ні	Так	Так		+	

Оскільки даний додаток буде використовуватись не часто, то його низька собівартість є серйозною перевагою через відсутність необхідності у підписці. Форма виконання та кросплатформність не впливають на його конкурентоспроможність. Основними перевагами є саме можливість налаштування алгоритму з-середини та покращення існуючих рішень.

4.2 Технологічний аудит ідеї проекту

Визначення технологічної здійсненності ідеї проекту передбачає аналіз технології виготовлення товару, наявність існуючих технологій, які можна використати та доступність цих засобів з точки зору фінансів проекту.

Таблиця 4.3 – Технологічна здійсненність ідеї проекту

№	Ідея проекту	Технології реалізації	Наявність технологій	Доступність технологій
1	Генератор розкладу	Python	Є у наявності	Доступно на більшості платформах. Безкоштовна.
		Pandas	Є у наявності	Доступно на більшості платформах. Безкоштовна.
		Numpy	Є у наявності	Доступно на більшості платформах. Безкоштовна.
Обрана технологія реалізації ідеї проекту: python + pandas				

У табл. 4.3 було наведено перелік технологій, які будуть використані для реалізації даного проекту. Мова написання алгоритму буде python, яка повністю безплатна та у вільному доступі. Обробка даних відбувається у numpy. Структури даних реалізовані через Pandas. Всі технології безплатні, чим і викликана низька ціна розробки.

4.3. Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

У наступній таблиці проведено аналіз наявності попиту, обсягу попиту та динаміку розвитку ринку.

Таблиця 4.4 – Попередня характеристика потенційного ринку стартап-проекту

<i>№ п/п</i>	<i>Показники стану ринку (найменування)</i>	<i>Характеристика</i>
1.	Кількість головних гравців, од	3
2.	Загальний обсяг продаж, грн/ум.од	1000 грн./ум.од
3.	Динаміка ринку (якісна оцінка)	Стабільна
4.	Наявність обмежень для входу (вказати характер обмежень)	Немає
5.	Специфічні вимоги до стандартизації та сертифікації	Немає
6.	Середня норма рентабельності в галузі (або по ринку), %	25%

Аналіз ринку показав, що загальний обсяг продаж від трьох головних гравців в середньому 1000 грн/ум.од. Динаміка ринку стабільна, обмежень нема, як і специфічних вимог. Середня норма рентабельності 25%. Тому можна вважати, що ринок є привабливим.

Надалі визначаються потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи (табл. 5).

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проекту

№	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
	Потреба в полегшенні процесу оптимізації роботи навчального закладу	Начальство навчальних закладів, організатори навчальних заходів, літніх шкіл	Цільова група бажає платити за товар, який не буде використовуватись регулярно	Рішення має бути інтуїтивно-зрозумілим для використання та мати достатній рівень ефективності, щоб замінити людину

В табл. 4.5 було описано цільову аудиторію даного проекту. Цільова аудиторія є дуже малою, проте її продукт вирішує її проблему повністю. Особливість проекту диктує його низьку ціну. Незважаючи на це, якість повинна бути на високому рівні, щоб цільова група купила продукт. Зрозумілий інтерфейс є також важливим. Налаштування повинні бути прості та інтуїтивні.

Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (табл. № 6-7). Фактори в таблиці подавати в порядку зменшення значущості.

Ідея продукту проста настільки, що дозволяє великій компанії реалізувати її без втрати ресурсів, або студенту створити безплатну реалізацію як проект. Також через особливості цільової аудиторії передбачено безплатні версії у майбутньому.

Таблиця 4.6 – Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
	Конкуренція	Вихід на ринок безплатної реалізації даної розробки у відкритому доступі.	1. Перейти на безплатну версію з платними додатковими функціями. 2. Обрати нову цільову аудиторію і зосередитися на ній
	Економічний	Зниження платіжної здатності цільової аудиторії	1. Планове зниження ціни. 2. Перехід на фріміум систему
	Зміна потреб користувачів	Користувачам необхідний інший функціонал	1. Передбачити можливість розширення функціоналу
	Законодавчі	Заборона використання даного інструменту від міністерства	1. Перехід на сервери Amazon

Юридична складова теж може бути впливовою, але технології висять над людьми, тому це не проблема.

У табл. 4.7 розглянуто можливості для стрімкого росту продукту. Основною можливістю є стрімкий ріст потреба у розвитку та збільшенні кваліфікації у цільової групи, а також у клієнтів цільової групи, що створить необхідність у даному продукті. Оскільки таких рішень на ринку мало, то агресивна реклама не буде мати конкурентів. Також розвитку сприятиме розробка готового API, що стало вже стандартом індустрії. Інші можливості є спільними для індустрії.

Таблиця 4.7 – Фактори можливостей

<i>№ п/ п</i>	<i>Фактор</i>	<i>Зміст можливості</i>	<i>Можлива реакція компанії</i>
1.	Стрімкий ріст попиту на інструменти формування розкладу	Наявність попиту на інструменти для формування розкладу	Агресивна рекламна компанія, коли тренд у стартовій точці.
2.	Поява нових великої кількості онлайн університетів.	На хвилі Coursera з'являються різні онлайн університети.	Перехід на API, чи WEB формат
3.	Стрімке зростання ринку	Компаніям, що тільки виходять на ринок, буде простіше отримати клієнтів	Змога запропонувати продукт більшій кількості потенційних користувачів
4.	Обслуговування додаткових груп споживачів	Поява нових потенційних груп споживачів	Змога розширити продукт для подальшого впровадження у нові галузі
5.	Розширення функціоналу, або розробка продукту, який буде включати в себе дайни пакет	Поява нового функціоналу, що привабить нових користувачів	Домовленість з компаніями, які використовують даний продукт про знижку за рекламу.

Надалі було проведено аналіз пропозиції: визначили загальні риси конкуренції на ринку (табл. 4.8).

У табл. 4.8 наведено аналіз конкуренції на ринку. Визначено, що най сервіс працює в середовищі нецінової, товарно-родової конкуренції, адже наші конкуренти частково повторюють функціонал нашого сервісу, тож конкуренція ведеться за рахунок якості надання послуг.

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції: Нецінова конкуренція.	Існує декілька конкурентів, які повторюють деякі функції нашої системи	Підтримка якості продукту та постійні нововведення
2. За рівнем конкурентної боротьби: Міжнародний.	Фірми-конкуренти знаходяться як в нашій країні так і в інших країнах.	Адаптація продукту як для вітчизняних так і для зарубіжних клієнтів.
3. За галузевою ознакою: внутрішньогалузева.	Продукт використовується лише всередині даної галузі.	Постійне вдосконалення продукту.
4. Конкуренція за видами товарів: товарно-родова.	Системи конкурентів виконують подібні функції але досить відрізняються від нашої	Створити продукт, врахувавши сильні і слабкі сторони конкурентів.
5. За характером конкурентних переваг: нецінова.	Збільшення функціональності в межах однієї системи та збільшення якості її роботи	Зниження ціни на продукт та підтримка його якості.
6. За інтенсивністю: марочна.	Бренди існують і конкурують.	PR, реклама, просування бренду.

Конкуренція відбувається не лише в середині країни, а й на міжнародному ринку надання послуг.

В табл. 4.9 проведено аналіз конкуренції в галузі за М. Портером.

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	<i>Навести перелік прямих конкурентів</i>	<i>Визначити бар'єри входження в ринок</i>	<i>Визначити фактори сили постачальників</i>	<i>Визначити фактори сили споживачів</i>	<i>Фактори загроз з боку замінників</i>
Висновки	Існує 3 конкуренти на ринку. Найбільш схожим за виконанням є конкурент 2, так як його рішення також представлено у двох варіантах: Веб сервісу та мобільного додатку.	Так, можливості для входу на ринок є, бо наше рішення поєднує в собі велику кількість можливостей, а також має зручний інтерфейс та являється кросплатформним	Постачальники відсутні	Важливим для користувача є зручність у користуванні	Товари-замінники можуть використати більш дешеву технологію створення ПЗ та зменшити собівартість товару

Було визначено конкурента, чий продукт найбільш подібний до нашого. Також було визначено та обґрунтовано можливості виходу нашого сервісу на ринок, наведено його основні переваги та фактори сили споживачів. Також було розглянуто фактори загрози з боку конкурентів.

За результатами аналізу таблиці робиться висновок щодо принципової можливості роботи на ринку з огляду на конкурентну ситуацію. Також робиться висновок щодо характеристик (сильних сторін), які повинен мати проект, щоб бути конкурентоспроможним на ринку. Другий висновок враховується при формулюванні переліку факторів конкурентоспроможності у п. 3.6. На основі аналізу конкуренції, проведеного в п. 3.5 (табл. 9), а та кожної із урахуванням характеристик ідеї проекту (табл. 2), вимог споживачів до товару (табл. 5) та факторів маркетингового середовища (табл. 6-7) визначається та обґрунтовується перелік факторів конкурентоспроможності. Аналіз оформлюється за табл. 10.

В табл. 4.10 наведено обґрунтування факторів конкурентоспроможності для нашого сервісу. Головними факторами конкурентоспроможності даного сервісу є те, що він існує як в варіанті веб-сервісу, так і у варіанту мобільного додатку. Також перевагою нашого сервісу над конкурентами є дуже зручний та простий інтерфейс, за допомогою якого можна за мінімальний час отримати потрібний результат.

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1.	Використання ПЗ у вигляді веб-сервісу та мобільного додатку	Зручне використання сервісу можливе з великої кількості пристроїв. Кожен з користувачів може відкрити даний сервіс у зручний для себе спосіб з максимальною зручністю.
2.	Простота інтерфейсу користувача	Інтуїтивно зрозумілий інтерфейс з простим доступом до найважливіших функцій даного сервісу.

В табл. 4.11 наведено порівняльний аналіз сильних та слабких сторін проекту в порівнянні з проектами конкурентів.

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін проекту

№	Фактор конкурентно-спроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з нашим підприємством						
			- 3	- 2	- 1	0	+ 1	+ 2	+ 3
1.	Гнучкість налаштування пріоритетів	18			+				
2.	Простота інтерфейсу	20	+						

Можна зробити висновок, що даний проект є досить перспективним, тому що він не поступається конкурентам, а іноді й перевершує їх, як наприклад у простоті інтерфейсу.

В табл. 4.12 наведено SWOT-аналіз стартап-проекту, базуючись на характеристиках проекту, що були надані в попередніх таблицях.

Таблиця 4.12 – SWOT-аналіз стартап-проекту

Сильні сторони: простий користувацький інтерфейс, конкурентна собівартість	Слабкі сторони: висока складність реалізації проекту через відсутність відкрити напрацювань
Можливості: у конкурента 2 виявлена відсутність налаштування додатку	Загрози: конкуренція, зміна потреб користувачів, зміни в законодавстві

Описано сильні сторони, такі як зручний інтерфейс та особливості реалізації, описано головний недолік, а саме високу складність реалізації. Надано інформацію про можливості та загрози. Основними можливостями є

невдачі конкурентів та збільшення фінансових можливостей у потенційних замовників реклами. Основними загрозами традиційно є конкуренція та зміни потреб користувачів.

В табл. 4.13 наведено альтернативні варіанти реалізації стартап-проекту. Наведено відсоток ймовірності отримання позитивного результату та приблизні терміни реалізації. З зазначених альтернатив перевага надається тій, яка має більшу ймовірність отримання результату та має більш стислий термін реалізації. Тож ми обираємо першу альтернативу.

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1.	Створення ПЗ використовуючи нейронні мережі	80%	18 місяців
2.	Створення ПЗ на основі класичних методів машинного навчання	30%	24 місяців

4.4 Розробка ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів. В табл. 4.14 описано основні цільові групи потенційних споживачів.

Таблиця 4.14 - Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1.	Університети	Спрощення роботи з високо пов'язаними даними	Великий	Існує 3 конкуренти, які надають схожі, але менш швидкі та менш результативні рішення.	Швидкодія, зручний користувацький інтерфейс
2.	Онлайн курси	Спрощення роботи з високо пов'язаними даними	Великий		Швидкодія, зручний користувацький інтерфейс
Які цільові групи обрано: обираємо інтернет-користувачі та підприємства					

Наведено орієнтовний попит в межах цільової групи, описано інтенсивність конкуренції в сегменту та орієнтовну складність виходу даного проекту на ринок. Основними цільовими групами було обрано університети та навчальні заклади, які зацікавлені у полегшенні своєї роботи.

В таблиці 4.15 зазначені базові стратегії розвитку.

Таблиця 4.15 – Визначення базової стратегії розвитку

№	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1.	Створення ПЗ використовуючи нейронні мережі	Ринкове позиціонування	Швидкодія, простота у користуванні, реалізація у двох варіантах (веб-сервіс та мобільний додаток)	Диференціація

Була обрана альтернатива розвитку проекту з використанням нейронних мереж. Стратегією охоплення ринку є ринкове позиціонування. Базовою стратегією розвитку є диференціація. Зазначено основні конкурентоспроможні позиції, а саме зручність простого інтерфейсу та наявність сервісу у двох варіантах.

Було визначено базову стратегію конкурентної поведінки як зайняття конкурентної ніші (табл. 4.16).

Визначимо стратегію позиціонування (табл. 4.17), що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект. Описано основні вимоги цільової аудиторії до товару, а саме простоту у користуванні та точність генерацій. Визначено базову стратегію розвитку (диференціація). Сформовано перелік ключових позицій конкурентоспроможності даного проекту.

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1.	Ні	Так	Буде, а саме: основною задачею є розробка ПЗ з використанням генетичного алгоритму конкуренти 1, 2, 3), форма виконання - додаток (конкурент 2)	Заняття конкурентної ніші

Таблиця 4.17 – Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1.	Простота інтерфейсу, швидкодія, точність результатів	Диференціація	Простота користувацького інтерфейсу дозволить отримувати необхідні дані.	Швидкодія, простота, якість генерацій

4.5 Розробка маркетингової програми

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у табл. 4.18 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1.	Реалізація, яка швидко та якісно генерує розклад	Виконує завдання	Перевага у зручності
2.	Кросплат-форменність	ПЗ однаково успішно працює на різних платформах та операційних системах	Перевага у зручності

В табл. 4.18 наведені ключові переваги концепції нашого проекту, а саме те, що проект являється кросплатформним. Основною вигодою для користувача є те, що додаток однаково добре працює на різних пристроях та системах, що забезпечує його перевагу у зручності використання перед його конкурентами.

У табл. 4.19 показано трирівневу маркетингову модель нашого проекту.

Таблиця 4.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Універсальна система генерації розкладу для навчального закладу.		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Зручність та простота користувацького інтерфейсу 2. Якість рекомендацій 3. Кросплатформенність	Не матеріальна	Технологічна
	Якість: згідно до стандарту ISO 4444 буде проведено тестування		
	Маркування відсутнє		
	Моя компанія: "RichardSoft"		
III. Товар із підкріпленням	2-денна пробна VIP версія		
	Постійна підтримка для користувачів		
За рахунок чого потенційний товар буде захищено від копіювання: наш сервіс буде захищений ліцензією.			

Описано основний задум товару, а саме, що це має бути універсальна система генерації розкладу. Було перераховано основні властивості продукту та його характеристики. Зазначено, що новим користувачам на 3-місячний період надається з додатковими послугами.

В табл. 4.20 показано означення цінових меж, якими необхідно керуватись при встановленні ціни на товар.

Таблиця 4.20 – Визначення меж встановлення ціни

№	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
	600-650\$	700-1200\$	100\$	50-51\$

Наступним кроком є визначення оптимальної системи збуту, в межах якої було прийняте рішення (табл. 4.21):

- Проводити збут власними силами,
- Однорівневий канал збуту.

Таблиця 4.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1.	Придбання підписки або покупка продукту	Продаж	0(напрямую), 1(через одного посередника)	Власна та через посередників

Отже, система приносить прибуток завдяки щомісячним внескам для подовження ліцензії та придбанням підписок, продаж буде виконуватись напряму або через одного посередника.

Останньою складовою маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (табл. 4.22).

Таблиця 4.22 – Концепція маркетингових комунікацій

<i>№ п/п</i>	<i>Специфіка поведінки цільових клієнтів</i>	<i>Канали комунікацій, якими кори- стуютьс я цільові клієнти</i>	<i>Ключові позиції, обрані для позиціонуван ня</i>	<i>Завдання рекламного повідомлення</i>	<i>Концепція реклам- ного звернення</i>
1.	Придбання ліцензії на користування, або підписка на час використання	Інтернет	Низька ціна	Показати переваги сервісу, у тому числі і перед конкурентами	Агресивна реклама

Отже, в табл. 4.22 наведено концепцію маркетингових комунікацій, було визначено, що придбання ліцензії на користування буде здійснюватись в мережі Інтернет, необхідним буде щомісячне її продовження, або повна покупка товару.

4.6 Висновки

В даному розділі було проведено аналіз програмного продукту у якості стартап-проекту. Можна зазначити що у проекті є можливість комерціалізації, через зростання попиту на автоматизацію та перенесення механічної роботи на комп'ютери, а також через збільшення попиту на розвиток та постійне самонавчання.

Було проведено аналіз ризиків та можливостей, які можуть виникнути. Основними загрозами, очікувано, виявились конкуренція та зміна потреб користувачів. Найбільш вдалимими можливостями для нас, звичайно ж, є невдачі наших конкурентів. Також гарною можливістю для росту є загальне «підняття» ринку.

На ринку наявна нецінова конкуренція, існує декілька фірм-конкурентів, але всі вони покривають лише якусь певну частину функціональності нашої системи, тому вихід на нього буде потребувати певних зусиль та капіталовкладень. Проте проект є доволі конкурентоспроможним завдяки своїй нижчій собівартості та простоті ідеї та реалізації. Через те, що він є повністю програмним, його розробка не потребує витрат на різноманітні матеріали та обладнання, необхідні для виготовлення корпусу, схем, тощо.

Для впровадження ринкової реалізації проекту слід обрати альтернативу, яка передбачає розробку програмного продукту за допомогою генетичних алгоритмів, а потім якісну рекламу та PR, сконцентровану навколо позитивних характеристик даного програмного продукту, таких як низька ціна, точність оптимізації, кросплатформність і т.д.

З огляду на проведений аналіз, можна чітко сказати, що подальша імплементація проекту є доцільною, адже він може знайти свою цільову аудиторію та зайняти місце на ринку.

ВИСНОВКИ

В ході виконання даної роботи було досліджено предметну область складання розкладів та розроблено алгоритм формування розкладу навчального закладу. Розглянуто проблематику, актуальність даної теми і її використання в наш час.

Було розглянуто різні варіанти вирішення поставленої задачі і вибрано остаточний алгоритм, за яким буде формуватися розклад, а саме генетичний алгоритм на основі цільової функції в якості фітнес функції оцінки якості розкладу. Було розроблено саму цільову функцію, яка відображає побажання як і викладачів, так і студентів, враховуючи їх пріоритети. Також було розроблено процедуру змішування, яка зберігає адекватність розкладу. Було розроблено та реалізовано математичну модель розкладу, яка близько відображає суть фізичного розпорядку роботи навчального закладу.

Було вибрано середовище програмування Jupyter Notebook та мова Python для реалізації цього завдання. Після чого було написано готовий додаток з графічним інтерфейсом, який дозволяє адміністратору зі зручністю користуватись базою даних та адмініструвати її. База даних була організована за допомогою MS SQL Server, який зберігає в собі всі важливі для обрахунків дані. Архітектура бази даних добре відображає зв'язки між різними даними про навчальний процес і зручно зберігає інформацію в розроблених таблицях.

Було реалізовано всі розроблені методи на мові програмування Python і Pandas. Повна архітектура повністю задовольняє всім вимогам алгоритму, реалізовує всі структури та гарантує виконання поставлених задач. Також ця реалізація є оптимізованою та швидкою. Надійність гарантується методами, які перевіряються розклад на адекватність. А збіжність цільової функції гарантує сам алгоритм.

Було перевірено розроблену програму на реальних даних, а саме розроблено алгоритм, який впорався з формуванням розкладу кафедри на 6

курсів та 15 груп та 28 викладачів. Формування розкладу в середньому за 13 ітерації за 4.5 сек. Жорсткі умови були дотримані, оптимізація розкладу та приближення до виконання всіх нежорстких умов було досягнуто. Отриманий з першого експерименту, розклад не відповідає всім нежорстким вимогам, які йому задані, проте час, потрачений на його побудову, цей недолік компенсує. А той факт, що після його генерації оператор може вручну змінити невідповідності, призводить до швидкого і якісного результату.

Дана реалізація має потенціал до розширення. Кількість дисциплін, викладачів, груп, аудиторій можна розширити з масштабу одного потоку до масштабу університету. Вимоги можна формувати для різних інстанцій.

Отримані дані про цільову функцію показують, що алгоритм працює правильно, проте з одною проблемою. Вона полягає у випадковій природі даного алгоритму. Практично всі методи містять якусь генерацію псевдовипадкових чисел, що залишає можливість до зациклення чи помилок. Проте дана проблема була вирішена введенням струсу, який розбавляє генофонд та запобігає застою.

Під час експерименту було вибрано розмір генерації у 8 розкладів. При збільшенні цього числа час виконання буде рости, проте якість буде також зростати, тому при правильному виборі цього числа та конфігурації машини, на якій буде запущений даний додаток, можна досягнути оптимального часу генерації гарного розкладу, а при подальшому розширенні проекту можна зменшити кількість зусиль з боку людини та швидкість обрахунків.

Таким чином, дослідження можна вважати плідним, а отриманий розклад лише трохи, але не принципово, відрізнявся від того, що був сформований людиною за довгий час. Швидкість і зручність алгоритму є його суттєвою перевагою.

ПЕРЕЛІК ПОСИЛАНЬ

1. Снитюк В.Є. Про особливості формування цільової функції та обмежень в задачі складання розкладу занять / Снитюк В.Є., Сіпко Є.Н. // Математичні машини і системи – 2014 - №3 – С. 67-76
2. Снитюк В.Є. Аспекти формування цільової функції в задачі складання розкладу занять у вищих навчальних закладах на основі суб'єктивних переваг / Снитюк В.Є., Сіпко Є.Н. // Автоматика. Автоматизація. Електротехнічні комплекси і системи - 2013 – №2 – С.98-104
3. Бевз С. В. Розробка автоматизованої системи формування розкладу магістратури / Бевз С. В., Войтко В. В., Бурбело С. М., Шоботенко А. М. // Інформаційні технології та комп'ютерна техніка – 2009 - №4 – С. 30-65
4. Бевз С. В. Автоматизація процесу формування розкладу сесії. / Бевз С.В., Войтко В.В., Бурбело С.М., Куба Т.О., Сухонос О.О.// Принципові концепції та структурування різних рівнів освіти з оптико-електронних інформаційно-енергетичних технологій – 2009 - №4 – С. 25-36
5. Офіційний сайт компанії Microsoft – Режим доступу : https://www.microsoftstore.com/store/msusa/en_US/pdp/Visual-Studio-Professional-2015/productID.323825200 - Дата доступу 13.06.2016
6. Астахова І.Ф. Створення розкладу навчальних занять на основі генетичного алгоритму / Астахова І.Ф., Фірас А.М. // Вісник воронежского державного університету, серія: «Системний аналіз и інформаційні технології». – 2013. – № 2. – С 93-99.
7. Деканова М.В. Математична модель и алгоритм побудови розкладу навчальних занять університету / Деканова М.В. // Вісник Полоцького державного університету. Серія С. – 2013. – №12. – С. 24-33.
8. Верьовкін В.И. Автоматизоване створення розкладів навчальних занять вишу с урахуванням складності дисциплін і втомленості

- студентів / Верьовкін В.И., Ісмагілова О.М., Атавін Т.А. // Доповіді ТУСУР. – 2009. – №1 (19), частина 1. – С. 221-225.
9. Бабкіна Т.С. Задача складання розкладу: рішення на основі багатоагентного підходу / Бабкіна Т.С. // Бізнес-інформатика. – 2008. – №1. – С.23-28.
10. Мулява І. Я. Вирішення задачі автоматизованого формування розкладу навчального закладу за допомогою генетичних алгоритмів // Міжнародний науковий журнал "Інтернаука". — 2018. — №9
11. Мулява І.Я. Програмна модель формування розкладу навчальних занять // Наука онлайн: Міжнародний електронний науковий журнал — 2018. — №5
12. Управління проектами [Електронний ресурс]. – Режим доступу: http://uk.wikipedia.org/wiki/Управління_проектами. – Назва з екрану.
13. Діаграма Ганта [Електронний ресурс]. – Режим доступу: http://uk.wikipedia.org/wiki/Діаграма_Ганта. – Назва з екрану.
14. Mean absolute error – Wikipedia [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Mean_absolute_error. – Назва з екрану.
15. Fonseca, C.M.; Fleming, P.J. Genetic algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Proceedings of the Fifth International Conference on Genetic Algorithms, Urbana-Champaign, IL, USA, 17–22 July 1993; Volume 93, pp. 416–423.

ЛІСТИНГ ПРОГРАМИ

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Diplom
{
    class Subj
    {
        public int subj_code { get; set; }
        public string subj { get; set; }
        public int subj_type { get; set; }
        public int grp_code { get; set; }

        public Subj() { }
        public Subj(string subj_new, int subj_type_new, int course, int group)
        {
            subj = subj_new;
            subj_type = subj_type_new;
            subj_code = Dictionary.Conv(subj) + subj_type ;
            grp_code = course * 10 + group;
        }
    }

    class Grp
    {
        public int grp_code { get; set; }
        public int grp { get; set; }
        public int course { get; set; }
        public int count { get; set; }
        public Grp() { }
        public Grp(int grp_new, int course_new, int count_new)
        {
            grp = grp_new;
            course = course_new;
            grp_code = course * 10 + grp;
            count = count_new;
        }
    }

    class Room
    {
        public int room { get; set; }
        public int type { get; set; }
        public int cnt { get; set; }
        public Room(int new_room, int count, int new_type)
        {
            room = new_room;
            cnt = count;
            type = new_type;
        }
    }
    class Teacher
    {
```

```

public string name { get; set; }
public string subj { get; set; }
public int labs { get; set; }
public Teacher(string name_n, string subj_n, int l, int p, int lab )
{
    name = name_n;
    subj = subj_n;
    lectures = l;
    practice = p;
    labs = lab;
}
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Diplom
{
    class Rozklad
    {
        public Node[, ] schedule;
        public Rozklad(Dictionary DC)
        {
            schedule = new Node[6, 6, DC.rooms_list.Count()];
            //Викладач
            Teacher chsn_teacher;
            //День
            int x;
            //Паpa
            int y; ;
            //Аудиторія
            int z;
            foreach (Subj subj in DC.subj_list)
            {
                do
                {
                    x = MainWindow.rand.Next(0, 6);
                    y = MainWindow.rand.Next(0, 6);
                    z = findRoom(subj, DC);
                    chsn_teacher = findTeacher(subj, DC);
                }
                while ((schedule[x, y, z] != null)
                    || (checkGrp(x, y, subj.grp_code))
                    || (checkTchr(x, y, chsn_teacher)));
                Node new_node = new Node(subj, chsn_teacher);
                schedule[x, y, z] = new_node;
            }
        }
    }

    internal Node Node
    {
        get
        {
            throw new System.NotImplementedException();
        }

        set
        {
        }
    }
}

```

```

public bool checkAll(Dictionary DC)
{
    int check;
    foreach(Teacher teacher in DC.teachers_list)
    {
        for(int i = 0; i < schedule.GetLength(0); i++)
            for(int j = 0; j < schedule.GetLength(1); j++)
            {
                check = 0;
                for (int z = 0; z < schedule.GetLength(2); z++)
                {
                    if (schedule[i, j, z] == null) continue;
                    if (schedule[i, j, z].teacher == teacher)
                        check++;
                }
                if (check > 1) return false;
            }
    }
    foreach (Grp group in DC.grp_list)
    {
        for (int i = 0; i < schedule.GetLength(0); i++)
            for (int j = 0; j < schedule.GetLength(1); j++)
            {
                check = 0;
                for (int z = 0; z < schedule.GetLength(2); z++)
                {
                    if (schedule[i, j, z] == null) continue;
                    if (schedule[i, j, z].subject.grp_code == group.grp_code)
                        check++;
                }
                if (check > 1) return false;
            }
    }
    //if (!(checkRooms(DC)))
        return true; //false
    //else return true;
}

private bool checkTchr(int x, int y, Teacher chsn_teacher)
{
    for (int i = 0; i < schedule.GetLength(2); i++)
    {
        if (schedule[x, y, i] != null)
        {
            if ((schedule[x, y, i].teacher == chsn_teacher))
                return true;
        }
    }
    return false;
}

private bool checkGrp(int x, int y, int grp_code)
{
    for (int i = 0; i < schedule.GetLength(2); i++)
    {
        if (schedule[x, y, i] != null)
        {
            if ((schedule[x, y, i].subject.grp_code == grp_code))
                return true;
        }
    }
    return false;
}

```

```

public bool checkRooms(Dictionary DC)
{
    for (int i = 0; i < schedule.GetLength(0); i++)
        for (int j = 0; j < schedule.GetLength(1); j++)
            {
                for (int z = 0; z < schedule.GetLength(2); z++)
                    {
                        if (schedule[i, j, z] == null) continue;
                        if (schedule[i, j, z].subject.subj_type != DC.rooms_list[z].type)
                            return true;
                    }
            }
    return false;
}

public Teacher findTeacher(Subj subj, Dictionary DC)
{
    int[] help = new int[3];
    int choose;
    List<Teacher> subj_oriented_teachers = new List<Teacher>();
    foreach (Teacher teacher in DC.teachers_list)
        {
            if (subj.subj == teacher.subj)
                {
                    help[0] = teacher.lectures;
                    help[1] = teacher.practice;
                    help[2] = teacher.labs;
                    if (help[subj.subj_type - 1] == 1)
                        {
                            subj_oriented_teachers.Add(teacher);
                        }
                }
        }
    choose = MainWindow.rand.Next(0, subj_oriented_teachers.Count());
    return subj_oriented_teachers[choose];
}

public int findRoom(Subj subj, Dictionary DC)
{
    int room_no;
    do
        {
            room_no = MainWindow.rand.Next(0, DC.rooms_list.Count());
        }
    while ((DC.rooms_list[room_no].cnt < DC.findGrp_count(subj.grp_code)) || (DC.rooms_list[room_no].type !=
subj.subj_type));
    return room_no;
}

}

}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using Excel = Microsoft.Office.Interop.Excel;

namespace Diplom
{
    class Generation
    {

```

```

int count;
public List<RF> main_gen = new List<RF>();
public List<RF> gen_list = new List<RF>();
public List<RF> new_gen_list = new List<RF>();

internal RF RF
{
    get
    {
        throw new System.NotImplementedException();
    }

    set
    {
    }
}

public void generate(Dictionary DC, int cnt)
{
    count = cnt;
    Rozklad new_roz;
    for (int i = 0; i < count; i++)
    {
        new_roz = new Rozklad(DC);
        gen_list.Add(new RF(new_roz, targetFunction(new_roz, DC)));
    }
    /*for (int i = 0; i < count; i++)
    {
        gen_list[i].value = targetFunction(gen_list[i].rozklad, DC);
    }*/
}

public double targetFunction(Rozklad schedule, Dictionary DC)
{
    Dictionary<int, int[]> dict = new Dictionary<int, int[]>();
    Dictionary<int, int> count_d = new Dictionary<int, int>();
    double res_1 = 0;
    foreach (Grp grp in DC.grp_list)
    {
        count_d.Add(grp.grp_code, 0);
    }
    foreach (Grp grp in DC.grp_list)
    {
        dict.Add(grp.grp_code, new int[6] { 0, 0, 0, 0, 0, 0 });
    }
    for (int i = 0; i < schedule.schedule.GetLength(0); i++)
        for (int j = 0; j < schedule.schedule.GetLength(1); j++)
            for (int k = 0; k < schedule.schedule.GetLength(2); k++)
                {
                    if (schedule.schedule[i, j, k] != null)
                    {
                        dict[schedule.schedule[i, j, k].subject.grp_code][i]++;
                    }
                }
    foreach (Grp grp in DC.grp_list)
        for (int j = 0; j < dict[grp.grp_code].GetLength(0); j++)
            if (dict[grp.grp_code][j] == 0)
                {
                    count_d[grp.grp_code]++;
                }
    foreach (Grp grp in DC.grp_list)
    {
        res_1 += count_d[grp.grp_code];
    }
    res_1 = 1 / (res_1 + 1);
}

```

```

int saturday = 0;
for (int j = 0; j < schedule.schedule.GetLength(1); j++)
    for (int k = 0; k < schedule.schedule.GetLength(2); k++)
        {
            if (schedule.schedule[5, j, k] != null)
                {
                    saturday++;
                }
        }
double res_2 = 0;
if (saturday == 0)
    res_2 = 1;
return (res_1 + res_2);
}

public Rozklad[] mixGen(Rozklad r1_old, Rozklad r2_old)
{
    Rozklad r1 = r1_old;
    Rozklad r2 = r2_old;
    int x1, y1, z1;
    int x2, y2, z2;
    int counter = 0;
    Node node1;
    do
    {
        x1 = MainWindow.rand.Next(0, 6);
        y1 = MainWindow.rand.Next(0, 6);
        z1 = MainWindow.rand.Next(0, r1.schedule.GetLength(2));
        if (counter != 50000)
            counter++;
        else return null;
    } while (r1.schedule[x1, y1, z1] == null);
    node1 = r1.schedule[x1, y1, z1];

    counter = 0;
    do
    {
        x2 = MainWindow.rand.Next(0, 6);
        y2 = MainWindow.rand.Next(0, 6);
        z2 = MainWindow.rand.Next(0, r1.schedule.GetLength(2));
        if (counter != 50000)
            counter++;
        else return null;
    } while (!(node1.Compare(r2.schedule[x2, y2, z2])));

    Node buff = null;
    buff = r1.schedule[x1, y1, z1];
    r1.schedule[x1, y1, z1] = r1.schedule[x2, y2, z2];
    r1.schedule[x2, y2, z2] = buff;
    r2.schedule[x1, y1, z1] = r2.schedule[x2, y2, z2];
    r2.schedule[x2, y2, z2] = buff;
    Rozklad[] res = new Rozklad[2];
    res[0] = r1;
    res[1] = r2;
    return res;
}

public void newGen(Dictionary DC)
{
    new_gen_list.Clear();
    var TF = new double[4];
    for (int i = 0; i < 4; i++)
    {
        TF[i] = targetFunction(gen_list[i].rozklad, DC);
    }
}

```



```

}
Rozklad[] mix = new Rozklad[2];
int first_par, second_par;
int steps = 0;
//
//var check = new bool[2];
//
while (new_gen_list.Count() < gen_list.Count())
{
    first_par = choosePar(TF);
    second_par = choosePar(TF);
    do
    {
        mix = mixGen(gen_list[first_par].rozklad, gen_list[second_par].rozklad);
        steps++;
        if (steps == 5000)
        {
            generate(DC, count);
            shakeGen(DC);
            MessageBox.Show("PZDC");
        }
        if (steps == 100000)
        {
            MessageBox.Show("PZDC");
            break;
        }
    } while (mix == null);

    for (int i = 0; i < 2; i++)
    {
        if (new_gen_list.Count() == gen_list.Count())
            break;
        if (mix[i].checkAll(DC))
            new_gen_list.Add(new RF(mix[i], targetFunction(mix[i], DC)));
    }

    steps++;
    if (steps == 5000)
    {
        generate(DC, count);
        shakeGen(DC);
        MessageBox.Show("PZDC");
    }
    if (steps == 100000)
    {
        MessageBox.Show("PZDC");
        break;
    }
    /*Стерти
    for (int i = 0; i < 2; i++)
    {
        check[i] = mix[i].checkAll(DC);
    }
    */
}
}

public void getBest(Dictionary DC)
{
    main_gen.Clear();
    foreach(RF rf in gen_list)
    {
        main_gen.Add(new RF(rf.rozklad, targetFunction(rf.rozklad, DC)));
    }
}

```

```

foreach(RF rf in new_gen_list)
{
    main_gen.Add(new RF(rf.rozklad, targetFunction(rf.rozklad, DC)));
}
main_gen = main_gen.OrderByDescending(RF => RF.value).ToList();
for(int i = 0; i < main_gen.Count()/2; i++)
{
    gen_list[i].rozklad = main_gen[i].rozklad;
    gen_list[i].value = main_gen[i].value;
}
}

public void shakeGen(Dictionary DC)
{
    new_gen_list.Clear();
    Rozklad new_roz;
    for (int i = 0; i < count; i++)
    {
        new_roz = new Rozklad(DC);
        new_gen_list.Add(new RF(new_roz, targetFunction(new_roz, DC)));
    }
}

public double getDelta(Dictionary DC)
{
    var TF = new double[gen_list.Count()];
    for(int i = 0; i < gen_list.Count(); i++)
    {
        TF[i] = gen_list[i].value;
    }
    return TF.Max() - TF.Min();
}

public int choosePar(double[] TF)
{
    normalize(ref TF);
    change(ref TF);
    TF[TF.GetLength(0) - 1] = 1;
    int rand_num = MainWindow.rand.Next(0, 101);
    for (int i = 0; i < 4; i++)
    {
        if (((double)rand_num / 100) <= TF[i])
            return i;
    }
    return -1;
}

public void change(ref double[] mass)
{
    for (int i = 0; i < mass.GetLength(0) - 1; i++)
    {
        mass[i + 1] += mass[i];
    }
}

public void normalize(ref double[] mass)
{
    double sum = mass.Sum();
    for (int i = 0; i < mass.GetLength(0); i++)
        mass[i] /= sum;
}

public void outputRozklad(Rozklad roz, Dictionary DC)
{

```

```

string file = @"C:\Users\igorm_000\Documents\Visual Studio
2015\Projects\Diplom\Diplom\bin\Debug\Rozklad.xlsx";
List<string> type_list = new List<string>();
type_list.Add("лекція");
type_list.Add("практика");
type_list.Add("лабораторна");
Excel.Application MyApp = new Excel.Application();
//MyApp.Visible = false;
Excel.Workbook MyBook = MyApp.Workbooks.Open(file);
Excel.Worksheet MySheet = (Excel.Worksheet)MyBook.Sheets[1];
MySheet.Cells.ClearContents();
MySheet.Columns.AutoFit();
foreach(Grp grp in DC.grp_list)
{
    MySheet.Cells[1, 1 + grp.grp] = "Група " + (grp.grp_code);
}
for(int i = 0; i < roz.schedule.GetLength(0); i++)
for (int j = 0; j < roz.schedule.GetLength(1); j++)
{
    switch (i)
    {
        case 0:
            MySheet.Cells[2 + 6 * i + j, 1] = "Понеділок " + (j + 1) + " пара";
            break;
        case 1:
            MySheet.Cells[2 + 6 * i + j, 1] = "Вівторок " + (j + 1) + " пара";
            break;
        case 2:
            MySheet.Cells[2 + 6 * i + j, 1] = "Середа " + (j + 1) + " пара";
            break;
        case 3:
            MySheet.Cells[2 + 6 * i + j, 1] = "Четвер " + (j + 1) + " пара";
            break;
        case 4:
            MySheet.Cells[2 + 6 * i + j, 1] = "П'ятниця " + (j + 1) + " пара";
            break;
        case 5:
            MySheet.Cells[2 + 6 * i + j, 1] = "Субота " + (j + 1) + " пара";
            break;
    }
}
for (int i = 0; i < roz.schedule.GetLength(0); i++)
for (int j = 0; j < roz.schedule.GetLength(1); j++)
for (int k = 0; k < roz.schedule.GetLength(2); k++)
{
    if (roz.schedule[i, j, k] != null)
    {
        if (roz.schedule[i, j, k].subject.grp_code == 41)
            MySheet.Cells[2 + 6 * i + j, 1 + 1] = roz.schedule[i, j, k].subject.subj + " " +
type_list[roz.schedule[i, j, k].subject.subj_type - 1] + "\n" + roz.schedule[i, j, k].teacher.name + "\n" +
DC.rooms_list[k].room;
        if (roz.schedule[i, j, k].subject.grp_code == 42)
            MySheet.Cells[2 + 6 * i + j, 1 + 2] = roz.schedule[i, j, k].subject.subj + " " +
type_list[roz.schedule[i, j, k].subject.subj_type - 1] + "\n" + roz.schedule[i, j, k].teacher.name + "\n" +
DC.rooms_list[k].room;
        if (roz.schedule[i, j, k].subject.grp_code == 2)
            MySheet.Cells[2 + 6 * i + j, 1 + 3] = roz.schedule[i, j, k].subject.subj + " " +
type_list[roz.schedule[i, j, k].subject.subj_type - 1] + "\n" + roz.schedule[i, j, k].teacher.name + "\n" +
DC.rooms_list[k].room;
    }

    //MySheet.Cells[2 + 6 * i + j, 1 + roz.schedule[i, j, k].subject.grp_code - 10] = roz.schedule[i, j,
k].subject.subj + " " + type_list[roz.schedule[i, j, k].subject.subj_type - 1] + "\n" + roz.schedule[i, j, k].teacher.name +
"\n" + DC.rooms_list[k].room;
}

```

```

        }
        MySheet.Columns.AutoFit();
        MySheet.Rows.AutoFit();
        MyBook.Save();
        MyBook.Close();
        MyApp.Quit();
    }
}
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Diplom
{
    class Dictionary
    {
        DataBase DB = new DataBase();
        public List<Subj> subj_list { get; set; }
        public List<Room> rooms_list { get; set; }
        public List<Grp> grp_list { get; set; }
        public List<Teacher> teachers_list { get; set; }

        internal Room Room
        {
            get
            {
                throw new System.NotImplementedException();
            }

            set
            {
            }
        }

        internal Grp Grp
        {
            get
            {
                throw new System.NotImplementedException();
            }

            set
            {
            }
        }

        internal Teacher Teacher
        {
            get
            {
                throw new System.NotImplementedException();
            }

            set
            {
            }
        }

        internal Subj Subj
        {

```

```

    get
    {
        throw new System.NotImplementedException();
    }

    set
    {
    }
}

public Dictionary()
{
    fillSubj();
    fillRooms();
    fillGrps();
    fillTeachers();
}

static public int Conv(string s)
{
    int code = 0;
    for (int i = 0; i < s.Length; i++)
    {
        code += (int)s[i];
    }
    return code;
}

public void fillSubj()
{
    subj_list = new List<Subj>();
    int i = int.Parse(DB.Ex_Select_Comm("SELECT count(*) FROM Discipline;").Rows[0][0].ToString());
    int lectures;
    int practice;
    int labs;
    while (i!=0)
    {
        lectures = int.Parse(DB.Ex_Select_Comm("SELECT lectures FROM Discipline WHERE Id_disc = " + i +
        "";").Rows[0][0].ToString());
        practice = int.Parse(DB.Ex_Select_Comm("SELECT practice FROM Discipline WHERE Id_disc = " + i +
        "";").Rows[0][0].ToString());
        labs = int.Parse(DB.Ex_Select_Comm("SELECT labs FROM Discipline WHERE Id_disc = " + i +
        "";").Rows[0][0].ToString());
        int iter = lectures + practice + labs;
        while (iter != 0)
        {
            for (int j = 0; j < lectures; j++)
            {
                Subj new_subj = new Subj(DB.Ex_Select_Comm("SELECT disc_name FROM Discipline WHERE
                Id_disc = " + i + "";").Rows[0][0].ToString(),
                1,
                int.Parse(DB.Ex_Select_Comm("SELECT course FROM Discipline WHERE Id_disc = " + i
                + "";").Rows[0][0].ToString()),
                int.Parse(DB.Ex_Select_Comm("SELECT group_no FROM Discipline WHERE Id_disc = "
                + i + "";").Rows[0][0].ToString())
                );
                subj_list.Add(new_subj);
                iter--;
            }

            for (int j = 0; j < practice; j++)
            {
                Subj new_subj = new Subj(DB.Ex_Select_Comm("SELECT disc_name FROM Discipline WHERE
                Id_disc = " + i + "";").Rows[0][0].ToString(),

```

```

                2,
                int.Parse(DB.Ex_Select_Comm("SELECT course FROM Discipline WHERE Id_disc = " + i
+ ";" ).Rows[0][0].ToString()),
                int.Parse(DB.Ex_Select_Comm("SELECT group_no FROM Discipline WHERE Id_disc = "
+ i + ";" ).Rows[0][0].ToString())
            );
            subj_list.Add(new_subj);
            iter--;
        }

        for (int j = 0; j < labs; j++)
        {
            Subj new_subj = new Subj(DB.Ex_Select_Comm("SELECT disc_name FROM Discipline WHERE
Id_disc = " + i + ";" ).Rows[0][0].ToString()),
                3,
                int.Parse(DB.Ex_Select_Comm("SELECT course FROM Discipline WHERE Id_disc = " + i
+ ";" ).Rows[0][0].ToString()),
                int.Parse(DB.Ex_Select_Comm("SELECT group_no FROM Discipline WHERE Id_disc = "
+ i + ";" ).Rows[0][0].ToString())
            );
            subj_list.Add(new_subj);
            iter--;
        }
    }
    i--;
}

public void fillRooms()
{
    rooms_list = new List<Room>();
    int i = int.Parse(DB.Ex_Select_Comm("SELECT count(*) FROM Rooms;").Rows[0][0].ToString());
    while (i != 0)
    {
        Room new_room = new Room(int.Parse(DB.Ex_Select_Comm("SELECT Id_room FROM
Rooms;").Rows[i-1][0].ToString()),
            int.Parse(DB.Ex_Select_Comm("SELECT places FROM Rooms;").Rows[i-1][0].ToString()),
            int.Parse(DB.Ex_Select_Comm("SELECT type FROM Rooms;").Rows[i-1][0].ToString())
        );
        rooms_list.Add(new_room);
        i--;
    }
}

public void fillGrps()
{
    grp_list = new List<Grp>();
    int i = int.Parse(DB.Ex_Select_Comm("SELECT count(*) FROM Groups;").Rows[0][0].ToString());
    while (i != 0)
    {
        Grp new_group = new Grp(int.Parse(DB.Ex_Select_Comm("SELECT group_no FROM Groups;").Rows[i -
1][0].ToString()),
            int.Parse(DB.Ex_Select_Comm("SELECT course FROM Groups;").Rows[i - 1][0].ToString()),
            int.Parse(DB.Ex_Select_Comm("SELECT cnt FROM Groups;").Rows[i - 1][0].ToString())
        );
        grp_list.Add(new_group);
        i--;
    }
}

public void fillTeachers()
{

```

```

teachers_list = new List<Teacher>();
int i = int.Parse(DB.Ex_Select_Comm("SELECT count(*) FROM Teacher;").Rows[0][0].ToString());
while (i != 0)
{
    Teacher new_teacher = new Teacher(DB.Ex_Select_Comm("SELECT pib FROM Teacher;").Rows[i -
1][0].ToString(),
        DB.Ex_Select_Comm("SELECT disc_name FROM Teacher;").Rows[i - 1][0].ToString(),
        int.Parse(DB.Ex_Select_Comm("SELECT lectures FROM Teacher;").Rows[i - 1][0].ToString()),
        int.Parse(DB.Ex_Select_Comm("SELECT practice FROM Teacher;").Rows[i - 1][0].ToString()),
        int.Parse(DB.Ex_Select_Comm("SELECT labs FROM Teacher;").Rows[i - 1][0].ToString())
    );
    teachers_list.Add(new_teacher);
    i--;
}
}
public int findMaxRoom()
{
    int maxvalue = 0;
    foreach(Room room in rooms_list)
    {
        if (room.room > maxvalue) maxvalue = room.room;
    }
    return maxvalue;
}
public int findRoom_count(int room)
{
    for (int i = 0; i < rooms_list.Count(); i++)
    {
        if (rooms_list[i].room == room)
            return rooms_list[i].cnt;
    }
    return 0;
}
public int findGrp_count(int grp_code)
{
    for (int i = 0; i < grp_list.Count(); i++)
    {
        if (grp_list[i].grp_code == grp_code)
            return grp_list[i].count;
    }
    return 0;
}
}
}
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Diplom
{
    public class DataBase
    {
        int flag=0;
        SqlConnection Conn;
        SqlConnection LoginConn;
        SqlDataAdapter MyAdapt;
        DataSet Ds;
        public DataBase()
    }
}

```

```

{
    Conn = new SqlConnection(Properties.Settings.Default.DB1ConnectionString);
}
public DataSet GetFullTables()
{
    Conn.Open();
    SqlCommand[] SqlCommands = new SqlCommand[2];
    SqlCommands[0] = new SqlCommand("Select * From Discipline", Conn);
    MyAdapt = new SqlDataAdapter(SqlCommands[0]);
    System.Data.DataSet dat_set = new System.Data.DataSet();
    MyAdapt.Fill(dat_set, "V_main");
    Conn.Close();
    return dat_set;
}
public DataTable Ex_Select_Comm(string cmd)
{
    Conn.Open();
    DataTable dt = new DataTable();
    SqlCommand SqlCommands = new SqlCommand(cmd, Conn);
    dt.Load(SqlCommands.ExecuteReader());
    Conn.Close();
    return dt;
}
public DataTable Ex_Select_LoginComm(string cmd)
{
    LoginConn.Open();
    DataTable dt = new DataTable();
    SqlCommand SqlCommands = new SqlCommand(cmd, LoginConn);
    dt.Load(SqlCommands.ExecuteReader());
    LoginConn.Close();
    return dt;
}
public void InsertComm(string Comm)
{
    SqlCommand command = new SqlCommand(Comm, Conn);
    Conn.Open();
    command.ExecuteNonQuery();
    Conn.Close();
}
public void LoginComm(string Comm)
{
    SqlCommand command = new SqlCommand(Comm, LoginConn);
    LoginConn.Open();
    command.ExecuteNonQuery();
    LoginConn.Close();
}

public int Access(string login, string password)
{
    LoginConn.Open();
    SqlCommand Comm = new SqlCommand("SELECT * FROM Admin WHERE Login = '" + login + "' AND
Password = '" + password + "';", LoginConn);
    MyAdapt = new SqlDataAdapter(Comm);
    DataSet dat_set = new DataSet();
    MyAdapt.Fill(dat_set, "Admin");
    if (dat_set.Tables[0].Rows.Count == 0)
    {
        Comm = new SqlCommand("SELECT * FROM Users WHERE Login = '" + login + "' AND Password = '" +
password + "';", LoginConn);
        MyAdapt = new SqlDataAdapter(Comm);
        MyAdapt.Fill(dat_set, "Users");
        if (dat_set.Tables[1].Rows.Count == 0)
        {

```



```
        LoginConn.Close();
        return 2;
    }
    else
    {
        LoginConn.Close();
        return 1;
    }
}

else
{
    LoginConn.Close();
    return 0;
}
}
}
```