МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЧЕРНІГІВСЬКА ПОЛІТЕХНІКА»

Навчально-науковий інститут електронних та інформаційних технологій
Кафедра інформаційних та комп'ютерних систем

Допущено до захисту
Завідувач кафедри
к.е.н., доцент Базилевич В.М.

«___» _____ 2020 р.

ВИПУСКНА КВАЛІФІКЦІЙНА РОБОТА
за освітньо-професійною програмою бакалавра

РОЗРОБКА БРАУЗЕРНОГО IDE ДЛЯ СВОРЕННЯ AMP-ДОДАТКІВ

Спеціальність 123 – Комп'ютерна інженерія
Галузь знань 12 – Інформаційні технології

Виконавець:
студент гр. КІ–162
Страмко Сергій Михайлович
(підпис)

Керівник:
ст.викладач
(посада) *(науковий ступінь, вчене звання)*
Бивойно Тарас Павлович
(підпис)

Чернігів 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЧЕРНІГІВСЬКА ПОЛІТЕХНІКА»

Навчально-науковий інститут електронних та інформаційних технологій
Кафедра інформаційних та комп'ютерних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
к.е.н., доцент Базилевич В.М.

"____"_____2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ
ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ БАКАЛАВРА

Страмка Сергія Михайловича

Тема роботи: СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СКЛАДАННЯ РОЗКЛАДУ УНІВЕРСИТЕТУ

Тему затверджено наказом ректора
від " " 2020 р. №____с

1. **Вхідні дані до роботи:**

1. Дані про аудиторії, викладачів, студентів.
2. Дані про навчальний план.
3. Вимоги студентів до розкладу,
4. Вимоги викладачів до розкладу.

2. **Зміст розрахунково-пояснювальної записки:**

1. Розглянути можливі варіанти реалізації генерації розкладу та обрати варіант для розробки.

2. Розробити алгоритм роботи програми генерації розкладу на його оцінки на основі переваг.
3. Розробити цільову функцію для оцінки розкладу.
4. Розробити план тестування розкладу.
5. Розробити програмну модель та протестувати її.
6. Розробити опис програми на мові C#.
7. Розробити тестовий додаток та виконати тестування його на реальних даних.

3. Демонстраційні матеріали:

7 слайдів для презентації роботи

4. Календарний план

№	Назва етапів роботи	Термін виконання	Примітки
1	Визначитися з тематикою роботи	10.02.2020	
2	Знайомство з проблемою, інформаційними джерелами, аналіз існуючих рішень	06.03.2020	
3	Аналіз задачі створення системи	01.04.2020	Звіт
4	Розробка бази даних та задач обробки даних	16.04.2020	
5	Розробка front-end проекту	18.05.2020	
6	Залік з переддипломної практики	18.05.2020	Звіт
7	Реалізація системи	01.06.2019	
8	Підготовка текстової частини і слайдів	01.06.2020	Звіт
9	Попередній захист дипломної роботи	01.06.2020	
10	Завершення оформлення, рецензування, перевірка на плагіат	15.06.20	
11	Захист дипломної роботи	з 15.06.20	

Завдання підготував:
керівник Бивойно Тарас Павлович

(підпис)

«___» _____2020 р.

Завдання одержав:
студент Страмко Сергій Михайлович

(підпис)

«___» _____2020 р.

АНОТАЦІЯ

Дана дипломна робота присвячена розробці програмного продукту для формування розкладу навчальних занять з використанням суб'єктивних переваг. Метою роботи є розробка програмного модуля, що базується на генетичних алгоритмах.

Запропоновано використання цільових функцій при визначенні якості розкладу. Розроблено метод генерації потенційних розкладів на основі використання генетичного алгоритму. Визначено структурні особливості та конструктивні елементи цільової функції.

Загальний обсяг роботи: 71 сторінок, 30 рисунків, 3 таблиць, 9 посилань, 1 додаток, на 15 сторінок.

Ключові слова: розклад занять, цільова функція, еволюційний алгоритм, генетичний метод оптимізації, урахування суб'єктивних вимог.

ANOTATYION

This project is devoted to developing software to form the schedule of training sessions with the use of subjective preferences. The aim is to develop a software module based on genetic algorithms.

The use of target functions in determining the quality of the schedule. The method of generating the potential schedules based on genetic algorithm. The structural features and structural elements of the objective function.

Total volume of work: 71 pages, 30 figures, 3 tables, 9 references, 1 extension in 15 pages.

Keywords: schedule, target function, evolutionary algorithm, genetic optimization method, taking into account the subjective requirements.

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

Дебагінг - процес вирішення помилок програми, при якому кожен крок програми у проблемному місці розглядається окремо.

MS – Microsoft – американська компанія розробки програмного забезпечення та комп'ютерних приладів.

MS SQL Server – Microsoft Simple Query Language Server – програма, яка реалізує обмін інформацією з базо даних через просто запити.

WPF - Windows Presentation Foundation – графічна підсистема .NET Framework 4.5, яка реалізує засоби інтерфейсу.

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

C# - об'єктно-орієнтована мова програмування, яка в основі має C/C++.

MS VS 2015 – Microsoft Visual Studio 2015 – середовище програмування для багатьох мов від компанії Microsoft.

ID – Identification – певне значення, яке однозначно ідентифікує певний запис в таблиці.

ЗМІСТ

ВСТУП

Складання розкладу є важливим завданням для університету. Задача складання розкладу розв’язується у багатьох галузях. Насамперед, при плануванні дискретного виробництва, організації пасажирських та 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.

Звісно ж, у кожному навчальному є людина, яка відповідає за організацію навчального процесу, але дана робота є складною для однієї особи. Велика кількість інформації, яку необхідно зібрати, багато різних умов, яких треба досягти та відповідальність, яка лежить на відповідальній людині є занадто тяжкими факторами для однієї особи, яка виконує це завдання паралельно з основними обов’язками. А враховуючи велику кількість аудиторій, предметів, викладачів, це завдання стає складнішим з розширенням навчального закладу. Це завдання є складним і відповідальним для людини, але може бути виконане комп’ютером набагато простіше і швидше.

Розклад сам по собі залежить від багатьох факторів. Їх можна розділити на об’єктивні(жорсткі) та суб’єктивні(непостійні) параметри. Об’єктивні – це база даних університету, в якій зберігається інформація про аудиторії та предмети. Суб’єктивні – це побажання студентів та викладачів.

Якщо ретельно оцінити проблематику, можна помітити, що завдання складання розкладу є складним для людини, бо треба тримати в голові всі аудиторії, при тому, що є різні види аудиторій, всі побажання студентів та викладачів, що означає необхідність цю інформацію і особистій формі зібрати, та необхідність з’єднати всі ці дані в розклад так, щоб всі були задоволені, але, як відомо, це майже ніколи не можливо досягти.

З іншого боку комп’ютер з цим завданням може легко впоратись. Зберігати інформацію про аудиторії в базі даних, застосувати певну функцію до бази, яка побудує розклад та оцінити даний розклад відносно побажань людей – проста

задача для машини. Задача складання розкладів викликає значний інтерес серед науковців, які працюють в університетах. Оскільки вони особисто зацікавлені в її розв’язанні та їх кількість є порівняно значною, то розроблена потужна множина відповідних моделей і методів.

Оптимальність у даному контексті визначає ефективність виробничих 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, якщо для промислових задач забезпечення оптимальності розкладу робіт відігравало 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 розклад поняттям неформалізованим, залежним від того, який критерій буде покладений в його основу.

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

1. АНАЛІЗ МЕТОДІВ І РІШЕННЯ ЗАДАЧІ ФОРМУВАННЯ РОЗКЛАДІВ

1.1 Що таке розклад

Розклад – це саме по собі поняття тривіальне з точки зору сучасного життя, а от задачу його формування тяжко назвати тривіальною. 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.

Формування розкладу я складною та клопіткою задачею тому, що вимагає від відповідальної людини врахування багатьох факторів: жорстких вимог та нежорстких. В даній роботі ми будемо зосереджувати свою увагу саме на розкладі занять навчального закладу, для якого жорсткими умовами будуть:

- предметні години, які треба відпрацювати,
- викладачі, які ці предмети ведуть,
- студенти,
- аудиторії, де ці заняття будуть проходити.

Також не таким і очевидним ресурсом буде час, який буде розділений між робочими днями та парами.

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.

Давайте розглянемо сам навчальний процес, щоб повністю оцінити та зрозуміти вимоги до розкладу.

1.1.1 Предмети

Предмет – це певна наукова дисципліна, яку проводить певний викладач певній групі студентів. Для спрощення завдання будемо вважати, що у однієї групи одну дисципліну може вести один викладач, тобто немає спільних для декількох груп пар.

Предмети мають декілька форм занять. До класичних треба віднести лекції, практики, лабораторні, семінари, самостійні заняття, екзамени, факультативи. У даній роботі ми обмежимось лекціями, практичними заняттями та лабораторними заняттями, бо вони є найважливішими та найчастішими.

Кожна форма заняття повинна у часі займати одну академічну годину, або пару. Ми опустимо варіанти, коли можу бути пів-пара, або більше.

1.1.2 Викладачі

Викладачі – це співробітники навчального закладу, які проводять заняття і є рушійною силою навчального процесу. Кожен викладач має свою посаду, наукове звання та ступінь, які прямо відповідають їх важливості та досягненням на кафедрі. Зрозуміло, чим вища посада, тим більш пріоритетні є вимоги даного викладача.

Викладач – жива людина, тому процес повинен бути строгий при формуванні розкладу, бо один викладач, не можу мати пару в двох групах в один і той самий час. А також один викладач не може бути в двох аудиторіях одночасно.

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 грошовий оклад, для спрощення обрахунків.

Також треба відвітити, що не всі форми занять можуть проводити всі викладачі. Лекції можуть вести тільки лектори-доценти, практики та лабораторні можуть вести аспіранти, чи нижчі за званням особи. Ці вимоги регулюються через навчальний план та за рішенням навчальної ради і повинні бути враховані як

жорсткі.

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, бо студенти – теж люди, проте є одна відмінність, яка зумовлена тим, що їх багато. А саме те, що якщо аудиторія мала, то вся група в неї не поміститься, тому це треба обов’язково враховувати.

Для полегшення задачі будемо не будемо враховувати те, що групи можуть поділятись на пів-групи на певних дисциплінах.

Щодо формування вимог все простіше, ніж з викладачами. Вимоги кожної групи будуть враховуватись залежно від голосування студентів. Тому кожні 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.

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.

1.1.4 Аудиторії

Про аудиторії частково уже було розказано раніше в розділі, проте треба уточнити багато деталей.

Кожна аудиторія – це фізична кімната в приміщенні навчального закладу, обладнана спеціальним чином для проведення занять. Кожна з кімнат облаштована під конкретний вид занять: лекційні аудиторії - для лекцій, лабораторії – для лабораторних.

Також 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 всіх навчальних закладів. І всі вони намагались вирішити це завдання створення розкладу автоматично, або хоча б автоматизовано.

Одною з таких систем є система розроблена Вінницьким університетом для формування розкладу магістрів та сесії описана в джерелах [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.

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 і 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.

Цільова функція, яка оцінює розклад в даній моделі має вигляд як на

формулі 1.1:

$$\delta = \max\left(\sum_{k=1}^{m_t} w_k \sum_{j=1}^{m_x} \frac{v_{kj}\pi_{kj}}{\sum_{i=1}^{m_x}\pi_{ki}} + \sum_{j=1}^{m_h} n_k\, h_k\right) \quad (1.1)$$

Де v_{kj} - індикаторна функція, яка визначає чи врахована вимога викладача для певної дисципліни, n_{kj} – пріоритет цієї вимоги, w_k – ваговий коефіцієнт рейтингу викладача, n_{kj}– пріоритети комплексу умов викладача.

Сам алгоритм розроблений був за таким принципом:

1. Набір інформації про дисципліни.
2. Набір та встановлення взаємозв’язків між викладачами і уроками.
3. Набір та встановлення взаємозв’язків між групами студентів і уроками.
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 1.1.

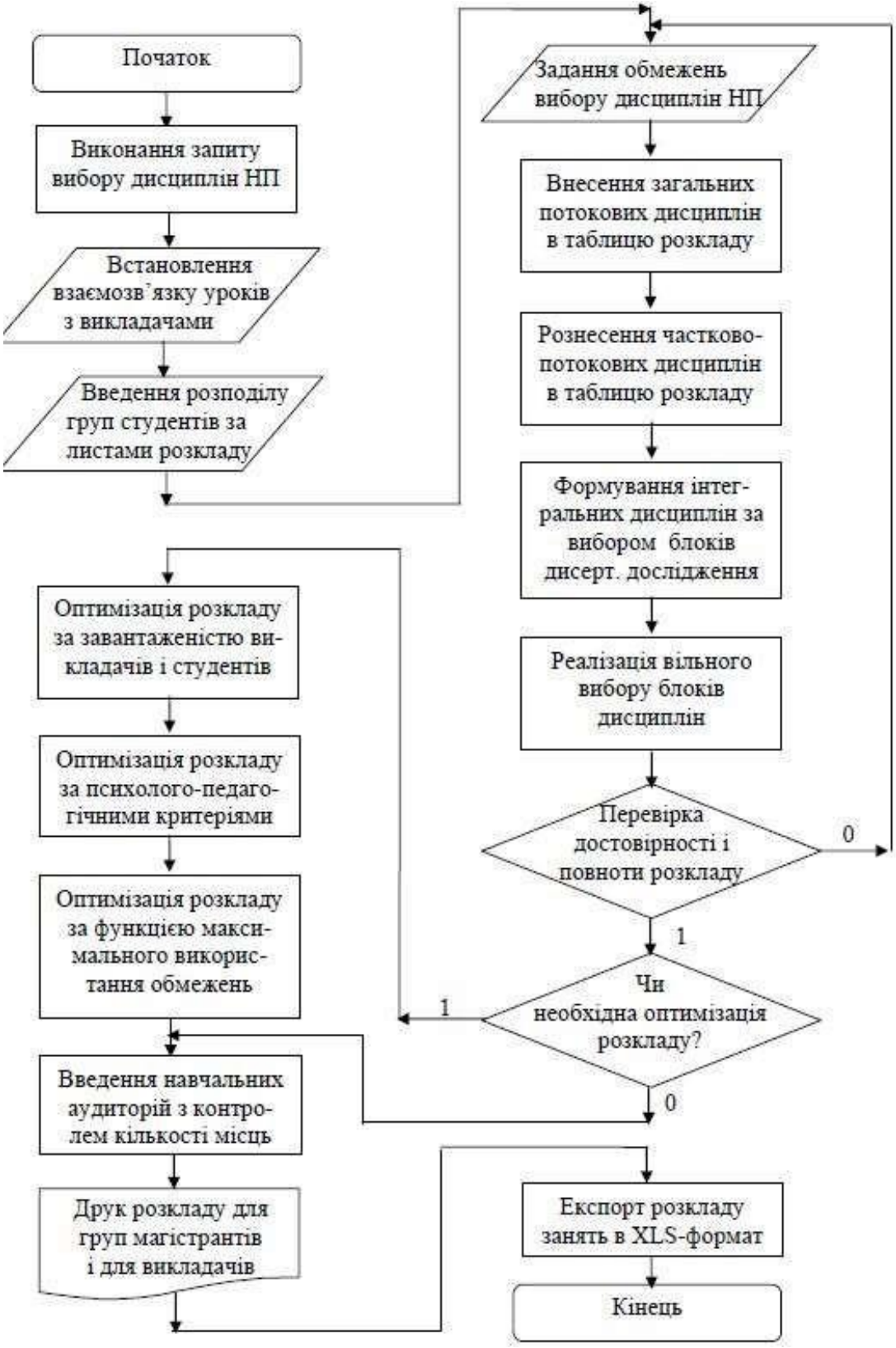


Рисунок 1.1 - Алгоритм реалізації методу формування розкладу магістратури.

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

1.3 Висновки

Оцінивши минулу інформацію та відомі методи реалізації поставленої задачі були визначені такі вимоги до формування розкладу:

- Розклад повинен відображати навчальний процес і бути адекватним.
- Адекватність гарантується виконанням жорстких умов.
- Розклад повинен відображати побажання студентів та викладачів.
- Пріоритети побажання різних викладачів повинні залежати від посади викладача на кафедрі.
- 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 є розклад. Його адекватність буде гарантуватись перевірками та самим алгоритмом. Після того як розклад буде сформований та оцінений, його треба оптимізувати певним методом та оцінити знову. Оптимізувати на оцінювати треба до тих пір, поки не буде досягнуто потрібної точності.

2 РОЗРОБКА МЕТОДІВ 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.

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. Повний опис алгоритму можна побачити на рис 2.1.

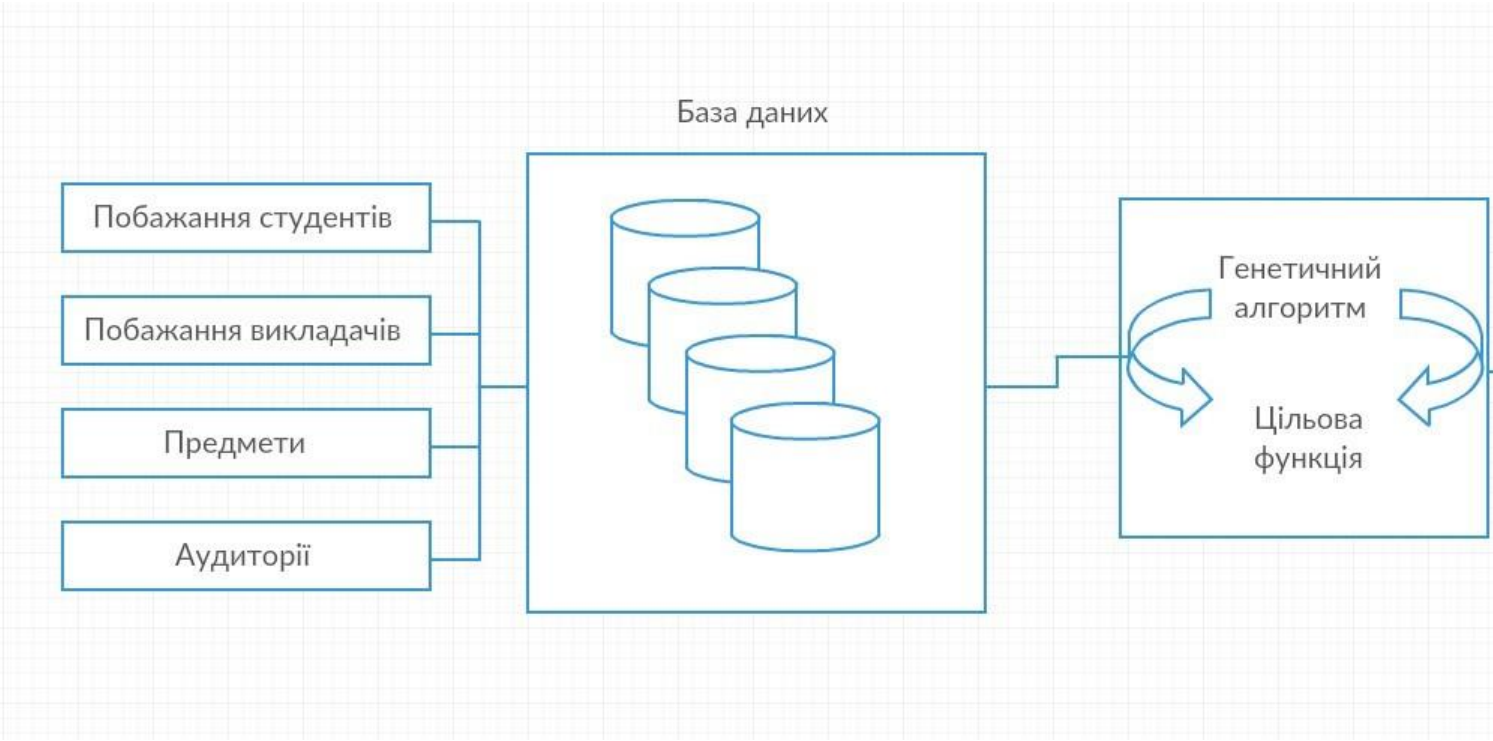


Рисунок 2.1 – Архітектура програми.

2.2 Розробка цільової функції

Формуючи цільову функцію треба врахувати багато факторів, які визначають на оцінюють розклад як ефективний, валідний 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 і 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 зросте.

З огляду на ці вимоги можна сформувати формулу (2.1), яка буде діяти на множині (2.2). Ця функція є дискретною з великою кількістю розривів. Залежить вона на пряму від виконання вимог розкладом. Гарантуються це завдяки двом індикаторним функціям.

$$F(r) = a_s \sum_{j=1}^{l} x_j \, X\{Z_j^v\} + a_L \sum_{j=1}^{K} y_i \sum_{i=1}^{M} X\{L_i \in T_i\} \sum_{j=1}^{K} d_{il}^{j} X\{Z_{il}^{T_i}\} \to max \quad (2.1)$$

$$r \in Ω(P, S, L, A) \quad (2.2)$$

Де r- розклад, a_s, a_L - вагові коефіцієнти, що вказують на пріоритети викладачів і студентів, як суб'єктів навчального процесу, x_j, y_i- пріоритети вимог студентів і викладачів, Z_j^v-$_{\mathbf{j}}$ вимоги груп студентів, L_i - викладачі, T_i –групи викладачів, $Z_{il}^{T_i}$- переваги викладачів, d_{il}^{j}- пріоритети таких побажань, l- кількість вимог студентів, K- кількість груп викладачів, які розподілені посадами, науковими ступенями та вченими званнями, M – кількість викладачів, і n - кількість викладачів в і -й групі, $Ω$ - область обмежень, P, L, A– множина навчальних дисциплін, 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. Кожен вектор буде множитись на вектор індикатор, який у відповідних позиціях буде мати «1» якщо вимога виконується, і «0», якщо – ні, як на формулі (2.3). Це забезпечить простоту реалізації, що дозволить програмі просто перевіряти певні умови і передати обрахунок якомусь іншому методу.

$$X\{Z_j^v\} = \begin{Bmatrix} 1, \text{якщо } Z_j^v \text{ виконується} \\ 0, \text{в іншому випадку} \end{Bmatrix} \quad (2.3)$$

Дана функція є великою і може спантеличити з першого погляду, 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 відповідних позиціях має коефіцієнт пріоритети кожного викладача. Спочатку іде перевірка виконання вимог, під час якої ці вектори множаться на відповідні індикаторні функції.

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 множимо 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.

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 на таблиці 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, а що ні, бо це залежить на пряму від того, з якого боку підійти до задачі, чи з якої сторони подивитися. У даній роботі первинним ключом було вибрано

комплект з Предмета, Типу заняття та Групи, у якій це заняття буде проводитись, бо саме ця інформація одночасно 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 не унікальним ключем.

Також можна зауважити, що один предмет може викладатися у декількох групах. Саме щоб анулювати так проблеми ми і вибираємо таку модель, бо якби ми вибрали іншу модель, виникли б інші проблеми.

Дана модель описана на таблиці 1 потребує спрощення, оптимізації на вдосконалення. Всю інформацію про розклад ми будемо розміщувати в паралелепіпед. Спочатку ми розділимо її на 4 групи, в які ми об'єднаємо певні поля як показано у формулах 2.4, 2.5, 2.6, 2.7.

X_1=<День> (2.4)

X_2=<Пара> (2.5)

X_3=<Аудиторія> (2.6)

Z=<Викладач – (Предмет – Тип) – (Курс – Група)> (2.7) 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 дисциплін. 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.

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 як у формулі (2.8).

$$r_n = (D_i, T_j, R_k, N_S) \quad (2.8)$$

Де D_i- день тижня, T_j- номер пари, R_k – аудиторія, N_S – ланка, яка з'єднує в собі ключ <Викладач – предмет – група>, r_n – Розклад. Звідси можна зробити

висновок, про розмірність кубу, який буде мати 3 виміри з четвертим у вузлах. Розміри кубу будуть статичні, оскільки в тижні всього 6 робочих днів, та в день може бути лише 6 пар, а кількість аудиторій буде братись з даних про кафедру, але про це в наступному розділі. Графічно можна зобразити розклад як показано на рисунку 2.2.

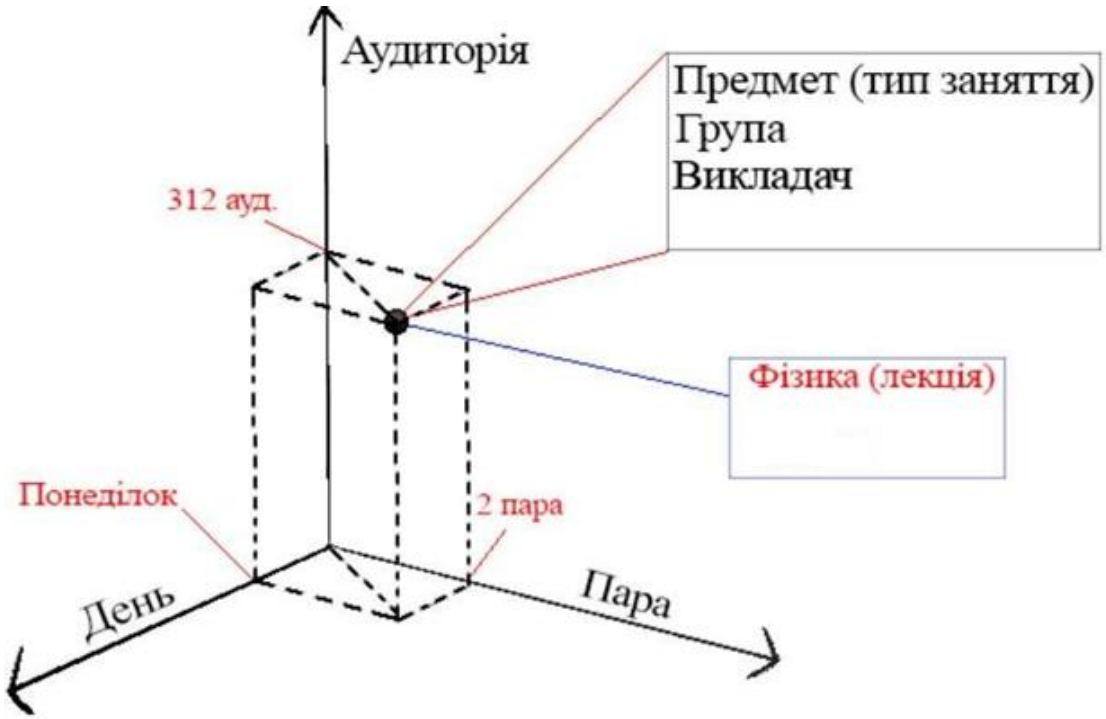


Рисунок 2.2. – Графічне уявлення розкладу

Для кращого розуміння цього рисунку 2.2 давайте розглянемо приклад. Нехай ключ складається з лекції з дисципліни «Фізика». Пара буде проводитись в 312 аудиторії в понеділок на другій парі. Тому як показано на рис. 2.2 у кубі за координатами (1, 2, 312), що відповідає першому дню тижня Понеділку, другій парі та відповідній аудиторії, буде знаходитись ланка <Фізика(лекція)>. Очевидно, що тепер в цей час і в цій аудиторії ніхто більше не може бути, бо це місце в кубі вже зайняте.

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 може бути лише в одній ланці одночасно гарантуватись даною схемою не може, тому це

буду гарантувати алгоритм генерації розкладу.

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.

Для виконання даної задачі було обрано Еволюційний 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.

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 також алгоритм показаний на рисунку 2.3:

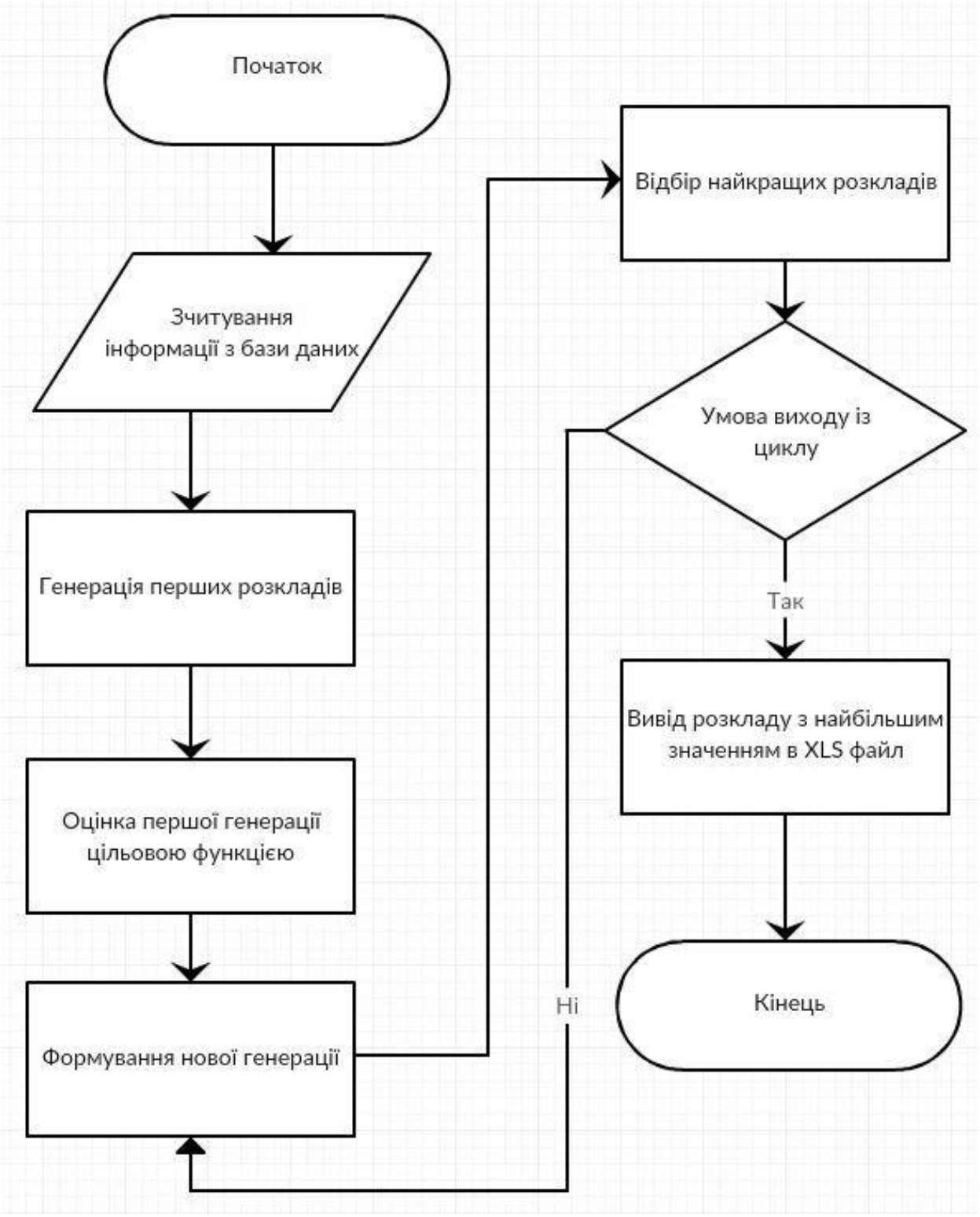


Рисунок 2.3 – Блок-схема алгоритму формування розкладу

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 швидко, на відміну від старих машин.

2.4.1 Перша генерація

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.

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

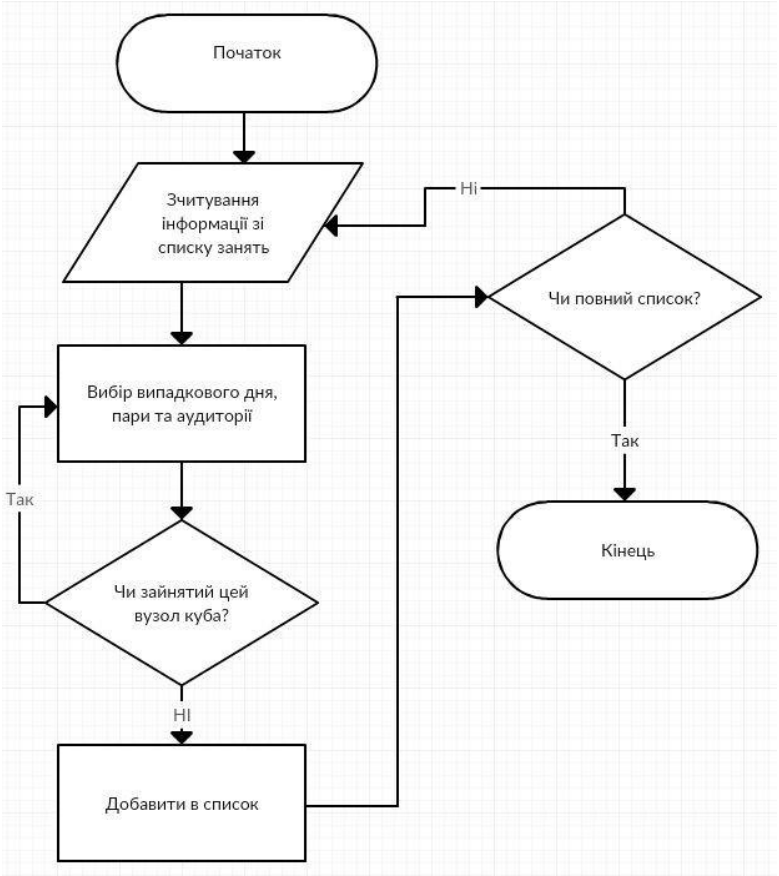


Рисунок 2.4 – Блок-схема формування першої генерації

2.4.2 Змішування та створення нової генерації

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 рисунку 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.

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.

Змішування двох розкладів відбувається так: спочатку чи випадково вибираємо координати куба (X1, Y1, Z1). Після чого ми 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 (X2, Y2, Z2). Цей процес можна побачити на рис. 2.6.

Після цього відбувається обмін, але дисциплінами, а координатами. Тобто дисципліну 1 з координатами (X1, Y1, Z1), яку ми вибрали на першому кроці, ми переміщуємо на позицію (X2, Y2, Z2), а дисципліну 2 з координатами (X2, Y2, Z2) ми переміщуємо за координатами (X1, Y1, Z1). Кінцевий результат змішування можна побачити на рис. 2.7.

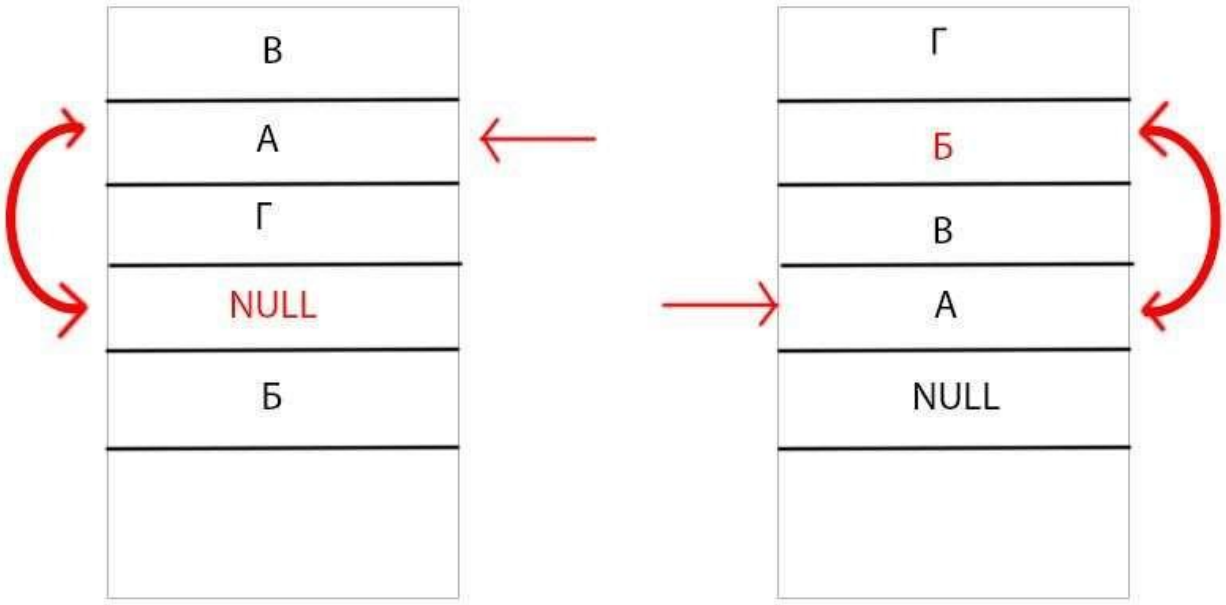


Рисунок 2.6 – Перший крок змішування

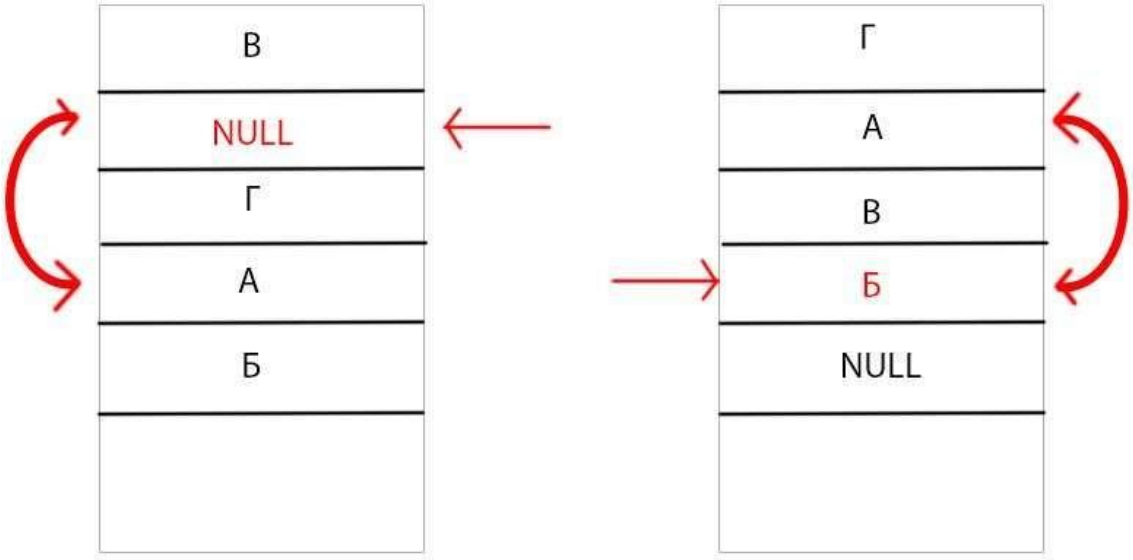


Рисунок 2.7 – Після змішування

Як видно на рисунку множина дисциплін не змінилася. Це дуже важливий момент. Справа в тому, що якщо ми робимо зміни в самому розкладі, не беручинічого ззовні, то у нас збережеться цілісність наукового плану, та будуть

гарантуватись жорсткі вимоги, які є реалізовані через куб.

Після цього іде перевірка валідності розкладу, яка проходить так. 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 одній аудиторії 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.

Загальну схему формування нових потомків можна побачити на рис 2.8.

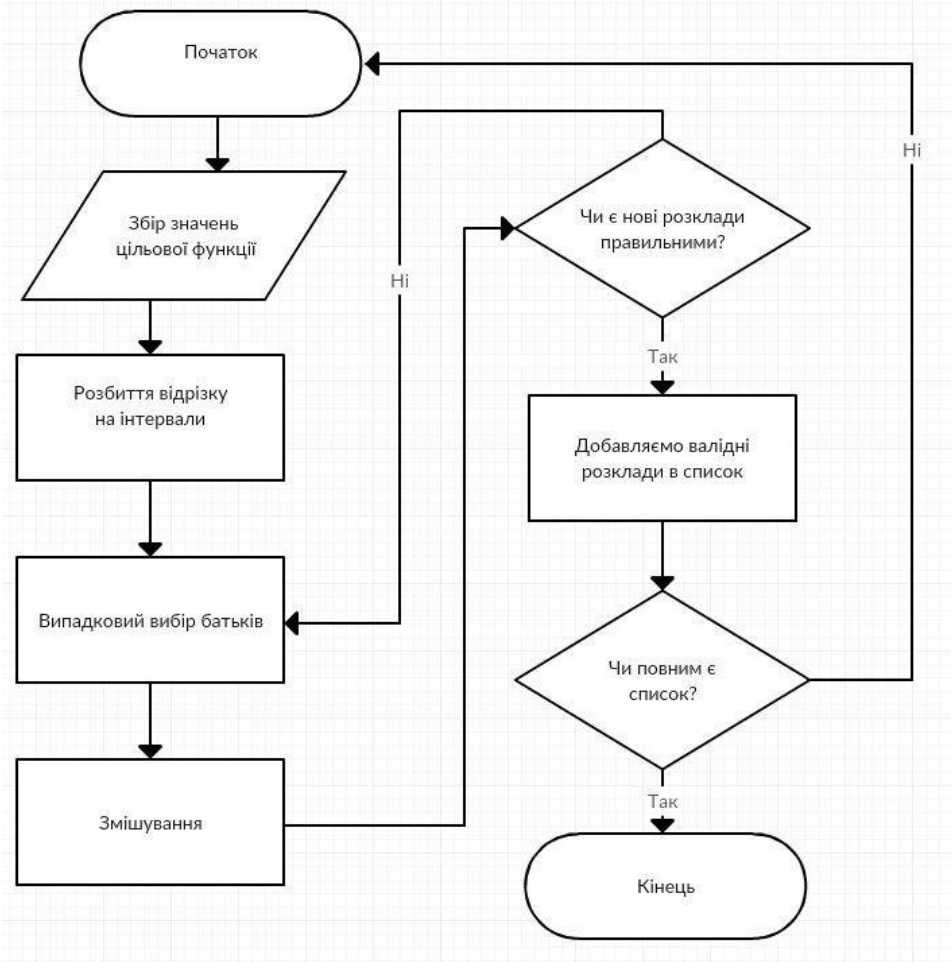


Рисунок 2.8 – Блок-схема формування потомства

2.4.3 Селекція та струс

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

Після закінчення кожної ітерації буде проводитись обрахунок різниці між найменшим значенням в списку генерації та найменшим як показано на формулі 2.9.

$$\max_i |\max_j F_{ij} - \min_j F_{ij}| < \varepsilon \tag{2.9}$$

Де F_{ij} – j-тий розклад на і-тій ітерації, ε – задана точність. Точність алгоритмуми задаємо виходячи з експериментальних досліджень. Та чисельної потужності системи, на якій програма запущена, бо чим більша точність, тим довше буде знаходитися рішення.

2.5 Висновки

В даному розділі було розглянуто основні алгоритми та моделі, які будуть використані для виконання даної задачі. Вибрано багатовимірний куб, як модель 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. РЕАЛІЗАЦІЯ ПРОГРАМНОЇ МОДЕЛІ ТА ЇЇ ТЕСТУВАННЯ

3.1 Вибір середовища програмування

Для виконання даної задачі нам потрібно вибрати хороше середовище програмування, яке буде виконувати наші вимоги, і також мати невисоку ціну. Щодо останнього можна зауважити, що більшість сучасних середовищ уже є в open-source, а ті що не у open-source уже стали безплатні, тому про ціну не вартує переживати.

Основні вимоги до середовища програмування для виконання даного проекту будуть:

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

Виходячи з цих вимог вибір був зупинений на пакеті MS Visual Studio 2015 Community з базою даних на основі MS SQL Server. Мовою програмування буде C#. Причини цього вибору:

- Безплатна платформа
- Дуже потужний інструментарій завдяки бібліотекам та вбудованим класам
- Зручний редактор коду з виділенням синтаксису, перевірки синтаксису в реальному часі та підказкам, які допомагають виправляти помилки
- Зручний графічний редактор класів, який дозволяє автоматизовано

генерувати потрібні зв’язки

- Можливість адмініструвати базу даних напряму з середовища
- Зручний конструктор інтерфейсу на основі WPF та XML
- Об’єктно-орієнтованість мови програмування допоможе реалізовувати велику кількість проблем
- Автоматична система стирання зайвих даних допоможе рівномірно розділяти ресурси системи
- Легкість написання коду на мові C#
- Можливість компіляції для мультиплатформного користування.

Опис програмної реалізації та архітектура бузи даних буде описана нижче. Вся реалізація була написана використовуючи лише інструменти MS VS 2015.

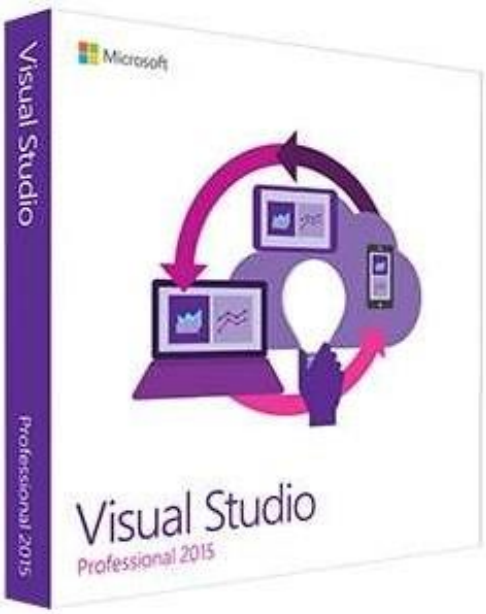


Рисунок 3.1 – MS Visual Studio 2015 [5]

3.2 Архітектура бази даних

Для того щоб реалізувати схему описану в розділі 2 нам треба створити базу даних яка буде зберігати всі дані про навчальний заклад, які нам потрібні для побудови розкладу. Виходячи з опису в таблиці 1, ми повинні створити такі таблиці:

1. Таблиця аудиторій
2. Таблиця викладачів
3. Таблиця груп студентів
4. Таблиця дисциплін.

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

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 таку базу даних побудовану на таких таблицях: Discipline – таблиця дисциплін, Teacher – таблиця викладачів, Rooms – таблиця аудиторій і Groups – таблиця студентів.

- Discipline
  - Id_disc – ID дисципліни
  - disc_name = назва дисципліни
  - course – курс, на якому проводиться
  - group_no – група, в якій проводиться
  - lectures – кількість лекцій в тиждень
  - practice – кількість пар в тиждень

- labs – кількість лабораторних в тиждень

— Teacher

- Id_tchr – ID викладача
- Pib – Ім’я викладача
- Degree – його ступінь
- disc_name – предмет, який він веде
- lectures – дорівнює 1, якщо він проводить лекції
- practice - дорівнює 1, якщо він проводить практичні
- Labs - дорівнює 1, якщо він проводить лабораторні.

— Groups

- Rooms Id_grp – ID групи
- Group_no – номер групи
- Course – курс
- Cnt – кількість студентів в групі.

— Rooms

- Id_room – номер аудиторії
- Type – тип аудиторії
- Places – кількість місць.

Повний опис бази даних разом з зв’язками 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.

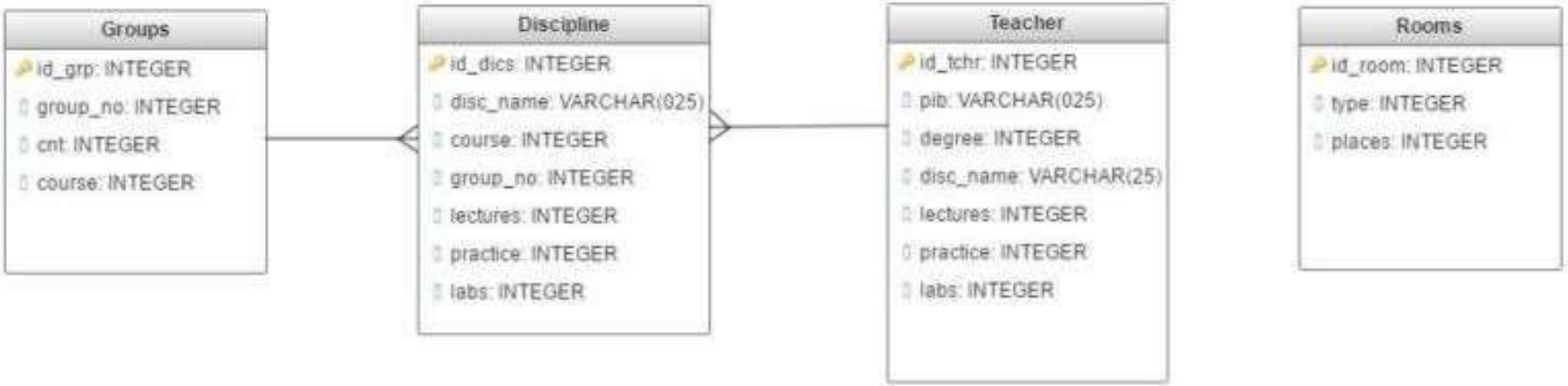


Рисунок 3.2 – Схема бази даних

3.3 Розробка опису програми на мові C#

Щоб реалізувати ці задачі ми створимо такі класи.

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 3.3. Діаграму залежностей можна розглянути на рис 3.5. У полях атрибутів цього класу можна побачити списки для кожної бази даних.

Рисунок 3.3 – Клас Dictionary

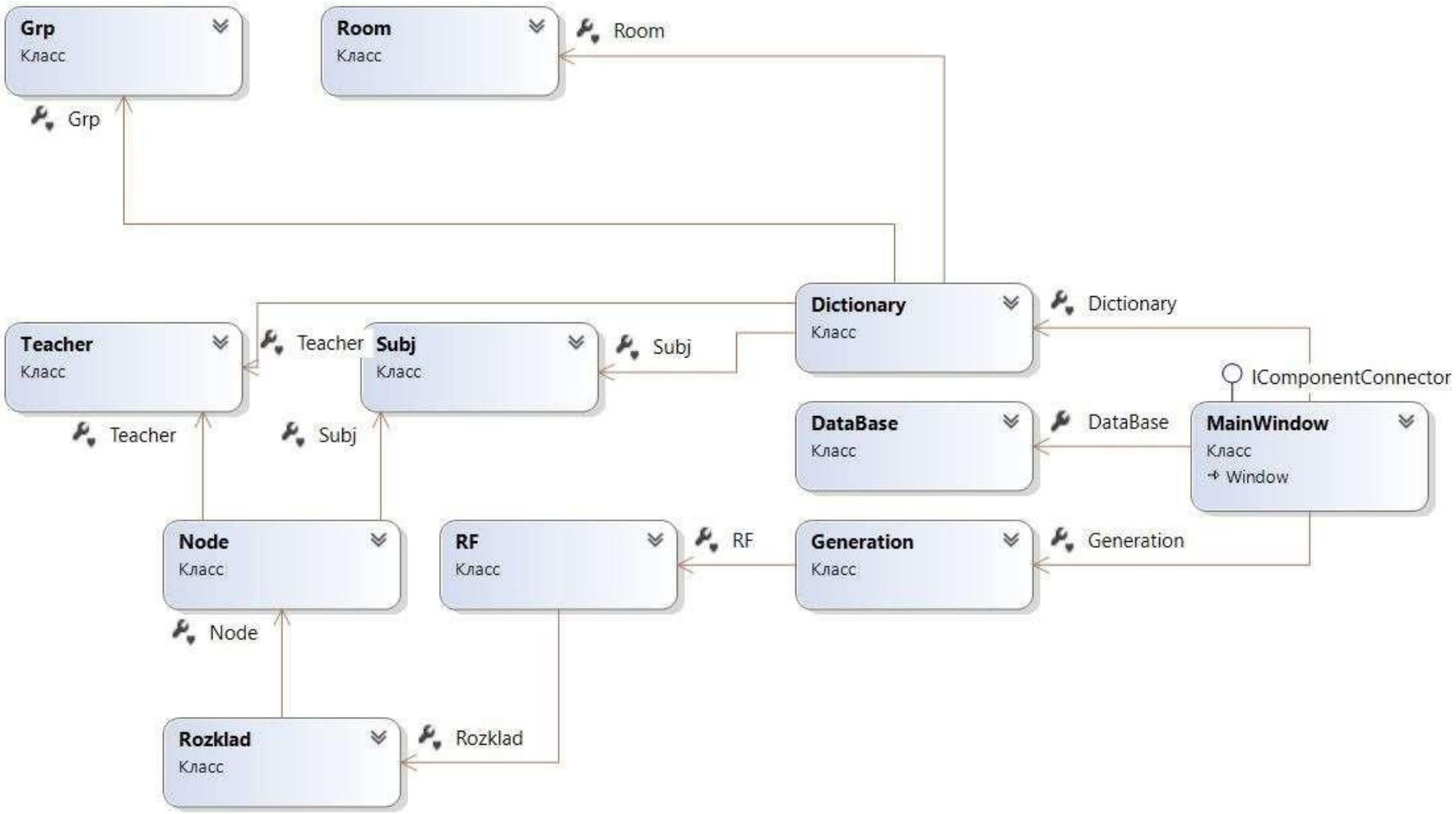


Рисунок 3.4 – Діаграма класів проекту

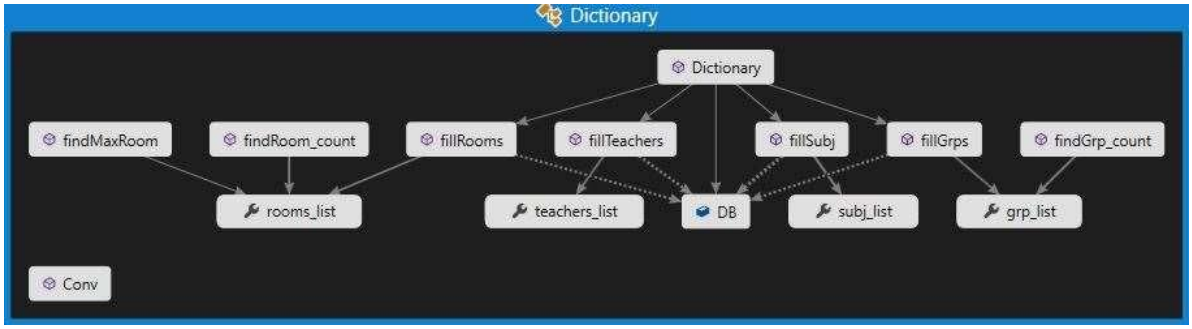


Рисунок 3.5 – Діаграма залежності класу Dictionary

Треба зауважити 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, як ланку розкладу. Наприклад, нехай у нас предмет Математика, який повинен проводитись в 21 групі. Повинно бути 2 лекції на тиждень та 1 практика, тому, виходячи з цього, у нас буде 3 записи в списку занять: 2 лекції та 1 практика з математики. Це полегшить нам подальшу генерацію розкладу.

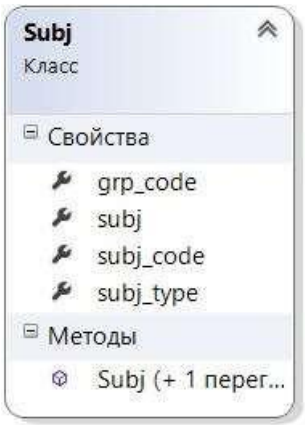


Рисунок 3.6 – Клас Subj

3.3.2 Клас Rozklad

Наступним класом, який ми розглянемо буде Rozlkad. Цей клас зберігає описаний в розділі 2 куб, який має розмірність 6 на 6 на кількість аудиторій. Діаграма залежностей показана на рис 3.7.

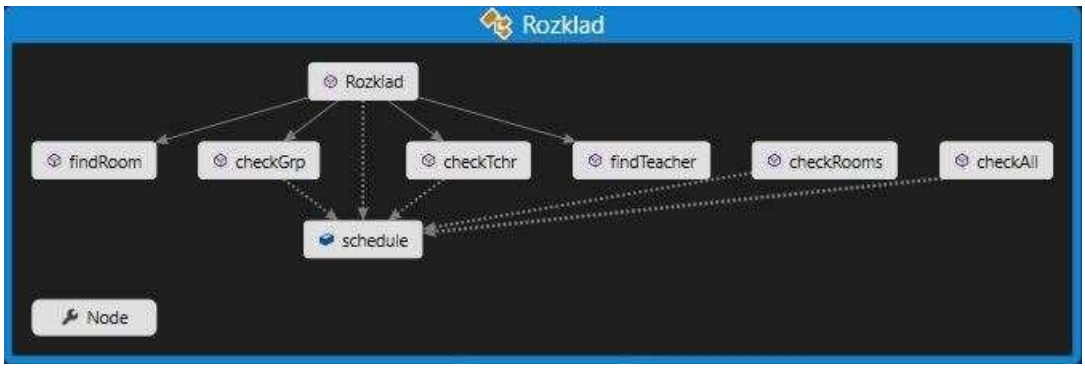


Рисунок 3.7 – Діаграма залежності класу Rozklad

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.

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. Його опис можна побачити на рис 3.8.

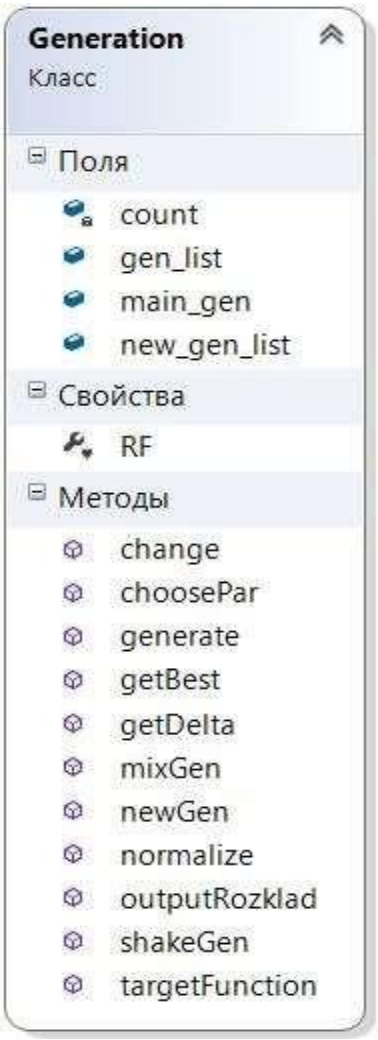


Рисунок 3.8 – Клас Rozklad

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

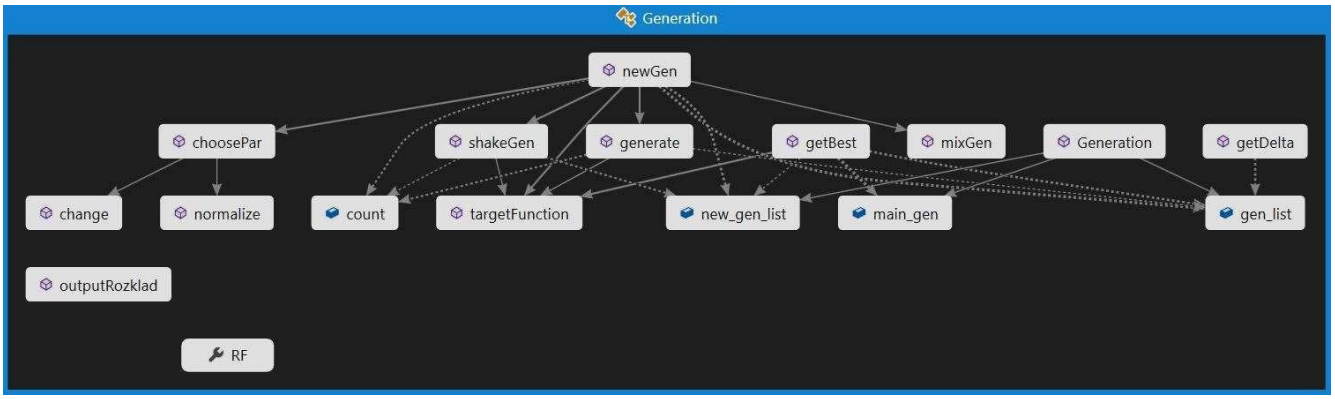


Рисунок 3.9 - Діаграма залежності класу Generation

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.

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.

Також вартує зауважити 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 мінятись. Загальна діаграма зв’язків між класами зображена на рис 3.10.

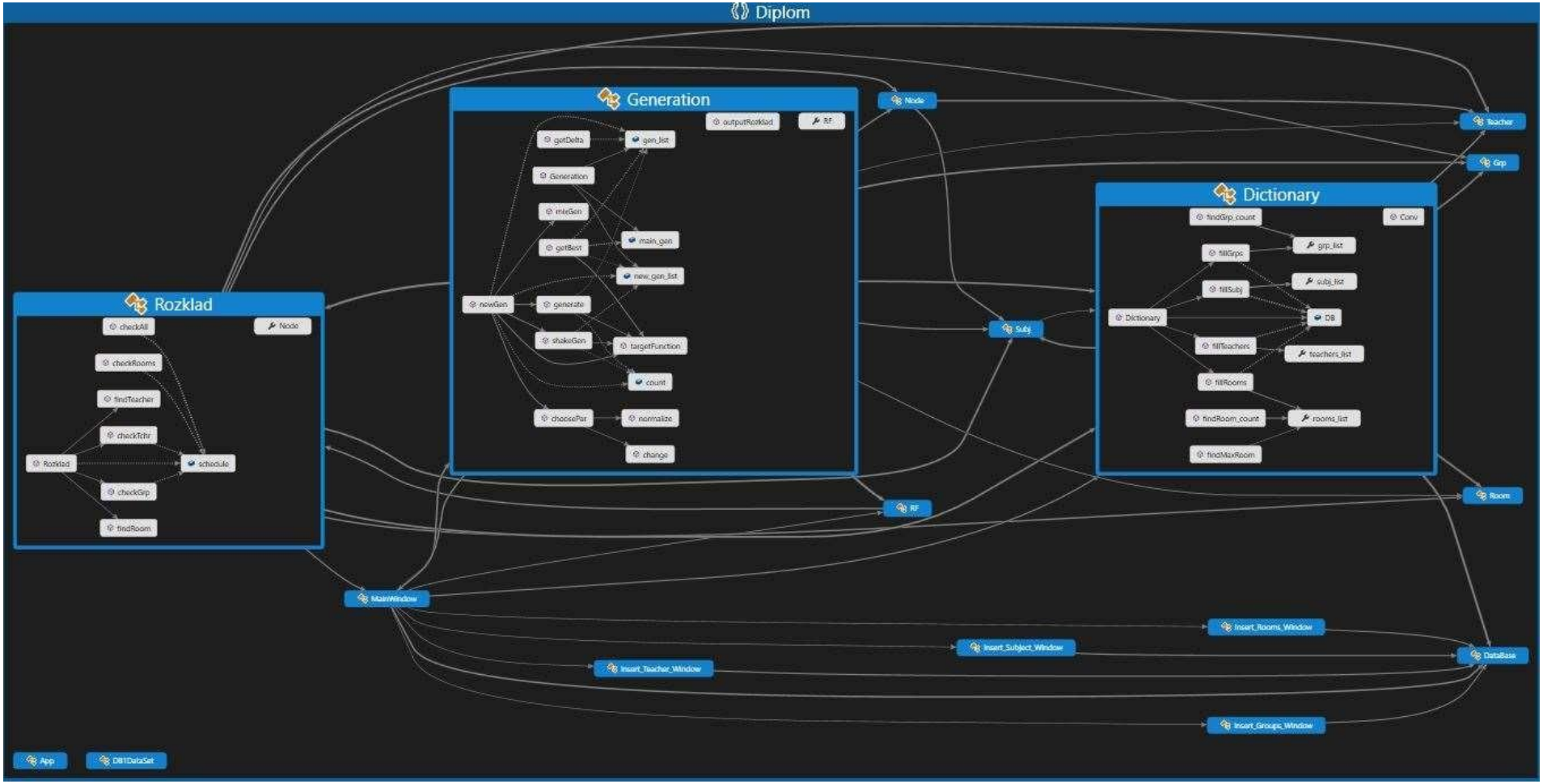


Рисунок 3.10 – Загальна діаграма залежності класів

3.4 Розробка плану тестування розкладу

Для того щоб перевірити роботу програми, треба проаналізувати цілі, які 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 якої є готовий розклад занять для навчального закладу.

Проводячи тестування по кожному пункту буде гарантувати нам адекватність результату, його правильність та впевненість у правильній роботі алгоритму.

3.5 Розробка тестового додатку

Інтерфейс програми складається 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 аудиторій, а також містить таблицю, де всі ці дані можна переглянути перед генерацією.

Натиск на всі кнопки введення будуть виводити окремі діалогові вікна, які показані на рис 3.12, де зручно прояснено метод користування цим інтерфейсом. Головна кнопка “Генерувати” при натисканні запускає всі необхідні для обрахунку алгоритми, які створюють першу генерацію, реалізують еволюційний алгоритм та відбираються найкращі розклади.

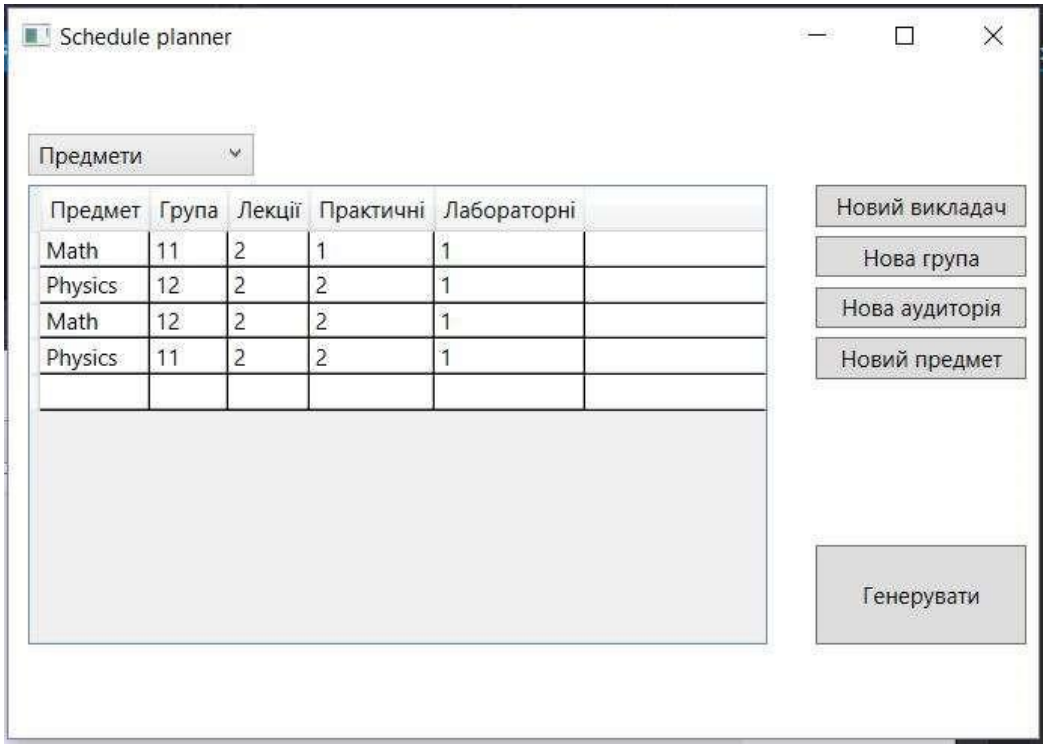


Рисунок 3.11 – Головне вікно

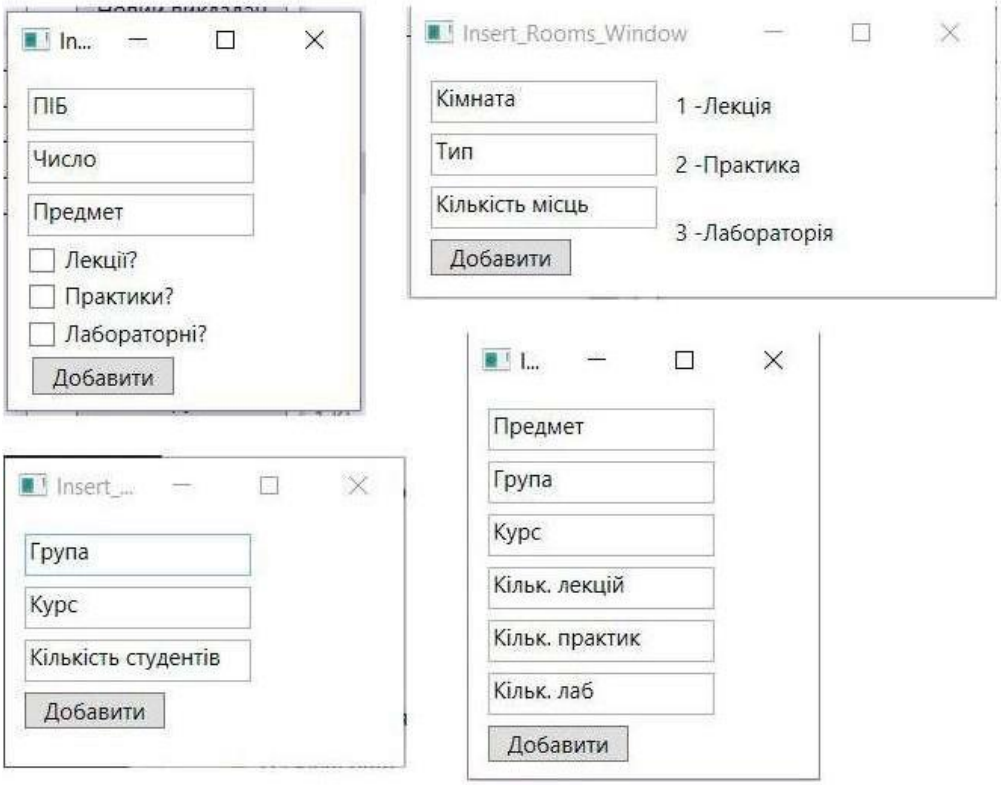
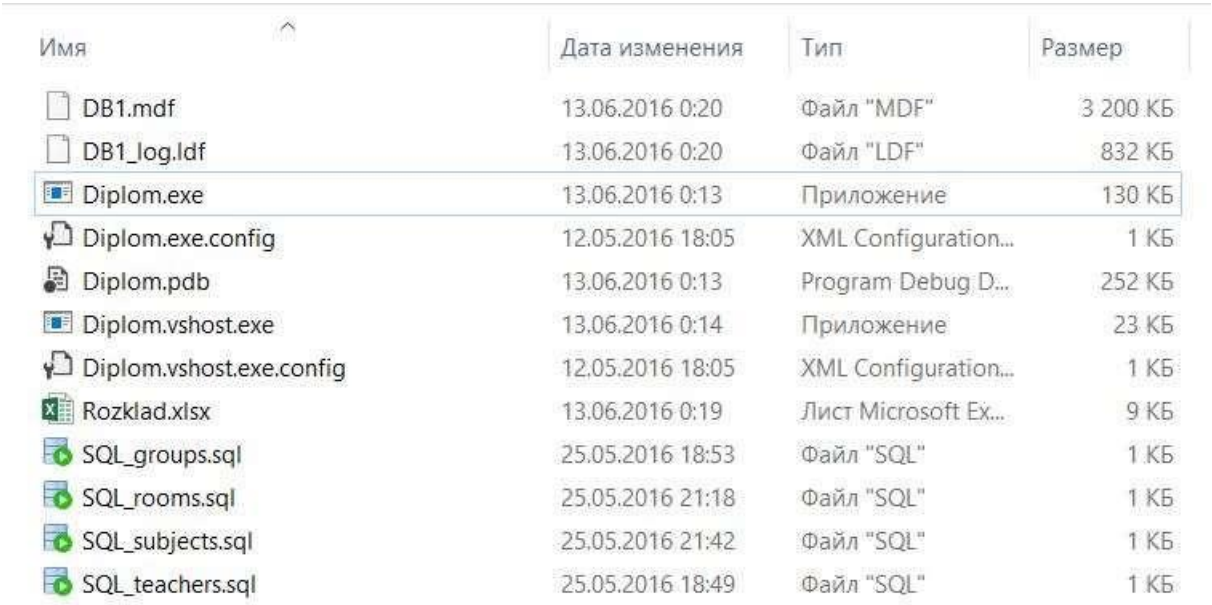


Рисунок 3.12 – Діалогові вікна введення даних

Після успішної генерації розкладу, він буде завантажений у XLSX файл на диску в папку з виконавчим 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 на рис 3.13. Тут видно і файли запитів, і файли бази даних, і виконавчі файли, і файл виводу розкладу.



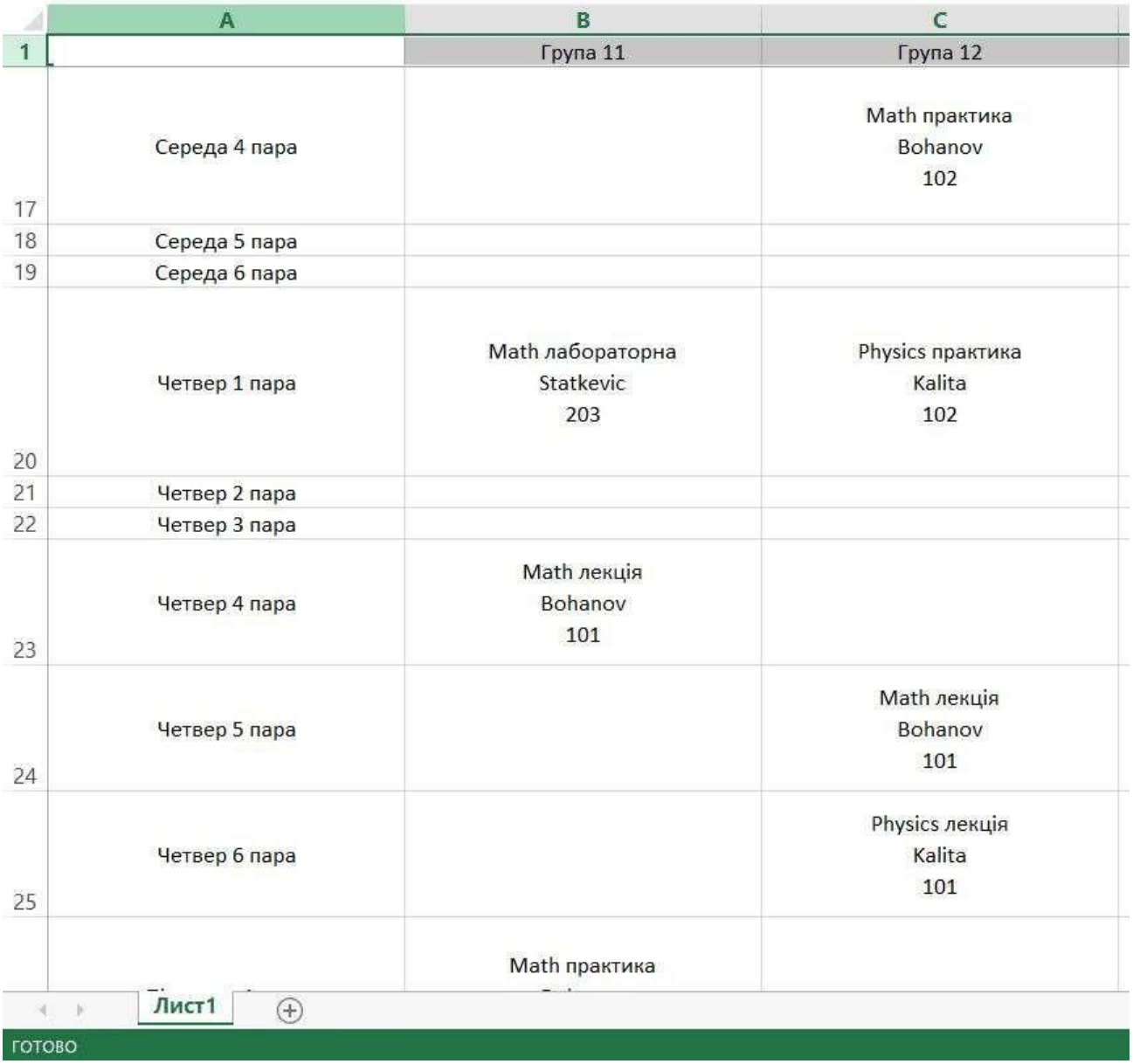
Имя	Дата изменения	Тип	Размер
DB1.mdf	13.06.2016 0:20	Файл "MDF"	3 200 КБ
DB1_log.ldf	13.06.2016 0:20	Файл "LDF"	832 КБ
Diplom.exe	13.06.2016 0:13	Приложение	130 КБ
Diplom.exe.config	12.05.2016 18:05	XML Configuration...	1 КБ
Diplom.pdb	13.06.2016 0:13	Program Debug D...	252 КБ
Diplom.vshost.exe	13.06.2016 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.

Треба також зауважити, що вимоги викладачів та студентів ніяк не можна зберігати в програмі після компіляції, і їх треба прописувати до компіляції. Це

зумовлено тим, що для реалізації такого функціоналу треба набагато більше часу і ресурсів, ніж були доступні під час цієї роботи.



	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 Statkevic 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

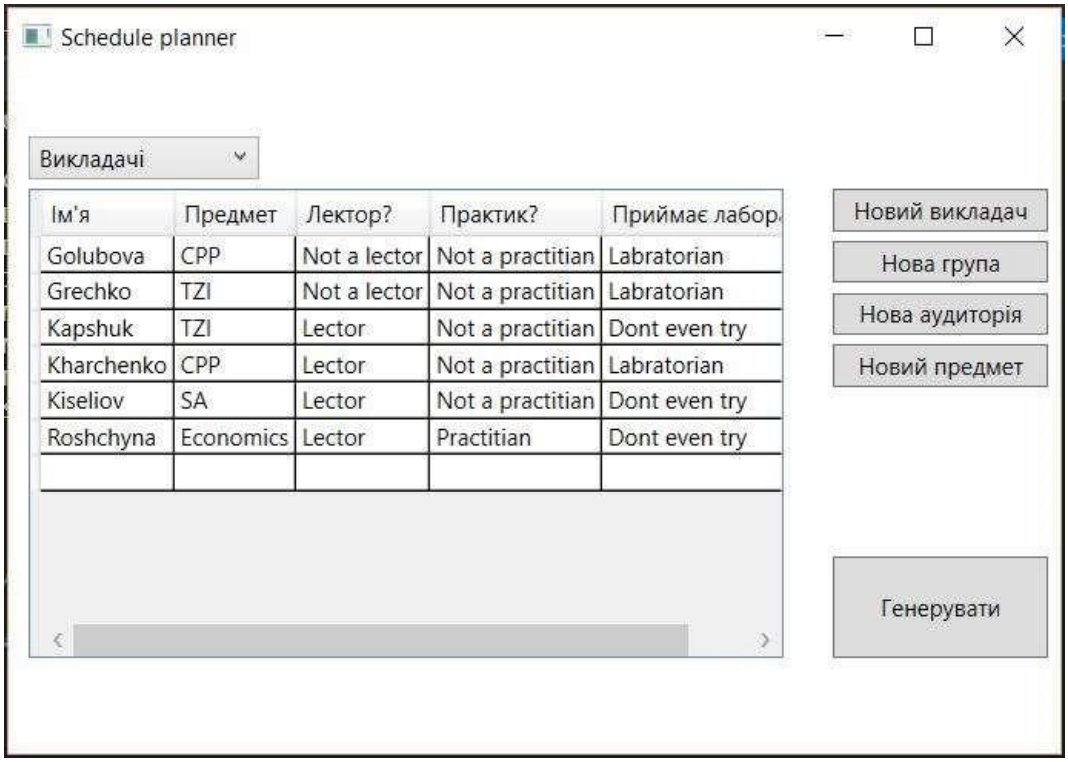


Рисунок 3.15 –Вихідні дані про викладачів

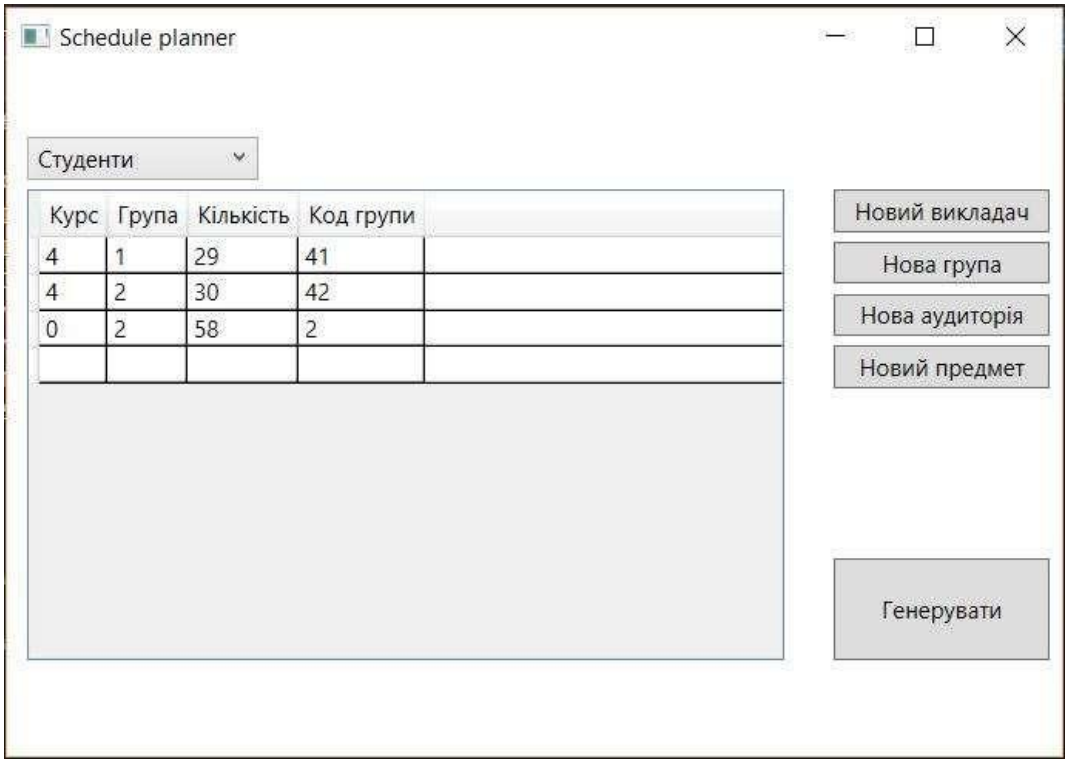


Рисунок 3.16 –Вихідні дані про студентів

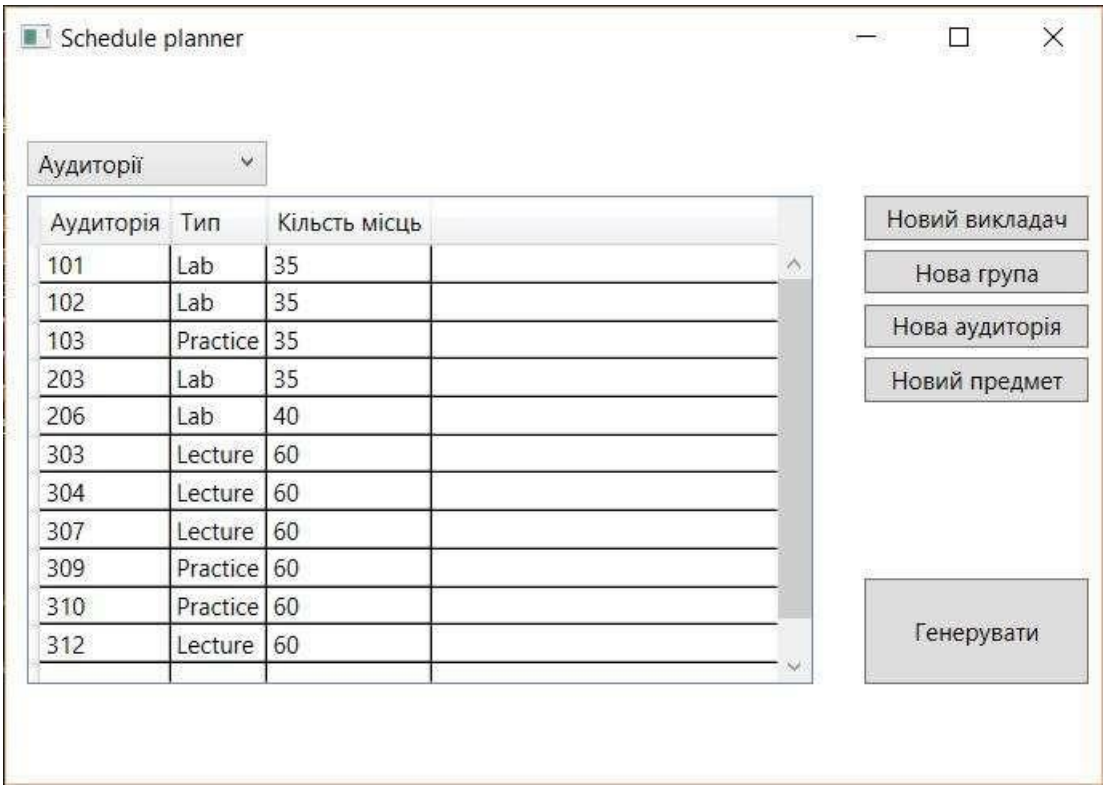


Рисунок 3.17 –Вихідні дані про аудиторії

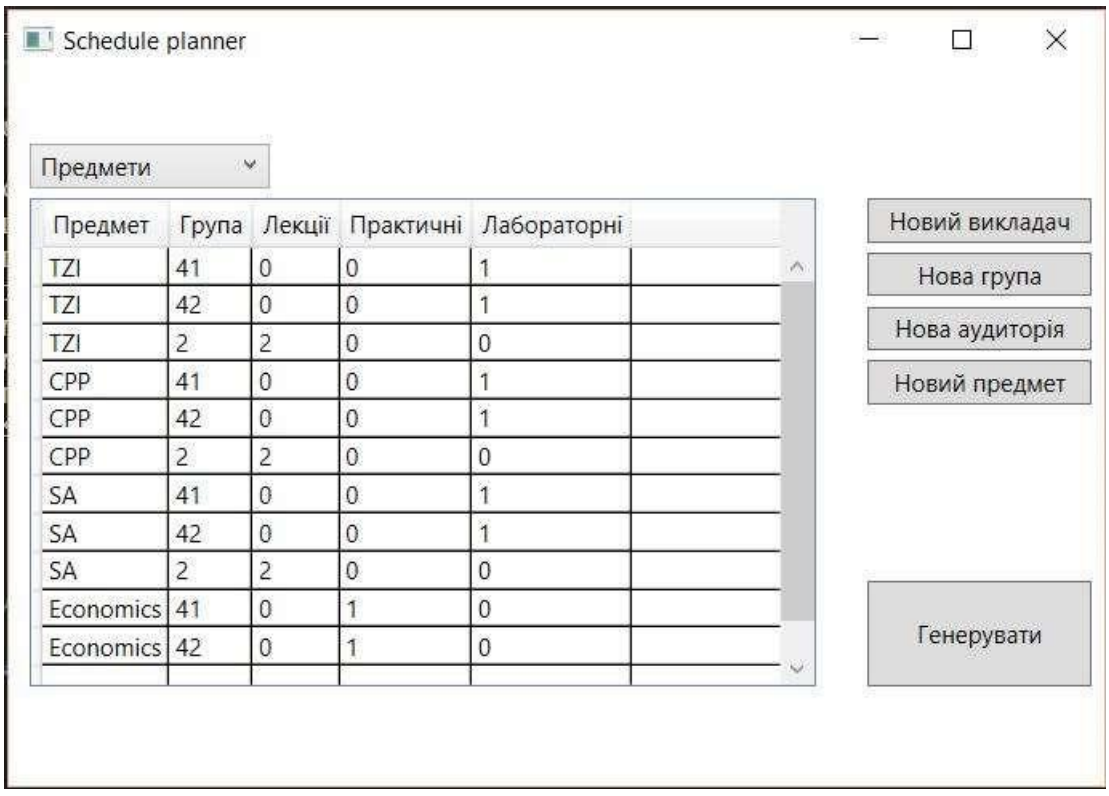


Рисунок 3.18 –Вихідні дані про навчальний план на тиждень

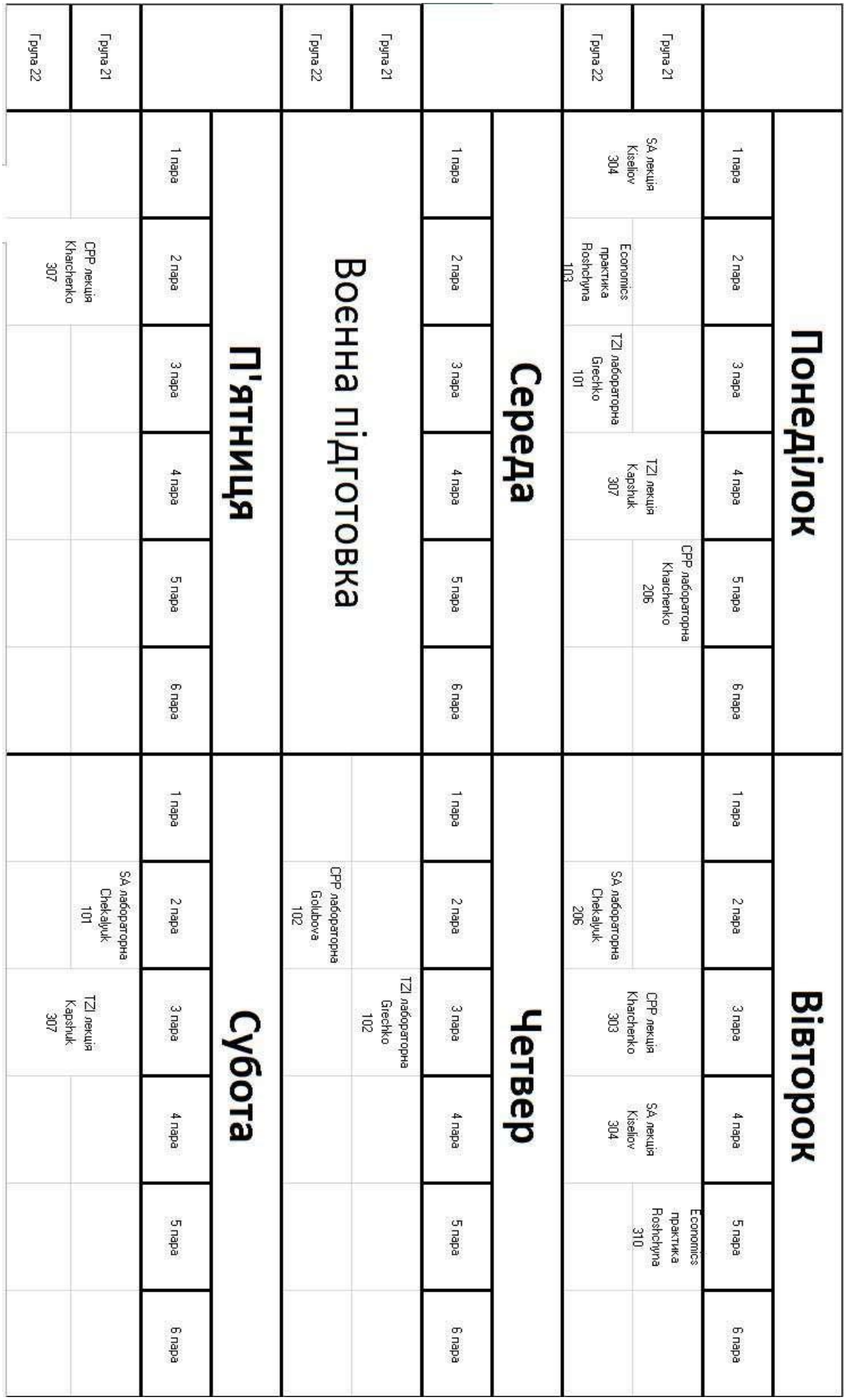


Рисунок 3.19 –Згенерований програмою розклад

Даним розкладом виконуються всі жорсткі вимоги: нема 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 2,356742412 секунди за 7 кроків.

З точки зору нежорстких вимог розклад чудово реалізує ці потреби, а саме наявність вихідного дня під час тижня, відсутність вікон, мала кількість перших пар, відсутність вікон у викладачів, пари в основному зосереджені в околі 2-3 пари та мінімум пар у суботу.

Виконання цих вимог показує, що генерація розкладу пройшла успішно. Також давайте поглянемо на графік залежності цільової функції від кількості кроків як показано на рис 3.20.

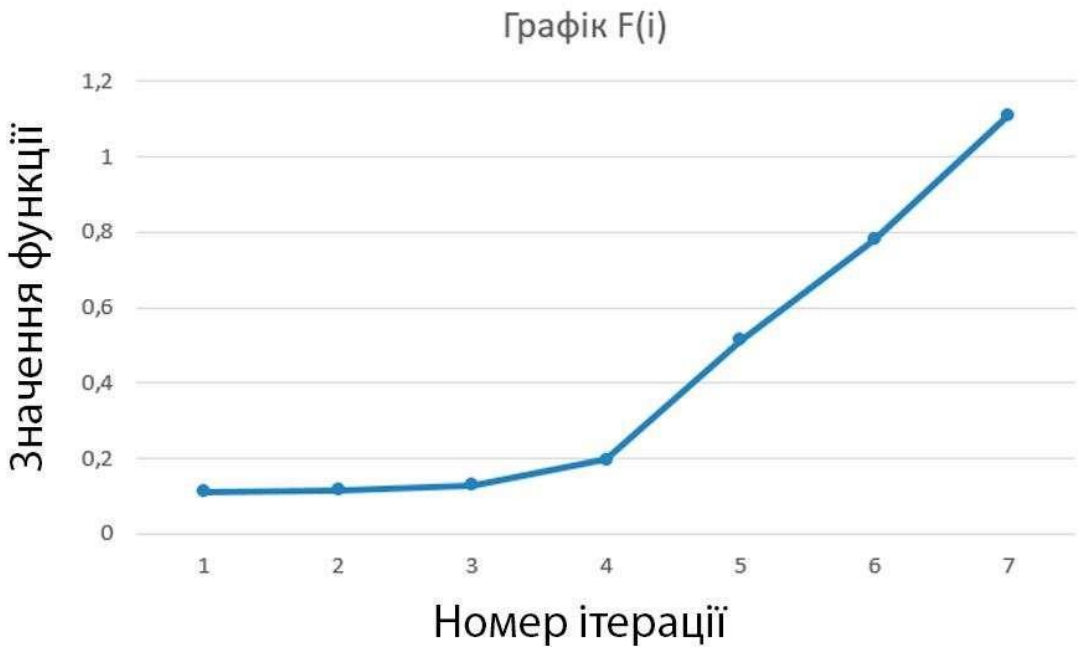


Рисунок 3.20 –Графік залежності середнього значення функції по всій генерації від ітерації еволюційного циклу

Також треба розглянути графік, який показує залежність різниці між мінімумом та максимумом цільової функції від ітерації як у формулі 2.9. Цей графік зображена на рис 3.21.

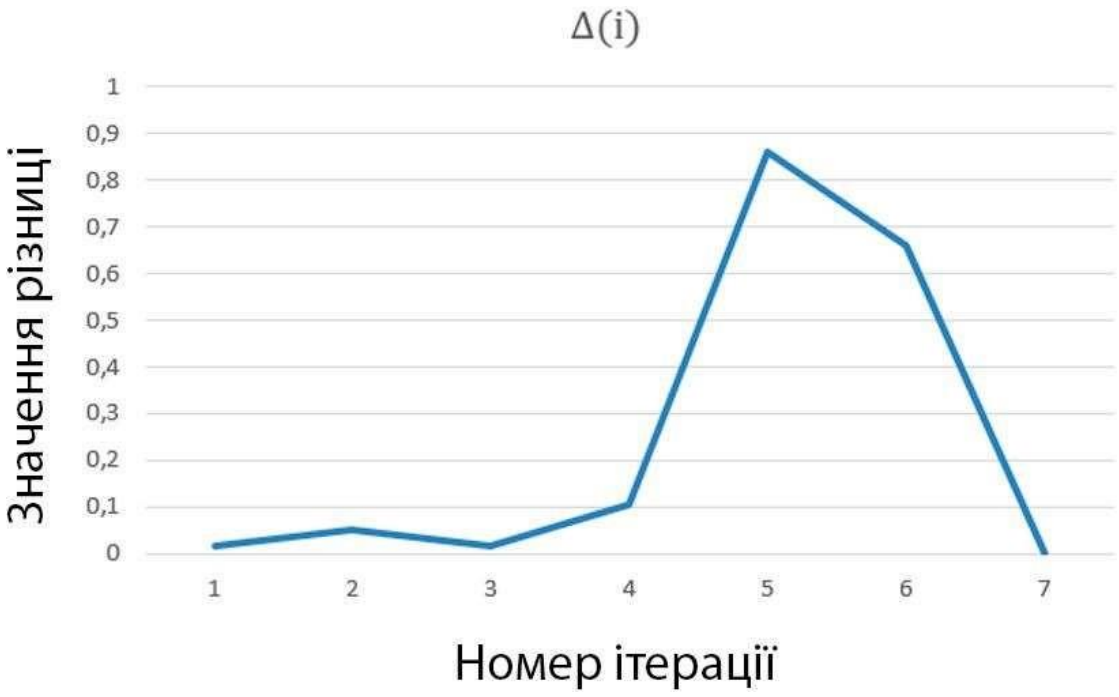


Рисунок 3.21 –Графік залежності залежність різниці між мінімумом та максимумом цільової функції від ітерації

3.7 Висновки

З огляду на отримані результати можна зробити наступні висновки. Програма формує розклад, який гарантовано відповідає жорстким вимогам. Жодні фізичні умови, чи логічні не є порушені під час генерації. 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 пріоритеті, а вимоги викладачів, які є вищими за званням були виконані навіть краще.

Щодо часу, то алгоритм зайняв дуже мало часу, щоб згенерувати горовий 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 підлягають до змін. А плани навчання міняються тільки трохи, але не сильно.

Щодо залежності цільової функції від ітерації видно, що середнє її значення росте з кожною ітерацією, що є чудовим доказом 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 змішуванні цього «альфа» розкладу з іншими в ітерації.

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

ВИСНОВКИ

В ході виконання даної роботи було досліджено предметну область складання розкладів та розроблено алгоритм формування розкладу навчального процесу. 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 еволюційний алгоритм, на основі цільової функції. Було розроблено саму цільову функцію, яка відображає побажання як і викладачів так і студентів, враховує пріоритети та побажання. Також було розроблено кроки генетичного алгоритму, які повинні вирішити задачу. Було розроблено та реалізовано математичну модель розкладу, яка близько відображає суть фізичного розпорядку роботи навчального закладу.

Було вибрано середовище програмування MS VS 2015 та мова C# 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 завдяки 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 таблиця.

Було реалізовано всі розроблені методи на мові програмування C# і SQL. Повна архітектура повністю задовольняє всім вимогам алгоритму, реалізовує всі структури та гарантує виконання поставлених задач. Також ця реалізація є оптимізовано та швидкою. Надійність гарантується методами, які перевіряються розклад на адекватність. А збіжність цільової функції гарантує сам алгоритм. І зібрано їх всіх в один додаток

Було перевірено розроблену програму на реальних даних та отримані хороші результати. Виконання жорстких умов було виконано, оптимізація розкладу та приближення до виконання всіх нежорстких умов було досягнуто. Звісно ж,

отриманий з першого експерименту, розклад не відповідає всім вимогам здорового глузду, 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.

Судячи з результатів дана реалізація має сильний потенціал до розширення. Кількість дисциплін, викладачів, груп, аудиторів можна розширити з масштабу одного потоку до масштабу університету. Вимоги можна формувати для різних інстанцій. Якщо в дану реалізацію імплементувати систему розпізнавання тексту, додаток зможе сам формувати вимоги без участі людини.

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

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

Тому виконання даного завдання можна вважати успішним, а отриманий розклад лише трохи, відрізнявся від того, який був би 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., Сухоносов 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 занятий університету / Деканова М.В. // Вісник Полоцкого 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.

ДОДАТОК А

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;
        //Пара
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 saturnday = 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)
            {
            saturnday++;
            }
            }
            double res_2 = 0; if
(saturnday == 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;
        }
        }
        }
        }
```