

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему АВТОМАТИЗОВАНА СИСТЕМА СКЛАДАННЯ ЕЛЕКТРОННОГО
РОЗКЛАДУ

AUTOMATED ELECTRONIC TIMETABLING SYSTEM

Виконав: студент 4 курсу, групи КНТ-137

Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Чернега І.С.

(прізвище та ініціали)

Керівник Неласа Г.В.

(прізвище та ініціали)

Рецензент Терещенко Е.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 20 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

Чернега Ілля Сергійович

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Автоматизована система складання електронного розкладу. Automated Electronic Timetabling System
керівник проєкту (роботи) Неласа Г.В., к.т.н, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
- затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103
2. Строк подання студентом проєкту (роботи) 14 травня 2021 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Реферат. Перелік скорочень та умовних познач. Вступ. 1. Аналіз предметної області. 2. Вибір та обґрунтування структури системи, яка проєктується. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора. Висновки. Перелік джерел посилання. Додатки.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень). Слайди презентації.

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Неласа Г.В, к.т.н. ,доцент		
Нормоконтроль	Камінська Ж. К., асистент		

7. Дата видачі завдання “20” лютого 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-3 тижні	Розділ 1
3	Розробка архітектури програми	4 тиждень	Розділ 2
4	Розробка програми	5-6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	7 тиждень	Розділи 4-5
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування.	8 тиждень	Додатки
7	Захист роботи	9 тиждень	

Студент

(підпис)

Чернега І. С.

(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Неласа Г.В

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
85 с., 10 табл., 21 рис., 4 дод., 22 джерел.

ЕЛЕКТРОННИЙ РОЗКЛАД, РОЗКЛАД ЗАНЯТЬ, СПИСКИ, ЗАМІНИ,
КАФЕДРА, ДИСПЕТЧЕР, НАВЧАЛЬНИЙ ВІДДІЛ.

Об'єкт дослідження – автоматизовані системи складання електронного розкладу.

Предмет дослідження – методи формування розкладу навчальних занять.

Мета роботи – створення програмної системи для складання електронного розкладу.

Актуальність та проблематика роботи полягає в тому, що в даний час в країні здійснюються інтенсивні реформи в системі вищої освіти з метою поліпшення якості освіти. Якість підготовки студентів і ефективність роботи викладацького складу залежить від рівня організації навчального процесу. Одним з критичних елементів цього процесу є розклад занять. Таким чином, проблема складання розкладу є важливою соціальною та економічною проблемою [1].

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, мова програмування C++, персональний комп'ютер з процесором процесор з тактовою частотою 1,8 ГГц. під управлінням операційної системи Microsoft Windows 7.

Результати. Створено програмний продукт, який дозволяє складати електронний розклад.

Висновки. Розроблено програмну систему яка дозволяє складати електронний розклад.

Галузь використання – формування розкладу занять у вищих навчальних закладах.

ABSTRACT

Explanatory note to the final qualifying work of the bachelor: 85 pages, 10 tables, 21 figures, 4 appendixes, 22 sources.

ELECTRONIC TIMESHEET, SCHEDULE, LIST, REPLACE, DEPARTMENT, DISPATCHERS, TRAINING DEPARTMENT.

The object of research is automated electronic scheduling systems.

The subject of research - methods of forming the schedule of educational classes.

The purpose of the work is to create a software system for compiling an electronic search schedule.

The urgency and problems of the work is that the country is currently undergoing intensive reforms in the higher education system in order to improve the quality of education. The quality of student training and the effectiveness of the teaching staff depends on the level of organization of the educational process. One of the critical elements of this process is the class schedule. Thus, the problem of scheduling is an important social and economic problem [1].

Materials, methods and technical means: structural and object-oriented programming, C ++ programming language, personal computer with a processor clocked at 1.8 GHz. running Microsoft Windows 7 operating system.

Results. A software product has been created that allows you to create an electronic schedule.

Conclusions. A software system has been developed that allows to compile an electronic schedule.

Area of use - the formation of classes in higher education.

ЗМІСТ

	С.
Перелік скорочень та умовних познач 8	8
Вступ 9	9
1 Аналіз предметної області 11	11
1.1 Основні поняття..... 11	11
1.2 Огляд існуючих програмних засобів, що вирішують аналогічні завдання..... 14	14
1.3 Постановка завдання до роботи 22	22
1.4 Висновок.....27	27
2 Вибір та обґрунтування структури системи, яка проєктується 28	28
2.1 Вибір технології розробки 28	28
2.2 Структурна схема програмного продукту 33	33
2.3 Висновок..... 34	34
3 Основні рішення щодо реалізації компонентів системи 35	35
3.1 Клас «списки» 35	35
3.2 Клас «розклад»..... 38	38
3.3 Висновки за розділом 3 40	40
4 Керівництво програміста 41	41
4.1 Призначення й умови використання 41	41
4.2 Вхідні і вихідні дані..... 41	41
4.3 Повідомленн..... 42	42
5 Керівництво оператора..... 44	44
5.1 Призначення й умови виконання програмного продукту 44	44
5.2 Виконання програмного продукту..... 44	44
5.3 Висновки за розділом 5 46	46
Висновки..... 47	47
Перелік джерел посилання 48	48
Додаток А Технічне завдання..... 50	50
Додаток Б Опис програмного продукту 56	56

Додаток В Текст файлів програми.....	63
Додаток Д Слайди презентації	79

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- API – Application Programming Interface;
БД – база даних;
ПІБ – прізвище, ім'я, по батькові;
ПП – програмний продукт;
СУБД – система управління базами даних;

ВСТУП

Актуальність роботи. В даний час в країні здійснюються інтенсивні реформи в системі вищої освіти з метою поліпшення якості освіти. Якість підготовки студентів і ефективність роботи викладацького складу залежить від рівня організації навчального процесу. Одним з критичних елементів цього процесу є розклад занять. Таким чином, проблема складання розкладу є важливою соціальною та економічною проблемою [1].

Складання розкладу занять є завдання оптимального управління такими ресурсами, як викладацький склад, аудиторний фонд, комп'ютерний парк навчального закладу і т. п. При цьому необхідно враховувати обов'язкові обмеження, такі як місткість аудиторій, санітарні вимоги до проведення занять, наявність сумісників, територіальні особливості розташування навчального закладу, а також численні додаткові вимоги, наприклад, відсутність вікон у студентів і викладачів.

Вихідними даними для розкладу є навчальний план, списки груп студентів, списки аудиторій, списки викладачів з предметів, календарні вимоги до розкладу. З математичної точки зору завдання складання розкладу відноситься до завдань цілочисельного програмування, складність яких зростає експоненціально з ростом числа змінних, що беруть участь в завданні (так звані NP-складні завдання). Для вирішення таких завдань розроблено багато точних (таких, що відносяться до класичної математики) методів і алгоритмів.

До класичних методів вирішення проблеми відносяться [2]:

- динамічне програмування;
- цілочисельне програмування;
- нелінійне програмування;
- імітаційне моделювання;
- метод гілок і меж;
- метод розмальовки графа;

- задача про призначення;
- та інші.

Загальним недоліком всіх класичних методів є складність і громіздкість структури використовуваної математичної моделі, а також занадто швидке, експоненціальне зростання часу, необхідного для виконання завдання, при збільшенні обсягів вихідних даних. Тому в даний час завдання, подібні складання розкладу, часто вирішуються за допомогою некласичних - евристичних методів, в основі яких лежать не точні математичні докази, а досвід і інтуїція [3].

Завдання автоматизації складання розкладу, на перший погляд, здається зрозумілим і простим, тому спроби його вирішення здійснювались багаторазово, і поки без особливого успіху. Оптимізація за суворими математичними моделями виявилася практично неефективною. В даний час вживаються різноманітні спроби вирішення цього завдання з позицій застосування евристичних алгоритмів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні поняття

Розклад занять вузу - це документ, що регламентує роботу всього навчального закладу, розподіляє зміст навчального плану на календарні дні навчального року. Основним завданням складання розкладу є планування методично правильного навчального процесу, з урахуванням взаємозв'язку навчальних дисциплін, можливостей студентів з навчання і побажань викладачів за графіком роботи [1] - [6].

Вихідна інформація для формування розкладу:

- структурно-логічні схеми дисциплін;
- сітки годин за курсами дисциплін;
- аудиторний фонд вузу;
- штатний розклад викладацького складу;
- перелік вимог, що пред'являються до розкладу.

Структурно-логічні схеми коротко і наочно відображають зміст основних тем, розділів навчальної дисципліни, логіку курсу в цілому і методику його викладу. Структурно-логічна схема представляє матеріал курсу в конкретній і структурованій формі, відображаючи зміст окремих питань теми або розділу у вигляді схем, графіків, креслень, формул, рівнянь, що допомагає учневі побачити особливості дисципліни, що вивчається.

Можна виділити наступні групи вимог до складання розкладу:

- методичні;
- організаційні;
- щодо забезпечення самостійної роботи студентів.

Вимоги також потрібно розділити на обов'язкові обмеження (без забезпечення яких навчальний процес неможливий) і додаткові вимоги, які впливають на якість навчального процесу. Додаткові вимоги також слід розділити за пріоритетами.

Головні методичні вимоги до розкладу:

- послідовність проведення навчальних занять відповідно до структурно-логічних схем дисциплін;
- рівномірність навчального навантаження;
- планування лекцій на перші години занять;
- планування від двох до чотирьох навчальних занять в день;
- чергування дисциплін в межах навчального дня;
- дотримання необхідного інтервалу часу між лекціями і практичними (лабораторними) заняттями;

Другорядні методичні вимоги до розкладу:

- відсутність повторень занять по одній дисципліні в один день;
- планування занять по складним дисциплінам на середину тижня;
- в п'ятницю кількість занять повинна бути менше, ніж в інші дні тижня;
- першим заняттям в понеділок має йти простий предмет.

Головні організаційні вимоги до розкладу:

- забезпечення всіх видів занять відповідними аудиторіями і технічними засобами;
 - виключення накладок в проведенні занять по викладачам і аудиторіям;
 - відсутність «вікон» в розкладі навчальних занять для студентів;
 - облік зайнятості викладачів на інших факультетах;
 - можливість взаємозамінності викладачів на випадок відсутності когось з них;
 - забезпечення відповідності числа студентів місткості аудиторії;
 - облік індивідуальних побажань викладачів-сумісників;
 - облік розташування аудиторій і відповідного часу на переміщення;
- Головні вимоги щодо забезпечення самостійної роботи студентів:
- планування не більше двох практичних занять в день;

- розподіл лекцій предмету з однаковою періодичністю.

Другорядні вимоги щодо забезпечення самостійної роботи студентів:

- рівномірний розподіл протягом тижня складних і трудомістких видів навчальної роботи;
- рівномірна завантаження студентів аудиторною роботою;
- виділення вільного дня (суботи) для виконання індивідуальних завдань [6].

Найбільш поширена схема складання розкладу зображена на рис. 1.1.

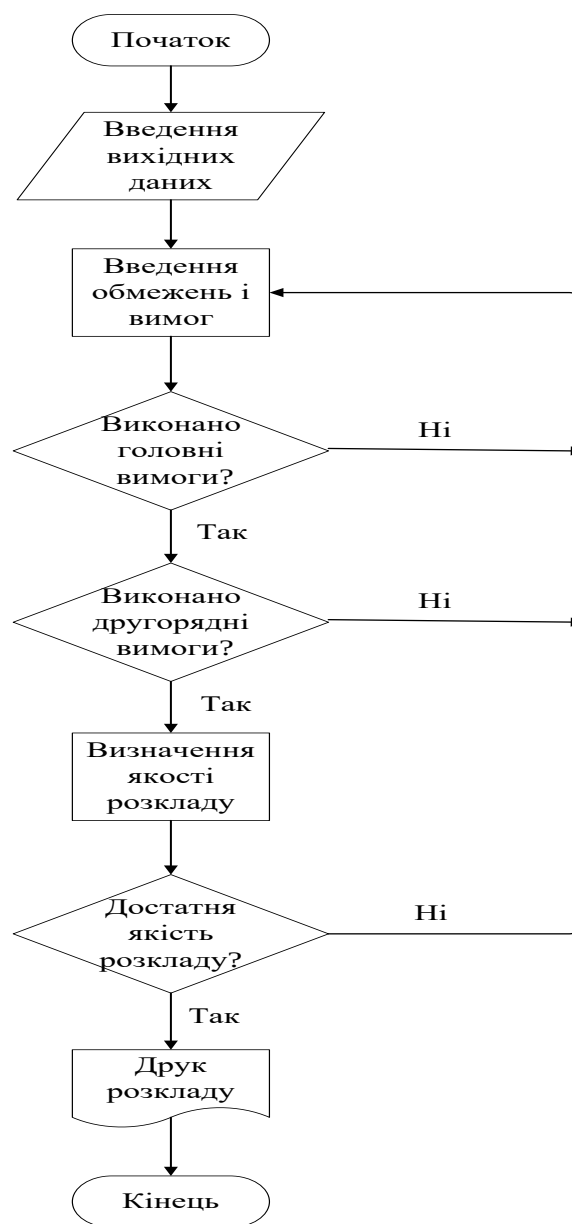


Рисунок 1.1 – Схема складання розкладу занять

Кафедри одержують навчальний план та іншу вихідну інформацію. Потім на кафедрах виконується підготовка відомостей до розкладу з урахуванням як критичних обмежень, (таких як навчальний план, аудиторний фонд), так додаткових вимог (таких як врахування побажань викладачів).

Потім ці дані беруться до роботи укладачем розкладу (диспетчером), і на їх основі складається розклад занять. Спочатку досягається виконання критичних вимог, потім другорядних, в порядку важливості. Отриманий розклад оцінюється на предмет досягнутого рівня якості. Процес цей ітеративний, в разі необхідності відбувається повернення до попереднього етапу, уточнення вимог і повторне виконання необхідних дій [5].

1.2 Огляд існуючих програмних засобів, що вирішують аналогічні завдання

Перші роботи на тему застосування комп'ютерів для складання розкладу занять з'явилися на початку 60-х років, таким чином історія вирішення цього завдання становить вже 50 років [7].

Перші розробки ставилися до великих, дуже дорогим комп'ютерів, і тому були, в основному теоретичними роботами.

За п'ятнадцять років (до 1980 року) було запропоновано багато оригінальних математичних моделей задачі складання розкладу (верхове розфарбування графів, хроматичний індекс графів і т.д.), але до реальних проблем це майже не мало відношення. Задача повністю не була вирішена, більш того, в 1976 році з'явилася робота математиків, в якій була доведена принципова труднощі вирішення цього завдання.

Другий підхід до розкладу занять було здійснено в дев'яності роки, на базі виключно швидкого зростання індустрії персональних комп'ютерів. За справу взялися не математики, а програмісти-практики. Було написано багато програм для складання розкладу, але чомусь ніяк не виходило змусити їх якісно працювати. Розклади, як і раніше, в основному, складалося вручну.

І вже на початку цього століття почався новий, поточний період вирішення завдання складання розкладу - але новому технічному і технологічному рівні.

У цих програм "третього покоління" можна відзначити такі особливості:

- врахування досвіду розробників попередніх поколінь;
- логічність і зручність інтерфейсу користувача;
- поєднання автоматичного і ручного режимів складання розкладу;
- часте використання не класичних, а евристичних методів розрахунку.

Розглянемо найбільш відомі на сьогодні в країнах СНД програмні засоби складання розкладу занять вузів.

1.2.1 Програмний продукт Ректор-ВУЗ

Ректор-ВУЗ - програма для складання розкладу занять в системі вищої професійної освіти. Основні переваги - врахування особливостей складання розкладу занять у вищих навчальних закладах, в тому числі: розподіл викладачів по кафедрах і груп за спеціальностями; планування розподілу навантажень по тижнях; звіти по розподілу навантажень кафедр і вузу за видами занять; автоматичний і ручний режими складання розкладу занять [8].

Головне вікно програми зображено на рис. 1.2.

Програма складається з чотирьох розділів: «Списки», «Навантаження», «Розклад» і «Заміни». Розділ «Списки» служить для введення, редагування і друку списків кафедр, спеціальностей, груп, дисциплін, аудиторій, викладачів і видів занять. Розділ «Навантаження» використовується для введення, редагування та друку навчальних планів по кожній спеціальності, навантажень викладачів, графіків розподілу годин по тижнях в межах семестру, звітів по завантаженню викладачів, кафедр і вузу в цілому. Розділ «Розклад» призначений для складання розкладу по групах, викладачам, аудиторіями і вузу в цілому. Розділ «Заміни» дозволяє оперувати замінами викладачів [8].

The screenshot shows the 'Ректор-ВУЗ - Primer2.rtt' application window. The top menu bar includes 'Файл', 'Настройки', 'Справка', 'Списки', 'Нагрузки', 'Расписание', and 'Замены'. The 'Расписание' (Schedule) tab is active. Below the menu, there are buttons for 'Группы', 'Преподаватели', 'Аудитории', and 'ВУЗ'. The main area displays a weekly schedule grid for the week of 31.08.09 to 06.09.09. The grid has columns for days (Понедельник, Вторник, Среда, Четверг) and rows for teachers (Волжанский О. П., Доброхотова Т. С., Зеленев И. М., Котегов Е. П., Лопаченко Е. П., Нещерет С. М., Поталов А. И., Хринов А. П.). The bottom table summarizes the schedule data.

Группа(поток)	Пар всего	Пар в неделю	+	#	%	Дисциплина	Преподаватель	Вид занятия	Аудитория	Спар	Позиц
0701	10/1	1/1				Техника экспер.	Котегов Е. П.	семинар		Нет	-
0701	8/1	1/1				Орган. химия	Котегов Е. П.	семинар		Нет	-
0701	10/1	1/1				Общая химия	Котегов Е. П.	семинар		Нет	-
0701	6/1	1/1				Техника экспер.	Котегов Е. П.	лекция		Нет	-
0701	11/1	1/1				Орган. химия	Котегов Е. П.	лекция		Нет	-
0701	17/1	1/1				Общая химия	Котегов Е. П.	лекция		Нет	-

Рисунок 1.2 – Головне вікно програми «Ректор-ВУЗ»

Розклад занять можна складати в автоматичному, ручному або комбінованому режимі; переходити від одного режиму до іншого можна в будь-який момент часу. При складанні розкладу в автоматичному режимі програма враховує всі сформульовані вимоги до розкладу. При складанні розкладу в ручному режимі програма підказує можливі варіанти розстановки уроків обраного викладача, можливі варіанти заповнення порожніх клітин в розкладі групи, стежить за кількістю місць в аудиторіях.

Готовий розклад занять однієї, всіх або деяких груп і викладачів можна зберегти у форматах Microsoft Word, Excel або HTML.

До недоліків програми можна віднести дещо застарілий інтерфейс та слабку документацію.

Ціна: ліцензія на 1 комп'ютер близько 5000 грн..

1.2.2 Програмний продукт Система Галактика Розклад навчальних занять

Система Галактика Розклад навчальних занять (РНЗ) призначена для автоматизації процесу складання розкладів навчальних занять в освітніх установах вищої та середньої професійної освіти. Основними користувачами

системи передбачаються співробітники навчально-методичних підрозділів освітніх установ, відповідальні за формування розкладу [9].

Головне вікно програми зображено на рис. 1.3.

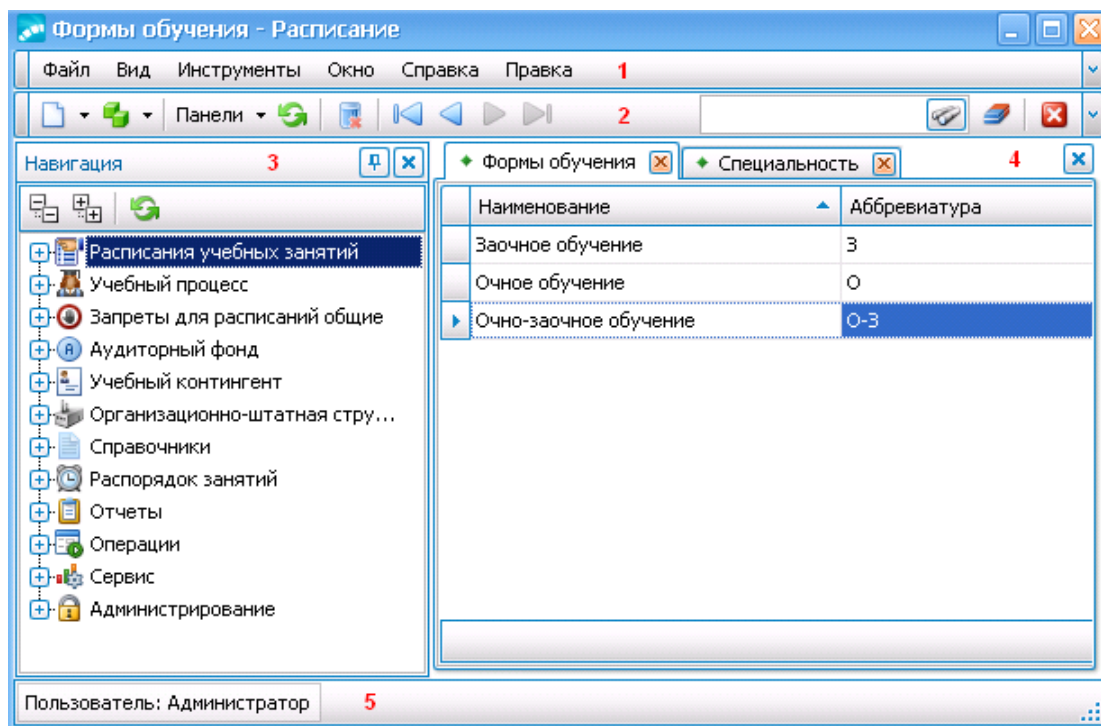


Рисунок 1.3 – Головне вікно програми Галактика РНЗ

Можливості системи Галактика: Розклад навчальних занять:

- ведення всієї довідкової інформації, необхідної для побудови розкладу (аудиторний фонд, викладачі, факультети, спеціальності і т. д.);
- імпорт даних із зовнішніх xml-файлів;
- використання excel-шаблонів для підготовки і завантаження даних;
- ведення та облік декількох розпорядків дзвінків для різних будівель, розкладів, днів тижня;
- облік змін в розкладі;
- контроль прив'язки аудиторного фонду до дисциплін, видів робіт, кафедр, факультетам;
- настройка пріоритетів використання ресурсів в розкладі (пріоритети дисциплін, викладачів, складність дисциплін і видів робіт);

- настройка і контроль тимчасової віддаленості будівель, їх пріоритетів при призначенні занять;
- підтримка різних об'єднань контингенту (підгрупа; група; потік), контроль їх перетину при складанні розкладу;
- ведення та використання графіків проходження дисциплін;
- облік послідовності вивчення дисциплін;
- створення заборон використання ресурсів в розкладі, включаючи побажання викладачів, доступність і бронювання аудиторного фонду;
- конструювання розкладу з ефективною програмною підтримкою і контролем дій оператора;
- автоматичне формування розкладу;
- настройка тривалості занять;
- експрес-оцінка загальних параметрів розкладу (змінність, тривалість навчального тижня, навчального дня);
- контроль реалізованості формованого розкладу, повноти визначеності ресурсів заняття, не допустити конфліктів їх призначення та ін .;
- розстановка пріоритетів, облік і контроль виконання понад два десятки показників якості розкладу;
- розрахунок інтегральних показників якості розкладу;
- ведення журналу оперативних змін;
- формування звітних форм [9].

Можливі кілька варіантів розгортання системи «Галактика Розклад навчальних занять»:

- система встановлюється на кожному клієнтському комп'ютері (менш переважно для роботи багатьох користувачів);
- система встановлюється на загальному сервері, а користувачі запускають її із загального каталогу (важливо зазначити, що система є СУБД-незалежною).
- система розгортається у вигляді web-служби.

Програма може мати україномовний інтерфейс.

Гарна і зрозуміла документація.

До недоліків програми можна віднести хіба що немаленьку вартість.

Ціна: базова модель від 35000 грн.

1.2.3 Програмний продукт «aSc Расписания»

«aSc Расписания» - одна з кращих комп'ютерних програм для складання розкладів уроків і занять в школах, гімназіях, ліцеях, ПТУ, ВУЗів та інших навчальних закладах. Більш ніж 150000 навчальних закладів в 170 країнах користуються цією програмою [10].

Головне вікно програми зображено на рис. 1.4.

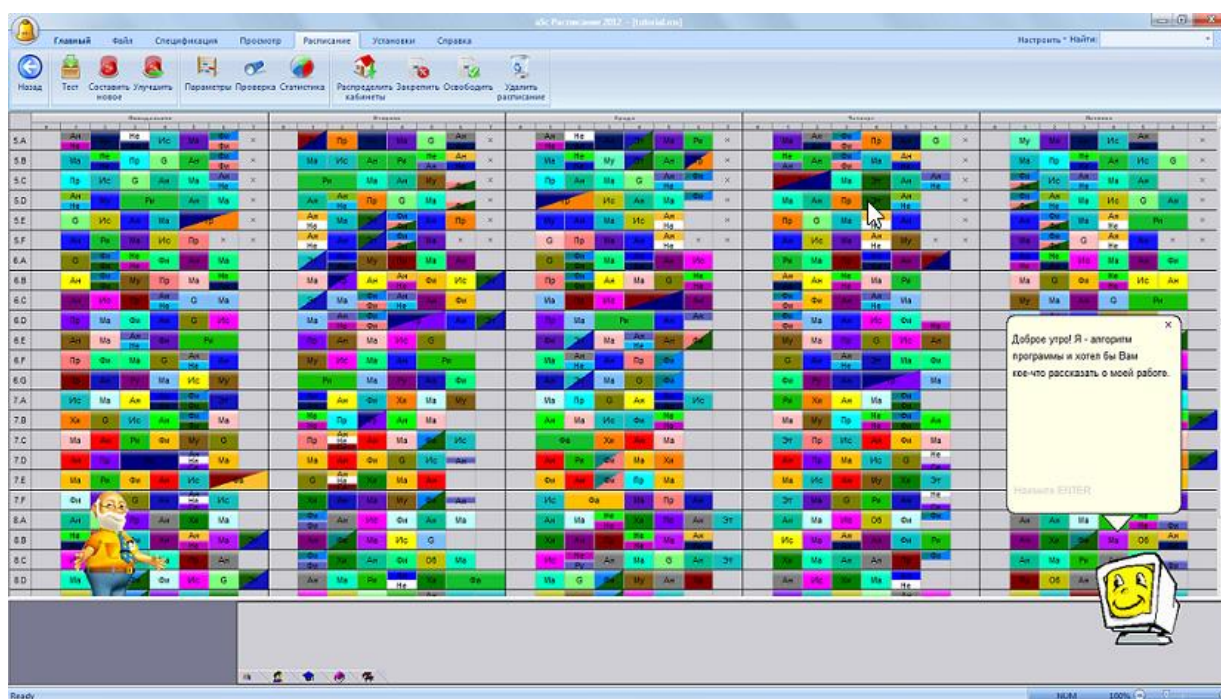


Рисунок 1.4 – Головне вікно програми «aSc Расписания»

Програма «aSc Розклади» дозволяє:

- ввести і зберегти всі основні дані: аудиторії, викладачі, кабінети, предмети, час дзвінків і ін.;
- визначити різні умови: час і розподіл занять, вільний час і вікна для викладачів, взаємозв'язок між різними дисциплінами, вибрати кабінети та ін.;

- на основі введених даних автоматично скласти оптимальний розклад; скласти загальні та індивідуальні розкладу для аудиторій, кабінетів і викладачів; вносити зміни вручну;
- складати і індивідуальні розкладу для учнів;
- експортувати дані в таблиці MS Excel і формат HTML або pdf для публікації в Інтернеті.
- замінити викладача у випадку відсутності або хвороби протягом вказаного періоду.

Програма «aSc Расписание» легка у використанні - вона працює в стандартному середовищі Microsoft Windows, має привабливий графічний інтерфейс.

Програма може мати україномовний інтерфейс.

Програма має дуже хорошу, докладну документацію, проілюстровану численними прикладами.

До недоліків можна віднести націленість програми все ж, в першу чергу, на шкільне, а не вузівська розклад [10].

Ну і європейська ціна (програма зі Словаччини).

Ціна: «aSc Расписание Премиум» - 14200 грн..

1.2.4 Порівняльна таблиця існуючих програм

Основні показники можливостей розглянутих вище програм для складання розкладу наведені в табл. 1.1.

Таблиця 1.1. Основні показники сучасних програмних продуктів з автоматизації складання розкладу навчальних занять

	«Ректор-ВУЗ»	«Галактика РНЗ»	aSc Расписание
Розробник	Павел Смекалов	Коропорація «Галаткита»	Applied Software Consultans
Операційна система	Windows (XP, Vista, 7)	Windows (XP, 7, 8, 10)	Windows (98, ME, NT, 2000, XP, 7, 8, 10), Apple iOS
Додаткове ПЗ	-	NET Framework v.4, СУБД (Oracle, Pervasive)	NET Framework v.4 і вище
Варіанти розгортання	клієнтський комп'ютер	клієнт, сервер, web- служба	клієнтський комп'ютер
Базовий період розкладу	тиждень	тиждень	тиждень
Вартість	5000 грн.	Базова модель від 35000 грн.	стандарт – 4500; преміум – 14200; PRO – 56000 грн.
Розподіл груп на підгрупи	2 підгрупи	необмежено	до 5 підгруп
Складання розкладу	автоматичне, ручне, комбіноване	автоматичне, ручне коригування	автоматичне, ручне коригування
Одночасна робота кількох операторів	-	+	-
Додаткові модулі, що не входять у базовий комплект	-	-	модуль «Заміни»
Документація	задовільна	добра	добра

Всі розглянуті програмні комплекси складають розклад в автоматичному режимі, з урахуванням введених обмежень, мають зручний інтерфейс, можливість коригування складеного розкладу при необхідності.

З таблиці видно, що безумовним лідером є програмний продукт "Галактика Руз". Одним з найважливіших переваг цієї програми є можливість одночасної роботи в мережі декількох диспетчерів. Однак, ця можливість потрібна лише великим вузам. Для невеликих вузів цілком достатнім інструментом для складання розкладу можуть бути і інші розглянуті програми.

Як орієнтир для нашої розробки також буде доцільно взяти програму «Галактика Руз». Зрозуміло, що за складністю результат кваліфікаційної роботи бакалавра і успішний комерційний програмний продукт не можуть бути порівнянними, але загальна концепція і структура цієї системи є найбільш підходящим зразком.

1.3 Постановка завдання до роботи

Розробка програмного забезпечення для складання розкладу занять - це складна алгоритмічна і програмна задача, розробка складної програми такого роду для великого вузу в рамках кваліфікаційної роботи бакалавра неможлива, тому визначимо точніше нашу задачу і її обмеження.

Складання розкладів зазвичай (принаймні, в невеликих вузах) виконується в кожному деканаті (факультеті) окремо. Візьмемо в якості посильної задачі розробку програми складання розкладу в деканаті невеликого факультету.

В такому випадку діаграма використання нашої підсистеми набере вигляду, показаного на рис. 1.5.

Вихідні дані для розкладу вводять навчальний відділ (навчальний план), кафедра (списки предметів і викладачів), деканат (списки груп і аудиторій).

Диспетчер (посадова особа з деканату) вводить вимоги до розкладу і виконує автоматизований розрахунок розкладу, можливо, декілька разів.

Викладачі та студенти мають доступ по читанню до складеного розкладу, кожен в своєму розрізі.

Створювана програма буде настільною програмою Windows, що розташована на клієнтських комп'ютерах і має доступ до центральної бази даних на сервері даних.

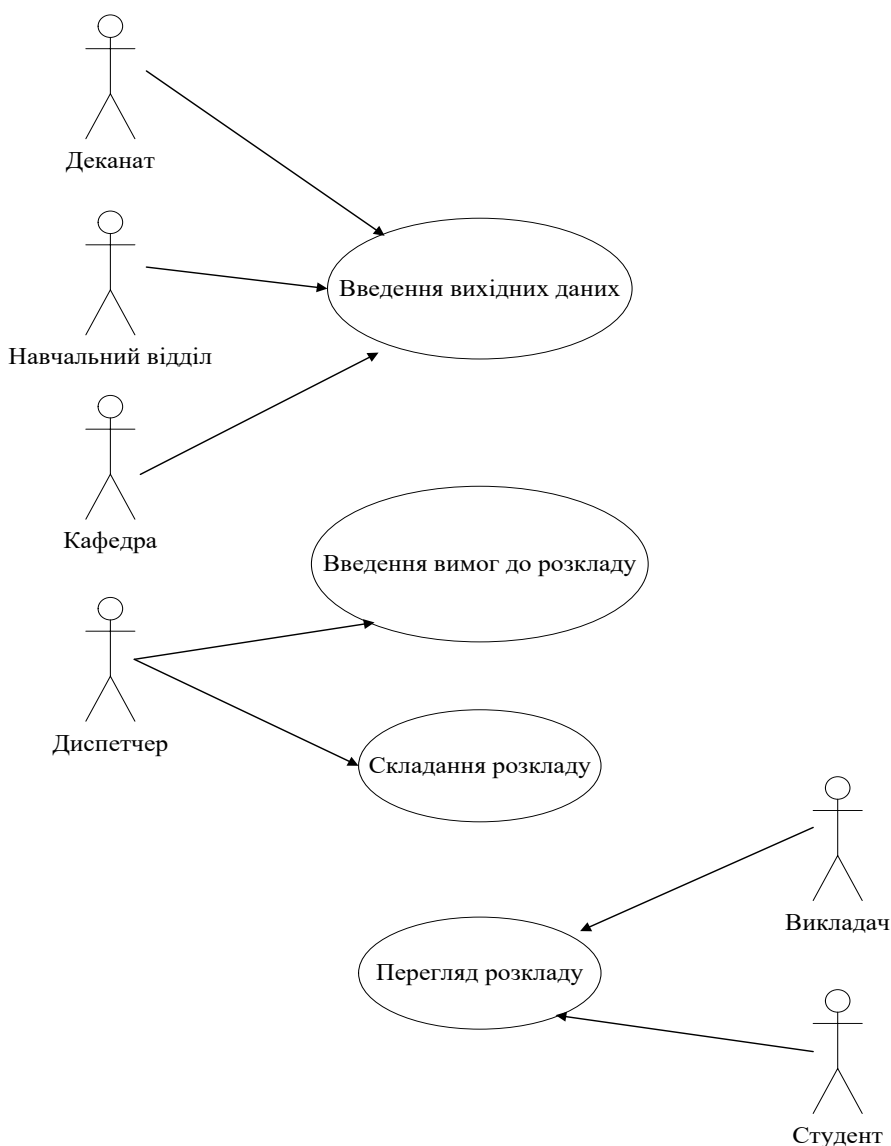


Рисунок 1.5 – Діаграма використання системи складання розкладу

Розглянемо оброблювану інформацію і завдання програми докладніше. Функції, що розробляється можна розбити на наступні блоки функцій:

- ведення бази даних;

- автоматичне складання розкладу;
- коригування розкладу;
- висновок розкладу.

Ведення бази даних включає в себе:

- ведення довідників (таблиць даних з довготривалою інформацією);
- ведення даних по навантаженню;
- ведення обмежень і вимог.

Ведення довідників включає в себе ведення наступних довідників:

- будівлі;
- аудиторії;
- групи;
- потоки студентів;
- факультети;
- кафедри;
- викладачі;
- форми навчання;
- спеціальності;
- семестри;
- курси;
- види робіт;
- види навчальної діяльності;
- типи аудиторій;
- цикли дисциплін;
- дисципліни;
- сітки пар.

Дані по навантаженню представляють собою навчальне навантаження, що підлягає розподілу по заняттях розкладу. Параметрами одиниці навантаження є:

- дисципліна;

- вид діяльності;
- викладач;
- контингент (потік, група, підгрупа);
- період навантаження;
- обсяг навантаження;
- аудиторія (при необхідності завдання певної аудиторії).

Ведення обмежень і вимог включає:

- обмеження по парам, аудиторіям і інтервалах часу. Обмеження можуть бути разові і постійні;
- дотримання послідовності вивчення дисциплін;
- відсутність вікон у груп;
- мінімізація вікон у викладачів;
- наявність вільних днів у викладачів.

Автоматичне складання розкладу включає:

- перекодування даних БД у внутрішнє представлення;
- налаштування параметрів генетичного алгоритму та фітнес-функції;
- запуск генетичного алгоритму;
- перекодування внутрішнього уявлення розкладу в дані БД.

Коригування розкладу включає:

- зміна занять в процесі навчального процесу за потребою.

Виведення розкладу включає:

- виведення розкладу для груп;
- виведення розкладу для викладачів.

Вимоги до інтерфейсу програми стандартні: зручність введення і коректування даних, зрозумілі повідомлення про помилки, розподіл доступу і повноважень користувачів [11] - [14].

Найбільш складним моментом розробки програми буде розробка функції автоматичного складання розкладу. Альтернативний потік подій:

Готовий програмний продукт може бути використаний без додаткових налагоджень і змін. Кінцевий користувач погоджує склад предметів з упорядником. Укладач вносить корективи до списків і складає або змінює розклад. Користувач отримує розклад у використання.

Післямови:

- зміни, внесені упорядником до складу розкладу, не підлягають відновленню;

- зміни програміста в програмі зберігаються на її ж розсуд;

- зміни в програмі впливають на роботу програми в подальшому;

Далі будується структурно-логічна схема системи у вигляді діаграми класів (рис. 1.6). Ця діаграма служить для представлення статичної структури моделі системи, різних взаємозв'язків між окремими сутностями предметної області, такими, як об'єкти і підсистеми, а також опису їх внутрішньої структури і типів відносин.

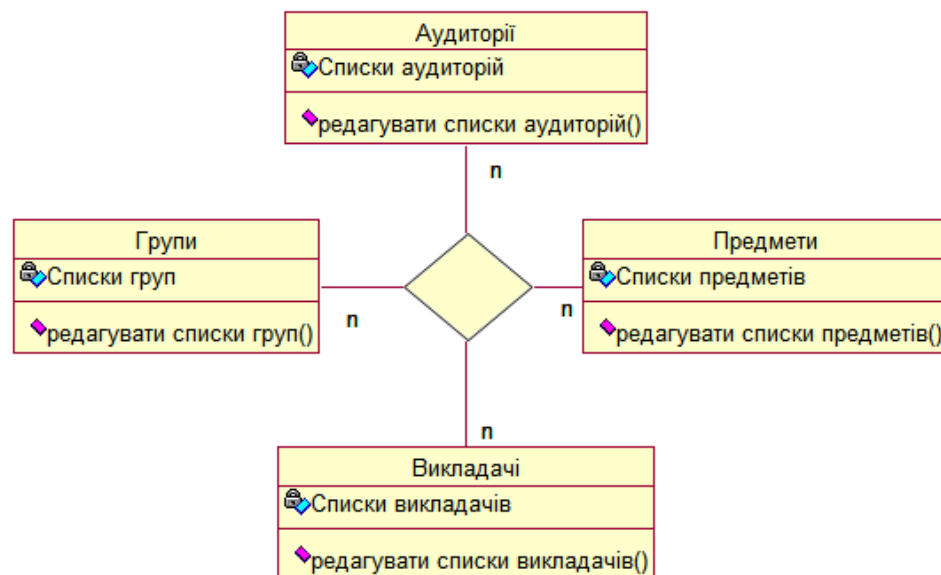


Рисунок 1.6 - Діаграма класів, клас «Списки»

Укладач та викладач узгодять навантаження, а саме кількість годин і предмети. Обидва компоненти відносяться до класу «користувач», при цьому володіють деякими унікальними здібностями. Як перших так і других може

бути більше одного з кожної сторони. Користувач являє собою клас, який використовує готовий розклад. Користувачів може бути будь-яка кількість. Розклад складається під час роботи упорядника з системою і редагування бази даних, ним же і перевіряється. Укладачів може бути декілька, а система і база даних усього в одному екземплярі. База даних взаємодіє з системою, яку контролює не менше одного адміністратора.

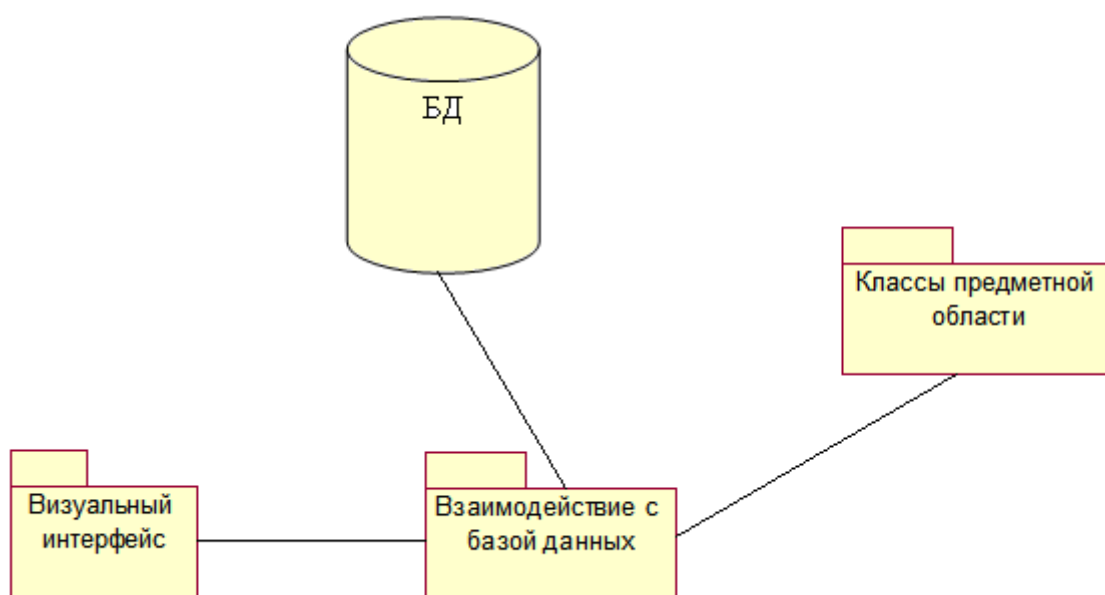


Рисунок 1.7- Діаграма пакетів

1.4 Висновок

У розділі були розглянуті основні проблеми, що виникають при складанні розкладу навчальних занять у вузі і можливі варіанти їх вирішення. Також були розглянуті провідні програмні продукти на ринку складення занять та проаналізовано їх можливості. На підставі розглянутого була визначені цілі випускної кваліфікаційної роботи бакалавра і виконана постановка задачі.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЄКТУЄТЬСЯ

2.1 Вибір технології розробки

В якості середовища розробки були обрані Microsoft Windows 7 а також фреймворк QT4 з використанням мови C++.

Всі вони відповідають політиці відкритого коду, що в свою чергу дає можливість усім бажаючим розвивати сподобався проєкт. Однак даний продукт розповсюджується під ліцензією GNU Lesser General Public License (GNU LGPL) [15].

Використання фреймворку QT4 дуже сильно розширює функціонал, додатки стають кроссплатформним; тривалі і дорогі розробки можна замінити готовими рішеннями.

«Відкрита» ліцензія не вимагає, щоб програмне забезпечення (ПЗ) надавалося безкоштовно. Тим не менш, багато хто з найбільш успішних проєктів відкритого ПЗ - безкоштовні. Поки що політика відкритого коду і безкоштовне програмне забезпечення поступаються популярністю платних аналогів із за того, що не завжди виявляються зручними для більшості користувачів. Але тим не менш платні продукти поступово витісняються з ринку програмного забезпечення.

Вихідні коди відкритих програм випускаються або як громадське надбання, чи на умовах «вільних» ліцензій - як, наприклад, GNU General Public License або BSD License. Вільна ліцензія дозволяє використовувати вихідний код програми для своїх потреб з мінімальними обмеженнями. Таким обмеженням може бути вимога посилатися на попередніх творців або вимога зберігати властивість відкритості при подальшому поширенні тієї ж самої або модифікованої відкритої програми. У деяких випадках (наприклад, Apache або FreeBSD) ці обмеження дуже малі, в інших (наприклад, GNU General Public License) досить поширювати ПЗ разом з вихідним кодом і текстом ліцензії, не змінюючи її [15] - [18].

Java – об'єктно-орієнтована мова програмування загального призначення. Відноситься к мовам з сильною типізацією. Творці реалізували принцип WORA: write once, run anywhere або «пиши один раз, запускай де завгодно». Це означає, що написаний на Java додаток можна запустити на будь-якій платформі, якщо на ній встановлене середовище виконання Java (JRE, Java Runtime Environment) [19].

На мові Java написано багато веб-додатків. Популярні фреймворки, в тому числі Spring та Stuts, використовуються для створення різних додатків в інтернеті: від комерційних проєктів до великих порталів, від освітніх платформ до урядових ресурсів [19].

Переваги мови Java:

- незалежність від платформи – написаний на Java додаток можна запустити на будь-якій платформі, якщо на ній встановлене середовище виконання Java JRE;
- легкість освоєння – простота синтаксису і раціональність, інтуїтивна зрозумілість коду;
- повна об'єктна орієнтованість – мова Java повністю об'єктно-орієнтована: всі сутності це об'єкти, а функції, в основному це методи класів;
- висока надійність – мова Java створена таким чином, щоб розробник коду не зміг створити проблемний програмний продукт. Для цього в Java є багато різних внутрішніх механізмів.

Недоліки мови Java:

- низька продуктивність – з-за того, що Java має багато різних внутрішніх механізмів, вона має низьку продуктивність;
- відсутність нативного дизайну - Java-інструменту для створення нативного дизайну немає;
- багатослівний і складний код – коли коду стає забагато, то він становиться складним для читання та розуміння [20].

C # – відноситься до сім'ї мов з C-подібним синтаксисом, з них його синтаксис найбільш близький до C ++ і Java. Мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів (в тому числі операторів явного і неявного приведення типу), делегати, атрибути, події, властивості, узагальнені типи і методи, ітератори, анонімні функції з підтримкою замикань, LINQ, виключення, коментарі у форматі XML [20].

Переваги мови C #:

- підтримка корпорації Майкрософт – на відміну від Java, якій не пішов на користь перехід у власність Oracle, C # добре розвивається завдяки зусиллям Microsoft;
- багато синтаксичного цукру. Синтаксичний цукор - це такі конструкції, які створені для полегшення написання і розуміння коду (особливо якщо це код іншого програміста) і не грають ролі при компіляції;
- середній поріг входження. Синтаксис схожий на C, C ++ або Java полегшує перехід для інших програмістів. Для новачків це також один з найперспективніших мов для вивчення [20].

Недоліки мови C #:

- орієнтованість, в основному, тільки на .NET (на Windows платформу);
- безкоштовність тільки для невеликих компаній, учнів і програмістів-одинаків. Для великих команд покупка ліцензій обійдеться недешево;
- наявність оператора go to [20];

Проаналізувавши роботи [18]-[21], було зроблено порівняльну характеристику мов програмування для вирішення поставленого завдання, яка представлена у табл. 2.1.

Звернемося тепер до СУБД. Найбільш популярні СУБД і їх характеристики наведені в табл. 2.2.

Таблиця 2.1 – Порівняння технологій розробки

Параметр			
Мова програмування	C++	C #	Java
Проблеми з безпекою	Ні	Ні	Можливі
Кросплатформеність	Так	Так	Так
Легкість освоєння	+	++	++
Рівень можливостей	+	++	++
Можна використовувати для frontend (клієнтська частина)	+	++	++
Можна використовувати для backend (серверна частина)	++	+	+
Ліцензія	—	.NET Core CLR: MIT / X11	GNU GPL
Незалежність від платформи	+	—	+
Наявність додаткових бібліотек та пакетів	+	+	—
Можливість паралельного програмування	+	+	+
Наявність взаємодії з іншими мовами програмування	+	+	+

Таблиця 2.2. Найбільш часто використовувані безкоштовні СУБД і СУБД, які мають безкоштовні версії

Компанія-виробник	Переваги	Недоліки
1	2	3
PostgreSQL	функціональна і безкоштовна СУБД з відкритим кодом, хороша підтримка з боку співтовариства розробників	місцями низька продуктивність при обробці великих обсягів інформації, мала популярність продукту

Продовження таблиці 2.3

1	2	3
Oracle	швидка обробка даних, легко інтегрувати в різні онлайн і оффлайн додатки, можливість створити велике сховище даних з мінімальними зусиллями. Oracle може впоратися з будь-якими масштабами інформації і кількістю користувачів, тому рівень масштабованості є одним з найбільш недосяжних для багатьох інших типів СУБД, великий вибір oracle інструментів	велика ціна, потужне обладнання для оптимальної роботи Oracle, високоспеціалізований персонал для підтримки бази даних, складності у вивченні функціоналу
MySQL	безкоштовна СУБД, має відкритий код, займає мало місця, легке адміністрування mysql з набором всього необхідного функціоналу, велика спільнота розробників, працює на безлічі платформ	незахищеність від втрати даних, підтримка тільки малих баз даних
SQL Server	величезний пакет корисних інструментів, які легко використовувати; стабільна робота з sql server базами даних; менші витрати на адміністрування sql server	працює тільки на платформах Microsoft, корпоративна версія може коштувати надто дорого
DB2	працює на безлічі платформ, стабільна робота бази даних з величезною кількістю функціональних особливостей, відмінна підтримка з боку IBM, діалект мови SQL більш гнучкий і має більшу кількість переваг, ніж SQL Server.	продуктивність не така висока як у Oracle, корпоративна версія може досягати завищених цін.

2.2 Структурна схема програмного продукту

Структурна схема програми зображена на (рис.2.1).

Створювана програма буде мати наступні підсистеми (модулі):

- модулі введення і коригування вихідних даних;
- модуль введення і коригування вимог до розкладу;
- модуль автоматичного складання розкладу;
- модулі відображення отриманого розкладу;
- модуль виведення на друк отриманого розкладу;
- модуль роботи з базою даних.

Користувач через меню головної (керуючої) форми викликає програмні модулі, необхідні для ведення даних та виконання інших функцій програми

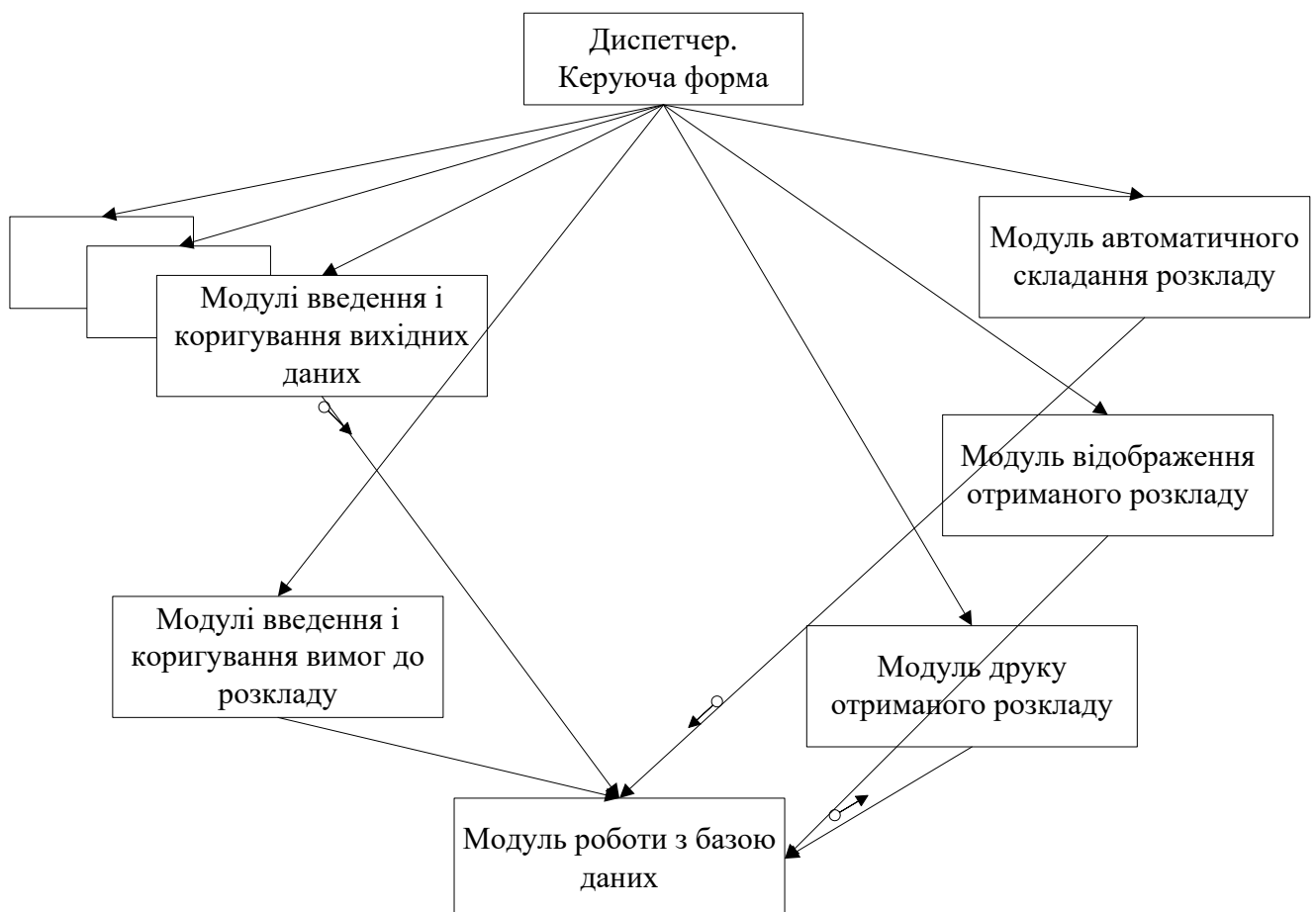


Рисунок 2.1 – Структурна схема програми

Модулі введення і коригування вихідних даних через модуль роботи з базою даних здійснюють обробку відповідних таблиць бази даних.

Аналогічно здійснює свої функції модуль введення і коригування вимог до розкладу.

Модуль автоматичного складання розкладу через модуль роботи з базою даних отримує необхідні дані, виконує складання розкладу і заносить сформований розклад у базу даних.

Модулі відображення та друку відповідно отримують необхідні дані і виводять їх у потрібному форматі на екран та принтер.

2.3 Висновок

У розділі були розглянуті сучасні інструменти для розробки програмного забезпечення і обрані: мова програмування, середовище розробки і СУБД для розробки автоматизованої системи складання розкладу.

В якості мови програмування проєкту обрано C++ - мову програмування, що має великий набір конструкцій і динамічно розвивається.

Була спроектована структурна схема програми, що розроблюється та складено опис модулів цієї програми.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

Для створення проєкту «Sheasy» було прийнято рішення про застосування об'єктно-орієнтованого підходу до написання програми. Це виразність, простота доповнення та розширення базових можливостей, модульність, зниження ймовірності випадкових логічних помилок. Основні переваги ООП. Всі об'єкти охопленої предметної області представлені у вигляді класів. Центральне місце у функціонуванні програми займають два класи, які є ядром системи: клас «база даних» (CDataBase) і клас «розклад» (CSchedule).

3.1 Клас «списки»

CDataBase - завдання цього класу полягає у наданні універсального для інших модулів програми інтерфейсу доступу до даних, незалежно від способу їх зберігання на жорсткому диску. Таким чином, клас CdataBase можна розглядати як «драйвер», за допомогою якого програма може використовувати як прості сховища інформації (CSV, XML), так і ресурси різних СУБД (SQLite, MySQL), звертаючись до них за допомогою одних і тих же функцій.

Клас CDataBase оперує списками примірників чотирьох різних підкласів: CTeachers (викладачі, вчителі), CSubjects (предмети, уроки), CGroups (групи, класи), CRooms (аудиторії): його ідентифікаційні дані і навантаження з предметів.— клас, що містить дані про кожного викладача CTeachers – Функції цього класу представлені в табл. 3.1;

Таблиця 3.1 - Функції класу CTeacher

Назва функції	Короткий опис
GetTitle();	Повертає ФІО викладача
GetId();	Повертає ідентифікатор викладача
GetSubjects();	Повертає навантаження викладача у вигляді списку предметів та годин по кожному предмету
SetSubject(subject);	Встановлює флаг зайнятості викладача на вказаному предметі
ReleaseSubject(subject);	Прибирає флаг зайнятості викладача з вказаного предмету

CGroups – клас, що містить дані про групи – назви, чисельність і навчальне навантаження. Функції цього класу представлені в табл. 3.2;

Таблиця 3.2 – Функції класу CGroup

Назва функції	Короткий опис
GetId();	Повертає ідентифікатор групи
GetTitle();	Повертає назву групи
GetStudentsCount();	Повертає кількість студентів у групі
GetLessons();	Повертає учбовий план групи у вигляді списку предметів та кількості годин
SetLesson(subject);	Встановлює флаг зайнятості групи на вказаному предметі
ReleaseLesson(subject);	Прибирає флаг зайнятості зі вказаного предмету

CRooms – клас, що містить дані про аудиторному фонді навчального закладу - ідентифікатори аудиторій, їх місткість і предмети, для проведення яких вони пристосовані. Функції цього класу представлені в табл. 3.3;

Таблиця 3.3 – Функції класу CRoom

Назва функції	Короткий опис
GetId()	Повертає ідентифікатор аудиторії
GetTitle();	Повертає символічну назву аудиторії
GetStudentsCount();	Повертає кількість місць у аудиторії
GetGroupsCount();	Повертає кількість груп, на яку розрахована аудиторія
GetSubjects();	Повертає список предметів, для аудиторії

CSubjects клас, що містить опис предметів, що викладаються в навчальному закладі – назви, кількість викладачів, які повинні вести цей предмет. Функції цього класу представлені в табл. 3.4;

Таблиця 3.4 – Функції класу CSubject

Назва функції	Короткий опис
GetId();	Повертає ідентифікатор об'єкта класу
GetTitle();	Повертає назву предмету
GetTeachersCount();	Повертає кількість викладачів необхідне для проведення занять
GetGroupsCount():	Повертає максимальну кількість груп, для яких може одночасно проводитись пара предмету

Функції, що представляють програмний інтерфейс класу CDataBase, наведені в табл. 3.5.

Таблиця 3.5 – Функції класу CDataBase

Назва функції	Короткий опис
FetchIds(DBEssenceType type)	Повертає список ідентифікаторів об'єктів вказаного типу
FetchStringField(qint32 id, DBEssenceType essence, DBEssenceFields field)	Повертає значення певного поля об'єкта у вигляді строкової змінної
GetTeacher(qint32 id) GetSubject(qint32 id) GetSubject(qint32 id) GetSubject(qint32 id)	Ці функції повертають вказівник на об'єкт певного типу у пам'яті по його ідентифікатору
GetTeachers(); GetSubjects(); GetGroups(); GetRooms();	Ці функції повертають списки вказівників на об'єкти у пам'яті певного типу

3.2 Клас «розклад»

Другим базовим класом системи є клас CSchedule (розклад). Його можна представити у вигляді трирівневої ієрархічної структури:

- CSchedule – узагальнюючий клас, містить кілька об'єктів класу CScheduleDay (за кількістю унікальних навчальних днів);
- CScheduleDay – клас, що містить дані про один навчальний день, містить кілька посилань на масиви об'єктів класу CScheduleLesson (по кількості уроків / пар на день). Функції цього класу представлені в табл. 3.6;
- CScheduleLesson – клас, що описує одну конкретну пару, що об'єднує аудиторію, предмет, групи і викладачів, що беруть участь в проведенні заняття. Функції цього класу представлені в табл. 3.7.

Таким чином, клас «розклад» являє собою складну структуру, здатну зберігати розклад занять навчального закладу, відстежуючи при цьому цілісність структури і прояв конфліктів, пов'язаних зі спробами багаторазового використання одного ресурсу.

Таблиця 3.6 – Функції класу CScheduleDay

Назва функції	Короткий опис
AddLesson(num);	Додати нові заняття зі вказаним номером
RemoveLesson(lesson)	Видалити заняття
Lessons(num)	Отримати список усіх занять, які проводяться на певній парі
TrySubject(num, obj); TryTeacher(num, obj); TryGroup(num, obj); TryRoom(num, obj);	Перевірка зайнятості відповідного ресурсу на вказаній парі

Таблиця 3.7 – Функції класу CScheduleLesson

Назва функції	Короткий опис
1	2
SetSubject(obj); SetTeacher(obj); SetGroup(obj); SetRoom(room);	Зробити вказаний об'єкт учасником навчального закладу

Продовження таблиці 3.7

ReleaseSubject(obj); ReleaseTeacher(obj); ReleaseGroup(obj); ReleaseRoom(room);	Звільнити вказаний об'єкт від занять
TrySubject(obj); TryTeacher(obj); TryGroup(obj); TryRoom(obj);	Перевірити чи може вказаний об'єкт бути учасником занять

3.3 Висновки за розділом 3

У цьому розділі було розглянуто склад таблиць бази даних. Також було розглянуто структуру та взаємодію класів програми.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення й умови використання

Програма призначена для швидкого та легкого складання розкладу занять у вищих навчальних закладах. Для її функціонування необхідно встановити продукт на комп'ютер, заповнити базу даних по групах, викладачах, студентах, аудиторіях. Для ефективної роботи з системою потрібен спеціаліст з області інформаційних технологій.

У проєкті було досягнуто можливість ручного складання розкладу, автоматичне складання з подальшим ручним редагуванням, зміна даних у базі даних, розраховувати навантаження, роздрукувати розклад, оптимізувати готовий розклад. Програмний продукт має інтуїтивно зрозумілий дружній інтерфейс, що робить її легкою в експлуатації..

4.2 Вхідні і вихідні дані

Вхідними даними є списки груп, аудиторій, викладачів та предметів, що містяться у окремих csv файлах, які мають табличну структуру. Це зроблено для того щоб не плутати користувача великою кількістю форм. Заповнювати данні можна за допомогою MSExcel або будь який інший аналог для Unix систем, що представляє собою дуже зручний інструмент для роботи з csv файлами.

Вихідними даними є ніщо інше як готовий розклад у зручному табличному вигляді. Також для зручності було зроблено можливість пересувати та відображати панелі таким чином, щоб можна було отримати максимум робочого простору.

4.3 Повідомлення

Найбільш цінною можливістю проєкту «Sheasy» є автоматичне складання розкладу навчального закладу за допомогою різних алгоритмів. Описані вище класи є ні що інше, як інтерфейс програміста, основа для створення таких алгоритмів. У цьому розділі наведемо кілька основних вимог до розробників модулів автоматичної обробки даних.

Джерелом даних для обробки служить клас `CDataBase` і його інтерфейсні функції. За допомогою них можна легко отримати списки необхідних для обробки об'єктів і їх властивості. Слід зазначити, що клас `CDataBase`, в даному контексті, може і повинен використовуватися в режимі «тільки для читання».

Результат своєї роботи модуль автоматичної обробки повинен зберегти всередині структури `CSchedule`. Клас `CSchedule` має внутрішні механізми перевірки і запобігання введення суперечливих даних. Його можливості можуть бути використані на будь-якому етапі виконання алгоритму. При цьому, не дивлячись на принципову можливість, алгоритм автоматичної обробки не повинен обходити заборони та обмеження, що виставляються модулем `CSchedule`.

Після того, як алгоритм завершив свою роботу, інтерфейсна частина програми відображає на екрані по запиту користувача таблиці, побудовані на основі внутрішньої структури класу `CSchedule`.

Під час роботи розробленого ПП дії користувача можуть призводити до помилок. ПП обробляє всі помилки, а також генерує повідомлення про них, для того, щоб користувач мав можливість їх виправити.

При введенні і редагуванні даних використовується роз'єднаний режим роботи з базою даних. Збереження змін повинен здійснювати користувач за допомогою кнопки "Зберегти". Якщо ж користувач виходить з форми редагування без збереження, йому видається нагадування – «Зберегти зміни».

Так само, при помилках користувача видається повідомлення про помилку. Приклад такого повідомлення (при неправильних параметрах з'єднання з БД) – «Підключення не встановлено тому, що кінцевий комп'ютер відкинув запит на підключення».

Server error – «Помилка серверу».

Server started on port... – Сервер запущений під портом ...

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення програми й умови виконання програмного продукту

Програма призначена для швидкого та легкого складання розкладу занять у вищих навчальних закладах. Для її функціонування необхідно встановити продукт на комп'ютер, заповнити базу даних по групах, викладачах, студентах, аудиторіях.

ПП повинен відповідати наступним мінімальним системним вимогам:

- операційна система Microsoft Windows 7;
- 64-розрядна версія операційної системи;
- центральний процесор з тактовою частотою 1,8 ГГц.
- 1 ГБ оперативної пам'яті;
- 1 ГБ вільного місця на жорсткому диску.

Керування програмою здійснюється за допомогою стандартних засобів керування – клавіатури та миші. Рекомендується використовувати для роботи з програмою монітор з розрішенням 1440 x 900 і більше.

5.2 Виконання програмного продукту

Для запуску програми оберіть файл Sheasy.exe та запустіть його.

Вам буде представлено головне вікно програми (рис 5.1), де стовпчиками буде представлено викладачів або групи в залежності від того яку вкладку обрати, а строками сітки будуть дні тижня.

Головне вікно містить строку меню у лівому верхньому кутку вікна, чотири рухомих фрейми з даними про групи, аудиторії, викладачів та предмети. Їх можна сховати або відобразити через меню (рис. 5.2) для зручності роботи, також їх можна перемістити.

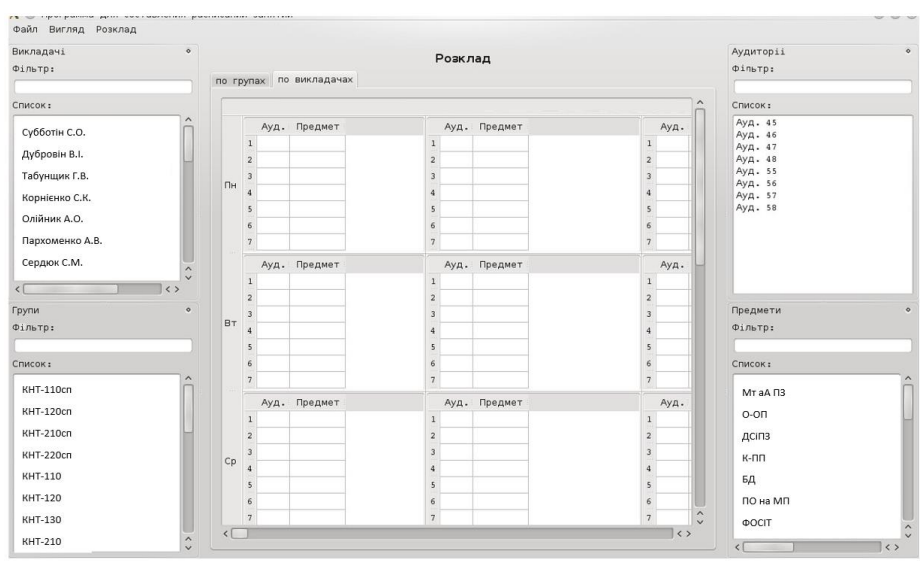


Рисунок 5.1 – Початок роботи

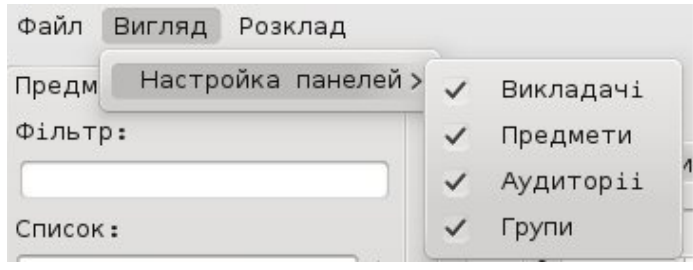


Рисунок 5.2 – меню для роботи з рухомими панелями

Розклад можна скласти вручну (рис. 5.3) за допомогою контекстного меню та орієнтуватись по групах або по викладачам, переключаючись за допомогою вкладок.

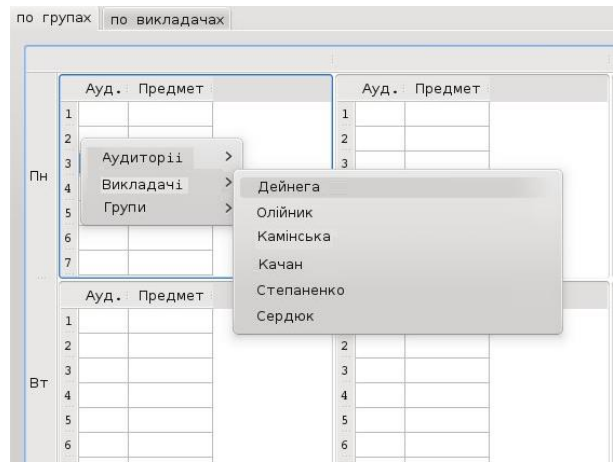


Рисунок 5.3 – Складання розкладу за допомогою контекстного меню

Також можна скласти розклад автоматичними засобами, які можна модифікувати та додавати у програму (рис. 5.4). На даний момент є два алгоритми складання розкладів – це метод прямого сортування та генетичний метод.

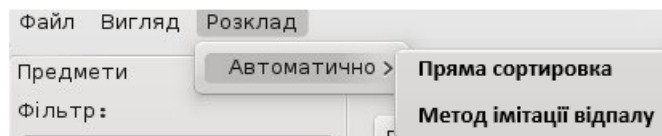


Рисунок 5.4 – вибір алгоритму

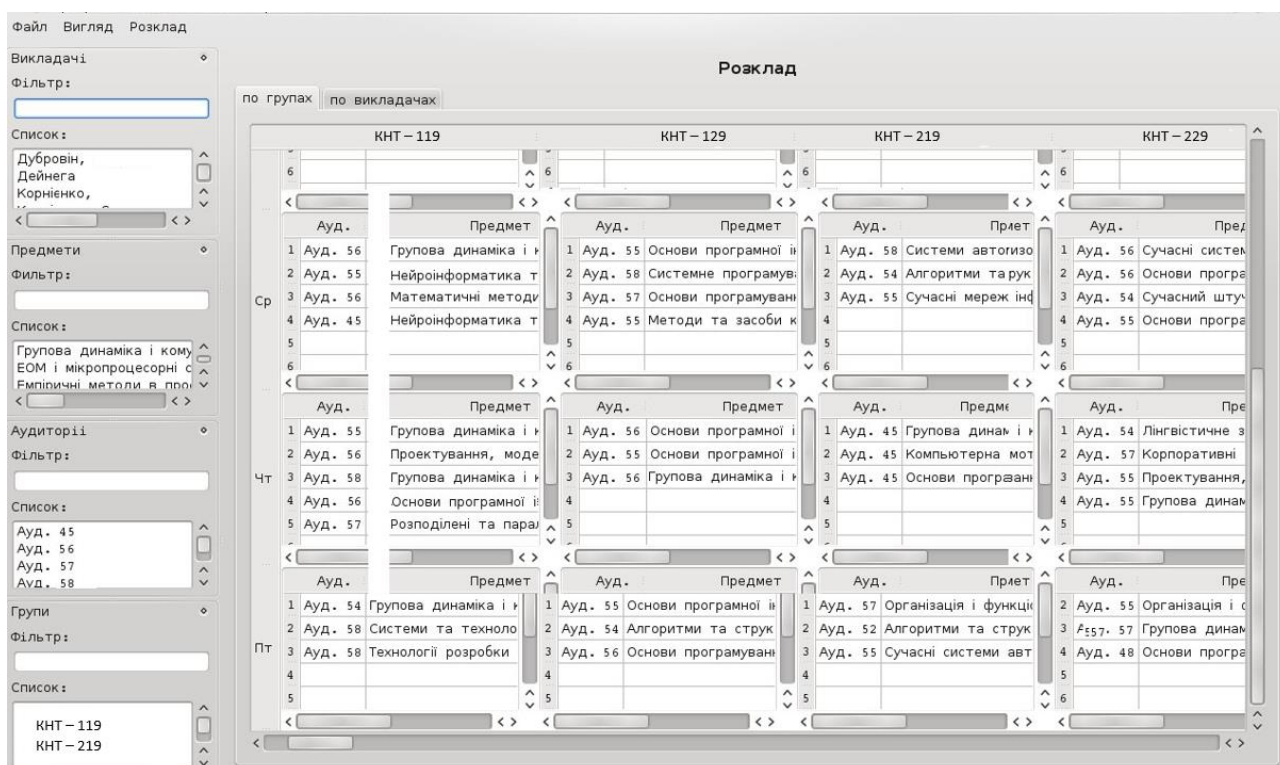


Рисунок 5.5 – Результати виконання програми

5.3 Висновки за розділом 5

У цьому розділі було приведені керівництво оператора. Дано детальний опис процесу роботи з програмою від її запуску до отримання результату.

Програма має простий та досить зручний інтерфейс в частині введення та редагування вхідних даних та ручного редагування розкладу.

ВИСНОВКИ

В процесі виконання кваліфікаційної роботи бакалавра було розроблено автоматизовану систему складання електронного розкладу.

Вихідними даними для розкладу є навчальний план, списки груп студентів, списки аудиторій, списки викладачів з предметів, календарні вимоги до розкладу. З математичної точки зору завдання складання розкладу відноситься до завдань цілочисельного програмування, складність яких зростає експоненціально з ростом числа змінних, що беруть участь в завданні (так звані NP-складні завдання). Для вирішення таких завдань розроблено багато точних (таких, що відносяться до класичної математики) методів і алгоритмів.

Програма має дружній інтерфейс, зручний у використанні, який не потребує багато часу на опанування принципів роботи з ним.

У кваліфікаційній роботі бакалавра була проаналізована предметна область, розроблена архітектура системи та на основі досліджень обране середовище розробки і функціонування програми.

В якості середовища розробки були обрані Windows 7, а також фреймворк QT4 з використанням мови C++.

Всі вони відповідають політиці відкритого коду, що в свою чергу дає можливість усім бажаючим розвивати сподобався проєкт. Однак даний продукт розповсюджується під ліцензією GNU Lesser General Public License (GNU LGPL).

Використання фреймворку QT4 дуже сильно розширює функціонал, додатки стають кроссплатформним; тривалі і дорогі розробки можна замінити готовими рішеннями.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Береговых Ю.В. Алгоритм составления расписания занятий / Ю.В. Береговых, Б.А. Васильев, Н.А. Володин // Штучний інтелект. — 2009. — № 2. — С. 50-56.
2. Сидорин А.Б. Методы автоматизации составления расписания занятий. Часть 1. Классические методы / А.Б. Сидорин, Л.В. Ликучева, А.М. Дворянкин // Известия ВГТУ. — 2009. — № 7. — С. 116-120.
3. Сидорин А.Б. Методы автоматизации составления расписания занятий. Часть 2. Эвристические методы оптимизации / А.Б. Сидорин, Л.В. Ликучева, А.М. Дворянкин // Известия ВГТУ. — 2009. — № 7. — С. 121-123.
4. Ерунов В. П. Формирование оптимального расписания учебных занятий в вузе / В.П. Ерунов, И.И. Морковин — Вестник ОГУ. — 2001. — № 3. — С. 55-63.
5. FET – Free Timetabling Software [Electronic resource]. – Access mode: <https://lalescu.ro/liviu/fet/>
6. Волков, С. В. Інформаційна система генерації та доступу до розкладу навчальних занять у комп'ютерних класах вищого навчального закладу [Текст] / С. В. Волков // Вісник Східноукраїнського національного університету імені Володимира Даля. – 2018. – №6 – С. 31–37.
7. Алгоритм раскраски графа – Библиотека алгоритмов на графах [Электрон. ресурс]. – Режим доступа: <http://urban-sanjoo.narod.ru/colours.html>.
8. Расписание занятий: "Ректор-ВУЗ" [Электрон. ресурс] / — Режим доступа: <http://rector.spb.ru/raspisanie-vuz-4u.php>.
9. Система Галактика: Расписание учебных занятий [Электрон.ресурс] — Режим доступа: <http://galaktika.ua/manuals/ruz/index.html>
10. aSc Расписания [Электрон. ресурс] / — Режим доступа: http://help.asctimetables.com/index.php?lang_id=7&id=173#441
11. Comparative Study of Ant Colony Algorithms for Multi-Objective Optimization / [J. Ning, C. Zhang, P. Sun et al.] // Information. – 2018. – Vol. 10, №

1. – Р. 4–11.

12. Субботін С. О. Інтелектуальні системи : Навчальний посібник / С. О. Субботін, А. О. Олійник ; під заг. ред. проф. С. О. Субботіна. – Запоріжжя : ЗНТУ, 2014. – 216 с.

13. Олійник, А. О. Інтелектуальний аналіз даних / А. О. Олійник, С. О. Субботін, О. О. Олійник. – Запоріжжя : ЗНТУ, 2012. – 278 с.

14. Клепко, В. Ю. Вища математика в прикладах і задачах / В. Ю. Клепко, В.Л. Голець – К. : Центр учбової літератури, 2009. – 162 с.

15. Прата, С. Язык программирования C++. Лекции и упражнения / С. Прата. – М. : Диалектика-Вильямс, 2016. – 1248 с.

16. Языки программирования. C Sharp [Електрон. ресурс]. – Режим доступа: <https://learn-code.ru/yazyki-programmirovaniya/c-sharp>.

17. Perkins, B. Hammer and J. Reid, Beginning c# 7 programming with visual studio / B. Perkins, J. Hammer, J. Reid. – Indianapolis : John Wiley & Sons, 2017. – 884 p

18. МакГрат, М. Программирование на Python для начинающих. Лекции и упражнения / М. МакГрат. – М. : Диалектика, 2015. – 192 с.

19. Плюсы и минусы программирования на Java [Електрон. ресурс]. – Режим доступа: <https://cutt.ly/JbKj3pq>.

20. Сьерра, К. Изучаем Java. Лекции и упражнения / К. Сьерра, Б. Бэйтс. – М. : Эксмо, 2017. – 720 с.

21. JetBrains .NET IDE hits RTM [Electronic resource] – Access mode: <https://blog.jetbrains.com/dotnet/2017/08/03/rider-2017-1-jetbrains-net-ide-hits-rtm/>

ДОДАТОК А
Технічне завдання

Вступ

Складання розкладу занять є завдання оптимального управління такими ресурсами, як викладацький склад, аудиторний фонд, комп'ютерний парк навчального закладу і т. п. При цьому необхідно враховувати обов'язкові обмеження, такі як місткість аудиторій, санітарні вимоги до проведення занять, наявність сумісників, територіальні особливості розташування навчального закладу, а також численні додаткові вимоги, наприклад, відсутність вікон у студентів і викладачів.

Вихідними даними для розкладу є навчальний план, списки груп студентів, списки аудиторій, списки викладачів з предметів, календарні вимоги до розкладу. З математичної точки зору завдання складання розкладу відноситься до завдань цілочисельного програмування, складність яких зростає експоненціально з ростом числа змінних, що беруть участь в завданні (так звані NP-складні завдання). Для вирішення таких завдань розроблено багато точних (таких, що відносяться до класичної математики) методів і алгоритмів.

A.1 Підстави для розробки

Підставою для розробки є завдання на дипломний проєкт бакалавра на тему “Автоматизована система складання електронного розкладу”, затверджене наказом Національного університету «Запорізька політехніка» №131 від 30 березня 2021 року.

A.2 Призначення розробки

Актуальність роботи. В даний час в країні здійснюються інтенсивні реформи в системі вищої освіти з метою поліпшення якості освіти. Якість підготовки студентів і ефективність роботи викладацького складу залежить від рівня організації навчального процесу. Одним з критичних елементів цього

процесу є розклад занять. Таким чином, проблема складання розкладу є важливою соціальною та економічною проблемою [1].

Розклад занять вузу - це документ, що регламентує роботу всього навчального закладу, розподіляє зміст навчального плану на календарні дні навчального року. Основним завданням складання розкладу є планування методично правильного навчального процесу, з урахуванням взаємозв'язку навчальних дисциплін, можливостей студентів з навчання і побажань викладачів за графіком роботи [6].

Вихідна інформація для формування розкладу:

- структурно-логічні схеми дисциплін;
- сітки годин за курсами дисциплін;
- аудиторний фонд вузу;
- штатний розклад викладацького складу;
- перелік вимог, що пред'являються до розкладу.

Структурно-логічні схеми коротко і наочно відображають зміст основних тем, розділів навчальної дисципліни, логіку курсу в цілому і методику його викладу. Структурно-логічна схема представляє матеріал курсу в конкретній і структурованій формі, відображаючи зміст окремих питань теми або розділу у вигляді схем, графіків, креслень, формул, рівнянь, що допомагає учневі побачити особливості дисципліни, що вивчається.

А.3 Основні вимоги до програми, яка розробляється

Програма призначена для швидкого та легкого складання розкладу занять у вищих навчальних закладах. Для її функціонування необхідно встановити продукт на комп'ютер, заповнити базу даних по групах, викладачах, студентах, аудиторіях..

ПП повинен відповідати наступним мінімальним системним вимогам:

- операційна система Microsoft Windows 7;
- 64-розрядна версія операційної системи;

- центральний процесор з тактовою частотою 1,8 ГГц.
- 1 ГБ оперативної пам'яті;
- 1 ГБ вільного місця на жорсткому диску.

Керування програмою здійснюється за допомогою стандартних засобів керування – клавіатури та миші. Рекомендується використовувати для роботи з програмою монітор з розрішенням 1440 x 900 і більше..

А.4 Основні вимоги до програми, що розробляється, початковим даним і результату

А.4.1 Вимоги до функціональних характеристик

Стадії та етапи розробки:

- аналіз існуючих систем, що забезпечують схожий функціонал;
- проектування функціоналу;
- проектування бази даних;
- проектування інтерфейсу;
- створення інтерфейсу;
- створення функціоналу;
- об'єднання інтерфейсу та функціоналу;
- тестування;
- захист випускної кваліфікаційної роботи бакалавра.

А.4.2 Вимоги до надійності

ПП повинен відповідати таким вимогам до надійності:

- захищати ПП від несанкціонованого доступу;
- захищати ПП від помилкових дій користувача;
- здійснювати обробку виняткових ситуацій.

ПП повинен нормально функціонувати без програмних збоїв та помилок

при безперервній роботі комп'ютера. У разі виникнення апаратних збоїв, нормальна робота ПП може бути відновлена після перезавантаження системи.

A.4.3 Умови експлуатації

Умови експлуатації:

- ПП не потребує обслуговуючого персоналу;
- для того, щоб використовувати ПП не потрібна додаткова кваліфікація.

A.4.4 Вимоги до маркування та упакування

Програмне забезпечення не потребує додаткових дій при маркуванні та упакуванні.

A.4.5 Вимоги до транспортування та збереження

Вимоги до транспортування й збереження є аналогічні вимогам, які висуваються до носія, на якому записаний ПП.

A.5 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- технічне завдання;
- керівництво програміста;
- керівництво оператора;
- опис програми.

А.6 Основні обмеження на установку і використання програмного продукту

Програмний продукт повинен використовуватись тільки на тих комп'ютерах, які задовільняють мінімальним вимогам та параметрам технічних засобів. Програмний продукт не конфліктує з іншими програмними продуктами та його використання може здійснюватись користувачем у будь-який момент часу.

ДОДАТОК Б
Опис програмного продукту

Б.1 Загальні відомості

Назва програмного продукту – «Sheasy».

Створений ПП застосовується для того, щоб вирішити проблему, пов'язану з часом доставки великої кількості ліків в замовленні автоматом видачі ліків, шляхом пошуку оптимального маршруту проходження маніпулятора за допомогою вирішення задачі комівояжера.

ПП «AutoPharm» має функцію авторизації користувачів, дозволяє додавати інформацію про замовлення у БД, визначати загальну кількість ліків в замовленні, будувати матрицю відстаней для полиць з ліками, які будуть включені в маршрут проходження маніпулятора автомату видачі ліків, а також знаходити оптимальний маршрут, який проходить маніпулятор та розраховувати його загальну відстань та час проходження.

ПП повинен відповідати наступним мінімальним системним вимогам:

- операційна система Microsoft Windows 7;
- 64-розрядна версія операційної системи;
- центральний процесор з тактовою частотою 1,8 ГГц.
- 1 ГБ оперативної пам'яті;
- 1 ГБ вільного місця на жорсткому диску.

Б.2 Функціональне призначення

Вхідними даними є списки груп, аудиторій, викладачів та предметів, що містяться у окремих csv файлах, які мають табличну структуру. Це зроблено для того щоб не плутати користувача великою кількістю форм. Заповнювати данні можна за допомогою MSExcel або будь який інший аналог для Unix систем, що представляє собою дуже зручний інструмент для роботи з csv файлами.

Вихідними даними є ніщо інше як готовий розклад у зручному табличному вигляді. Також для зручності було зроблено можливість пересувати

та відображати панелі таким чином, щоб можна було отримати максимум робочого простору.

Б.3 Повідомлення

Найбільш цінною можливістю проєкту «Sheasy» є автоматичне складання розкладу навчального закладу за допомогою різних алгоритмів. Описані вище класи є ні що інше, як інтерфейс програміста, основа для створення таких алгоритмів. У цьому розділі наведемо кілька основних вимог до розробників модулів автоматичної обробки даних.

Джерелом даних для обробки служить клас `CDataBase` і його інтерфейсні функції. За допомогою них можна легко отримати списки необхідних для обробки об'єктів і їх властивості. Слід зазначити, що клас `CDataBase`, в даному контексті, може і повинен використовуватися в режимі «тільки для читання».

Результат своєї роботи модуль автоматичної обробки повинен зберегти всередині структури `CSchedule`. Клас `CSchedule` має внутрішні механізми перевірки і запобігання введення суперечливих даних. Його можливості можуть бути використані на будь-якому етапі виконання алгоритму. При цьому, не дивлячись на принципову можливість, алгоритм автоматичної обробки не повинен обходити заборони та обмеження, що виставляються модулем `CSchedule`.

Після того, як алгоритм завершив свою роботу, інтерфейсна частина програми відображає на екрані по запиті користувача таблиці, побудовані на основі внутрішньої структури класу `CSchedule`.

Під час роботи розробленого ПП дії користувача можуть призводити до помилок. ПП обробляє всі помилки, а також генерує повідомлення про них, для того, щоб користувач мав можливість їх виправити.

При введенні і редагуванні даних використовується роз'єднаний режим роботи з базою даних. Збереження змін повинен здійснювати користувач за

допомогою кнопки "Зберегти". Якщо ж користувач виходить з форми редагування без збереження, йому видається нагадування – «Зберегти зміни».

Так само, при помилках користувача видається повідомлення про помилку. Приклад такого повідомлення (при неправильних параметрах з'єднання з БД) – «Підключення не встановлено тому, що кінцевий комп'ютер відкинув запит на підключення».

Server error – «Помилка серверу».

Server started on port... – Сервер запущений під портом ...

Б.4 Структурна схема програмного продукту

Структурна схема програми зображена на (рис. Б.1).

Створювана програма буде мати наступні підсистеми (модулі):

- модулі введення і коригування вихідних даних;
- модуль введення і коригування вимог до розкладу;
- модуль автоматичного складання розкладу;
- модулі відображення отриманого розкладу;
- модуль виведення на друк отриманого розкладу;
- модуль роботи з базою даних.

Користувач через меню головної (керуючої) форми викликає програмні модулі, необхідні для ведення даних та виконання інших функцій програми

Модулі введення і коригування вихідних даних через модуль роботи з базою даних здійснюють обробку відповідних таблиць бази даних.

Аналогічно здійснює свої функції модуль введення і коригування вимог до розкладу.

Модуль автоматичного складання розкладу через модуль роботи з базою даних отримує необхідні дані, виконує складання розкладу і заносить сформований розклад у базу даних.

Модулі відображення та друку відповідно отримують необхідні дані і виводять їх у потрібному форматі на екран та принтер.

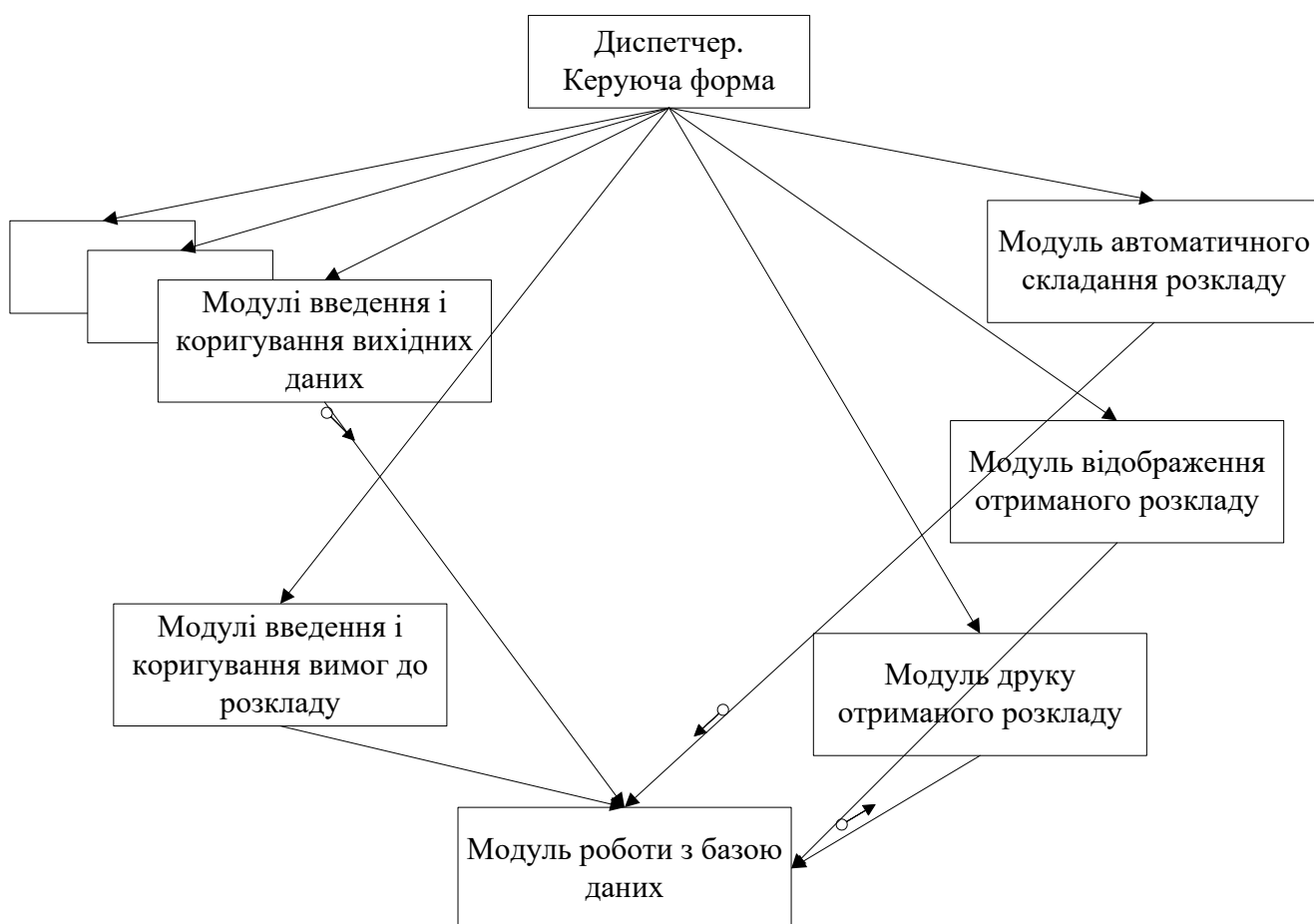


Рисунок Б.1 – Структурна схема програми.

Б.6 Приклади реалізації

Для запуску програми оберіть файл Sheasy.exe та запустіть його.

Вам буде представлено головне вікно програми (рис. Б.2), де стовпчиками буде представлено викладачів або групи в залежності від того яку вкладку обрати, а строками сітки будуть дні тижня.

Головне вікно містить строку меню у лівому верхньому кутку вікна, чотири рухомих фрейми з даними про групи, аудиторії, викладачів та предмети. Їх можна сховати або відобразити через меню (рис. Б.3) для зручності роботи, також їх можна перемістити.

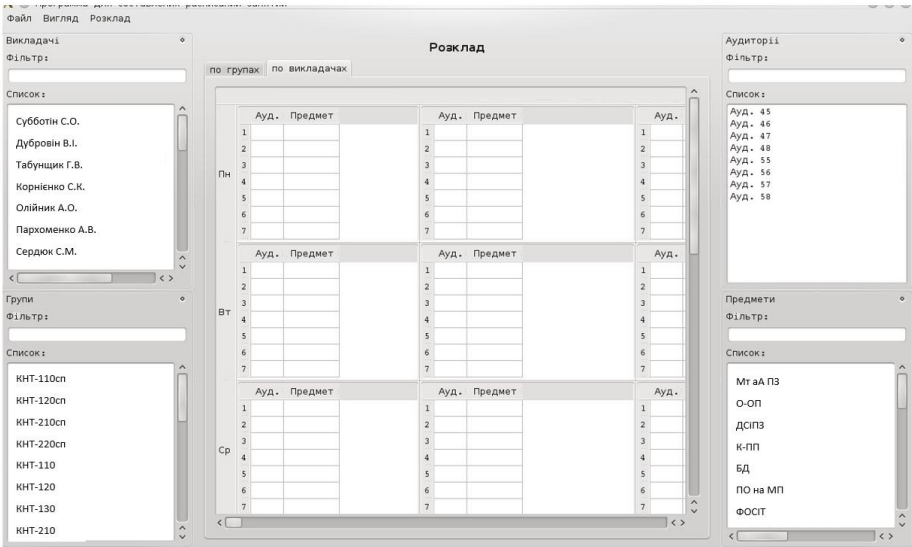


Рисунок Б.2 – Початок роботи

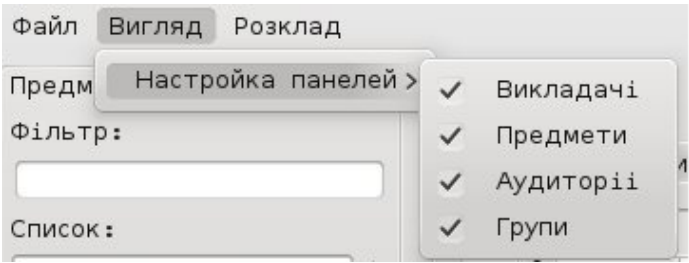


Рисунок Б.3 – меню для роботи з рухомими панелями

Розклад можна скласти вручну (рис.Б.4) за допомогою контекстного меню та орієнтуватись по групах або по викладачам, переключаючись за допомогою вкладок.

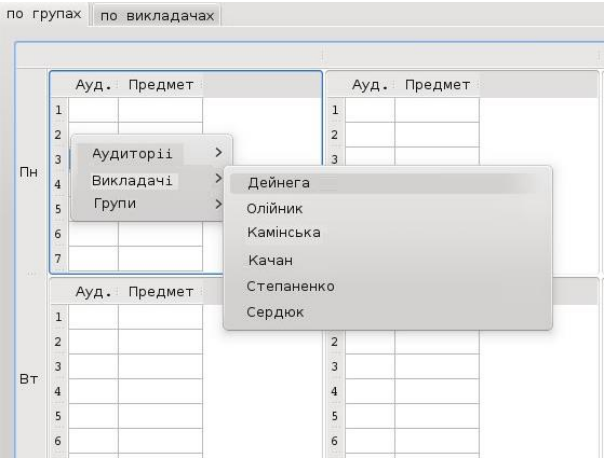


Рисунок Б.4 – Складання розкладу за допомогою контекстного меню

Також можна скласти розклад автоматичними засобами, які можна модифікувати та додавати у програму (рис. Б.5). На даний момент є два алгоритми складання розкладів – це метод прямої сортировки та генетичний метод.

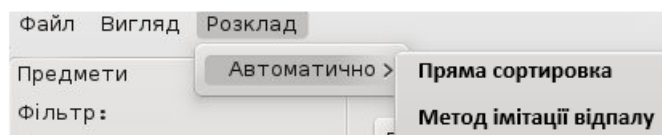


Рисунок Б.5 – вибір алгоритму

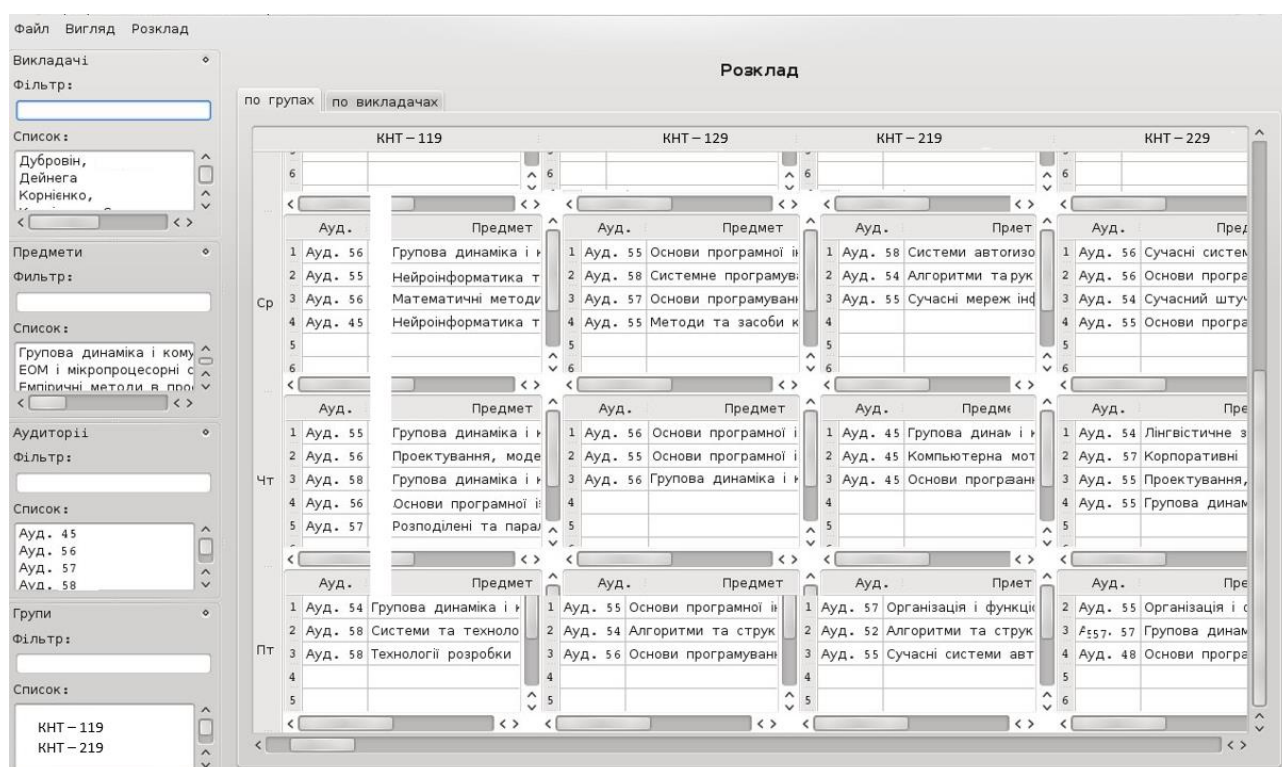


Рисунок Б.6 – Результати виконання програми

ДОДАТОК В
Текст файлів програми

В.1 Файл `cautogreedy.cpp`

```
#include <QTime>
#include <iostream>
#include "cautogreedy.h"

CAutoGreedy::CAutoGreedy(CDataBase *db, CSchedule *sh)
{
    this->db = db;
    this->sh = sh;
}

void CAutoGreedy::Start() {

    QList<CSubject*> subjects;
    QList<CGroup*> groups;
    QList<CRoom*> rooms;
    QList<CTeacher*> teachers;
    CSubject* subject = NULL;
    CTeacher* teacher = NULL;
    CGroup* group = NULL;
    CRoom* room = NULL;
    CScheduleLesson* lesson;

    do {

        for (int lnum=1; lnum<=LESSONS_COUNT; lnum++) {
            for (int dayNum=1; dayNum<=DAYS_COUNT; dayNum++) {

                subjects = GetSubjects();
                groups = sh->Day(dayNum)->GetFreeGroups(lnum);
                rooms = sh->Day(dayNum)->GetFreeRooms(lnum);
                teachers = sh->Day(dayNum)->GetFreeTeachers(lnum);

                lesson = sh->Day(dayNum)->AddLesson(lnum);

                subject = NULL;
                room = NULL;
                teacher = NULL;
                group = NULL;
            }
        }
    } while (true);
}
```

```

while (!subject || !room || !teacher || !group) {

    if (!subjects.count() || !groups.count() || !rooms.count() || !teachers.count()) break;

    subject = ChooseSubject(subjects, lesson);
    if (!subject) continue;
    lesson->SetSubject(subject);

    group = ChooseGroup(groups, lesson);
    if (!group) continue;
    lesson->SetGroup(group);

    teacher = ChooseTeacher(teachers, lesson);
    if (!teacher) continue;
    lesson->SetTeacher(teacher);

    room = ChooseRoom(rooms, lesson);
    if (!room) continue;
    lesson->SetRoom(room);
}
}
}

} while (subjects.count() && groups.count() && teachers.count() && rooms.count());
}

```

```

QList<CSubject*> CAutoGreedy::GetSubjects() {

    SGroupLessons lessons;
    QList<CSubject*> rv;
    QList<CGroup*> groups;

    groups = db->GetGroups();

    for (int i=0; i<groups.count(); i++) {

        for (int k=0; k<groups.at(i)->GetLessons()->count(); k++) {

            lessons = groups.at(i)->GetLessons()->at(k);
            if (rv.indexOf(lessons.subject) == -1) {

                for (int t=0; t<lessons.hours_left.count(); t++) {

```

```

        if (lessons.hours_left.at(t) > 0) {
            rv.append(lessons.subject);
            break;
        }
    }
}
}
}
}

return rv;
}

```

```

CSubject* CAutoGreedy::ChooseSubject(QList<CSubject*> &subjects, CSheduleLesson* lesson) {

```

```

    CSubject* tmp;

```

```

    if (!subjects.count()) return NULL;

```

```

    tmp = subjects.takeFirst();

```

```

    subjects.append(tmp);

```

```

    subjects.takeAt(qrand()%(subjects.count()));

```

```

    for (int i=0; i<subjects.count(); i++) {
        if (lesson->TrySubject(subjects.at(i))) {
            return subjects.at(i);
        }
    }
}

```

```

    return NULL;
}

```

```

CTeacher* CAutoGreedy::ChooseTeacher(QList<CTeacher*> &teachers, CSheduleLesson* lesson) {

```

```

    CTeacher* tmp;

```

```

    if (!teachers.count()) return NULL;

```

```

    tmp = teachers.takeFirst();

```

```

    teachers.append(tmp);

```

```

    teachers.takeAt(qrand()%(teachers.count()));

```

```

    for (int i=0; i<teachers.count(); i++) {
        if (lesson->TryTeacher(teachers.at(i))) {
            return teachers.at(i);
        }
    }

    return NULL;
}

CGroup* CAutoGreedy::ChooseGroup(QList<CGroup*> &groups, CScheduleLesson* lesson) {

    CGroup* tmp;

    if (!groups.count()) return NULL;

    tmp = groups.takeFirst();
    groups.append(tmp);

    groups.takeAt(qrand()%(groups.count()));

    for (int i=0; i<groups.count(); i++) {
        if (lesson->TryGroup(groups.at(i))) {
            return groups.at(i);
        }
    }

    return NULL;
}

CRoom* CAutoGreedy::ChooseRoom(QList<CRoom*> &rooms, CScheduleLesson* lesson) {

    CRoom* tmp;

    if (!rooms.count()) return NULL;

    tmp = rooms.takeFirst();
    rooms.append(tmp);

    rooms.takeAt(qrand()%(rooms.count()));

    for (int i=0; i<rooms.count(); i++) {

```

```

        if (lesson->TryRoom(rooms.at(i))) {
            return rooms.at(i);
        }
    }

    return NULL;
}

```

В.2 Файл cdatabase.cpp

```

#include "cdatabase.h"

#include <QDir>
#include <QTextStream>
#include <QTextCodec>

#include <QStringList>

CDataBase::CDataBase(QString dirpath, QObject *parent) :
    QObject(parent)
{

    file_teachers.setFileName(dirpath + QString("/teachers.csv"));
    file_groups.setFileName(dirpath + QString("/groups.csv"));
    file_subjects.setFileName(dirpath + QString("/subjects.csv"));
    file_rooms.setFileName(dirpath + QString("/rooms.csv"));

    if (!file_teachers.exists()) {
        file_teachers.open(QIODevice::WriteOnly);
        file_teachers.close();
    }
    if (!file_subjects.exists()) {
        file_subjects.open(QIODevice::WriteOnly);
        file_subjects.close();
    }
    if (!file_rooms.exists()) {
        file_rooms.open(QIODevice::WriteOnly);
        file_rooms.close();
    }
    if (!file_groups.exists()) {
        file_groups.open(QIODevice::WriteOnly);

```

```

        file_groups.close();
    }

    Reload();
}

CDataBase::~CDataBase() {

    file_teachers.close();
    file_subjects.close();
    file_rooms.close();
    file_groups.close();
}

void CDataBase::Reload() {

    _LoadSubjects();
    _LoadRooms();
    _LoadTeachers();
    _LoadGroups();
}

void CDataBase::_LoadTeachers() {

    QString buffer;
    QString title;
    QString subjects;
    QStringList subjects_list;
    QList<CSubject*> p_subjects;
    CSubject* p_subject;
    quint32 id;
    quint32 k;
    quint8 hours;

    teachers.clear();

    file_teachers.open(QIODevice::ReadOnly | QIODevice::Text);

    QTextStream in(&file_teachers);
    in.setCodec("Windows-1251");

    while(!in.atEnd()) {

```

```

subjects_list.clear();

buffer = in.readLine();

id = buffer.section(QString(";"),0,0).toInt();

title = buffer.section(QString(";"),1,1);
if (title[0] == "") title.remove(0,1);
if (title[title.length()-1] == "") title.remove(title.length()-1,1);

subjects = buffer.section(QString(";"),2,2);
if (subjects[0] == "") subjects.remove(0,1);
if (subjects[subjects.length()-1] == "") subjects.remove(subjects.length()-1,1);

teachers.append(CTeacher(id, title));

subjects_list = subjects.split(",");

for (qint32 i=0; i<subjects_list.count(); i++) {

    hours = subjects_list.at(i).section(":", 0, 0).toShort();

    k = 1;
    p_subjects.clear();

    while (subjects_list.at(i).section(":", k, k).toInt()) {
        p_subject = GetSubject(subjects_list.at(i).section(":", k, k).toInt());
        p_subjects.append(p_subject);
        k++;
    }

    teachers.last().AddSubjects(hours, p_subjects);
}

file_teachers.close();
}

void CDataBase::_LoadSubjects() {

    QString buffer;

```

```

QString title;
qint32 id;
qint8 teachers_count;
qint8 groups_count;

subjects.clear();

file_subjects.open(QIODevice::ReadOnly | QIODevice::Text);

QTextStream in(&file_subjects);
in.setCodec("Windows-1251");

while(!in.atEnd()) {

    buffer = in.readLine();
    id = buffer.section(";", 0, 0).toInt();

    title = buffer.section(";", 1, 1);
    teachers_count = buffer.section(";", 2, 2).toShort();
    groups_count = buffer.section(";", 3, 3).toShort();

    if (title[0] == "") title.remove(0,1);
    if (title[title.length()-1] == "") title.remove(title.length()-1,1);

    subjects.append(CSubject(id, title, teachers_count, groups_count));
}

file_subjects.close();
}

void CDataBase::_LoadGroups() {

    QString buffer;
    QString title;
    QString subjects;
    QStringList subjects_list;
    qint8 students_count;
    CSubject* p_subject;
    QList<qint8> hours;
    qint32 id;
    qint32 k;

```

```

groups.clear();

file_groups.open(QIODevice::ReadOnly | QIODevice::Text);

QTextStream in(&file_groups);
in.setCodec("Windows-1251");

while(!in.atEnd()) {

    subjects_list.clear();

    buffer = in.readLine();

    id = buffer.section(QString(";"),0,0).toInt();

    title = buffer.section(QString(";"),1,1);
    if (title[0] == "") title.remove(0,1);
    if (title[title.length()-1] == "") title.remove(title.length()-1,1);

    students_count = buffer.section(QString(";"),2,2).toInt();

    subjects = buffer.section(QString(";"),3,3);
    if (subjects[0] == "") subjects.remove(0,1);
    if (subjects[subjects.length()-1] == "") subjects.remove(subjects.length()-1,1);

    groups.append(CGroup(id, title, students_count));

    subjects_list = subjects.split(",");

    for (qint32 i=0; i<subjects_list.count(); i++) {

        p_subject = GetSubject(subjects_list.at(i).section(":", 0, 0).toInt());

        k = 1;
        hours.clear();

        while (subjects_list.at(i).section(":", k, k).toShort()) {
            hours.append(subjects_list.at(i).section(":", k, k).toShort());
            k++;
        }

        groups.last().AddLesson(p_subject, hours);
    }
}

```

```

    }
}

file_groups.close();
}

void CDataBase::_LoadRooms() {

    QString buffer;
    QString title;
    quint32 id;
    quint8 students_count;
    quint8 groups_count;
    QString subjects;
    QStringList subjects_list;
    CSubject* p_subject;

    rooms.clear();

    file_rooms.open(QIODevice::ReadOnly | QIODevice::Text);

    QTextStream in(&file_rooms);
    in.setCodec("Windows-1251");

    while(!in.atEnd()) {

        subjects_list.clear();

        buffer = in.readLine();

        id = buffer.section(QString(";"),0,0).toInt();
        students_count = buffer.section(QString(";"),2,2).toInt();
        groups_count = buffer.section(QString(";"),3,3).toInt();

        title = buffer.section(QString(";"),1,1);
        if (title[0] == "") title.remove(0,1);
        if (title[title.length()-1] == "") title.remove(title.length()-1,1);

        subjects = buffer.section(QString(";"),4,4);
        if (subjects[0] == "") subjects.remove(0,1);
        if (subjects[subjects.length()-1] == "") subjects.remove(subjects.length()-1,1);
    }
}

```

```

rooms.append(CRoom(id, title, students_count, groups_count));

subjects_list = subjects.split(",");

for (qint32 i=0; i<subjects_list.count(); i++) {

    p_subject = GetSubject(subjects_list.at(i).toInt());

    rooms.last().AddSubject(p_subject);
}
}

file_rooms.close();
}

```

```

QList<qint32> CDataBase::FetchIds(DBEssenceType type) {

    QList<qint32> ids;

    ids.clear();

    switch (type) {
    case E_Teacher:
        for (int i=0; i<teachers.count(); i++) {
            ids.append(teachers.at(i).GetId());
        }
        break;
    case E_Subject:
        for (int i=0; i<subjects.count(); i++) {
            ids.append(subjects.at(i).GetId());
        }
        break;
    case E_Group:
        for (int i=0; i<groups.count(); i++) {
            ids.append(groups.at(i).GetId());
        }
        break;
    case E_Room:
        for (int i=0; i<rooms.count(); i++) {
            ids.append(rooms.at(i).GetId());
        }
        break;
    }
}

```

```

    }

    return ids;
}

QString CDataBase::FetchStringField(qint32 id, DBEssenceType type, DBEssenceFields field) {

    void* p;

    switch (type) {
    case E_Teacher:
        p = this->GetTeacher(id);
        break;
    case E_Subject:
        p = this->GetSubject(id);
        break;
    case E_Group:
        p = this->GetGroup(id);
        break;
    case E_Room:
        p = this->GetRoom(id);
        break;
    default:
        return QString("");
    }

    if (!p) return QString("");

    switch (type) {
    case E_Teacher:
        switch(field) {
        case EF_title:
            return ((CTeacher*)p)->GetTitle();
        }
    case E_Subject:
        switch(field) {
        case EF_title:
            return ((CSubject*)p)->GetTitle();
        }
    case E_Group:
        switch(field) {
        case EF_title:

```

```

        return ((CGroup*)p)->GetTitle();
    }
case E_Room:
    switch(field) {
        case EF_title:
            return ((CRoom*)p)->GetTitle();
        }
    default:
        return QString("");
    }
}

CTeacher* CDataBase::GetTeacher(qint32 id) {

    for (int i=0; i<teachers.count(); i++)
        if (teachers[i].GetId() == id)
            return &teachers[i];

    return 0;
}

CSubject* CDataBase::GetSubject(qint32 id) {

    for (int i=0; i<subjects.count(); i++)
        if (subjects[i].GetId() == id)
            return &subjects[i];

    return 0;
}

CGroup* CDataBase::GetGroup(qint32 id) {

    for (int i=0; i<groups.count(); i++)
        if (groups[i].GetId() == id)
            return &groups[i];

    return 0;
}

CRoom* CDataBase::GetRoom(qint32 id) {

```

```

    for (int i=0; i<rooms.count(); i++)
    if (rooms[i].GetId() == id)
        return &rooms[i];

    return 0;
}

QList<CTeacher*> CDataBase::GetTeachers() {

    QList<CTeacher*> rv;
    QList<qint32> ids;

    ids = FetchIds(E_Teacher);

    for (int i=0; i<ids.count(); i++) {
        rv.append(GetTeacher(ids.at(i)));
    }

    return rv;
}

QList<CSubject*> CDataBase::GetSubjects() {

    QList<CSubject*> rv;
    QList<qint32> ids;

    ids = FetchIds(E_Subject);

    for (int i=0; i<ids.count(); i++) {
        rv.append(GetSubject(ids.at(i)));
    }

    return rv;
}

QList<CGroup*> CDataBase::GetGroups() {

    QList<CGroup*> rv;
    QList<qint32> ids;

    ids = FetchIds(E_Group);

```

```

    for (int i=0; i<ids.count(); i++) {
        rv.append(GetGroup(ids.at(i)));
    }

    return rv;
}

QList<CRoom*> CDataBase::GetRooms() {

    QList<CRoom*> rv;
    QList<qint32> ids;

    ids = FetchIds(E_Room);

    for (int i=0; i<ids.count(); i++) {
        rv.append(GetRoom(ids.at(i)));
    }

    return rv;
}

```

ДОДАТОК Д
Слайди презентації

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Кафедра програмних засобів

АВТОМАТИЗОВАНА СИСТЕМА СКЛАДАННЯ ЕЛЕКТРОННОГО РОЗКЛАДУ

Виконав:

студент групи КНТ-137

І.С. Чернега

Керівник:

к.т.н

Г.В. Неласа



Рисунок Д.1 – Слайд № 1

2

Мета, об'єкт, предмет дослідження

- **Об'єкт дослідження** – автоматизовані системи складання електронного розкладу.
- **Предмет дослідження** – методи формування розкладу навчальних занять.
- **Мета роботи** – створення програмної системи для складання електронного розкладу.
- **Актуальність** та проблематика роботи полягає в тому, що в даний час в країні здійснюються інтенсивні реформи в системі вищої освіти з метою поліпшення якості освіти. Якість підготовки студентів і ефективність роботи викладацького складу залежить від рівня організації навчального процесу. Одним з критичних елементів цього процесу є розклад занять. Таким чином, проблема складання розкладу є важливою соціальною та економічною проблемою [1].

Рисунок Д.2 – Слайд № 2

3

Огляд існуючих мов програмування

Параметр			
Мова програмування	C++	C #	Java
Проблеми з безпекою	Ні	Ні	Можливі
Кросплатформеність	Так	Так	Так
Легкість освоєння	+	++	++
Рівень можливостей	+	++	++
Можна використовувати для frontend (клієнтська частина)	+	++	++
Можна використовувати для backend (серверна частина)	++	+	+
Ліцензія	–	.NET Core CLR: MIT / X11	GNU GPL
Незалежність від платформи	+	–	+
Наявність додаткових бібліотек та пакетів	+	+	–
Можливість паралельного програмування	+	+	+
Наявність взаємодії з іншими мовами програмування	+	+	+

Рисунок Д.3 – Слайд № 3

4

Системи управління базами даних

Компанія-виробник	Переваги	Недоліки
PostgreSQL	функціональна і безкоштовна СУБД з відкритим кодом, хороша підтримка з боку співтовариства розробників	місцями низька продуктивність при обробці великих обсягів інформації, мала популярність продукту
Oracle	швидка обробка даних, легко інтегрувати в різні онлайн і оффлайн додатки, можливість створити велике сховище даних з мінімальними зусиллями. Oracle може впоратися з будь-якими масштабами інформації і кількістю користувачів, тому рівень масштабованості є одним з найбільш недосяжних для багатьох інших типів СУБД, великий вибір oracle інструментів	велика ціна, потужне обладнання для оптимальної роботи з Oracle, високоспеціалізований персонал для підтримки бази даних, складності у вивченні функціоналу
MySQL	безкоштовна СУБД, має відкритий код, займає мало місця, легке адміністрування mysql з набором всього необхідного функціоналу, велика спільнота розробників, працює на безпечі платформ	незахищеність від втрати даних, підтримка тільки малих баз даних
SQL Server	величезний пакет корисних інструментів, які легко використовувати, стабільна робота з sql server базами даних; менші витрати на адміністрування sql server	працює тільки на платформах Microsoft, корпоративна версія може коштувати надто дорого
DB2	працює на безпечі платформ, стабільна робота бази даних з величезною кількістю функціональних особливостей, відмінна підтримка з боку IBM, діалект мови SQL більш гнучкий і має більшу кількість переваг, ніж SQL Server.	продуктивність не така висока як у Oracle, корпоративна версія може досягати завищених цін.

Рисунок Д.4 – Слайд № 4

5

Аналіз ПП складання розкладу навчальних занять

	«Ректор-ВУЗ»	«Галактика РНЗ»	aSc Расписание
Розробник	Павел Смекалов	Корпорація «Галактика»	Applied Software Consultants
Операційна система	Windows (XP, Vista, 7)	Windows (XP, 7, 8, 10)	Windows (98, ME, NT, 2000, XP, 7, 8, 10), Apple iOS
Додаткове ПЗ	-	NET Framework v.4, СУБД (Oracle, Pervasive)	NET Framework v.4 і вище
Варіанти розгортання	клієнтський комп'ютер	клієнт, сервер, web-служба	клієнтський комп'ютер
Базовий період розкладу	тиждень	тиждень	тиждень
Вартість	5000 грн.	35000 грн.	стандарт – 4500 грн.
Розподіл груп на підгрупи	2 підгрупи	необмежено	до 5 підгруп
Складання розкладу	автоматичне, ручне, комбіноване	автоматичне, ручне коригування	автоматичне, ручне коригування
Одночасна робота кількох операторів	-	+	-
Додаткові модулі, що не входять у базовий комплект	-	-	модуль «Заміни»
Документація	задовільна	добра	добра

Рисунок Д.5 – Слайд №5

6

Діаграма використання системи складання розкладу

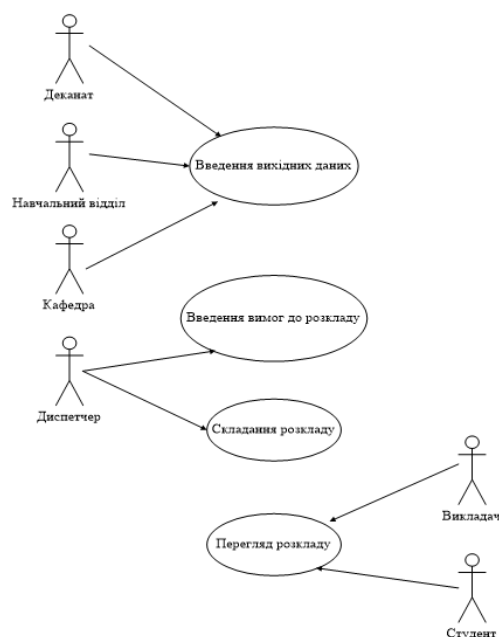


Рисунок Д.6 – Слайд № 6

7

Структурна схема програми

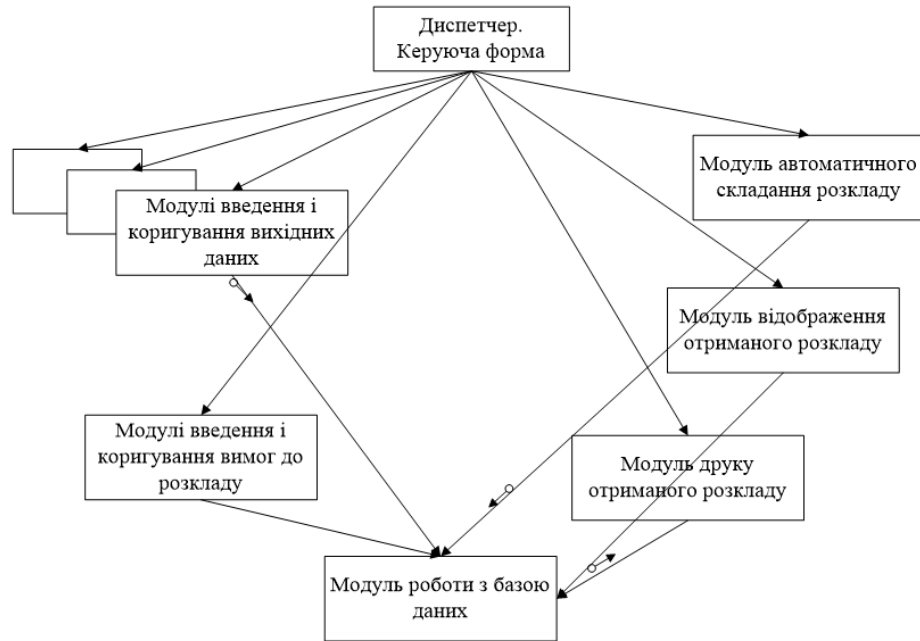


Рисунок Д.7 – Слайд № 7

8

Діаграма класів, клас «Списки»

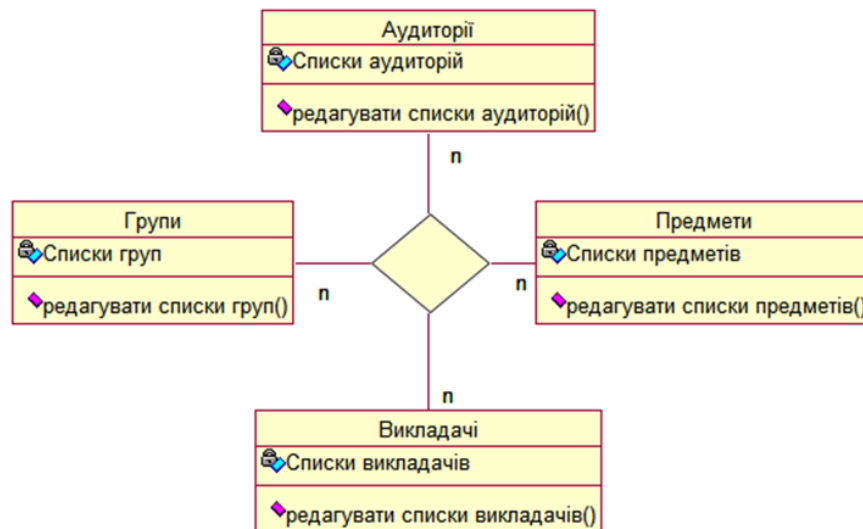


Рисунок Д.8 – Слайд № 8

Інтерфейсні вікна програми

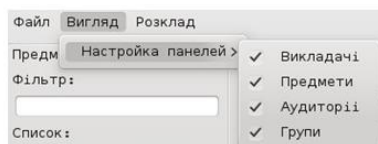
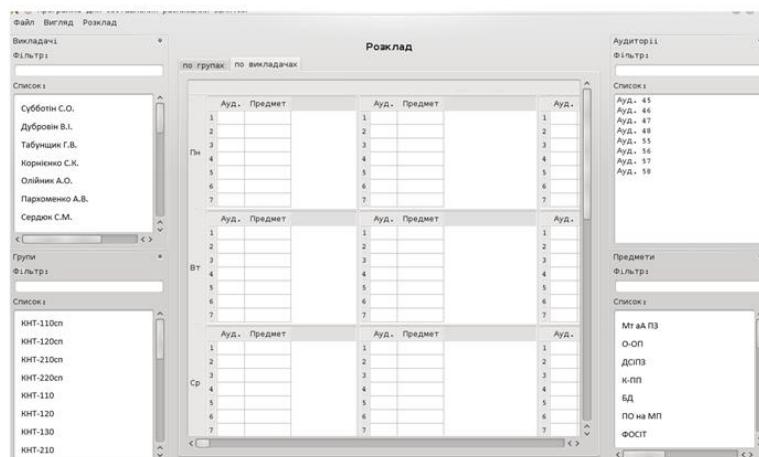


Рисунок Д.9 – Слайд № 9

Інтерфейсні вікна програми

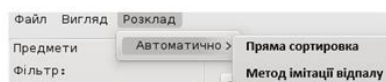
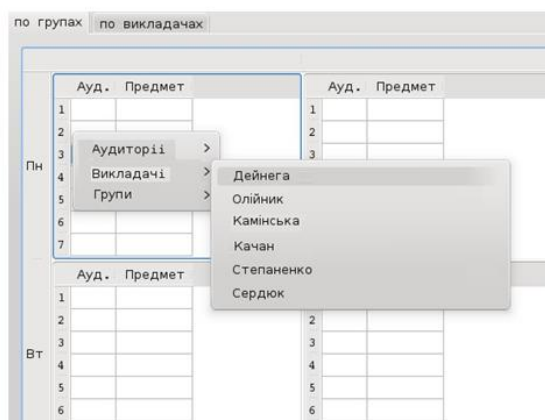


Рисунок Д.10 – Слайд № 10

Інтерфейсні вікна програми

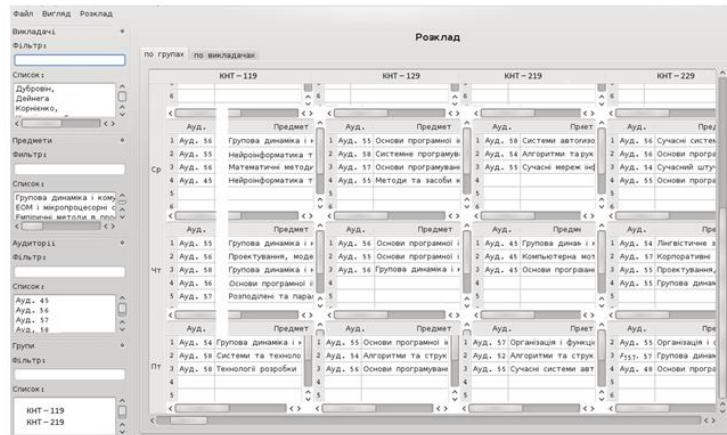


Рисунок Д.11 – Слайд № 11

Висновки

В процесі виконання кваліфікаційної роботи бакалавра було розроблено автоматизовану систему складання електронного розкладу.

Вихідними даними для розкладу є навчальний план, списки груп студентів, списки аудиторій, списки викладачів з предметів, календарні вимоги до розкладу. З математичної точки зору завдання складання розкладу відноситься до завдань цілочисельного програмування, складність яких зростає експоненціально з ростом числа змінних, що беруть участь в завданні (так звані NP-складні завдання). Для вирішення таких завдань розроблено багато точних (таких, що відносяться до класичної математики) методів і алгоритмів.

Програма має дружній інтерфейс, зручний у використанні, який не потребує багато часу на опанування принципів роботи з ним.

У кваліфікаційній роботі бакалавра була проаналізована предметна область, розроблена архітектура системи та на основі досліджень обране середовище розробки і функціонування програми.

Рисунок Д.12 – Слайд № 12