Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Дубовенка Єгора Андрійовича*
(ПІБ)

академічної групи *121-19ск-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка інтерактивної тестувально навчальної системи з використанням DotNet Framework*

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	*доцент. Кабак Л.В.*			
розділів:				
спеціальний	*доцент. Кабак Л.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-19ск-1* *Дубовенко Єгора Андрійовича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка інтерактивної тестувально навчальної системи з використанням DotNet Framework*

затверджена наказом ректора НТУ «ДП» від «18» травня 2022 р. № 268-с

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ *доц. Кабак Л.В.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Дубовенко Є.А.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *13.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 100 с., 22 рис., 10 табл., 3 дод., 17 джерела.

Об'єкт розробки: програмний додаток для навчання студентів дисципліни алгоритми та структури даних на C#.

Мета кваліфікаційної роботи: розробка програмного додатку для спрощення процесу навчання студентів і роботи викладачів. Додаток дозволить спростити та автоматизувати процес розробки і проходження тестового контролю для студентів, надання лекційних матеріалів студентам, освоєння алгоритмів сортування студентами завдяки можливості візуалізації.

У вступі наводиться аналіз сучасного стану проблеми, проводиться уточнення постановки завдання, мети кваліфікаційної роботи та галузі її застосування, а також обґрунтовується актуальність теми.

У першому розділі виконується дослідження предметної галузі та наявних рішень заданої задачі. Визначається актуальність завдання та призначення розробки, розроблюється постановка завдання.

У другому розділі обирається платформа для розробки, виконується проектування програми і її розробка, наводиться опис алгоритму і структури функціонування системи, визначаються вхідні і вихідні дані, наводяться характеристики складу параметрів технічних засобів, описується робота програми.

В економічному розділі обчислюється трудомісткість розробленого програмного продукту, виконується обчислення вартості роботи зі створення програмного додатку та визначається час на його створення.

Практичне значення роботи полягає у наданні користувачам можливості не тільки автоматизувати стандарті процеси навчання і ведення даних користувачів системи а також спростити процес пізнання важкого матеріалу через візуалізацію процесів сортування і обробки масивів.

Актуальність розробки даного програмного продукту обумовлюється відсутністю можливості перегляду процесу обробки масиву даних у аналогічних засобів та високим рівнем складності розуміння даних процесів без наочного прикладу.

Список ключових слів: ПРОГРАМА, ІНТЕРАКТИВНА, КЛІЄНТ, СЕРВЕР, ЕОМ, АЛГОРИТМ, ТЕОРЕТИЧНІ ВІДОМОСТІ, ТЕСТОВИЙ КОНТРОЛЬ, ТИП ОБЛІКОВОГО ЗАПИСУ, АДМІНІСТРАТОР, ВИКЛАДАЧ, СТУДЕНТ, C#.

ABSTRACT

Explanatory note: 100 pages, 22 figures, 10 tables, 3 appendices, 17 sources.

Object of development: a software application for teaching students the discipline of algorithms and data structures in C #.

The purpose of the qualification work: development of a software application to simplify the process of teaching students and teachers. The application will simplify and automate the process of developing and passing test tests for students, providing lecture materials to students, mastering sorting algorithms by students due to the possibility of visualization.

The introduction provides an analysis of the current state of the problem, clarifies the task, the purpose of the qualification work and the scope of its application, as well as substantiates the relevance of the topic.

In the first section, a study of the subject area and the available solutions to the problem. The urgency of the task and the purpose of development are determined, the task statement is developed.

In the second section the platform for development is chosen, the program design and its development is carried out, the description of algorithm and structure of functioning of system is given, input and output data are defined, characteristics of structure of parameters of technical means are given, work of the program is described.

In the economic section, the complexity of the developed software product is calculated, the cost of work on creating a software application is calculated and the time for its creation is determined.

The practical significance of the work is to provide users with the ability not only to automate the standard learning processes and data management of system users, but also to simplify the process of learning heavy material through the visualization of sorting and processing of arrays.

The relevance of the development of this software product is due to the lack of ability to view the process of processing an array of data in similar tools and the high level of complexity of understanding these processes without a clear example.

List of keywords: PROGRAM, INTERACTIVE, CLIENT, SERVER, COMPUTER, ALGORITHM, THEORETICAL INFORMATION, TEST CONTROL, TYPE OF ACCOUNT, ADMINISTRATION, ADMINISTRATION

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

WinForms – Windows Forms;

MVC – Model-View-Controler;

ООП – об’єктно-орієнтоване програмування;

ADO.NET – ActiveX Data Object для .NET;

SQL – Structured Query Language.

ВСТУП

Навчання є не від’ємною частиною нашого життя. Сучасна людина вивчає великі об’єми інформації та навчається вирішувати складні задачі, але цей процес не завжди і не кожному дається легко, особливо коли мова йде про вивчення програмування або алгоритмів вирішення задач, які реалізуються за допомогою мов програмування. Серед алгоритмів, які викликають складність розуміння, наприклад, у студентів є алгоритми сортування. Під сортуванням розуміють процес перестановки об’єктів множини у певному порядку. Мета сортування – полегшити пошук елементів у відсортованій множині.

Історія виникнення алгоритмів сортування дуже пов'язана з розвитком електронно-обчислювальних машин. Перші прототипи сучасних методів сортування використовувались для прискорення обробки даних перепису населення в США і обробляли інформацію записану на перфокартах, вони обробляли близько п’ятдесяти перфокарт за хвилину. З часом потужність електронно-обчислювальних машин зростала, але зростав і об’єм даних, які необхідно було відсортувати, тому почали з’являтися нові більш сучасні та ефективні методи сортування, які дозволяли використовувати ресурси електронно-обчислювальних машин більш ефективно.

В наш час існує безліч алгоритмів сортування, всі вони відрізняються швидкістю, ефективністю та призначенням. Їх різноманіття та складність деяких із них не дозволяє швидко зрозуміти їх принцип та ускладнює процес навчання. Але засвоєння найбільш використовуваних та найбільш ефективних алгоритмів сортування є не від’ємною частиною навчання спеціалістів з розробки програмного забезпечення. Тому виникає необхідність у розробці такого програмного забезпечення, яке дозволило б полегшити цей процес та зробити його більш наочним та ефективним. Ось чому цей проект буде мати практичну цінність та успішну реалізацію, адже візуалізація роботи сучасних, складних алгоритмів сортування, аналіз їх швидкості та ефективності, а також

їх недоліків може спростити їх вивчення та дасть змогу студентам навчатися за допомогою новітніх технологій.

Метою даного проекту є розробка клієнт-серверного додатка який буде застосовуватись в процесі навчання студентами для перегляду анімації алгоритмів сортування, проходження тестів з теоретичних питань та ознайомлення з теоретичними відомостями щодо алгоритмів. Також додаток зможуть використовувати викладачі для адміністрування даних студентів, розробки тестового контролю та теоретичних відомостей. Адміністратор додатка матиме можливість керувати обліковими записами викладачів та підпорядкуванням груп студентів відповідним викладачам.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Алгоритм сортування:

Алгоритм сортування – це алгоритм призначений для впорядкування елементів масиву. Іноді елементи масиву можуть зберігати не єдине значення а декілька полів. В такому випадку перше значення називають ключем сортування а друге корисними даними, тобто саме ключ є критерієм сортування. На практиці в якості ключа часто виступає число, а в інших полях зберігаються будь-які дані, що ніяк не впливають на роботу алгоритму.

Методи сортування можна поділити на внутрішні та зовнішні. Внутрішнє сортування використовується для даних, які можна завантажити в оперативну пам'ять, таке сортування є більш гнучким щодо використання структур даних. При зовнішньому сортуванні дані в оперативну пам'ять не вміщуються, таке сортування спрямовано на досягнення результату за умов обмежених ресурсів.

Алгоритми сортування оцінюються за швидкістю виконання та ефективністю використання пам'яті:

Час(обчислювальна складність) - основний параметр, що характеризує швидкодію алгоритму. Для значної кількості алгоритмів середній і найгірший час впорядкування n-елементного масиву є O(n^2) — це пов'язано з тим, що в них передбачені перестановки елементів, що стоять поряд . Такі алгоритми зазвичай є стабільними, хоча і не ефективними для великих масивів. Інший клас алгоритмів здійснює впорядкування за час O(n log n). В цих алгоритмах використовується можливість обміну елементів, що знаходяться на будь-якій відстані один від одного.

Пам'ять - ряд алгоритмів потребує виділення додаткової пам'яті під тимчасове зберігання даних. Як правило, ці алгоритми вимагають O(log n) пам'яті. При оцінці не враховується місце, яке займає вихідний масив і незалежні від вхідної послідовності витрати, наприклад, зберігання коду

програми (оскільки це споживає O(1) пам'яті). Алгоритми сортування, які не споживають додаткової пам'яті, відносять до так званих сортувань на місці.

Властивості алгоритмів сортування:

Стійкість - стійке сортування не змінює взаємного розташування елементів з однаковими ключами.

Природність поведінки - ефективність методу сортування під час обробки вже впорядкованих чи частково впорядкованих даних. Алгоритм веде себе природно, якщо враховує цю характеристику вхідної послідовності та працює краще.

Використання операції порівняння - алгоритми, які використовують для сортування порівняння елементів між собою, називаються заснованими на порівняннях. Мінімальна трудомісткість найгіршого випадку для цих алгоритмів становить O(n log n), але вони відрізняються гнучкістю застосування [1].

Клієнт-серверний додаток:

Клієнт і сервер це програмне забезпечення. Зазвичай ці програми розташовані на різних обчислювальних машинах іноді вони можуть бути розміщені на одній машині, вони взаємодіють між собою через мережу за допомогою мережевих протоколів.

Програма-сервер очікує від клієнтських програм запити а у відповідь надає їм певні данні для користування.

Оскільки одна програма-сервер може виконувати запити від багатьох програм-клієнтів, її розміщують на спеціально виділеній обчислювальній машині тому продуктивність цієї машини має бути високою.

Програма-клієнт ініціює запити на необхідні для неї дані або послуги також клієнт має зрозуміти відповідь тобто вміст та форматування даних відправлених сервером.

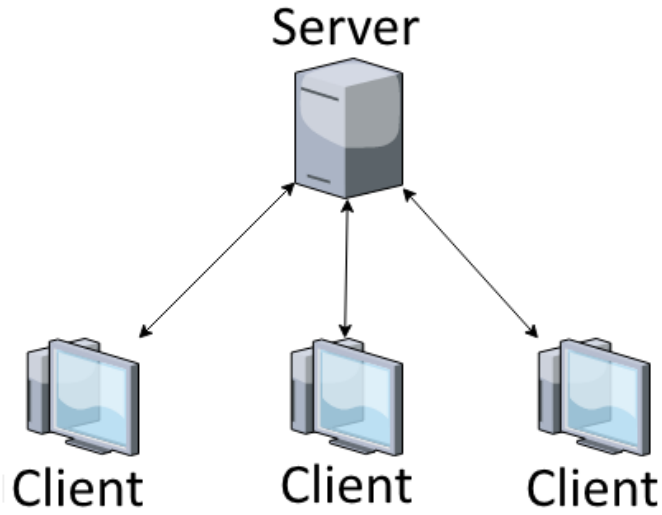

Рис. 1.1. Клієнт-серверна взаємодія

1.2. Призначення розробки та галузь застосування

Оскільки проект є інтерактивною тестувально навчальною системою його призначенням є оптимізація та покращення якості навчання. Додаток є вузько спеціалізованим тобто призначений для вивчення дисципліни алгоритми та структури даних і може використовуватись як викладачами для покращення організації процесу навчання так і студентами для спрощення сприйняття учбових матеріалів.

Розробка не має вікових обмежень, оскільки може використовуватись не тільки в державних навчальних закладах вищої освіти але і в приватних установах для підготовки учнів за спеціальностями інженерія програмного забезпечення, комп'ютерні науки або комп'ютерна інженерія.

На підставі всього вище перерахованого можна зробити висновок що областю застосування розробленого застосунка є будь-яка навчальна або педагогічна діяльність в межах дисципліни алгоритми та структури даних.

1.3. Підстава для розробки

Підставами для розробки та виконання кваліфікаційної роботи) є:

– освітня програма 121 Інженерія програмного забезпечення;

– навчальний план та графік навчального процесу;

– наказ ректора Національного технічного університету «Дніпровська політехніка» № 268-с від 18.05.2022 р;

– завдання на кваліфікаційну роботу на тему «Розробка інтерактивної тестувально навчальної системи з використанням Dot Net Framework».

1.4. Постановка завдання

Метою даної кваліфікаційної роботи є розробка інтерактивної тестувально навчальної системи, яка призначена для вивчення принципів роботи алгоритмів внутрішнього сортування. Додаток може застосовуватись для використання в якості допоміжного програмного забезпечення для викладання дисципліни алгоритми та структури даних, а також для допомоги студентам у вивченні даної дисципліни. До основних функцій розроблюваного застосунку буде входити: управління обліковими записами користувачів, візуалізація алгоритмів сортування, розробка тестового контролю для студентів та публікація теоретичних відомостей. Додаток буде розділяти повноваження користувачів на три типи облікових записів (адміністратор, викладач, студент), після проходження авторизації користувач отримує повноваження відповідно до типу його облікового запису.

Для досягнення поставленої задачі необхідно вирішити основні завдання:

- вивчити предметну область задачі яку необхідно розв’язати;
- правильно спроектувати базу даних додатка;
- розробити бібліотеку класів для візуалізації алгоритмів сортування.

В цілому додаток буде забезпечувати оптимізацію процесу навчання та полегшення сприйняття учбового матеріалу студентами.

1.5. Вимоги до функціональних характеристик

Програма, яка розробляється призначена для використання у навчанні студентами (які вивчають дисципліну алгоритми та структури даних) і викладачами.

В додатку повинні бути передбачені: ведення облікових записів користувачів, перегляд та редагування теоретичних відомостей щодо алгоритмів сортування, візуалізація та аналіз алгоритмів сортування, а також можливість створення та проходження періодичного тестового контролю щодо отримання знань студентами.

Кожен з користувачів повинен мати тип облікового запису, а саме адміністратор, викладач, студент і повинен мати змогу виконувати певні функції, бути наділений певними правами та мати певні обмеження.

Адміністратор – повинен мати можливість створювати та видаляти облікові записи викладачів, змінювати дані будь-якого облікового запису. Також адміністратор повинен мати змогу проводити тестування програми, виконуючи сортування з заданими даними та обраним алгоритмом сортування, але він не повинен мати можливість редагування теоретичних відомостей щодо алгоритмів сортування.

Викладач – повинен мати змогу виконувати редагування теоретичних відомостей щодо алгоритмів сортування, редагування тестового контролю студентів та перегляд результатів його проходження, а також переглядати результати сортування масивів даних за обраним алгоритмом сортування, але викладач не може здійснювати додавання, видалення та редагування облікових записів адміністраторів або інших викладачів.

Студент – повинен мати змогу переглядати результати сортування масивів даних за обраним алгоритмом сортування, проходження тестового контролю щодо отримання знань та перегляд своїх результатів, перегляд теоретичних відомостей щодо алгоритмів сортування, але студент не повинен мати можливість додавання, видалення та редагування облікових записів інших користувачів. Також студент не повинен мати можливість редагувати теоретичні відомості про алгоритми сортування.

Інтерфейс розроблюваної програми повинен бути багато віконним та використовувати наступні компоненти:

– головне меню для надання різних можливостей управління всією програмою для кожного типу користувача;

– прапорці для надання користувачу можливості вибору щодо виконання або не виконання певних дій;

– випадаючий список для вибору користувачем необхідного методу сортування або облікового запису;

– таблиця для відображення необхідних даних з бази даних;

– кнопка для переходу між вікнами або виконання обраних дій;

– панель інструментів для відображення найчастіше використовуваних команд щодо пошуку або сортування;

– спливаючі підказки для спрощення навігації по програмі та прискорення роботи з нею;

– рядок стану для відображення інформації про користувача та тип його облікового запису.

1.6. Вимоги до інформаційної безпеки

Розроблюваний програмний додаток є багатокористувацьким, тому повинен забезпечувати розділення користувачів за типами облікових записів і надавати кожному відповідні повноваження та накладати певні обмеження. Перш ніж працювати з додатком кожен користувач повинен пройти авторизацію для підтвердження особи.

Додаток повинен забезпечувати перевірку введених даних облікових записів користувачів на коректність, тому поля для введення повинні мати певні обмеження і використовувати функції для перевірки даних.

У випадку використання програми вперше користувач матиме можливість зареєструватися в ньому для створення облікового запису.

Також програмний додаток повинен забезпечити цілісність даних та надання користувачам мінімальних привілеїв достатніх для виконання поставлених перед ними завдань відповідно до їх ролі в системі.

1.7. Вимоги до складу та параметрів технічних засобів

Для використання програми необхідно мати персональний комп’ютер або ноутбук з наступними мінімальними технічними характеристиками:

- операційна система Windows 10;
- оперативний запам’ятовувальний пристрій об’ємом 2 ГБ або більше;
- процесор з тактовою частотою 1 ГГц або вище;
- не менше 20 Гб вільного місця на жорсткому диску;
- наявність маніпулятора-миші та клавіатури;
- .Net Framework версії 4.8 або вище.

1.8. Вимоги до інформаційної та програмної сумісності

Розроблюваний додаток призначений для функціонування під управлінням операційних систем сімейства Windows. Отже для коректної роботи програми необхідно мати встановлену операційну систему Microsoft Windows не нижче версії Windows 7 з встановленою платформою .NET Framework, не нижче версії .NET Framework 4.8.

Для успішного виконання запитів клієнтського додатка до бази даних, на сервері повинно бути встановлено програмне забезпечення Microsoft SQL Server 2019.

Оскільки в програмі передбачається зберігання теоретичних відомостей про алгоритми сортування для коректної роботи програми необхідно мати встановлений текстовий редактор Microsoft Word.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Функціональне призначення системи: автоматизація навчальної діяльності студентів і викладачів які займаються дисципліною алгоритми та структури даних, автоматизація процесу розробки тестового контролю, автоматизація зберігання та ведення теоретичних відомостей, автоматизація проходження тестового контролю, автоматизація ведення облікових даних, візуалізація алгоритмів сортування, отримання інтуїтивно зрозумілого інтерфейсу тощо.

Для реалізації інформаційної системи розроблена бібліотека класів Algorithm призначена для візуалізації алгоритмів які обробляють масиви числових даних. Розроблена бібліотека дозволяє також відображати такі параметри як: кількість перестановок елементів, кількість перестановок однакових елементів, об’єм пам’яті який використовується алгоритмом, час виконання алгоритму.

Також розроблена бібліотека має можливість подальшого використання для розширення переліку анімацій алгоритмів даного додатка, або для реалізації інших додатків які потребують анімації обробки числових масивів.

2.2. Опис застосованих математичних методів

Оскільки особливості предметної області розв'язуваної задачі не передбачають застосування математичних методів у розроблюваному програмному продукті не було застосовано математичних методів.

2.3. Опис використаної архітектури та шаблонів проектування

Програмний додаток реалізовано на основі архітектурного шаблону проектування Model-view-controller (Модель–представлення–контролер), який передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для

відокремлення даних (моделі) від інтерфейсу користувача так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача [2].

Мета шаблону - гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

В цілому, логіка роботи шаблону MVC наведена на рис. 2.3.

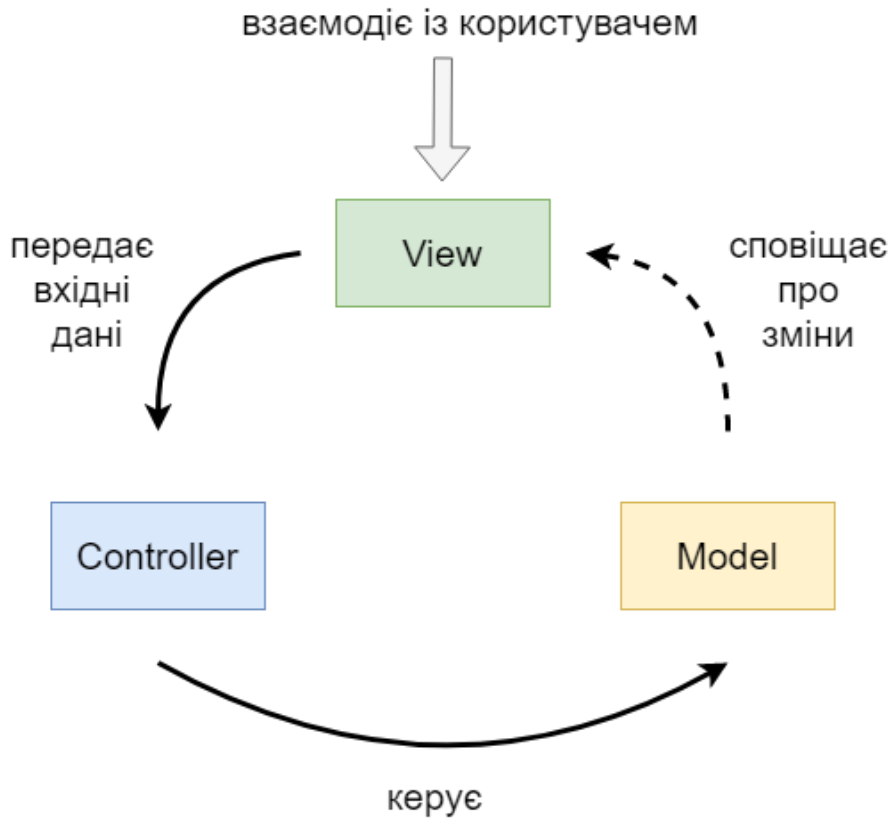

Рис. 2.1. Взаємодія компонентів шаблону MVC

Як можна побачити на рис. 2.3, кожен компонент моделі відповідає за свою конкретну задачу:

1. Model – надає дані і реагує на команди контролера, змінюючи свій стан.

2. View – відповідає за відображення даних моделі користувачеві, реагуючи на зміни моделі.

3. ViewModel – інтерпретує дії користувача, сповіщаючи модель про необхідність змін [3].

2.4. Опис використаних технологій та мов програмування

Мова програмування:

Існує велика кількість мов програмування для розробки додатків з типом архітектури клієнт-сервер, наприклад такі як C++, Java, Python, C#.

C# – об'єктно-орієнтована мова програмування, розроблена в 1998-2001 роках групою інженерів компанії Microsoft, як мова розробки додатків для платформи Microsoft .NET Framework [4].

Мова програмування C# відноситься до сімейства мов з C-подібним синтаксисом, синтаксис C# найбільш схожий на C++ та Java, і при цьому надає відсутні в Java потужні функції, наприклад обнуляємі типи значень, перерахування, делегати, лямбда-вирази і прямий доступ до пам'яті. Також мова програмування C# багато чого перейняла від своїх предків, наприклад типи даних та синтаксис багатьох операторів дуже схожий на C та C++ тому багато розробників, які програмували на C++ або Java без особливої складності можуть перейти на C#, але синтаксис C# спрощує багато складнощів C++. C# підтримує універсальні методи і типи, які забезпечують більш високий рівень безпеки і продуктивності, а також ітератори, що дозволяють визначати в класах колекцій власну поведінку ітерації, яку можна легко застосувати в клієнтському коді. Вирази LINQ (Language Integrated Query – мова інтегрованих запитів) створюють дуже зручну мовну конструкцію для суворо типізованих запитів.

C# є об'єктно-орієнтованою мовою, а значить підтримує інкапсуляцію, успадкування і поліморфізм [5].

Процес побудови програмних додатків в C# простіше в порівнянні з C або C++, але більш гнучкий, ніж в Java. Окремі файли заголовка не використовуються, і немає необхідності оголошувати методи і типи в певному

порядку. Вихідний файл C# може визначити будь-яке число класів, структур, інтерфейсів і подій. Синтаксис C# дуже багатий, але при цьому простий і зручний у вивченні.

C# досить тривалий час твердо займає позицію в лідируючій десятці найпопулярніших мов програмування. Програмісти, які не є розробниками під Windows, теж починають вивчати C#, оскільки тепер він працює на Mac і Linux.

Гнучкість мови C# є величезною перевагою, в порівнянні з іншими мовами програмування. Різноманітність додатків, які можуть бути розроблені за допомогою C# практично безмежна, наприклад: додатки для Windows, мобільні додатки, веб-додатки, ігри, додатки для Android і iOS, які розробляються за допомогою додаткових фрейм ворків, таких як Xamarin або Mono.

Звичайно, все це можливо виконувати і за допомогою інших мов програмування, але зазвичай, в таких випадках, використовуються сторонні інструменти інших розробників. Програмісти, які працюють з C# мають великий набір інструментів, які підтримуються Microsoft для розробки будь-якого типу програми [6].

Завдяки розвитку візуального програмування на C# та CASE-засобів вдалося значно спростити та прискорити процес розробки користувацьких інтерфейсів та програмного продукту в цілому.

Генерування користувацьких інтерфейсів відбувається на основі графічного інтерфейсу створеного в діалоговому режимі. В процесі створення інтерфейсу середовище надає можливість створювати форми, на яких можна розміщувати як візуальні, так і не візуальні компоненти. Розміщення компонентів відбувається за допомогою перетягування їх з панелі елементів на форму, також за допомогою миші можна змінювати розміри компонентів та їх розташування. У відповідних діалогових вікнах можна змінити задані за замовчуванням значення властивостей цих компонентів і створити обробники подій для них. При цьому в процесі проектування завжди видно результат – зображення форми і розташованих на ній компонентів. Але головна перевага візуального програмування у тому, що підчас проектування форми і

розміщення на ній компонентів автоматично формується код програми, включаючи в неї фрагменти, які описують додані компоненти та їх властивості та події. Тобто проектування, фактично, зводиться до розміщення компонентів на формі, завдання їх властивостей та написання обробників подій, що дозволяє за хвилини або години зробити те, на що раніше йшли місяці роботи [7].

Середовище розробки:

Для використання будь-якої сучасної мови програмування необхідне програмне забезпечення яке буде надавати необхідні інструменти розробнику а також виконувати функції компілятора та інтерпретатора.

Існує багато різноманітних середовищ розробки для написання програм на C#, але найбільш популярною з них є Microsoft Visual Studio.

Microsoft Visual Studio – програмний продукт компанії Microsoft який являється інтегрованим середовищем розробки програмного за-безпечення, в якому є все необхідне для проектування, тестування та налагодження доповнень. Microsoft Visual Studio дозволяє розробляти як консольні додатки, так і додатки з графічним інтерфейсом, а також веб-сайти, веб-додатки та веб-служби.

Microsoft Visual Studio, включає в себе редактор коду з підтрим-кою технології IntelliSense, яка дозволяє автоматично доповнювати програмний код, вбудований відлачик, редактор форм для спрощення створення графічного інтерфейсу додатку, веб-редактор, дизайнер класів та дизайнер схеми бази даних. У верхній частині вікна середовища знаходиться панель інструментів, яка містить швидкі кнопки, що дублюють найчастіше використовувані команди меню, в нижній частині вікна знаходиться список помилок, який з’являється у випадку наявності помилок в програмному коді і містить код помилки та її опис. Ліворуч розташована панель елементів, що містить палітру компонентів, які відсортовані по категоріях [8]. У правій частині вікна знаходиться панель властивостей, яка містить всі властивості і події компонентів та оглядач рішень, у якому відображаються всі файли проекту. У центрі знаходиться вікно форми і вікно редактора кодів, що зображено на рисунку 2.2.

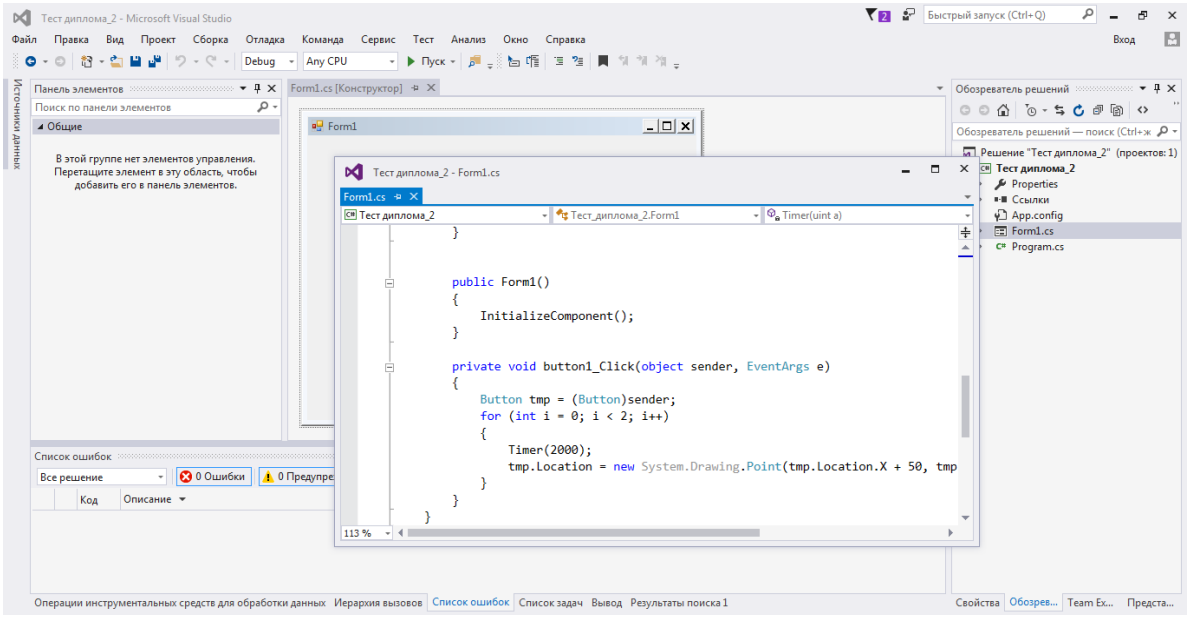

Рис. 2.2. – Інтерфейс середовища розробки Microsoft Visual Studio 2019

База даних:

База даних – організована відповідно до певних правил і підтримувана в пам'яті комп'ютера сукупність даних, що характеризує актуальний стан деякої предметної області і використовувана для задоволення інформаційних потреб користувачів.

На сьогоднішній день бази даних зберігають великі об’єми різноманітної інформації обробляти її вручну було б дуже складно та займало б величезну кількість часу, тому для спрощення обробки та внесення змін в бази даних використовують спеціальні запити, які надають змогу користувачу переглядати та редагувати данні в базах даних ці запити побудовані за допомогою спеціальної мови SQL[9].

SQL (Structured query language) – мова структурованих запитів розроблена компанією IBM, яка застосовується для створення, модифікації та управління даними в реляційній базі даних, керованій відповідною системою керування базами даних.

За допомогою мови SQL можна виконувати наступні операції:

– створення бази даних;

– створення та редагування структури бази даних;

– додавання даних в базу даних;

– оновлення даних;

– видалення даних;

– перегляд даних.

З точки зору прикладного інтерфейсу існують два різновиди команд SQL:

– інтерактивний SQL – використовується в спеціальних утилітах, що дозволяють в інтерактивному режимі вводити запити з використанням команд SQL, посилати їх для виконання на сервер і отримувати результати в призначеному для цього вікні;

– вбудований SQL – використовується в прикладних програмах, дозволяючи їм посилати запити до сервера і обробляти отримані результати, в тому числі комбінуючи set-орієнтований і record-орієнтований підходи.

Мова SQL призначена для маніпулювання даними в реляційних базах даних, визначення структури баз даних і для управління правами доступу до даних в багатокористувацькому середовищі.

Тому, в мову SQL в якості складових частин входять:

– мова маніпулювання даними (Data Manipulation Language, DML);

– мова визначення даних (Data Definition Language, DDL);

– мова керування даними (Data Control Language, DCL).

Підкреслимо, що це не окремі мови, а різні групи команд однієї мови. Такий поділ проведено тільки з точки зору різного функціонального призначення цих команд [10].

На сьогоднішній день існує багато систем, які підтримують мову SQL але найпоширенішою з них є Microsoft SQL Server.

SQL Server – система керування базами даних, розроблена корпорацією Microsoft у 1988 році.

В SQL Server використовуються наступні технології:

– компонент Database Engine являє собою основну службу для зберігання, обробки та забезпечення безпеки даних;

– служби SQL Server Data Quality Services (DQS) є рішенням для очищення даних на основі знань, дозволяють створити базу знань, а потім

виконати в ній виправлення даних і видалення дублікатів за допомогою як автоматизованих, так і інтерактивних засобів;

– служби Analysis Services – це платформа аналітичних даних і набір засобів для бізнес-аналітики на особистому рівні, рівні робочої групи та організації;

– служби Integration Services є платформою для створення високопродуктивних рішень з інтеграції даних, в тому числі пакетів для зберігання даних, що забезпечують витяг, перетворення і завантаження даних;

– Master Data Services - це рішення SQL Server для управління основними даними, рішення, яке базується на основі Master Data Services, дозволяє забезпечити правильність інформації, використовуваної для побудови звітів і виконання аналізу;

– служби Reporting Services пропонують засоби створення корпоративних звітів з підтримкою веб-інтерфейсу, які дозволяють включати в звіти дані з різних джерел, публікувати звіти в різноманітних форматах, а також централізовано керувати безпекою і підписками[11].

Технологія доступу до даних ADO.NET:

При розробці додатку, який взаємодіє з базою даних необхідно використовувати одну з існуючих технологій мови програмування C#.

Для реалізації у програмному додатку роботи з базою даних була обрана технологія ADO.NET, яка забезпечує можливість підключення та управління базою даних із програмного додатка.

ADO.NET надає узгоджений доступ до таких джерел даних, як SQL Server і XML, а також до джерел даних, що надаються за допомогою OLE DB і ODBC. Призначені для користувача функції, можуть використовувати ADO.NET для з'єднання з цими джерелами даних і для отримання, обробки і поновлення наявних в них даних[12].

ADO.NET розділять доступ до даних і обробку даних на дискретні компоненти, які можуть використовуватися окремо або разом. ADO.NET включає постачальників даних .NET Framework для з'єднання з базою даних,

виконання команд і отримання результатів. Ці результати, поміщені в об'єкт ADO.NET DataSet, обробляються безпосередньо, щоб вони могли бути надані користувачеві нерегламентованим чином, об'єднані з даними з багатьох джерел або передавання даних між рівнями. Об'єкт DataSet також може незалежно використовуватися постачальником даних .NET Framework для управління локальними для додатка даними або даними, джерелом яких є XML.

Класи ADO.NET є в System.Data.dll і інтегруються з класами XML, наявними в System.Xml.dll. Для розробників, які пишуть керований код, ADO.NET надає функціональний набір, подібний до функціонального набору, який надають об'єкти даних ActiveX (ADO) розробникам моделей об'єктів власних компонентів (COM). Для доступу до даних в додатку .NET рекомендується використовувати ADO.NET, а не ADO.

ADO.NET надає найпряміший спосіб доступу до даних в .NET Framework. Для більш високого рівня абстракції, який дозволяє додаткам працювати з концептуальною моделлю, а не базовою моделлю зберігання.

Постачальниками даних .NET Framework є компоненти, які спеціально сконструйовані для обробки даних і швидкого, однопрохідного доступу до даних тільки для читання. Об'єкт Connection забезпечує обмін даними з джерелом даних. Об'єкт Command дозволяє звертатися до команд бази даних для повернення даних, зміни даних, виконання збережених процедур і передачі або отримання відомостей про параметри. DataReader забезпечує високопродуктивний потік даних з джерела даних. Нарешті, DataAdapter надає міст між об'єктом DataSet і джерелом даних. DataAdapter використовує об'єкти Command для виконання команд SQL на джерелі даних для завантаження DataSet з даними і узгодження змін даних, виконаних в DataSet, знову з джерелом даних.

DataSet – клас в ADO.NET спеціально сконструйований для доступу до даних незалежно від джерела даних. Тому він може бути використаний з багатьма і різними джерелами даних, з XML-даними або для управління даними, локальними для додатка. DataSet містить колекцію одного або

декількох об'єктів DataTable, що складаються з рядків і стовпців даних, доступ до яких можна отримати за допомогою перевантаженого оператора індексації, а також первинний ключ, зовнішній ключ, обмеження і пов'язані відомості про данні в об'єктах DataTable [13].

На рисунку 2.3 ілюстровано структуру класу .NET Framework і DataSet.

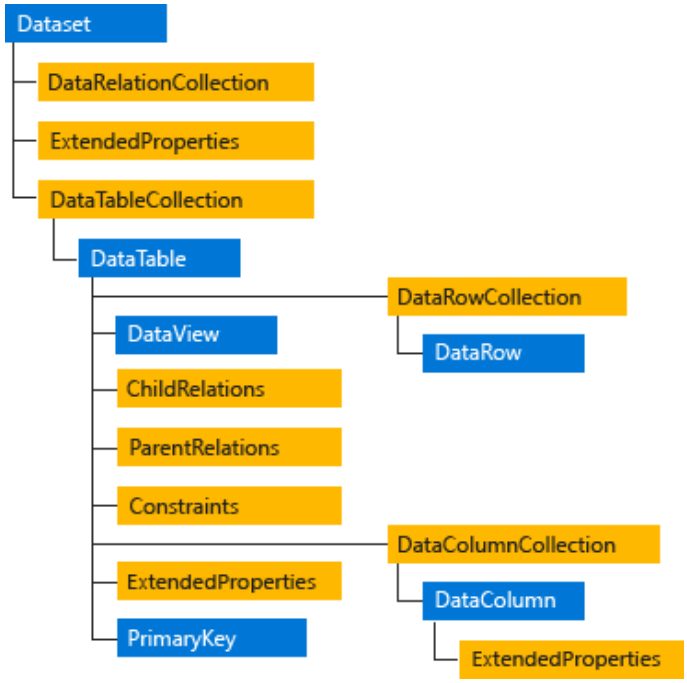

Рис. 2.3. – Структура класу DataSet

Завдання DataSet:

– локальне хешування даних в додатку для подальшої обробки. Якщо потрібно тільки зчитувати результати запиту, клас DataReader підходить як найкраще;

– віддалена взаємодія з даними між рівнями або з веб-служби XML;

– динамічна взаємодія з даними, наприклад прив'язка до елементу управління Windows Forms або комбінування і зв'язування даних з декількох джерел;

– виконання інтенсивної обробки, що не вимагає відкритого з'єднання з джерелом даних, що звільняє з'єднання для використання іншими клієнтами.

Якщо функціональність, що надається класом DataSet, не потрібна, можна підвищити продуктивність програми, використовуючи клас DataReader для

повернення даних в однопрохідному режимі тільки для читання. Незважаючи на те, що DataAdapter використовує DataReader для заповнення вмісту DataSet, за допомогою DataReader, можна підвищити продуктивність, оскільки без використання набору даних можна зберегти пам'ять, яка буде використовуватися DataSet і уникнути обробки даних, необхідної для створення і заповнення вмісту DataSet [14].

2.5. Опис структури програми та алгоритмів її функціонування

Структура розробленого проекту наведена на рис. 2.4.

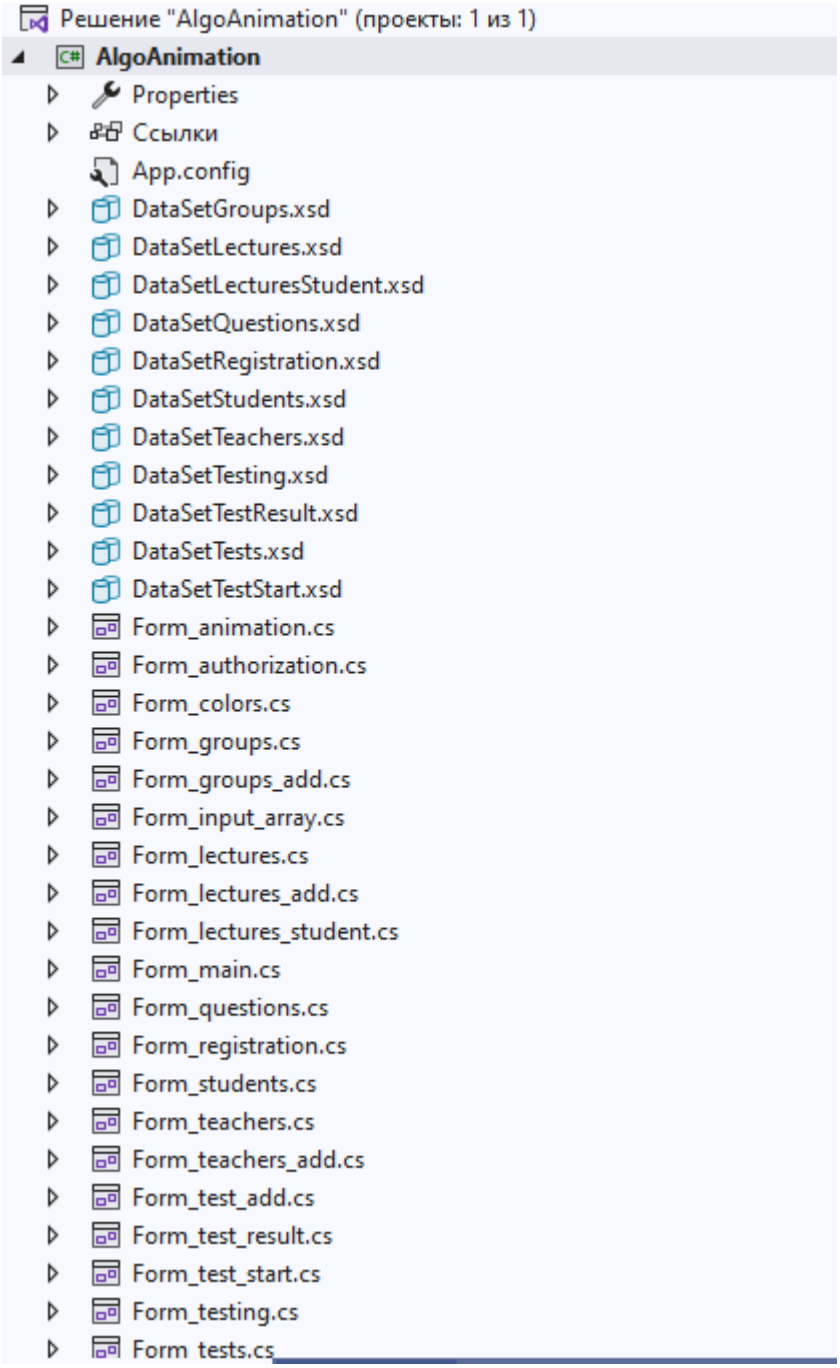

Рис. 2.4. Структура проекту

Проект складається з тридцяти двох файлів:

– App.config – файл в якому зберігаються рядки підключення до бази даних;

– DataSetGroups.xsd – файл в якому зберігається структура набору даних з переліком всіх груп в базі даних;

– DataSetLectures.xsd – файл в якому зберігається структура набору даних з переліком теоретичних відомостей з бази даних;

– DataSetLecturesStudent.xsd – файл в якому зберігається структура набору даних з переліком теоретичних відомостей з бази даних;

– DataSetQuestions.xsd – файл в якому зберігається структура набору даних з переліком запитань з бази даних;

– DataSetRegistration.xsd – файл в якому зберігається структура набору даних з переліком всіх груп в базі даних;

– DataSetStudents.xsd – файл в якому зберігається структура набору даних з переліком студентів з бази даних;

– DataSetTeachers.xsd – файл в якому зберігається структура набору даних з переліком викладачів з бази даних;

– DataSetTesting.xsd – файл в якому зберігається структура набору даних з переліком запитань і відповідей з бази даних;

– DataSetTestResult.xsd – файл в якому зберігається структура набору даних з переліком результатів тестування студентів з бази даних;

– DataSetTests.xsd – файл в якому зберігається структура набору даних з переліком тестів і запитань з бази даних;

– DataSetTestStart.xsd – файл в якому зберігається структура набору даних з переліком тестів і викладачів з бази даних;

– Form_animation.cs – файл в якому зберігається код вікна перегляду анімації;

– Form_authorization.cs – файл в якому зберігається код вікна авторизації;

– Form_colors.cs – файл в якому зберігається код вікна налаштування кольорів анімації;

– Form_groups.cs – файл в якому зберігається код вікна з інформацією про студентські групи;

– Form_groups_add.cs – файл в якому зберігається код вікна додавання або редагування інформації про студентські групи;

– Form_input_array.cs – файл в якому зберігається код вікна редагування вхідного масиву;

– Form_lectures.cs – файл в якому зберігається код вікна з інформацією про теоретичні відомості;

– Form_lectures_add.cs – файл в якому зберігається код вікна додавання або редагування інформації про теоретичні відомості;

– Form_lectures_student.cs – файл в якому зберігається код вікна з інформацією про теоретичні відомості;

– Form_main.cs – файл в якому зберігається код головного вікна програми;

– Form_questions.cs – файл в якому зберігається код вікна з інформацією про запитання до тестів;

– Form_registration.cs – файл в якому зберігається код вікна реєстрації студентів;

– Form_students.cs – файл в якому зберігається код вікна з інформацією про студентів;

– Form_teachers.cs – файл в якому зберігається код вікна з інформацією про викладачів;

– Form_teachers_add.cs – файл в якому зберігається код вікна додавання або редагування інформації про студентів;

– Form_test_add.cs – файл в якому зберігається код вікна додавання або редагування інформації про тести;

– Form_test_result.cs – файл в якому зберігається код вікна з інформацією про пезультати проходження тестів студентами;

– Form_test_start.cs – файл в якому зберігається код вікна вибору тестового контролю для проходження;

– Form_testing.cs – файл в якому зберігається код вікна проходження тестового контролю;

– Form_tests.cs – файл в якому зберігається код вікна з інформацією про тести розроблені кожним викладачем.

Спроектована база даних містить 10 пов’язаних між собою таблиць:

– groups – Групи;

– students – Студенти;

– admins – Адміністратор;

– teachers – Викладачі;

– groups_teachers – Зв’язок між групами і викладачами;

– lectures – Теоретичні відомості;

– tests – Тести;

– questions – Запитання до тестів;

– answers – Відповіді до запитань;

– tests_students – Зв’язок між студентами і тестами, з результатами проходження тестового контролю.

Нижче можна побачити фізичну модель створюваної бази даних (рис. 2.5).

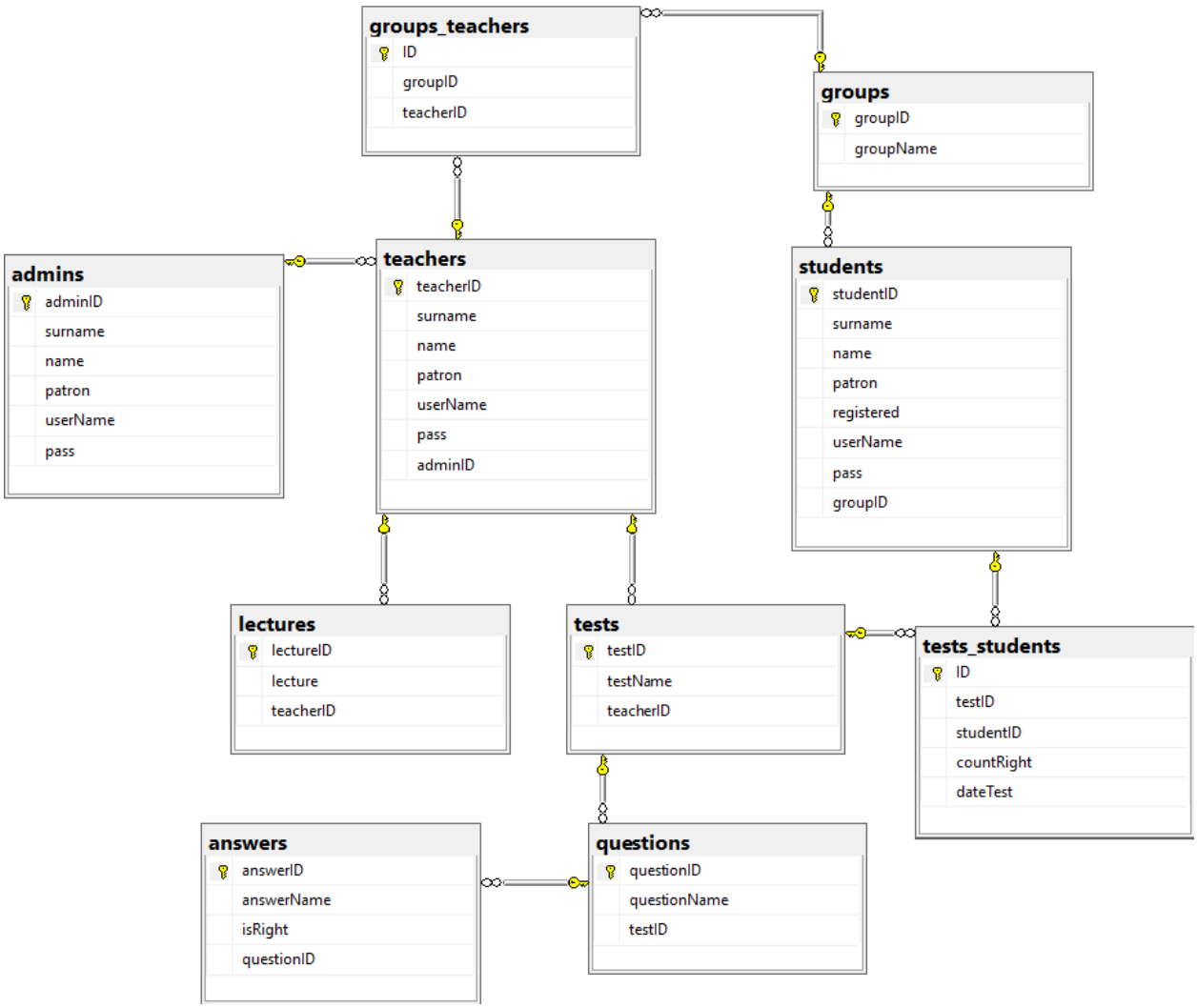

Рис. 2.5. Фізична модель бази даних

Розглянемо таблиці у вищезазначеній базі даних 2.1 – 2.10.

Таблиця 2.1

Студенти

Назва поля	Призначення поля	Тип даних	Ключ
studentID	ID студента	int	PK
name	Ім'я	nvarchar(30)	
surname	Прізвище	nvarchar(30)	
patron	По-батькові	nvarchar(30)	
registered	Помітка про реєстрацію	bit	
userName	Логін	nvarchar(30)	
pass	Пароль	nvarchar(30)	
groupID	ID групи	int	FK

Таблиця 2.2

Групи

Назва поля	Призначення поля	Тип даних	Ключ
GroupID	ID групи	int	PK
groupName	Назва	nvarchar(20)	

Таблиця 2.3

Адміністратор

Назва поля	Призначення поля	Тип даних	Ключ
adminID	ID адміністратора	int	PK

name	Ім'я	nvarchar(30)	
surname	Прізвище	nvarchar(30)	
patron	По-батькові	nvarchar(30)	
userName	Логін	nvarchar(30)	
pass	Пароль	nvarchar(30)	

Таблиця 2.4

Викладачі

Назва поля	Призначення поля	Тип даних	Ключ
teacherID	ID викладача	int	PK
name	Ім'я	nvarchar(30)	
surname	Прізвище	nvarchar(30)	
patron	По-батькові	nvarchar(30)	
userName	Логін	nvarchar(30)	
pass	Пароль	nvarchar(30)	
adminID	ID адміністратора	int	FK

Таблиця 2.5

Зв’язок групи з викладачем

Назва поля	Призначення поля	Тип даних	Ключ
ID	ID зв’язка групи з викладачем	int	PK
groupID	ID групи	int	FK
teacherID	ID викладача	int	FK

Таблиця 2.6

Теоретичні відомості

Назва поля	Призначення поля	Тип даних	Ключ
lectureID	ID лекції	int	PK
lectureName	Назва	nvarchar(50)	
lecture	Бітове представлення файлу	varbinary(max)	
teacherID	ID викладача	int	FK

Таблиця 2.7

Тести

Назва поля	Призначення поля	Тип даних	Ключ
testID	ID тесту	int	PK
testName	Назва тесту	nvarchar(20)	
teacherID	ID викладача	int	FK

Таблиця 2.8

Запитання

Назва поля	Призначення поля	Тип даних	Ключ
questionID	ID запитання	int	PK
questionName	Назва	varchar(max)	
testID	ID тесту	int	FK

Таблиця 2.9

Відповіді

Назва поля	Призначення поля	Тип даних	Ключ
answerID	ID відповіді	int	PK
answerName	Відповідь	varchar(max)	
isRight	Помітка про правильність відповіді	bit	
questionID	ID запитання	int	FK

Таблиця 2.10

Зв’язок результату тестування зі студентом

Назва поля	Призначення поля	Тип даних	Ключ
ID	ID зв’язка результату тестування зі студентом	int	PK
testID	ID тесту	int	FK
studentID	ID студента	int	FK
countRight	Кількість правильних відповідей	int	
dateTest	Дата	date	

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Програмне забезпечення отримує вхідні дані шляхом зчитування інформації з полів введення і таблиць з якими працює користувач:

- тип облікового запису користувача;
- логін і пароль користувача;
- масив даних для сортування;
- метод сортування даних;
- шлях до файлу з теоретичними відомостями;
- тестові запитання і відповіді на них.

Вихідні дані:

- кількість балів у гравця й у комп’ютера на даний момент.
- відсортований масив даних;
- результати аналізу алгоритму сортування;
- анімація сортування;
- результати проходження тестового контролю;
- дані облікових записів користувачів;
- теоретичні відомості.

2.7. Опис розробленого програмного продукту

2.7.1. Використані технічні засоби

Розроблюване програмне забезпечення створювалося, та призначене для ЕОМ за наступними технічними характеристиками:

- оперативний запам’ятовувальний пристрій об’ємом 2 ГБ або більше;
- процесор з тактовою частотою 1 ГГц або вище;
- жорсткий диск об’ємом не менше 20 ГБ;
- наявність маніпулятора-миші та клавіатури;

– монітор SVGA.

2.7.2. Використані програмні засоби

Під час розробки даного застосунку були використані наступні програмні засоби:

– Visual Studio 2019 Community;
– SQL Server 2019 Developer;
– Git, GitHub, GithubDesktop;
– .NET Framework 4.8.

Також були використані програмні можливості надані мовою програмування C#. Візуальна частина програмного додатку виконана за допомогою технології Windows Forms.

2.7.3. Виклик та завантаження програми

Оскільки розроблений додаток використовує базу даних для зберігання інформації, перед встановленням програми необхідно встановити СУБД SQL Server 2019 для успішного виконання запитів клієнтського додатка до БД.

Після цього програмний додаток необхідно інсталювати запустивши файл «InstallAlgoAnimation», та слідувати інструкціям інсталятора.

Після встановлення для запуску програмного додатку необхідно запустити на виконання файл «AlgoAnimation.exe».

2.7.4. Опис інтерфейсу користувача

Після запуску додатка з’являється вікно авторизації користувача (рис. 2.6), на якому розташовано текстові поля для введення логіну і паролю, кнопки входу і реєстрації, прапорець для відображення паролю, випадаючий список для вибору типу облікового запису.

Рис. 2.6. Вікно авторизації користувача

Якщо користувач не має облікового запису потрібно перейти у вікно реєстрації, для цього потрібно натиснути кнопку «Зареєструватись», після цього відкриється вікно реєстрації студента (рис. 2.7). В даному вікні розташовані поля для вводу даних студента (ПІБ, логін, пароль), випадаючий список для вибору групи до якої бажає долучитись студент, кнопки реєстрації та повернення до вікна авторизації.

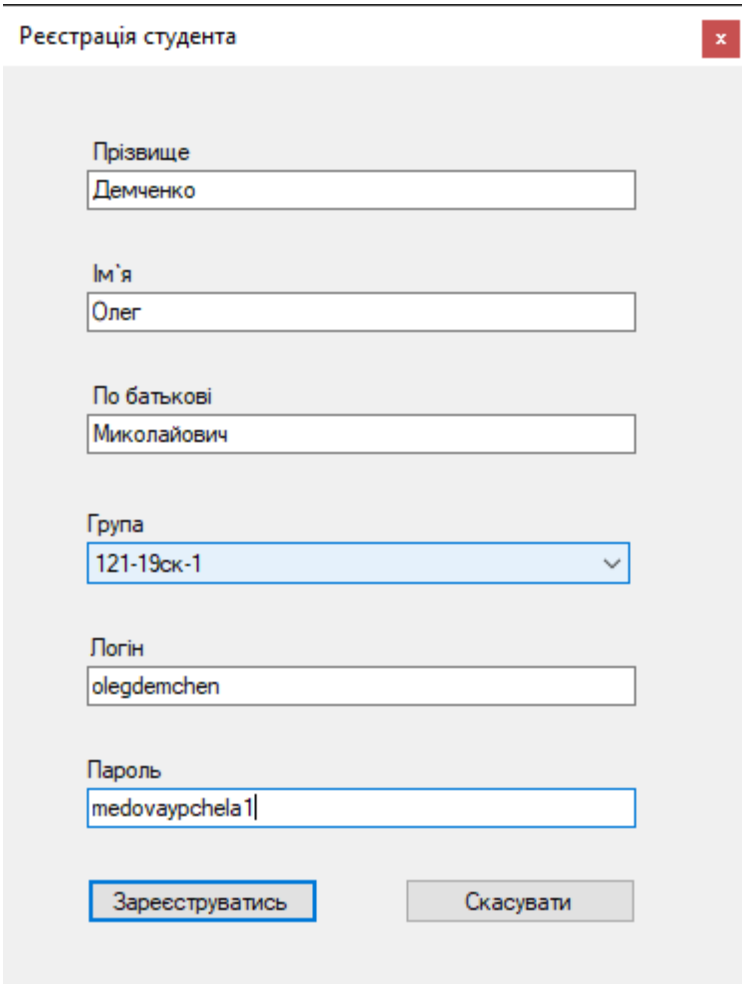

Рис. 2.7. Вікно реєстрації студента

Після введення даних студента необхідно натиснути кнопку «Зареєструватись», якщо введені дані є не коректними тобто поля вводу порожні або логін студента вже існує, з’явиться повідомлення про помилку. У випадку успішної реєстрації з’явиться повідомлення про реєстрацію даного студента. Після реєстрації студента, викладач який закріплений за обраною студентом групою має підтвердити реєстрацію в додатку.

Якщо користувач має обліковий запис він має можливість ввести облікові дані у вікні авторизації і натиснути кнопку «Вхід». У випадку введення не коректних даних тобто не правильного логіну або паролю буде відображено повідомлення про помилку. Якщо авторизація пройдена успішно користувач отримає доступ до додатка з урахуванням повноважень його облікового запису і з’явиться головне вікно програми (рис. 2.8). В даному вікні ліворуч розташована панель інструментів для налаштування анімації, в нижній частині

вікна присутній рядок стану для відображення інформації про користувача(ПІБ, тип облікового запису), праворуч розташована діаграма яка відображає данні вхідного масиву, у верхній частині вікна присутнє головне меню для управління додатком.

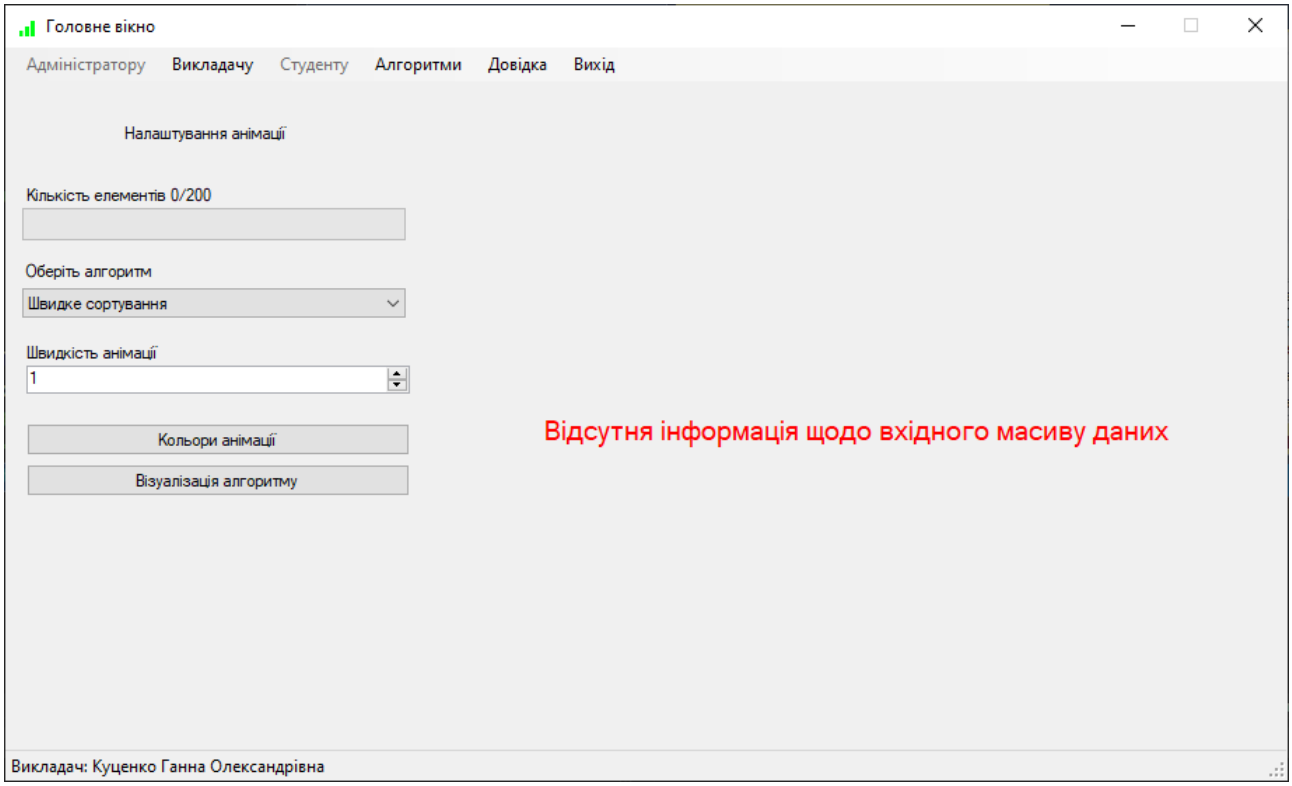

Рис. 2.8. Головне вікно програми

Після того як користувач отримує доступ до головного вікна програми він може виконувати будь-які дії в межах наданих йому повноважень. При цьому додаток блокує можливість доступу до певних пунктів головного меню, доступ до яких виходить за межі повноважень користувача.

Однією з головних функцій додатка є візуалізація роботи алгоритмів сортування, для того щоб згенерувати потрібну анімацію необхідно виконати налаштування анімації та згенерувати вхідний масив даних.

Для налаштування анімації в головному вікні програми за допомогою випадаючого списку потрібно обрати алгоритм для візуалізації та швидкість відтворення анімації за допомогою числового поля. Також потрібно натиснути на кнопку «Кольори анімації» для переходу у вікно налаштування кольорів

анімації (рис. 2.9). В даному вікні розташований перелік кольорів для кожного алгоритму сортування і кнопки «Зберегти», «Скасувати».

Рис. 2.9. Вікно налаштування кольорів анімації

Для того щоб змінити обраний колір потрібно натиснути на нього і відкриється діалогове вікно зміни кольору. Після внесення необхідних змін потрібно натиснути кнопку «Зберегти». Таким чином налаштування анімації завершено.

Для редагування вхідного масиву потрібно перейти до вікна редагування вхідного масиву обравши пункт меню «Алгоритми» - «Вхідні данні», після чого буде відображено вікно редагування вхідного масиву (рис. 2.10). В верхній частині даного вікна розташована діаграма даних вхідного масиву, в нижній частині вікна присутня панель інструментів для редагування вхідного масиву.

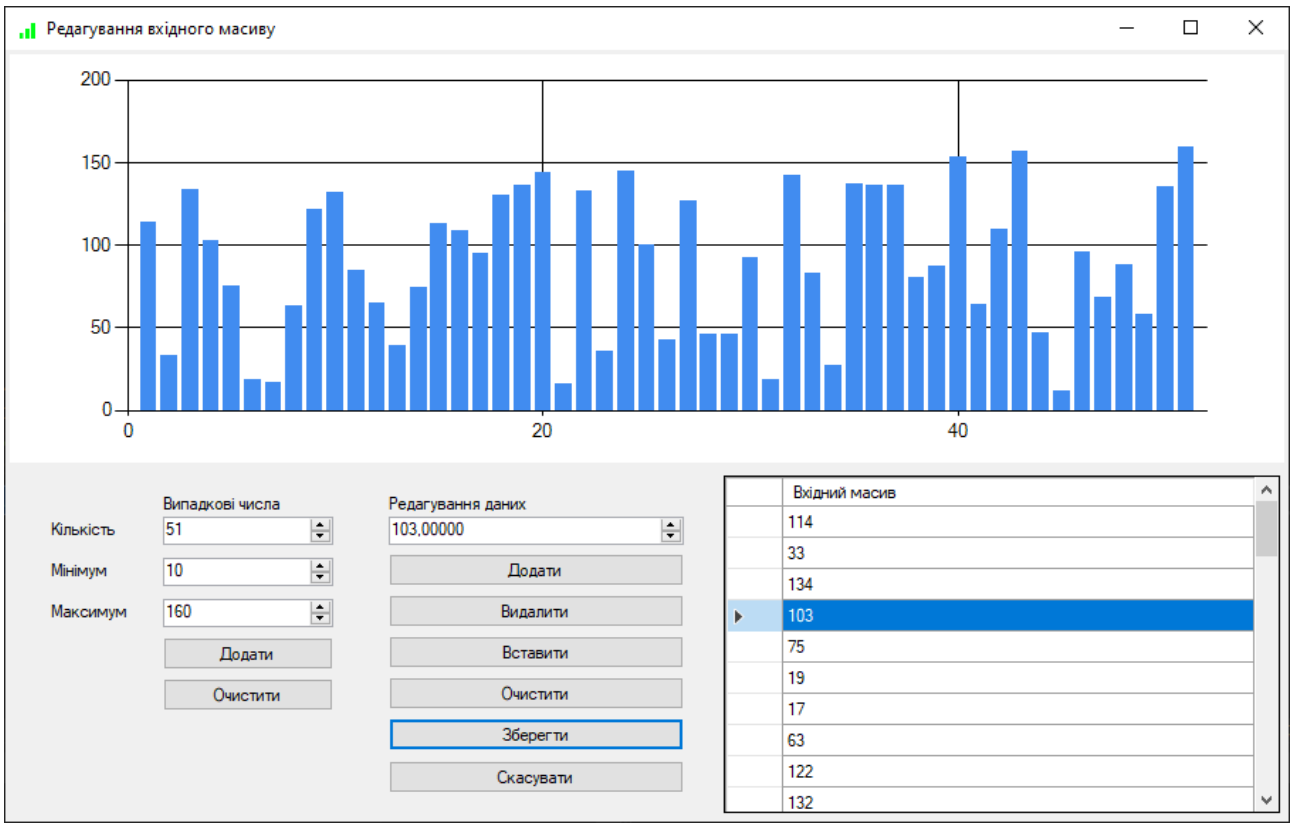

Рис. 2.10. Вікно редагування вхідного масиву

В даному вікні користувач може використовувати декілька варіантів створення вхідного масиву тобто вводити дані самостійно або генерувати випадкові числа, також можливо використовувати обидва варіанти разом.

Для генерації випадкових чисел потрібно ввести данні про кількість випадкових чисел та діапазон значень в текстові поля на панелі інструментів зліва, після цього натиснути кнопку «Додати». Якщо необхідно змінити значення полів генерації випадкових чисел на значення за замовченням користувач може натиснути кнопку «Очистити».

Для самостійного редагування вхідного масиву в панелі інструментів є список елементів масиву в правій частині вікна і текстове поле для редагування обраного елементу списку. Також для вставки чисел до масиву, додавання і видалення елементів є відповідні кнопки на панелі інструментів «Вставити», «Додати», «Видалити». Якщо необхідно очистити список елементів масиву потрібно натиснути кнопку «Очистити». Для збереження або виходу з вікна редагування вхідного масиву є кнопки «Зберегти» і «Скасувати». Після

повернення до головного вікна на панелі інструментів буде відображено на скільки заповнений вхідний масив також в правій частині з’явиться діаграма яка відображає елементи масиву.

Для візуалізації алгоритму необхідно в головному вікні програми натиснути кнопку «Візуалізація алгоритму», після цього відкриється вікно перегляду анімації обраного алгоритму (рис. 2.11). В даному вікні розташовано діаграму для відображення анімації в верхній частині вікна, панель управління відтворенням анімації, рядок стану для відображення інформації щодо анімації алгоритму(об’єм пам’яті, кількість перестановок, кількість перестановок однакових елементів, час виконання алгоритму).

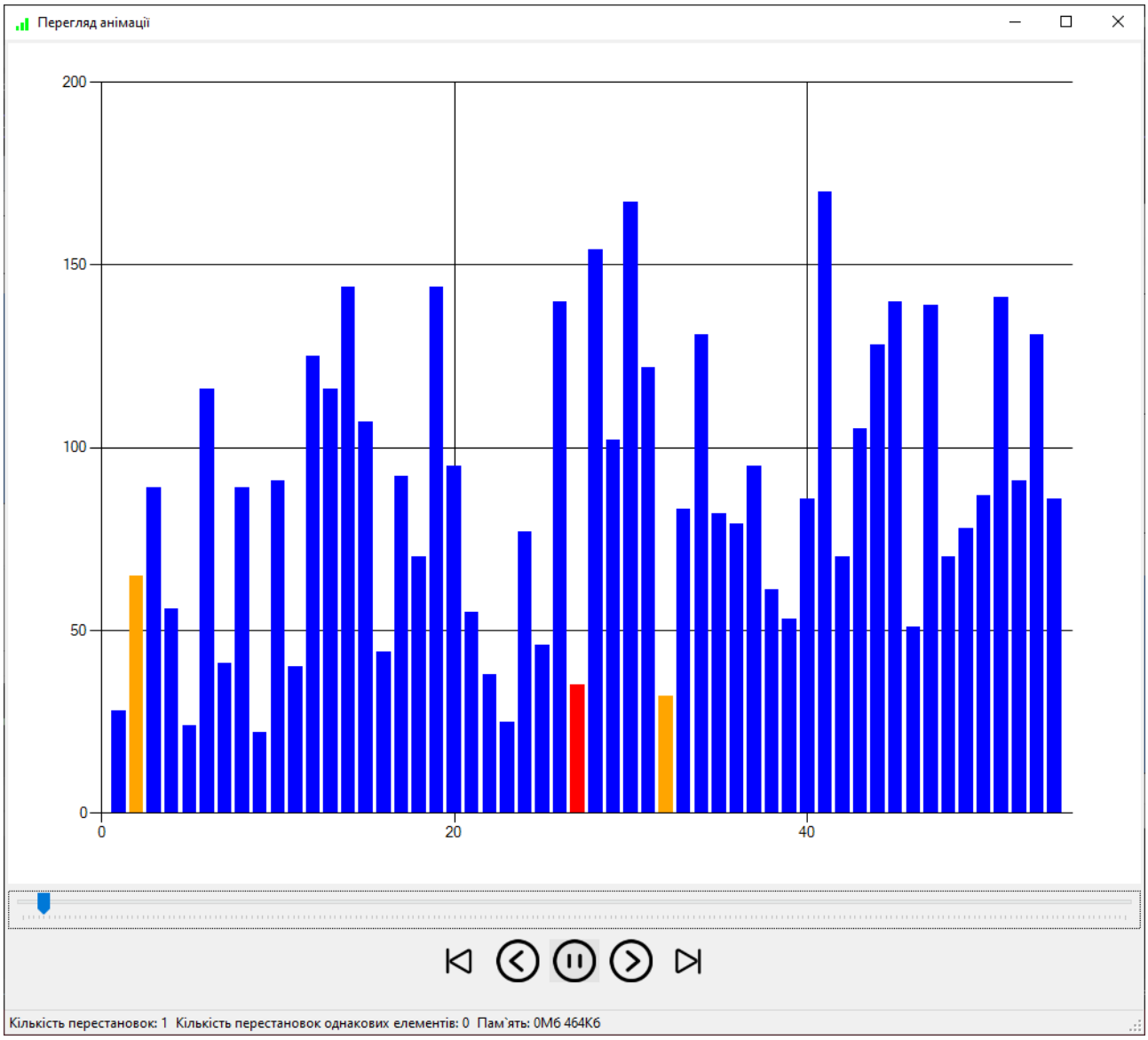

Рис. 2.11. Вікно перегляду анімації

Оскільки додаток розділяє користувачів на три типи (Адміністратор, Викладач, Студент), кожен з них має свої повноваження і може використовувати певні функції до яких не мають доступ інші користувачі.

Адміністратор:

Користувач типу адміністратор має можливість ведення списку студентських груп та інформації про облікові записи викладачів.

Після того як користувач пройшов авторизацію і отримав доступ до облікового запису адміністратора, для редагування інформації щодо студентських груп необхідно перейти до вікна студентські групи (рис. 2.12) обравши пункт меню «Адміністратору» - «Студентські групи» в головному вікні програми. У вікні для редагування інформації щодо студентських груп розташований список існуючих груп в правій частині вікна, панель інструментів і меню для управління даними.

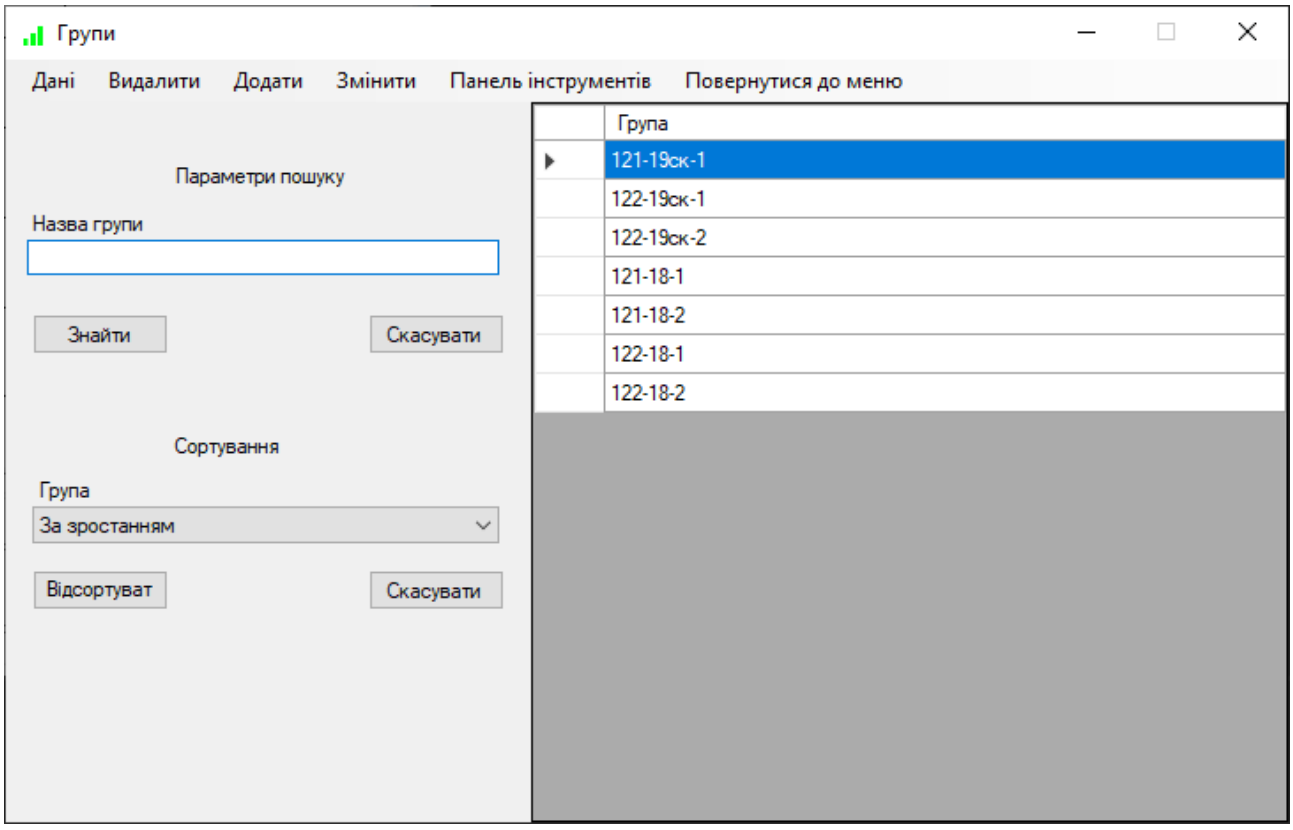

Рис. 2.12. Вікно студентські групи

Для додавання, видалення і редагування інформації щодо груп у верхній частині вікна присутні відповідні пункти меню «Додати», «Змінити»,

«Видалити». Збереження або оновлення даних виконується за допомогою пункту меню «Дані». Для повернення до головного вікна програми використовується пункт меню «Повернутися до меню».

Якщо необхідно виконати пошук або сортування елементів списку потрібно використовувати відповідні поля на панелі інструментів.

Для ведення даних викладачів потрібно перейти до вікна викладачі (рис. 2.13). В даному вікні розташована таблиця персональних даних кожного викладача, список закріплених за викладачем груп, панель інструментів, меню для управління даними.

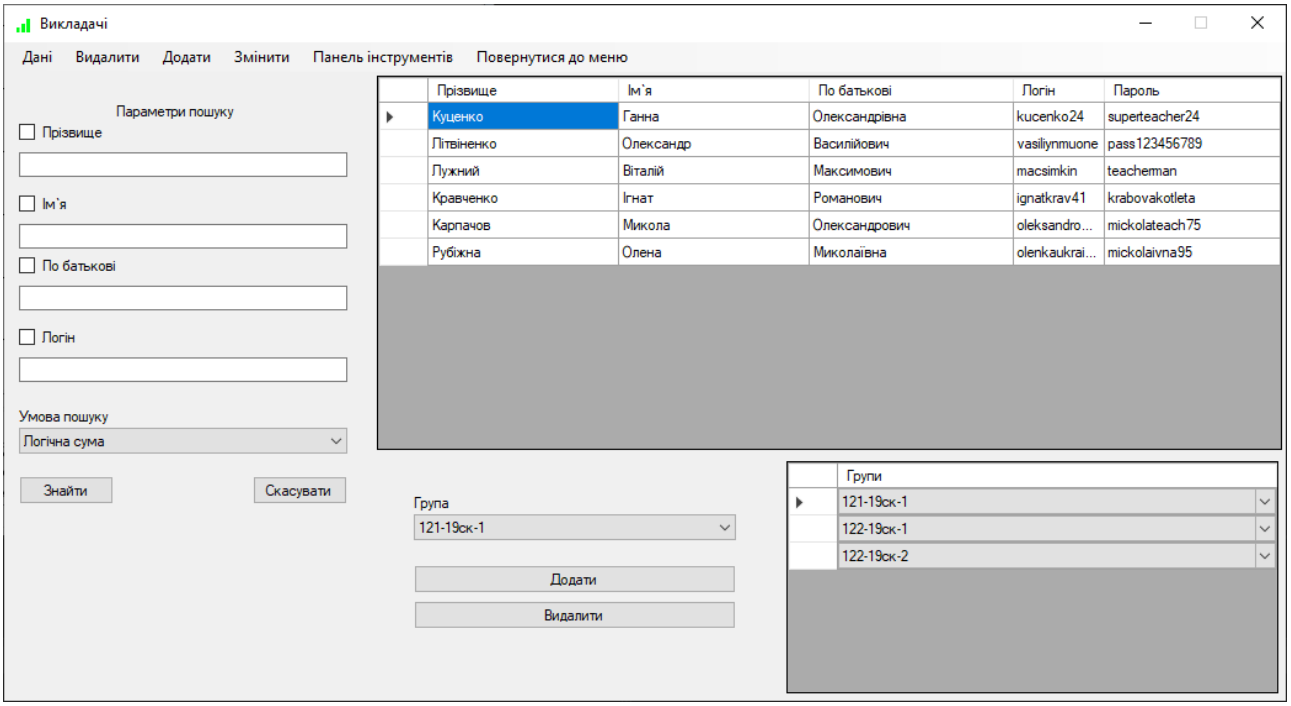

Рис. 2.13. Вікно викладачі

Для додавання, видалення і редагування інформації щодо викладачів у верхній частині вікна присутні відповідні пункти меню «Додати», «Змінити», «Видалити». Збереження або оновлення даних виконується за допомогою пункту меню «Дані». Для повернення до головного вікна програми використовується пункт меню «Повернутися до меню».

Якщо необхідно виконати пошук даних викладачів потрібно використовувати відповідні поля на панелі інструментів.

Окрім персональних даних за викладачами повинні бути закріплені студентські групи. Для закріплення групи за певним викладачем потрібно обрати в таблиці викладача і обрати групу з випадаючого списку в нижній частині вікна, після цього натиснути кнопку «Додати». Для видалення групи зі списку її потрібно виділити і натиснути кнопку «Видалити».

Викладач:

Користувач типу викладач має можливість створення тестового контролю, ведення теоретичних відомостей, ведення даних студентів.

Після того як користувач пройшов авторизацію і отримав доступ до облікового запису викладача, для розробки тестового контролю і запитань до тестів, потрібно перейти до вікна створення тестів (рис. 2.14) обравши пункт меню «Викладачу» - «Тести» - «Тести і запитання». В даному вікні розташований список тестів створених викладачем, панель інструментів для управління списком запитань до обраного тесту, текстове поле для перегляду і редагування запитань, меню для управління даними щодо тестів.

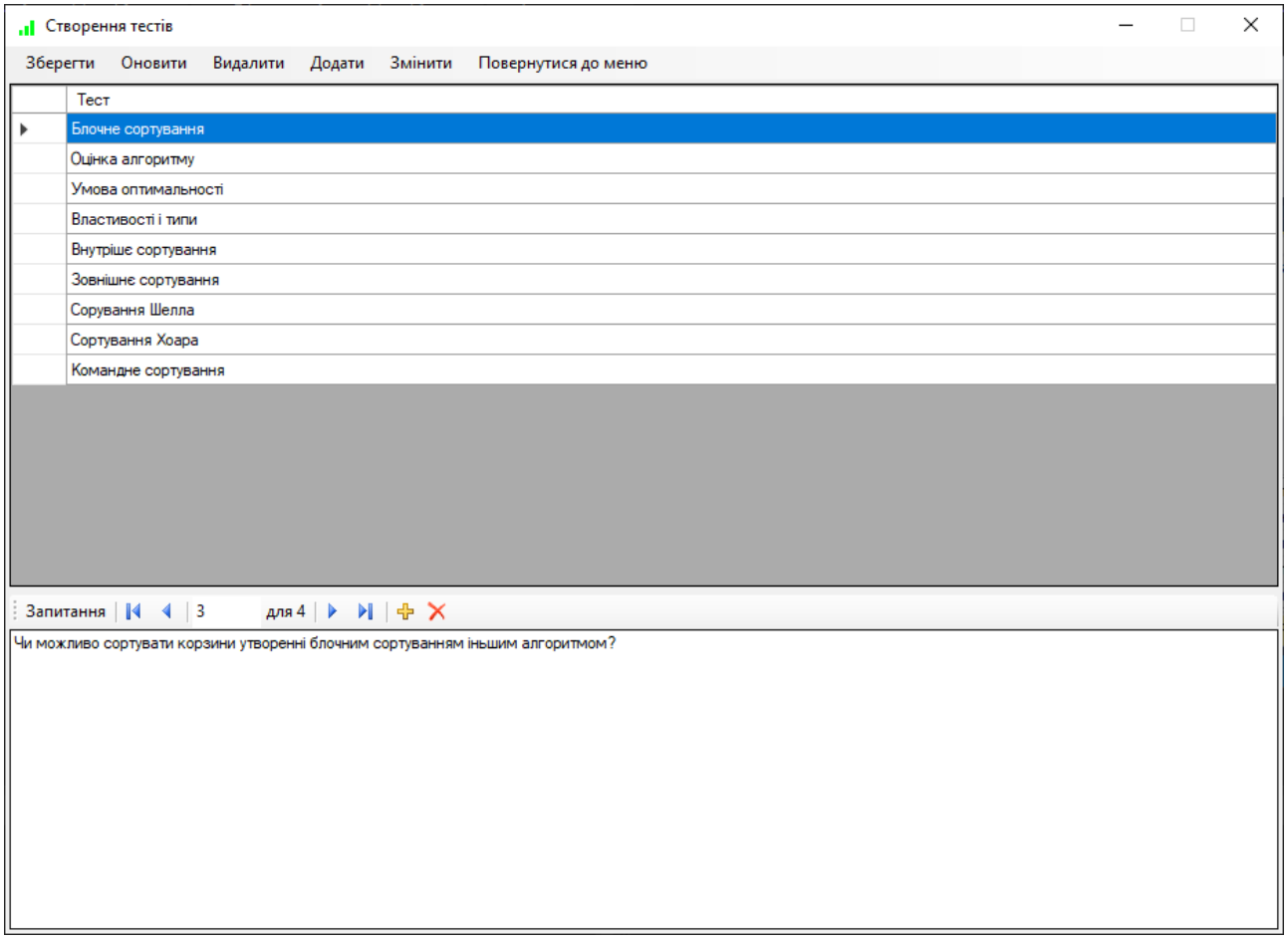

Рис. 2.14. Вікно створення тестів

Для додавання, видалення і редагування інформації щодо тестів у верхній частині вікна присутні відповідні пункти меню «Додати», «Змінити», «Видалити». Збереження або оновлення даних виконується за допомогою пунктів меню «Зберегти», «Оновити». Для повернення до головного вікна програми використовується пункт меню «Повернутися до меню».

Оскільки тестовий контроль передбачає наявність запитань до тестів і варіантів відповідей до них, необхідно перейти до вікна створення варіантів відповідей (рис. 2.15) для додавання відповіді на кожне запитання. Для переходу до вікна створення варіантів відповідей потрібно в головному вікні програми обрати пункт меню «Викладачу» - «Тести» - «Варіанти відповідей». Вданому вікні розташовано випадаючий список для вибору назви тесту, панелі інструментів для управління даними щодо запитань та варіантів відповідей до обраного тесту, текстові поля для редагування запитань і варіантів відповідей.

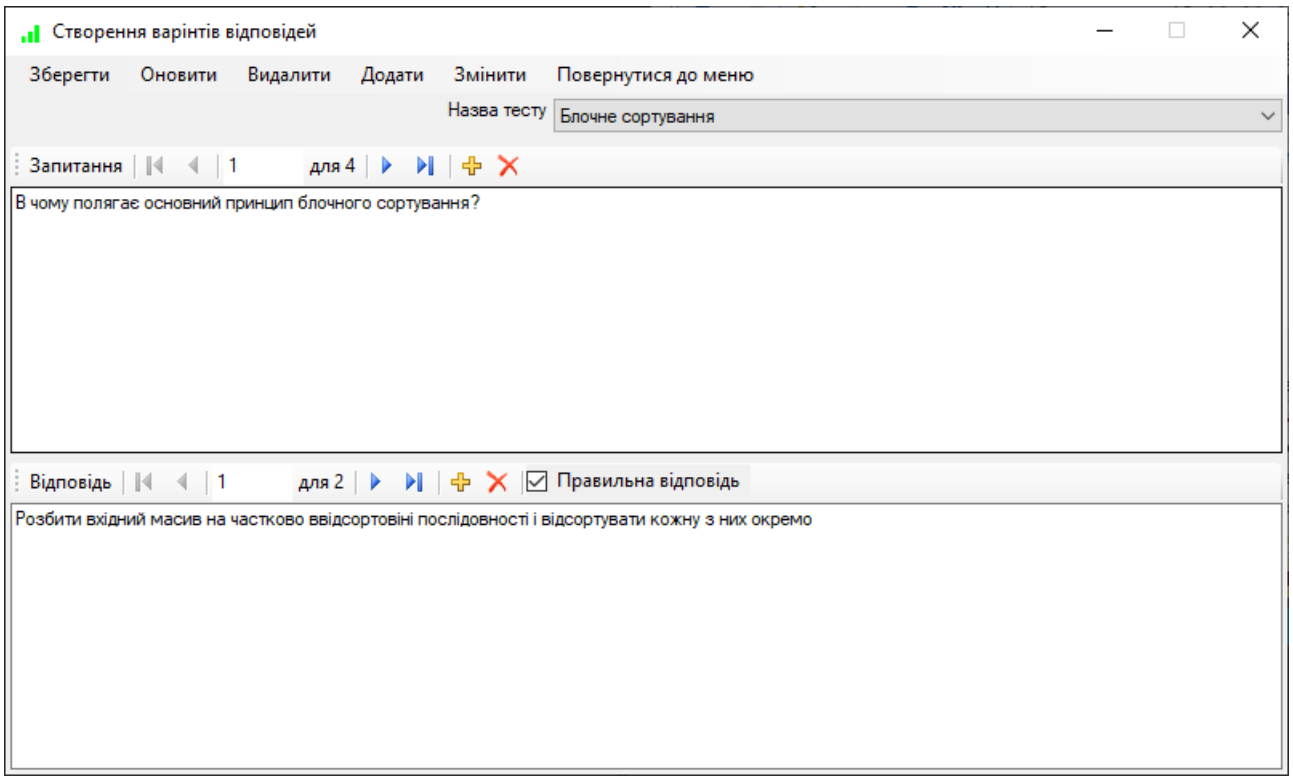

Рис. 2.15. Вікно створення варіантів відповідей

Для додавання, видалення і редагування запитань до тестів у верхній частині вікна присутні відповідні пункти меню «Додати», «Змінити»,

«Видалити». Збереження або оновлення даних виконується за допомогою пунктів меню «Зберегти», «Оновити». Для повернення до головного вікна програми використовується пункт меню «Повернутися до меню».

Окрім розробки тестового контролю для студентів викладач має можливість ведення даних самих студентів, для цього потрібно перейти до вікна студенти (рис. 2.16) обравши пункт меню «Викладачу» - «Студенти» в головному вікні програми. У вікні студенти розташовано панель інструментів для здійснення пошуку по записах, таблиці з персональними даними студентів і результатами проходження тестового контролю.

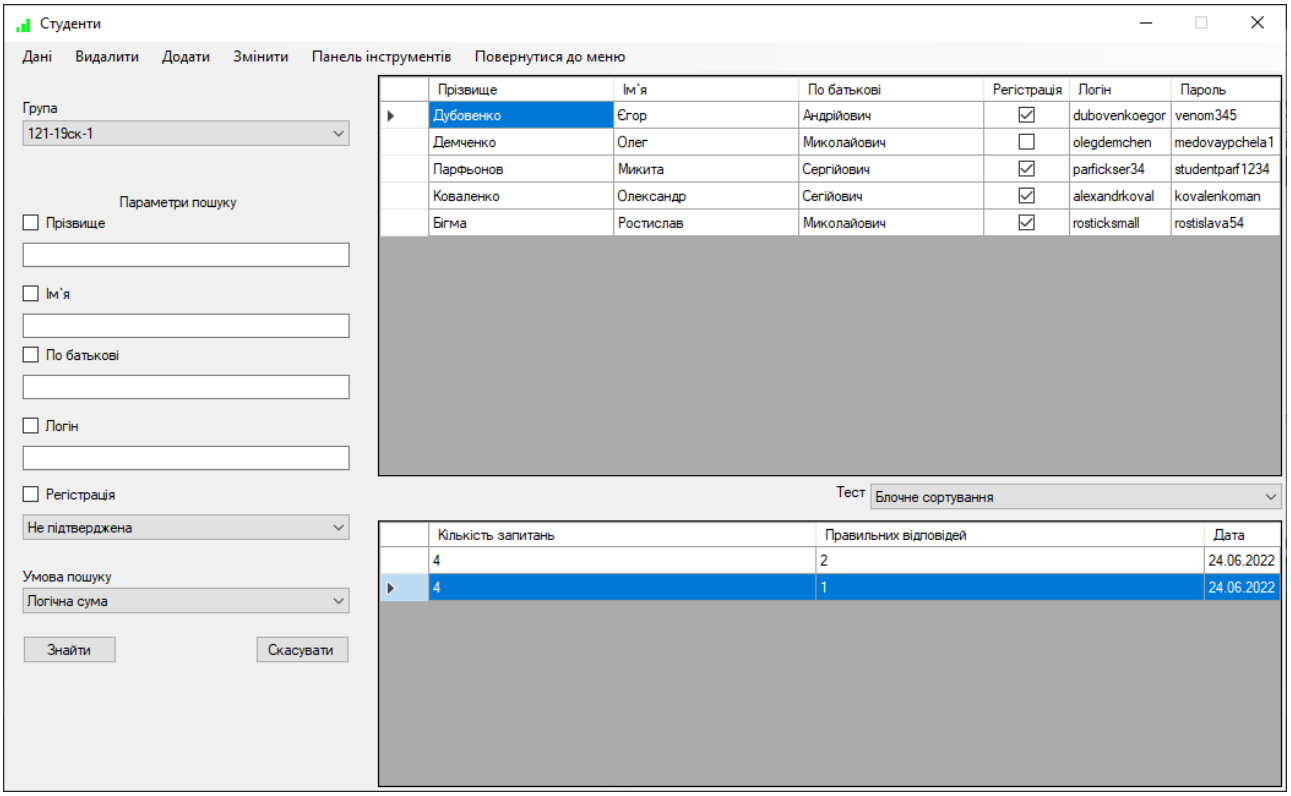

Рис. 2.16. Вікно студенти

Для додавання, видалення і редагування інформації щодо студентів у верхній частині вікна присутні відповідні пункти меню «Додати», «Змінити», «Видалити». Збереження або оновлення даних виконується за допомогою пунктів меню «Зберегти», «Оновити». Для повернення до головного вікна програми використовується пункт меню «Повернутися до меню». Пошук і фільтрація студентів за групами виконується за допомогою панелі інструментів, яка розташована в лівій частині вікна.

Викладач також має можливість створення теоретичних відомостей з якими будуть ознайомлюватись студенти. Для переходу до вікна теоретичні (рис. 2.17) відомості необхідно обрати пункт меню «Викладачу» - «Теоретичні відомості» в головному вікні програми. В даному вікні розташований список для перегляду назв теоретичних відомостей, панель інструментів, меню для управління даними щодо теоретичних відомостей.

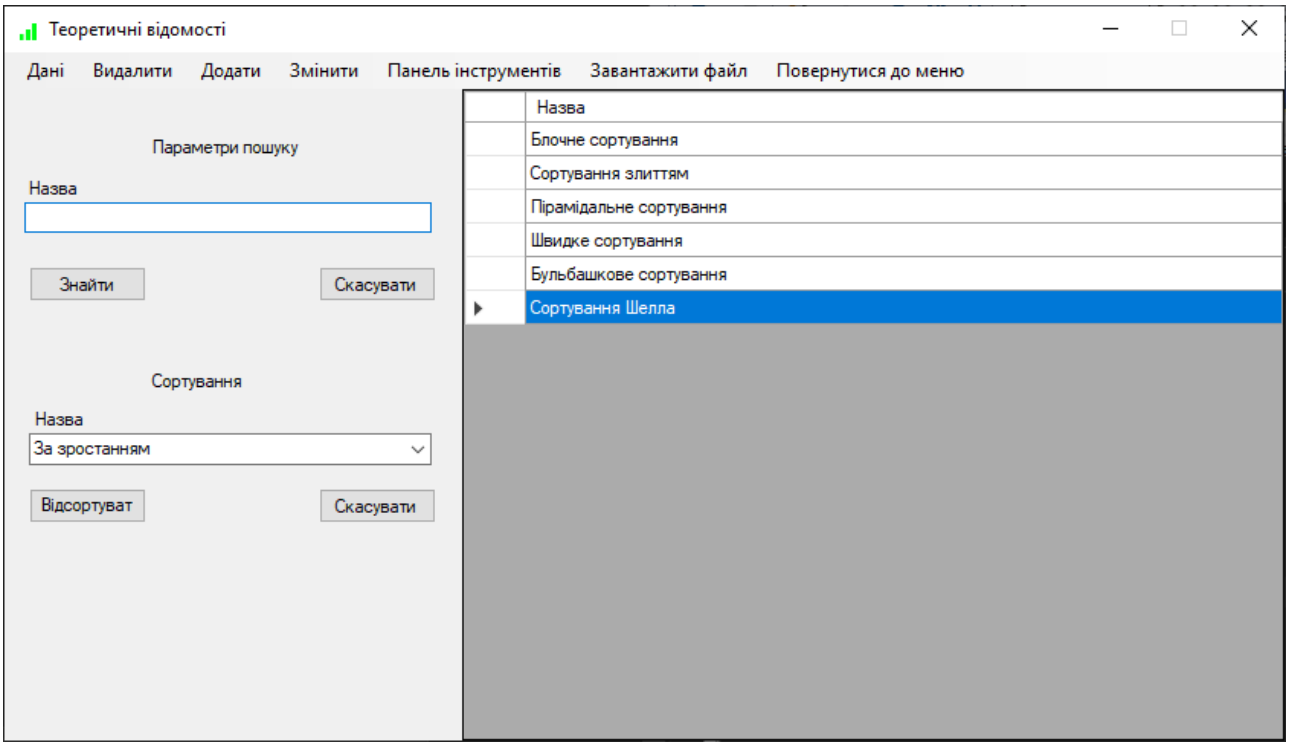

Рис. 2.17. Вікно теоретичні відомості

Для додавання, видалення і редагування інформації щодо теоретичних відомостей теоретичних відомостей у верхній частині вікна присутні відповідні пункти меню «Додати», «Змінити», «Видалити». Збереження або оновлення даних виконується за допомогою пункту меню «Дані». Для повернення до головного вікна програми використовується пункт меню «Повернутися до меню». Пошук і сортування даних виконується за допомогою панелі інструментів, яка розташована в лівій частині вікна. Якщо є необхідність завантажити файл з теоретичними відомостями потрібно обрати його у списку і натиснути пункт меню «Завантажити файл» в верхній частині вікна.

Студент:

Користувач студент має можливість проходити тестування розроблене викладачами, переглядати свої результати, переглядати теоретичні відомості.

Перед проходженням тестового контролю спочатку необхідно перейти до вікна вибору тестів і обрати тест, для цього потрібно обрати пункт меню «Студенту» - «Тестування». У вікні вибору тесту (рис. 2.18) розташовано список тестів кожного викладача закріпленого за студентською групою, текстові поля для відображення даних викладачів, панель інструментів для переміщення по списку викладачів, кнопки «Почати тестування» і «Скасувати».

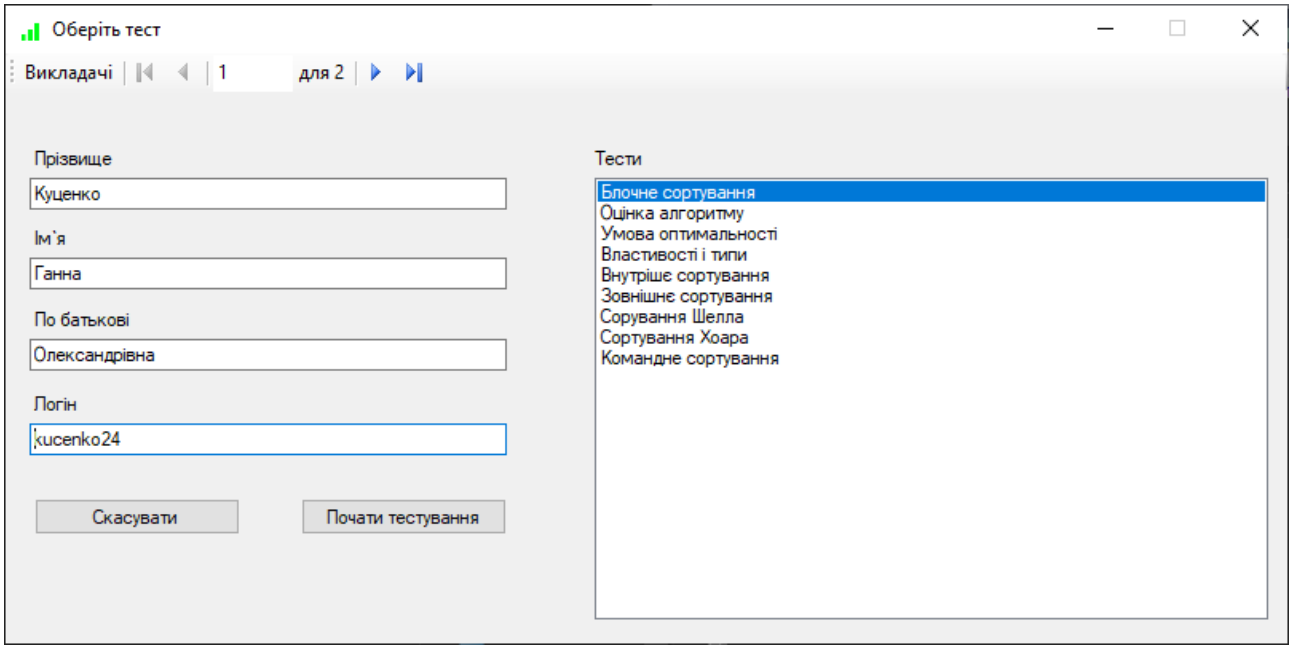

Рис. 2.18. Вікно вибору тесту

Для початку проходження тестового контролю потрібно обрати тест за допомогою списку і натиснути кнопку «Почати тестування», після цього відкриється вікно тестування (рис. 2.19). В даному вікні розташовано текстові поля для перегляду запитань і варіантів відповідей, панелі інструментів для переходу по списку запитань і варіантів відповідей.

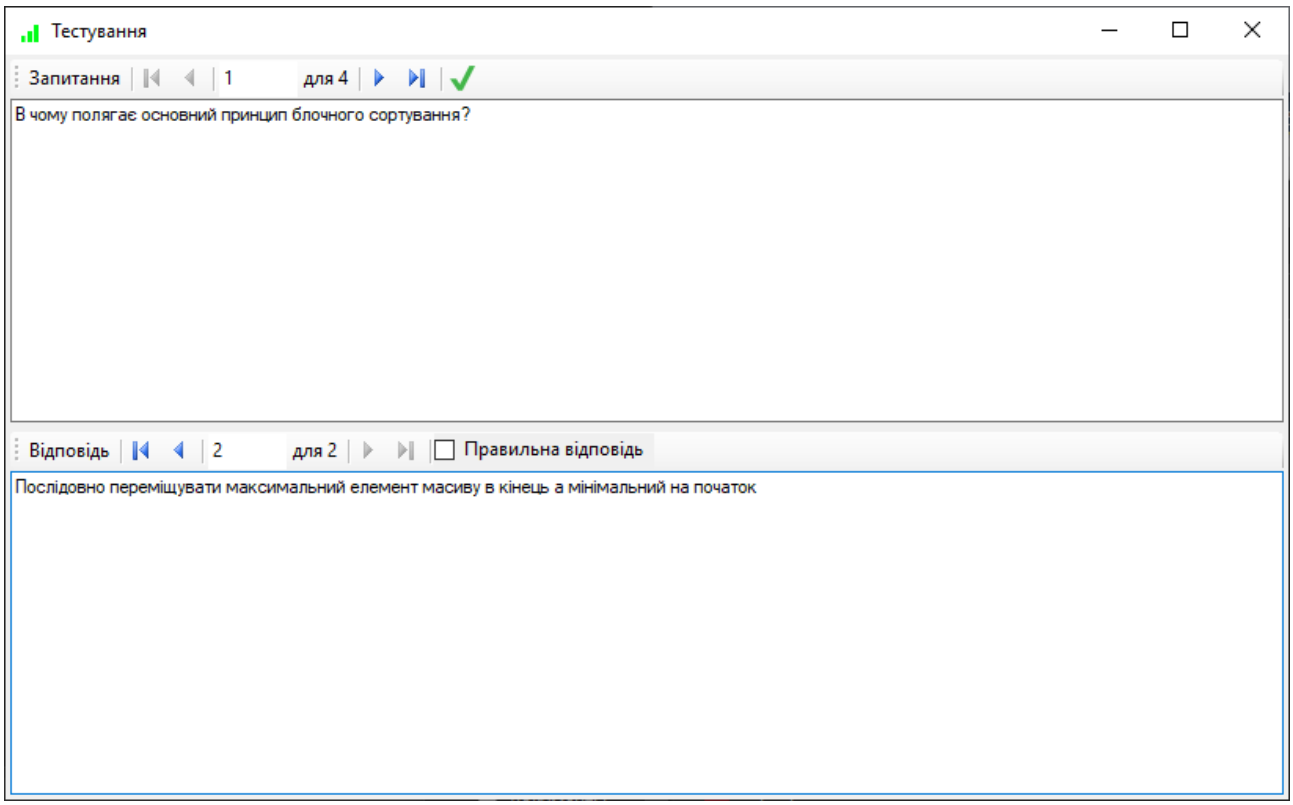

Рис. 2.19. Вікно тестування

Підчас перегляду варіантів відповідей до кожного запитання, існує необхідність відмічати правильні і не правильні відповіді, для цього на панелі управління списком варіантів відповідей, розташовано прапорець за допомогою якого студент може обирати правильні відповіді. Після проходження тестового контролю потрібно натиснути кнопку у вигляді галочки на пенделі управління списком запитань, таким чином результат проходження тестового контролю буде збережено і відображено у вигляді повідомлення студенту.

Окрім складання тестів студент може переглядати свої результати тестового контролю, для цього необхідно перейти до вікна результати тестування (рис. 2.20), обравши пункт меню «Студенту» - «Результати тестування». В даному вікні розташована таблиця для відображення результатів проходження тестового контролю обраного тесту, випадаючий список тестів обраного викладача, текстові поля для відображення даних викладачів, панель інструментів для переходу по списку викладачів.

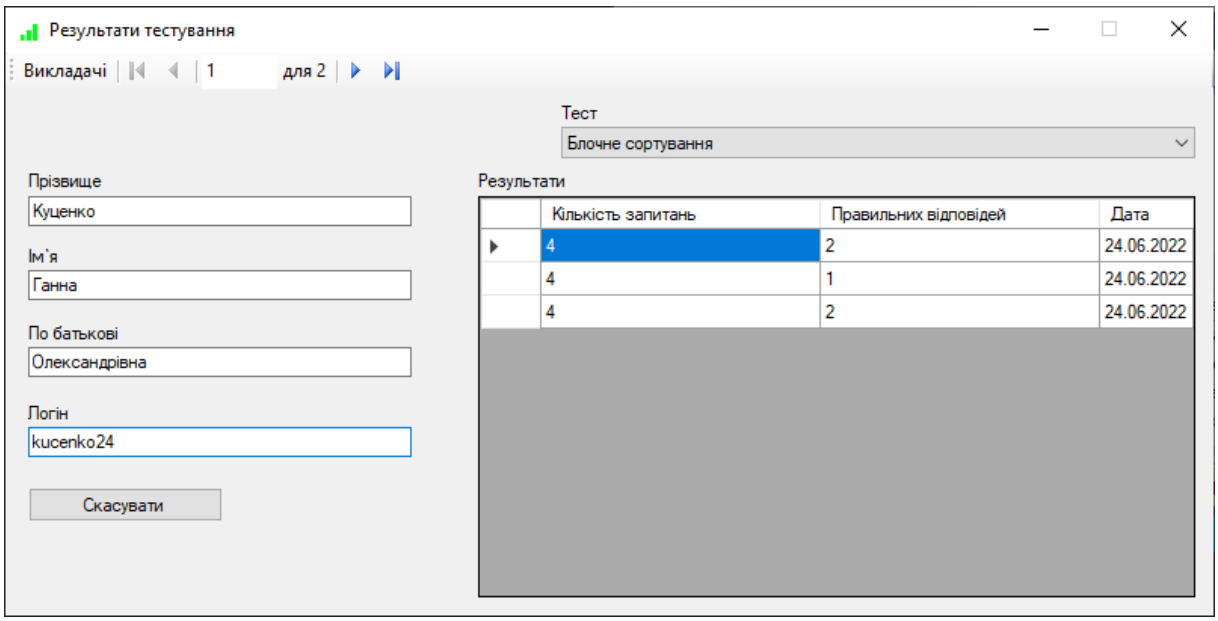

Рис. 2.20. Вікно результати тестування

Також студент має доступ до теоретичних відомостей які були розроблені викладачами закріпленим за відповідною групою студента. Для переходу до теоретичних відомостей необхідно обрати пункт меню «Студенту» - «Теоретичні відомості» в головному вікні програми. У вікні теоретичні відомості (рис. 2.21) розташовано список теорії розробленої обраним викладачем, текстові поля для відображення даних викладача, панель інструментів для переходу по списку викладачів, кнопки «Скасувати», «Завантажити файл».

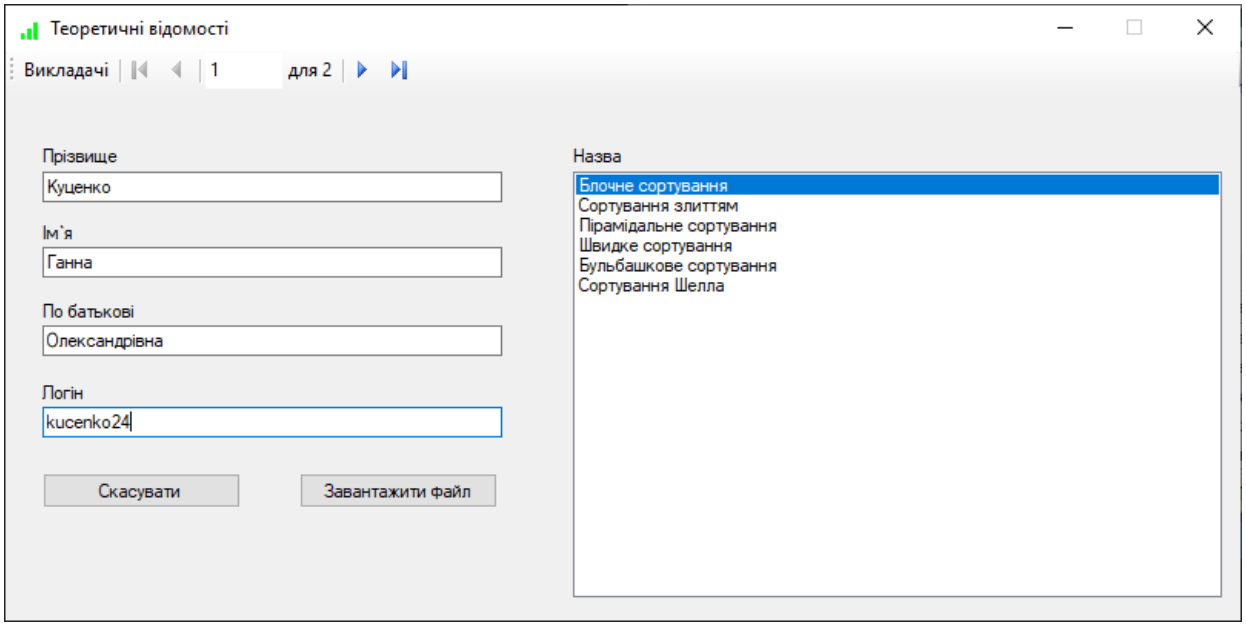

Рис. 2.21. Вікно теоретичні відомості

Для завантаження файлу з теоретичними відомостями необхідно обрати його у списку і натиснути кнопку «Завантажити файл», якщо потрібно повернутися до головного вікна програми необхідно натиснути кнопку «Скасувати»

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Обчислення трудомісткості розробки програмного забезпечення

Вхідні дані [15-16]:

q – передбачуване число операторів – 850.

C – коефіцієнт складності програми – 1,6.

p – коефіцієнт корекції програми в ході її розробки – 0,07.

B – коефіцієнт збільшення витрат – 1,2.

k – коефіцієнт кваліфікації програміста – 0,8.

$С_{ПР}$– середня годинна заробітна плата програміста, грн/год. Згідно зі статистичними даними з сайту DOU.ua [17], для програміста кваліфікації Junior Software Engineer на C#/.NET складає в середньому 925$ на місяць. При 40 годинному робочому тижні та чинному курсі Національного банку України 1 долар = 29,46 гривень, отримаємо значення 155 грн/год.

B_k– число виконавців – 1.

F_p– місячний фонд робочого часу (при 40 годинному робочому тижні $F_p = 176$ годин).

$С_{Мч}$ – вартість машино-години ЕОМ, грн/год – 4,7.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою (3.1):

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50),

t_u – витрати праці на дослідження алгоритму рішення задачі,

t_a – витрати праці на розробку блок-схеми алгоритму,

t_n – витрати праці на програмування по готовій блок-схемі,

$t_{\text{отл}}$ – витрати праці на налагодження програми на ЕОМ,

$t_{\text{д}}$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється. Умовне число операторів (підпрограм) визначається за формулою (3.2):

$$Q = q * C * (1 + p), \tag{3.2}$$

де q – передбачуване число операторів,

C – коефіцієнт складності програми,

p – коефіцієнт кореляції програми в ході її розробки.

$$Q = 850 * 1{,}6 * (1 + 0{,}07) = 1455$$

Витрати праці на вивчення опису задачі t_u визначається за формулою (3.3) з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q * B}{57 * k}, \text{людино} - \text{годин}, \tag{3.3}$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі,

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

$$t_u = \frac{1455 * 1{,}2}{57 * 0{,}8} = 29{,}1, \text{людино} - \text{годин}.$$

Витрати праці на розробку алгоритму рішення задачі обчислюються за формулою (3.4):

$$t_a = \frac{Q}{20 * k}, \text{людино} - \text{годин}. \tag{3.4}$$

$$t_a = \frac{1455}{20 * 0{,}8} = 90{,}95, \text{людино} - \text{годин}.$$

Витрати на складання програми по готовій блок-схемі визначаються за формулою (3.5):

$$t_n = \frac{Q}{25 * k}, \text{людино} - \text{годин}. \quad (3.5)$$

$$t_n = \frac{1455}{25 * 0{,}8} = 72{,}76, \text{людино} - \text{годин}.$$

Витрати праці на налагодження програми на ЕОМ обчислюються за формулою (3.6):

$$t_{\text{отл}} = \frac{Q}{5 * k}, \text{людино} - \text{годин}. \quad (3.6)$$

$$t_{\text{отл}} = \frac{1455}{5 * 0{,}8} = 363{,}8, \text{людино} - \text{годин}.$$

Витрати праці на підготовку документації обчислюються за формулою (3.7):

$$t_{\text{д}} = t_{\text{др}} + t_{\text{до}}, \quad (3.7)$$

де $t_{\text{др}}$ – трудомісткість підготовки матеріалів і рукопису визначається формулою (3.8).

$$t_{\text{др}} = \frac{Q}{15 * k}, \text{людино} - \text{годин}, \quad (3.8)$$

$$t_{\text{др}} = \frac{1455}{18 * 0{,}8} = 101{,}05, \text{людино} - \text{годин}.$$

$t_{до}$ – трудомісткість редагування, печатки й оформлення документації визначається формулою (3.9).

$$t_{до} = 0{,}75 * t_{др}, людино - годин. \quad (3.9)$$

$$t_{до} = 0{,}75 * 101{,}05 = 75{,}79, людино - годин.$$

Звідси витрати праці на підготовку документації:

$$t_{д} = 101{,}05 + 75{,}79 = 783{,}46, людино - годин.$$

Трудомісткість розробки ПЗ, відповідно, дорівнює:

$$t = 50 + 29{,}1 + 90{,}95 + 72{,}76 + 363{,}8 + 783{,}46 = 783{,}46, людино - годин$$

3.2. Обчислення витрат на створення програмного забезпечення

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ та визначаються за формулою (3.10):

$$К_{ПО} = З_{ЗП} + З_{МВ}, *грн.* \quad (3.10)$$

Заробітна плата виконавців визначається за формулою (3.11):

$$З_{ЗП} = t * С_{ПР}, грн, \quad (3.11)$$

де: t – загальна трудомісткість, людино-годин,

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

$$З_{ЗП} = 783{,}46 * 155 = 121436{,}48, грн.$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} * С_{мч}, грн, \quad (3.12)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год,

$С_{мч}$ - вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 363{,}8 * 4{,}7 = 1709{,}86, грн.$$

Відповідно, витрати на створення ПЗ становлять:

$$К_{ПО} = 121436{,}48 + 1709{,}86 = 123146{,}35, грн.$$

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k * F_p}, міс, \quad (3.13)$$

де B_k - число виконавців,

F_p - місячний фонд робочого часу.

$$T = \frac{783{,}46}{1 * 176} = 4{,}4, міс.$$

Висновки: трудомісткість розробленого програмного додатку становить 783,46 людино-години. Обчислена вартість роботи по створенню програми дорівнює 123146,35 гривень. Визначений затрачений час на створення програмного забезпечення становить 4,4 місяців.

ВИСНОВКИ

Метою кваліфікаційної роботи є створення програмного забезпечення призначеного спростити та пришвидшити процес навчання і забезпечати високу якість сприйняття теоретичного матеріалу студентами.

База даних, призначена для зберігання та маніпулювання даними, представлена у вигляді іменованої сукупності даних, організованих за певними правилами, та містить загальні правила опису, зберігання та маніпулювання даними.

Програма призначена для використання в державних та приватних навчальних закладах в процесі вивчення дисципліни алгоритми та структури даних і надає можливість виконувати наступні дії:

– авторизуватися в системі;

– створювати обліковий запис у системі;

– виконувати налаштування та перегляд анімації роботи алгоритмів сортування;

– створювати і редагувати облікові записи користувачів;

– створювати зв’язок між користувачами за допомогою додавання студентських груп;

– додавати теоретичні відомості до системи;

– завантажувати теоретичні відомості;

– розробляти тестовий контроль для студентів;

– проводити заходи тестового контролю;

– зберігати і переглядати результати проходження тестового контролю.

Актуальність поставленої задачі обумовлюється необхідністю надавати пояснення принципів роботи алгоритмів сортування студентам і періодично проводити тестовий контроль для перевірки якості засвоєння теоретичного матеріалу, а також складністю пояснень принципів роботи алгоритмів без наочного прикладу.

Структура програми являє собою клієнтський додаток, написаний мовою програмування високого рівня C#, що взаємодіє з базою даних «dbAlgo» за допомогою технології ADO.NET. База даних розроблена мовою SQL в системі управління базами даних Microsoft SQL Server 2019.

В «Економічному розділі» обчислено трудомісткість розробленого програмного додатку 783,46 людино-годин, та вартість роботи по створенню програми 123146,35 гривень. Визначено, що затрачений час на створення програмного додатку становить 4,4 місяців.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Алгоритми сортування. Вікіпедія / URL: https://ru.wikipedia.org/wiki (дата звернення 24.05.2022).

2. Паттерн MVC. Metaint.com / URL: https://metanit.com/sharp/wpf/22.1.php (дата звернення 24.05.2022).

3. Model-View-Controler (MVC) Explained. Atmosera / URL: https://www.atmosera.com/blog/model-view-viewmodel-mvvm-explained/ (дата звернення 24.05.2022).

4. Мэтью Мак-Дональд: Windows Forms в .NET 4.5 с примерами на C# 5.0 для профессионалов, 4-е издание: Windows Forms in .NET 4.5, 4th edition. — М.: «Вильямс», 2013. — 1024 с.

5. Мэтью Мак-Дональд: Windows Forms в .NET 3.5 с примерами на C# Windows Form in C# 2008: Windows Forms with .NET 3.5. — 2-ое. — М.: «Вильямс», 2008. — С. 25. — 928 с.

6. Симан М. Внедрение зависимостей в.NET. / Марк Симан., 2014. – 464 с. – (2). – (Для профессионалов).

7. Основні принципи ООП. / URL : https://training.epam.ua/#!/News/275?lang=ua (дата звернення 24.05.2022).

8. Хейлсберг А. Язык программирования C# / А. Хейлсберг. – Питер, 2016. – 784 с.

9. Command-line interface (CLI). / URL: https://www.techtarget.com/searchwindowsserver/definition/command-line-interface-CLI (дата звернення 25.05.2022).

10. Common Language Runtime (CLR). | Microsoft Docs / URL: https://docs.microsoft.com/en-us/dotnet/standard/clr (дата звернення 25.05.2022).

11. Microsoft Visual Studio | Visual Studio / URL: https://visualstudio.microsoft.com/ru/ (дата звернення 25.05.2022).

12. Технологія доступу до даних ADO.NET. | Microsoft Docs / URL: https://docs.microsoft.com/ru-ru/dotnet/framework/data/adonet/ (дата звернення 25.05.2022).

13. SQL:2008 now an approved ISO International Standard | SYBASE / URL : https://web.archive.org/web/20110628130925/http://iablog.sybase.com/paulley/2008/07/sql2008-now-an-approved-iso-international-standard/ (дата звернення 26.05.2022).

14. Технічна документація SQL Server | Microsoft Docs / URL: https://docs.microsoft.com/ru-ru/sql/sql-server/?view=sql-server-ver16/ (дата звернення 26.05.2022).

15. Методичні рекомендації до виконання кваліфікаційних робіт здобувачів першого рівня вищої освіти спеціальності 121 Інженерія програмного забезпечення / О.С. Шевцова, І.М. Удовик; Д : НТУ «Дніпровська політехніка», 2021. – 65 с.

16. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи ” / Уклад. О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 11 с.

17. Статистика зарплат програмістів в Україні | DOU.UA / URL: https://jobs.dou.ua/salaries/?period=2021-12&position=Middle%20SE&technology=C#/.NET&city=3/(дата звернення 07.06.2022).

КОД ПРОГРАМИ

Form_animation.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Diagnostics;
using System.Windows.Forms;
//using Algorithm;


namespace AlgoAnimation
{

   public partial class Form_animation : Form
   {
      public int QueqSortColorMid;
      public int QueqSortColorLeftRight;
      public int QueqSortColorBackground;

      public int ShellSortColorLeftRight;
      public int ShellSortColorBackground;

      public int HeapSortColorLeftRight;
      public int HeapSortColorBackground;

      public int MergeSortColorLeftRight;
      public int MergeSortColorBackground;

      public int BucketSortColorMid;
      public int BucketSortColorLeftRight;
      public int BucketSortColorBackground;
      public int BucketSortColorBucket1;
      public int BucketSortColorBucket2;
      public int BucketSortColorBucket3;
      public int BucketSortColorBucket4;
      public int BucketSortColorBucket5;

      int Index;
      double[] Array;


      Image play;
      Image stop;

      IAlgorithm algorithm;
      Algorithm Algo;
```

```
        int IndexBackground;
        int[] Colors;
        Dictionary<int, Analitic> Items;
        List<int> Keys;
        int LastKeyIndex;

        void AnimationStop()
        {
            this.pictureBoxPlay.BackgroundImage = this.play;
            this.timerTime.Stop();
        }
        void AnimationPlay()
        {
            this.pictureBoxPlay.BackgroundImage = this.stop;
            this.timerTime.Start();
        }

        void ShowItem(Analitic Item)
        {
            this.statusStripAnalitic.Items[0].Text = String.Format("Кількість перестановок: {0}",
Item.CountShifts.ToString());
            this.statusStripAnalitic.Items[1].Text = String.Format("Кількість перестановок однакових елементів: {0}", Item.CountEqualShifts.ToString());
            int[] memo = Item.MemoryToArray(2);
            this.statusStripAnalitic.Items[2].Text = string.Format("Пам`ять: {0}Мб {1}Кб", memo[0],
memo[1]);
        }
        int GetLeftKey(int Key, List<int> Keys)
        {
            for (int i = 0; i < Keys.Count - 1; i++)
            {
                if (Keys[i] < Key && Keys[i + 1] > Key) return i;
            }
            return -1;
        }
        void ShowLineData(LineData ld)
        {
            for (int i = 0; i < ld.Count; i++) chart1.Series[0].Points.AddY(ld[i]);
        }
        void ShowLineColor(int ColorBackground, int Count)
        {
            int i = 0;
            int j = Count - 1;
            do
            {
                chart1.Series[0].Points[i].Color = Color.FromArgb(ColorBackground);
                chart1.Series[0].Points[j].Color = Color.FromArgb(ColorBackground);

                i++;
                j--;
            } while (i <= j);
        }
        void ShowLineColor(LineColor lc, int[] Colors)
        {
            int i = 0;
            int j = lc.Count - 1;
```

```
            do
            {
                chart1.Series[0].Points[i].Color = Color.FromArgb(Colors[lc[i]]);
                chart1.Series[0].Points[j].Color = Color.FromArgb(Colors[lc[j]]);

                i++;
                j--;
            } while (i <= j);
        }

        public Form_animation(int interval, int Index, double[] Array)
        {
            InitializeComponent();
            this.play = Image.FromFile("play.png");
            this.stop = Image.FromFile("pause.png");
            this.pictureBoxPlay.BackgroundImage = this.play;
            ////////////////////////////////////////////////////////////
            this.timerTime.Interval = interval;
            this.Index = Index;
            this.Array = Array;

        }

        private void trackBar1_ValueChanged(object sender, EventArgs e)
        {
            TrackBar TR = (TrackBar)sender;
            ILine line = Algo[TR.Value];

            if (Items != null && Items.Count > 0)
            {
                if (Items.ContainsKey(line.ID))
                {
                    LastKeyIndex = this.Keys.IndexOf(line.ID);
                    this.ShowItem(Items[line.ID]);
                }
                else
                {
                    int ID = (line.ID > this.Keys[this.LastKeyIndex]) ? this.Keys[this.LastKeyIndex] :
this.Keys[this.LastKeyIndex - 1];
                    this.ShowItem(Items[ID]);
                }
            }

            if (TR.Value != TR.Maximum) this.statusStripAnalitic.Items[3].Text = String.Empty;
            else
            {
                TimeSpan Time = this.algorithm.Time;
                string strTime = string.Format("Час виконання: секунд {0} мілісекунд {1} тіків {2}",
Time.Seconds, Time.Milliseconds, Time.Ticks);
                this.statusStripAnalitic.Items[3].Text = strTime;
            }

            if (line.Type == LineType.LineData)
            {
                this.chart1.Series[0].Points.Clear();
                this.ShowLineData((LineData)line);
                this.ShowLineColor(Colors[IndexBackground], this.chart1.Series[0].Points.Count);
```

```
            return;
        }

        if (line.Type == LineType.LineColor)
        {
            this.ShowLineColor((LineColor)line, Colors);
            return;
        }

        LineGroup lg = (LineGroup)line;
        foreach (ILine l in lg)
        {
            if (l.Type == LineType.LineData)
            {
                this.chart1.Series[0].Points.Clear();
                this.ShowLineData((LineData)l);
                this.ShowLineColor(Colors[IndexBackground], this.chart1.Series[0].Points.Count);
            }
            else
            {
                this.ShowLineColor((LineColor)l, Colors);
            }
        }
    }
    private void Form_animation_Load(object sender, EventArgs e)
    {

        switch (Index)
        {
            case 0:
                QueqSort Queqsort = new QueqSort();
                Queqsort.SetColorBackground(this.QueqSortColorBackground);
                Queqsort.SetColorMid(this.QueqSortColorMid);
                Queqsort.SetColorLeftRight(this.QueqSortColorLeftRight);
                this.algorithm = Queqsort;
                break;
            case 1:
                HeapSort Heapsort = new HeapSort();
                Heapsort.SetColorBackground(this.HeapSortColorBackground);
                Heapsort.SetColorLeftRight(this.HeapSortColorLeftRight);
                this.algorithm = Heapsort;
                break;
            case 2:
                BucketSort Bucketsort = new BucketSort();
                Bucketsort.SetColorBackground(this.BucketSortColorBackground);
                Bucketsort.SetColorLeftRight(this.BucketSortColorLeftRight);
                Bucketsort.SetColorMid(this.BucketSortColorMid);
                Bucketsort.SetColorBucket1(this.BucketSortColorBucket1);
                Bucketsort.SetColorBucket2(this.BucketSortColorBucket2);
                Bucketsort.SetColorBucket3(this.BucketSortColorBucket3);
                Bucketsort.SetColorBucket4(this.BucketSortColorBucket4);
                Bucketsort.SetColorBucket5(this.BucketSortColorBucket5);
                this.algorithm = Bucketsort;
                break;
            case 3:
                ShellSort Shellsort = new ShellSort();
                Shellsort.SetColorBackground(this.ShellSortColorBackground);
```

```
            Shellsort.SetColorSelected(this.ShellSortColorLeftRight);
            this.algorithm = Shellsort;
            break;
        case 4:
            MergeSort Mergesort = new MergeSort();
            Mergesort.SetColorBackground(this.ShellSortColorBackground);
            Mergesort.SetColorLeftRight(this.HeapSortColorLeftRight);
            this.algorithm = Mergesort;
            break;
    }

    this.algorithm.DoSort(Array);
    this.Algo = (Algorithm)this.algorithm;


    this.Colors = this.algorithm.GetColors();
    this.IndexBackground = this.algorithm.IndexBackground;

    this.Items = this.Algo.Items;
    this.Keys = this.Items.Keys.ToList();

    this.trackBar1.Maximum = this.Algo.Count - 1;
    this.trackBar1.Minimum = 0;

    this.trackBar1_ValueChanged(this.trackBar1, e);

}


private void pictureBox_MouseMove(object sender, MouseEventArgs e)
{
    PictureBox picture = (PictureBox)sender;
    picture.BackColor = Color.FromArgb(-1842205);
}
private void pictureBox_MouseLeave(object sender, EventArgs e)
{
    PictureBox picture = (PictureBox)sender;
    picture.BackColor = Color.FromArgb(-986896);
}

private void timerTime_Tick(object sender, EventArgs e)
{
    Timer t = (Timer)sender;
    if (this.trackBar1.Value == this.trackBar1.Maximum)
    {
        this.AnimationStop();
        return;
    }
    this.trackBar1.Value++;
}

private void pictureBoxPlay_Click(object sender, EventArgs e)
{
    Image Background = this.pictureBoxPlay.BackgroundImage;
    if (Background.Equals(this.play))
    {
        this.AnimationPlay();
```

```
            }
            else
            {
                this.AnimationStop();
            }
        }
        private void pictureBoxFirst_Click(object sender, EventArgs e)
        {
            this.trackBar1.Value = this.trackBar1.Minimum;
        }
        private void pictureBoxLast_Click(object sender, EventArgs e)
        {
            this.trackBar1.Value = this.trackBar1.Maximum;
        }
        private void pictureBoxPrevious_Click(object sender, EventArgs e)
        {
            if (this.trackBar1.Value > this.trackBar1.Minimum) this.trackBar1.Value--;
        }
        private void pictureBoxNext_Click(object sender, EventArgs e)
        {
            if (this.trackBar1.Value < this.trackBar1.Maximum) this.trackBar1.Value++;
        }
    }
```

Form_authorization.cs

```
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace AlgoAnimation
{
    public partial class Form_authorization : Form
    {
        DataTable accounts;
        public Form_authorization()
        {
            InitializeComponent();

            DataColumn name = new DataColumn("name", typeof(string));
            DataColumn value = new DataColumn("value", typeof(string));

            accounts = new DataTable();
            accounts.Columns.AddRange(new DataColumn[] { name, value });

            DataRow admin = accounts.NewRow();
            admin["name"] = "admins";
            admin["value"] = "Адміністратор";
            accounts.Rows.Add(admin);

            DataRow teacher = accounts.NewRow();
```

```
            teacher["name"] = "teachers";
            teacher["value"] = "Викладач";
            accounts.Rows.Add(teacher);

            DataRow student = accounts.NewRow();
            student["name"] = "students";
            student["value"] = "Студент";
            accounts.Rows.Add(student);

            this.StartPosition = FormStartPosition.CenterScreen;
            this.Text = "Вхід";

        }

        private void Form_authorization_Load(object sender, EventArgs e)
        {
            this.button_authorization.Text = "Вхід";
            this.button_register.Text = "Зареєструватися";

            this.comboBox_accounts.DataSource = accounts;
            this.comboBox_accounts.ValueMember = "name";
            this.comboBox_accounts.DisplayMember = "value";
            this.comboBox_accounts.SelectedIndex = 0;
            this.comboBox_accounts.DropDownStyle = ComboBoxStyle.DropDownList;
        }

        private void button_authorization_Click(object sender, EventArgs e)
        {
            string diplomaConnectionString = ConfigurationManager.ConnectionStrings["diploma"].ConnectionString;
            string queryString = String.Format("select * from {0} where userName = '{1}' and pass = '{2}'", comboBox_accounts.SelectedValue.ToString(), textBox_login.Text, textBox_pass.Text);

            using (SqlConnection connection = new SqlConnection(
              diplomaConnectionString))
            {
                SqlCommand command = new SqlCommand(queryString, connection);
                command.Connection.Open();

                SqlDataReader resalt = command.ExecuteReader();
                DataRowView SelectedRow = (DataRowView)this.comboBox_accounts.SelectedItem;
                if (!resalt.HasRows) MessageBox.Show("Не вірний логін або пароль!", "Помилка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                else
                {
                    resalt.Read();
                    if (resalt.FieldCount == 8 && !resalt.GetBoolean(4))
                    {
                        MessageBox.Show("Не вірний логін або пароль!", "Помилка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                        return;
                    }
                    int UserID = resalt.GetInt32(0);
                    ///////Исправить///////////////
                    string UserType = SelectedRow["value"].ToString();
                    string Surnsme = resalt.GetString(1);
                    string UserName = resalt.GetString(2);
```

```
                string Patron = resalt.GetString(3);

                this.Hide();
                Form_main form = new Form_main(UserID, UserType, Surnsme, UserName, Patron);
                if (form.ShowDialog() == DialogResult.OK) this.Show();
                else this.Close();
            }
        }
    }

    private void button_register_Click(object sender, EventArgs e)
    {
        Form_registration form = new Form_registration();
        this.Hide();
        form.ShowDialog();
        this.Show();
    }

    private void checkBoxShowPass_CheckedChanged(object sender, EventArgs e)
    {
        CheckBox checkBox = (CheckBox)sender;
        if (checkBox.Checked) this.textBox_pass.PasswordChar = '\0';
        else this.textBox_pass.PasswordChar = 'x';
    }
  }
}
```

Сервіс для навігації

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Controls;

namespace WpfApp2TypeDiabet.Services
{
  public class NavigationService
  {
    //подія зміни сторінки
    public event Action<Page> OnPageChanged;
    //стек для збереження історії відвідувань
    private Stack<Page> navigationHistory;
    //метод для перевірки можливості повернутися на попередню сторінку
    public bool CanGoBack => navigationHistory.Any();
    //конструктор класу
    public NavigationService()
    {
      navigationHistory = new Stack<Page>();
    }
    //метод навігації на певну сторінку
    public void Navigate(Page page)
    {
      OnPageChanged?.Invoke(page);
      navigationHistory.Push(page);
    }
    //метод повернення на попередню сторінку в історії відвідувань
```

```
        public void GoBack()
        {
            navigationHistory.Pop();
            var page = navigationHistory.Peek();
            OnPageChanged?.Invoke(page);
        }
    }
}
```

Form_colors.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace AlgoAnimation
{
    public partial class Form_colors : Form
    {
        public Form_colors()
        {
            InitializeComponent();
        }

        public int QueqSortColorMid
        {
            get { return this.pictureBoxQueqSortMid.BackColor.ToArgb(); }
            set { this.pictureBoxQueqSortMid.BackColor = Color.FromArgb(value); }
        }
        public int QueqSortColorLeftRight
        {
            get { return this.pictureBoxQueqSortLeftRight.BackColor.ToArgb(); }
            set { this.pictureBoxQueqSortLeftRight.BackColor = Color.FromArgb(value); }
        }

        public int QueqSortColorBackground
        {
            get { return this.pictureBoxQueqSortBackground.BackColor.ToArgb(); }
            set { this.pictureBoxQueqSortBackground.BackColor = Color.FromArgb(value); }
        }

        public int ShellSortColorLeftRight
        {
            get { return this.pictureBoxShellSortLeftRight.BackColor.ToArgb(); }
            set { this.pictureBoxShellSortLeftRight.BackColor = Color.FromArgb(value); }
        }

        public int ShellSortColorBackground
        {
            get { return this.pictureBoxShellSortBackground.BackColor.ToArgb(); }
            set { this.pictureBoxShellSortBackground.BackColor = Color.FromArgb(value); }
```

```
}

public int HeapSortColorLeftRight
{
   get { return this.pictureBoxHeapSortLeftRight.BackColor.ToArgb(); }
   set { this.pictureBoxHeapSortLeftRight.BackColor = Color.FromArgb(value); }
}

public int HeapSortColorBackground
{
   get { return this.pictureBoxHeapSortBackground.BackColor.ToArgb(); }
   set { this.pictureBoxHeapSortBackground.BackColor = Color.FromArgb(value); }
}

public int MergeSortColorLeftRight
{
   get { return this.pictureBoxMergeSortLeftRight.BackColor.ToArgb(); }
   set { this.pictureBoxMergeSortLeftRight.BackColor = Color.FromArgb(value); }
}

public int MergeSortColorBackground
{
   get { return this.pictureBoxMergeSortBackground.BackColor.ToArgb(); }
   set { this.pictureBoxMergeSortBackground.BackColor = Color.FromArgb(value); }
}

public int BucketSortColorMid
{
   get { return this.pictureBoxBucketSortMid.BackColor.ToArgb(); }
   set { this.pictureBoxBucketSortMid.BackColor = Color.FromArgb(value); }
}
public int BucketSortColorLeftRight
{
   get { return this.pictureBoxBucketSortLeftRight.BackColor.ToArgb(); }
   set { this.pictureBoxBucketSortLeftRight.BackColor = Color.FromArgb(value); }
}

public int BucketSortColorBackground
{
   get { return this.pictureBoxBucketSortBackground.BackColor.ToArgb(); }
   set { this.pictureBoxBucketSortBackground.BackColor = Color.FromArgb(value); }
}

public int BucketSortColorBucket1
{
   get { return this.pictureBoxBucketSortBucket1.BackColor.ToArgb(); }
   set { this.pictureBoxBucketSortBucket1.BackColor = Color.FromArgb(value); }
}
public int BucketSortColorBucket2
{
   get { return this.pictureBoxBucketSortBucket2.BackColor.ToArgb(); }
   set { this.pictureBoxBucketSortBucket2.BackColor = Color.FromArgb(value); }
}
public int BucketSortColorBucket3
{
   get { return this.pictureBoxBucketSortBucket3.BackColor.ToArgb(); }
   set { this.pictureBoxBucketSortBucket3.BackColor = Color.FromArgb(value); }
```

```
        }
        public int BucketSortColorBucket4
        {
            get { return this.pictureBoxBucketSortBucket4.BackColor.ToArgb(); }
            set { this.pictureBoxBucketSortBucket4.BackColor = Color.FromArgb(value); }
        }
        public int BucketSortColorBucket5
        {
            get { return this.pictureBoxBucketSortBucket5.BackColor.ToArgb(); }
            set { this.pictureBoxBucketSortBucket5.BackColor = Color.FromArgb(value); }
        }

        private void buttonOK_Click(object sender, EventArgs e)
        {
            this.DialogResult = DialogResult.OK;
        }

        private void pictureBoxStartColorDialog_Click(object sender, EventArgs e)
        {
            PictureBox picture = (PictureBox)sender;
            this.colorDialogColor.Color = picture.BackColor;
            if (this.colorDialogColor.ShowDialog() == DialogResult.OK) picture.BackColor = this.colorDialogColor.Color;
        }

        private void Form_colors_Load(object sender, EventArgs e)
        {

        }
    }
}
```

Form_groups.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace AlgoAnimation
{
    public partial class Form_groups : Form
    {
        public Form_groups()
        {
            InitializeComponent();

            DataTable TableSort = new DataTable("TableSort");
            DataColumn value = new DataColumn("value", typeof(System.String));
            DataColumn text = new DataColumn("text", typeof(System.String));
            TableSort.Columns.AddRange(new DataColumn[] { value, text });
            TableSort.PrimaryKey = new DataColumn[] { value };
```

```
            DataRow up = TableSort.NewRow();
            up[0] = "ASC";
            up[1] = "За зростанням";
            TableSort.Rows.Add(up);

            DataRow down = TableSort.NewRow();
            down[0] = "DESC";
            down[1] = "За спаданням";
            TableSort.Rows.Add(down);

            this.comboBoxSort.DataSource = TableSort;
            this.comboBoxSort.ValueMember = "value";
            this.comboBoxSort.DisplayMember = "text";
            this.comboBoxSort.SelectedIndex = 0;
        }

        private void GroupAdd()
        {
            ////////////////////////////////добавление групы//////////////
            Form_groups_add form = new Form_groups_add("Додати групу","Додати");
            if (form.ShowDialog() == DialogResult.OK)
            {
                DataSetGroups.groupsRow row =
(DataSetGroups.groupsRow)((DataRowView)this.groupsBindingSource.AddNew()).Row;
                row.groupName = form.Group;
                this.groupsBindingSource.EndEdit();
            }
        }
        private void GroupChenge()
        {
            if (this.groupsBindingSource.Current != null)
            {
                DataSetGroups.groupsRow row =
(DataSetGroups.groupsRow)((DataRowView)this.groupsBindingSource.Current).Row;
                Form_groups_add form = new Form_groups_add("Редагувати групу", "Зберегти");
                form.Group = row.groupName;
                if (form.ShowDialog() == DialogResult.OK)
                {
                    row.groupName = form.Group;
                    this.groupsBindingSource.EndEdit();
                }
            }
        }
        private void GroupDelete()
        {
            if (this.groupsBindingSource.Current != null)
            {
                ((DataRowView)this.groupsBindingSource.Current).Row.Delete();
            }
            this.groupsBindingSource.EndEdit();
        }

        private void Form_groups_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"dataSetGroups.groups". При необходимости она может быть перемещена или удалена.
```

```
            this.groupsTableAdapter.Fill(this.dataSetGroups.groups);
        }

        private void buttonSearch_Click(object sender, EventArgs e)
        {
            string strSearch = string.Format("groupName = '{0}'", this.textBoxSearch.Text);
            this.groupsBindingSource.Filter = strSearch;
        }

        private void buttonCancel_Click(object sender, EventArgs e)
        {
            this.groupsBindingSource.RemoveFilter();
        }

        private void buttonSort_Click(object sender, EventArgs e)
        {
            this.groupsBindingSource.Sort = String.Format("groupName {0}",
this.comboBoxSort.SelectedValue);
        }

        private void buttonSortCancel_Click(object sender, EventArgs e)
        {
            this.groupsBindingSource.RemoveSort();
        }

        private void зберегтиToolStripMenuItem_Click(object sender, EventArgs e)
        {

            this.groupsTableAdapter.Update(this.dataSetGroups.groups);
        }

        private void оновитиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.groupsTableAdapter.Fill(this.dataSetGroups.groups);
        }

        private void додатиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.GroupAdd();
        }

        private void змінитиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.GroupChenge();
        }

        private void видалитиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.GroupDelete();
        }

        private void панельІнструментівToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.splitContainer1.Panel1Collapsed = !this.splitContainer1.Panel1Collapsed;
        }

        private void повернутисьДоМенюToolStripMenuItem_Click(object sender, EventArgs e)
```

```
            {
                this.DialogResult = DialogResult.OK;
            }
        }
    }
```

Form_groups_add.cs

```
    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.Data;
    using System.Drawing;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;
    using System.Windows.Forms;

    namespace AlgoAnimation
    {
        public partial class Form_groups_add : Form
        {
            public Form_groups_add(string Title, string ButtonOK)
            {
                InitializeComponent();
                this.Text = Title;
                this.buttonOK.Text = ButtonOK;
            }

            public string Group { get { return this.textBoxGroup.Text; } set { this.textBoxGroup.Text =
value; } }

            private void buttonOK_Click(object sender, EventArgs e)
            {
                this.DialogResult = DialogResult.OK;
            }

            private void buttonCancel_Click(object sender, EventArgs e)
            {

            }
        }
    }
```

Form_input_array.cs

```
    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.Data;
    using System.Drawing;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;
    using System.Windows.Forms;

    namespace AlgoAnimation
```

```
{
    public partial class Form_input_array : Form
    {
        public List<double> Array;
        public Form_input_array(List<double> Array)
        {
            InitializeComponent();
            this.Array = new List<double>(Array);
        }

        private void DataGridCountChanged(object sender)
        {
            int max = 200;
            DataGridView dataGrid = (DataGridView)sender;
            this.numericUpDownCount.Maximum = max - dataGrid.Rows.Count;
            bool OverFlow = dataGrid.Rows.Count >= max;
            this.buttonAdd.Enabled = !OverFlow;
            this.buttonInsert.Enabled = !OverFlow;
        }

        private void ChartRefresh(object sender)
        {
            DataGridView dataGrid = (DataGridView)sender;
            this.chart1.Series[0].Points.Clear();
            foreach (DataGridViewRow row in dataGrid.Rows)
this.chart1.Series[0].Points.AddY(row.Cells[0].Value);
        }


        private void buttonAdd_Click(object sender, EventArgs e)
        {
            DataGridView dataGrid = this.dataGridViewInputArray;
            if (dataGrid != null)
                dataGrid.Rows.Add(new object[] { 1 });
        }

        private void buttonDelete_Click(object sender, EventArgs e)
        {
            DataGridView dataGrid = this.dataGridViewInputArray;
            if (dataGrid != null && dataGrid.CurrentRow != null)
                dataGrid.Rows.Remove(dataGrid.CurrentRow);
        }

        private void buttonInsert_Click(object sender, EventArgs e)
        {
            DataGridView dataGrid = this.dataGridViewInputArray;
            if (dataGrid != null && dataGrid.CurrentRow != null)
                dataGrid.Rows.Insert(dataGrid.CurrentRow.Index, new object[] { 1 });
            else
                this.buttonAdd_Click(sender, e);
        }

        private void numericUpDownArrayItem_ValueChanged(object sender, EventArgs e)
        {
            DataGridView dataGrid = this.dataGridViewInputArray;
            if (dataGrid != null && dataGrid.CurrentRow != null)
            {
```

```
            dataGrid.CurrentRow.Cells[0].Value = this.numericUpDownArrayItem.Value;
            this.ChartRefresh(dataGrid);
        }
    }

    private void dataGridViewInputArray_SelectionChanged(object sender, EventArgs e)
    {
        DataGridView dataGrid = this.dataGridViewInputArray;
        if (dataGrid != null && dataGrid.CurrentRow != null)
            this.numericUpDownArrayItem.Value =
decimal.Parse(dataGrid.CurrentRow.Cells[0].Value.ToString());
    }

    private void buttonClear_Click(object sender, EventArgs e)
    {
        this.dataGridViewInputArray.Rows.Clear();
    }

    private void dataGridViewInputArray_RowsAdded(object sender,
DataGridViewRowsAddedEventArgs e)
    {
        this.DataGridCountChanged(sender);
        this.ChartRefresh(sender);
    }

    private void dataGridViewInputArray_RowsRemoved(object sender,
DataGridViewRowsRemovedEventArgs e)
    {
        this.DataGridCountChanged(sender);
        this.ChartRefresh(sender);
    }

    private void buttonGenerate_Click(object sender, EventArgs e)
    {
        int count = (int)this.numericUpDownCount.Value;
        if (count > 0)
        {
            DataGridView dataGrid = this.dataGridViewInputArray;
            if (dataGrid != null)
            {
                int min = (int)this.numericUpDownMin.Value;
                int max = (int)this.numericUpDownMax.Value;
                if (min <= max)
                {
                    Random rnd = new Random();
                    for (int i = 0; i < count; i++) dataGrid.Rows.Add(new object[] { rnd.Next(min, max) });
                }
            }
        }
    }

    private void Form_input_array_Load(object sender, EventArgs e)
    {
        DataGridView dataGrid = this.dataGridViewInputArray;
        if (dataGrid != null)
            foreach (double value in this.Array)
            {
```

```
                dataGrid.Rows.Add(new object[] { value });
            }
        }

        private void dataGridViewInputArray_CellValueChanged(object sender,
DataGridViewCellEventArgs e)
        {
            //this.ChartRefresh(sender);
            //DataGridView dataGrid = this.dataGridViewInputArray;
            //if (dataGrid != null && dataGrid.CurrentRow != null)
this.chart1.Series[0].Points[e.RowIndex].SetValueY(dataGrid.Rows[e.RowIndex].Cells[0].Value);
        }

        private void buttonSave_Click(object sender, EventArgs e)
        {
            this.Array.Clear();
            int count = this.chart1.Series[0].Points.Count;
            for (int i = 0; i < count; i++) this.Array.Add(this.chart1.Series[0].Points[i].YValues[0]);
            this.DialogResult = DialogResult.OK;
        }

        private void buttonExit_Click(object sender, EventArgs e)
        {
            this.DialogResult = DialogResult.OK;
        }

        private void buttonReset_Click(object sender, EventArgs e)
        {
            this.numericUpDownCount.Value = this.numericUpDownCount.Minimum;
            this.numericUpDownMin.Value = this.numericUpDownMin.Minimum;
            this.numericUpDownMax.Value = this.numericUpDownMax.Minimum;
        }
    }
}
```

Form_lectures.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;

namespace AlgoAnimation
{
    public partial class Form_lectures : Form
    {
        int TeacherID;
        public Form_lectures(int TeacherID)
        {
            InitializeComponent();
            this.TeacherID = TeacherID;
```

```
            DataTable TableSort = new DataTable("TableSort");
            DataColumn value = new DataColumn("value", typeof(System.String));
            DataColumn text = new DataColumn("text", typeof(System.String));
            TableSort.Columns.AddRange(new DataColumn[] { value, text });
            TableSort.PrimaryKey = new DataColumn[] { value };

            DataRow up = TableSort.NewRow();
            up[0] = "ASC";
            up[1] = "За зростанням";
            TableSort.Rows.Add(up);

            DataRow down = TableSort.NewRow();
            down[0] = "DESC";
            down[1] = "За спаданням";
            TableSort.Rows.Add(down);

            this.comboBoxSort.DataSource = TableSort;
            this.comboBoxSort.ValueMember = "value";
            this.comboBoxSort.DisplayMember = "text";
            this.comboBoxSort.SelectedIndex = 0;
        }


        private void LectureAdd()
        {
            ////////////////////////////////добавление групы//////////////
            Form_lectures_add form = new Form_lectures_add("Додати теоретичні відомості",
"Додати");
            if (form.ShowDialog() == DialogResult.OK)
            {
                DataSetLectures.lecturesRow row =
(DataSetLectures.lecturesRow)((DataRowView)this.lecturesBindingSource.AddNew()).Row;
                row.lectureName = form.Title;
                row.teacherID = this.TeacherID;
                string filePath = form.File;
                row.lecture = File.ReadAllBytes(filePath);

                this.lecturesBindingSource.EndEdit();
            }
        }
        private void LectureChenge()
        {
            if (this.lecturesBindingSource.Current != null)
            {
                DataSetLectures.lecturesRow row =
(DataSetLectures.lecturesRow)((DataRowView)this.lecturesBindingSource.Current).Row;
                Form_lectures_add form = new Form_lectures_add("Редагувати теоретичні відомості",
"Зберегти");

                form.Title = row.lectureName;

                if (form.ShowDialog() == DialogResult.OK)
                {
                    row.lectureName = form.Title;
                    string filePath = form.File;
                    row.lecture = File.ReadAllBytes(filePath);
```

```
                this.lecturesBindingSource.EndEdit();
            }
        }
    }
    private void LectureDelete()
    {
        if (this.lecturesBindingSource.Current != null)
        {
            ((DataRowView)this.lecturesBindingSource.Current).Row.Delete();
        }
        this.lecturesBindingSource.EndEdit();
    }



    private void додатиToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.LectureAdd();
    }

    private void видалитиToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.LectureDelete();
    }

    private void змінитиToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.LectureChenge();
    }

    private void Form_lectures_Load(object sender, EventArgs e)
    {
        // TODO: данная строка кода позволяет загрузить данные в таблицу "dataSetLectures.lectures". При необходимости она может быть перемещена или удалена.
        this.lecturesTableAdapter.FillBy(this.dataSetLectures.lectures, this.TeacherID);

    }


    private void buttonSearch_Click(object sender, EventArgs e)
    {
        string strSearch = string.Format("lectureName = '{0}'", this.textBoxSearch.Text);
        this.lecturesBindingSource.Filter = strSearch;
    }

    private void buttonCancel_Click(object sender, EventArgs e)
    {
        this.lecturesBindingSource.RemoveFilter();
    }

    private void buttonSort_Click(object sender, EventArgs e)
    {
        this.lecturesBindingSource.Sort = String.Format("lectureName {0}", this.comboBoxSort.SelectedValue);
    }
```

```
        private void buttonSortCancel_Click(object sender, EventArgs e)
        {
            this.lecturesBindingSource.RemoveSort();
        }


        private void переглянутиToolStripMenuItem_Click(object sender, EventArgs e)
        {

            if (this.lecturesBindingSource.Current != null)
            {
                DataSetLectures.lecturesRow row =
(DataSetLectures.lecturesRow)((DataRowView)this.lecturesBindingSource.Current).Row;
                if (this.saveFileDialogFile.ShowDialog() == DialogResult.OK)
                {
                    string filePath = this.saveFileDialogFile.FileName;
                    File.WriteAllBytes(filePath, row.lecture);
                }
            }

        }

        private void повернутисьДоМенюToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.DialogResult = DialogResult.OK;
        }

        private void панельІнструментівToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.splitContainer1.Panel1Collapsed = !this.splitContainer1.Panel1Collapsed;
        }

        private void зберегтиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.lecturesTableAdapter.Update(this.dataSetLectures.lectures);
        }

        private void оновитиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.lecturesTableAdapter.FillBy(this.dataSetLectures.lectures, this.TeacherID);
        }
    }
}
```

Form_lectures_student.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
```

```
namespace AlgoAnimation
{
   public partial class Form_lectures_student : Form
   {
      int StudentID;
      public Form_lectures_student(int StudentID)
      {
         InitializeComponent();
         this.StudentID = StudentID;
      }

      private void buttonCancel_Click(object sender, EventArgs e)
      {
         this.DialogResult = DialogResult.OK;
      }

      private void Form_lectures_student_Load(object sender, EventArgs e)
      {
         // TODO: данная строка кода позволяет загрузить данные в таблицу "dataSetLecturesStudent.lectures". При необходимости она может быть перемещена или удалена.
         this.lecturesTableAdapter.Fill(this.dataSetLecturesStudent.lectures);
         // TODO: данная строка кода позволяет загрузить данные в таблицу "dataSetLecturesStudent.teachers". При необходимости она может быть перемещена или удалена.
         this.teachersTableAdapter.FillBy(this.dataSetLecturesStudent.teachers, this.StudentID);

      }

      private void buttonOK_Click(object sender, EventArgs e)
      {

         if (this.fKteacherlecturesBindingSource.Current != null)
         {
            DataSetLecturesStudent.lecturesRow row = (DataSetLecturesStudent.lecturesRow)((DataRowView)this.fKteacherlecturesBindingSource.Current).Row;
            if (this.saveFileDialogFile.ShowDialog() == DialogResult.OK)
            {
               string filePath = this.saveFileDialogFile.FileName;
               File.WriteAllBytes(filePath, row.lecture);
            }
         }

      }
   }
}
```

Form_main.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
namespace AlgoAnimation
{
   public partial class Form_main : Form
   {

      int UserID;
      List<double> array = new List<double>();


      private int QueqSortColorMid = Color.Red.ToArgb();
      private int QueqSortColorLeftRight = Color.Orange.ToArgb();
      private int QueqSortColorBackground = Color.Blue.ToArgb();

      private int ShellSortColorLeftRight = Color.Orange.ToArgb();
      private int ShellSortColorBackground = Color.Blue.ToArgb();

      private int HeapSortColorLeftRight = Color.Orange.ToArgb();
      private int HeapSortColorBackground = Color.Blue.ToArgb();

      private int MergeSortColorLeftRight = Color.Orange.ToArgb();
      private int MergeSortColorBackground = Color.Blue.ToArgb();

      private int BucketSortColorMid = Color.Red.ToArgb();
      private int BucketSortColorLeftRight = Color.Orange.ToArgb();
      private int BucketSortColorBackground = Color.Blue.ToArgb();
      private int BucketSortColorBucket1 = Color.Green.ToArgb();
      private int BucketSortColorBucket2 = Color.Brown.ToArgb();
      private int BucketSortColorBucket3 = Color.DimGray.ToArgb();
      private int BucketSortColorBucket4 = Color.Cornsilk.ToArgb();
      private int BucketSortColorBucket5 = Color.Aquamarine.ToArgb();

      public Form_main(int UserID, string UserType, string Surname, string UserName, string Patron)
      {
         InitializeComponent();

         this.comboBoxAnimationType.SelectedIndex = 0;

         this.UserID = UserID;
         this.statusStripUser.Items[0].Text = String.Format("{0}: {1} {2} {3}", UserType, Surname,
UserName, Patron);

         switch (UserType)
         {
            case "Адміністратор":
               this.menuStrip1.Items[1].Enabled = false;
               this.menuStrip1.Items[2].Enabled = false;
               break;
            case "Викладач":
               this.menuStrip1.Items[0].Enabled = false;
               this.menuStrip1.Items[2].Enabled = false;
               break;
            case "Студент":
               this.menuStrip1.Items[0].Enabled = false;
               this.menuStrip1.Items[1].Enabled = false;
               break;
         }
      }
```

```
        private void label2_Click(object sender, EventArgs e)
        {

        }

        private void ToolStripMenuItemInputArray_Click(object sender, EventArgs e)
        {
            Form_input_array form = new Form_input_array(this.array);
            this.Hide();
            if (form.ShowDialog() == DialogResult.OK)
            {
                this.Show();
                this.array.Clear();
                foreach (double value in form.Array) this.array.Add(value);

                this.labelArrayInfo.Visible = (this.array.Count <= 0) ? true : false;
                this.labelArrayCount.Text = String.Format("Кількість елементів {0}/200",
this.array.Count);
                this.progressBarArrayLenght.Value = this.array.Count;

                this.chart1.Series[0].Points.Clear();
                foreach (double value in this.array) this.chart1.Series[0].Points.AddY(value);

            }
            else this.Close();
        }

        private void buttonColors_Click(object sender, EventArgs e)
        {
            Form_colors form = new Form_colors();
            form.QueqSortColorMid = this.QueqSortColorMid;
            form.QueqSortColorLeftRight = this.QueqSortColorLeftRight;
            form.QueqSortColorBackground = this.QueqSortColorBackground;

            form.ShellSortColorLeftRight = this.ShellSortColorLeftRight;
            form.ShellSortColorBackground = this.ShellSortColorBackground;

            form.HeapSortColorLeftRight = this.HeapSortColorLeftRight;
            form.HeapSortColorBackground = this.HeapSortColorBackground;

            form.MergeSortColorLeftRight = this.MergeSortColorLeftRight;
            form.MergeSortColorBackground = this.MergeSortColorBackground;

            form.BucketSortColorMid = this.BucketSortColorMid;
            form.BucketSortColorLeftRight = this.BucketSortColorLeftRight;
            form.BucketSortColorBackground = this.BucketSortColorBackground;
            form.BucketSortColorBucket1 = this.BucketSortColorBucket1;
            form.BucketSortColorBucket2 = this.BucketSortColorBucket2;
            form.BucketSortColorBucket3 = this.BucketSortColorBucket3;
            form.BucketSortColorBucket4 = this.BucketSortColorBucket4;
            form.BucketSortColorBucket5 = this.BucketSortColorBucket5;



            if (form.ShowDialog() == DialogResult.OK)
            {
```

```
        this.QueqSortColorMid = form.QueqSortColorMid;
        this.QueqSortColorLeftRight = form.QueqSortColorLeftRight;
        this.QueqSortColorBackground = form.QueqSortColorBackground;

        this.ShellSortColorLeftRight = form.ShellSortColorLeftRight;
        this.ShellSortColorBackground = form.ShellSortColorBackground;

        this.HeapSortColorLeftRight = form.HeapSortColorLeftRight;
        this.HeapSortColorBackground = form.HeapSortColorBackground;

        this.MergeSortColorLeftRight = form.MergeSortColorLeftRight;
        this.MergeSortColorBackground = form.MergeSortColorBackground;

        this.BucketSortColorMid = form.BucketSortColorMid;
        this.BucketSortColorLeftRight = form.BucketSortColorLeftRight;
        this.BucketSortColorBackground = form.BucketSortColorBackground;
        this.BucketSortColorBucket1 = form.BucketSortColorBucket1;
        this.BucketSortColorBucket2 = form.BucketSortColorBucket2;
        this.BucketSortColorBucket3 = form.BucketSortColorBucket3;
        this.BucketSortColorBucket4 = form.BucketSortColorBucket4;
        this.BucketSortColorBucket5 = form.BucketSortColorBucket5;
    }
}

private void buttonAnimationStart_Click(object sender, EventArgs e)
{
    int interval = 1000 / ((int)this.numericUpDownAnimationSpeed.Value);
    int Selection = this.comboBoxAnimationType.SelectedIndex;
    Form_animation form = new Form_animation(interval, Selection, this.array.ToArray());

    form.QueqSortColorMid = this.QueqSortColorMid;
    form.QueqSortColorLeftRight = this.QueqSortColorLeftRight;
    form.QueqSortColorBackground = this.QueqSortColorBackground;

    form.ShellSortColorLeftRight = this.ShellSortColorLeftRight;
    form.ShellSortColorBackground = this.ShellSortColorBackground;

    form.HeapSortColorLeftRight = this.HeapSortColorLeftRight;
    form.HeapSortColorBackground = this.HeapSortColorBackground;

    form.MergeSortColorLeftRight = this.MergeSortColorLeftRight;
    form.MergeSortColorBackground = this.MergeSortColorBackground;

    form.BucketSortColorMid = this.BucketSortColorMid;
    form.BucketSortColorLeftRight = this.BucketSortColorLeftRight;
    form.BucketSortColorBackground = this.BucketSortColorBackground;
    form.BucketSortColorBucket1 = this.BucketSortColorBucket1;
    form.BucketSortColorBucket2 = this.BucketSortColorBucket2;
    form.BucketSortColorBucket3 = this.BucketSortColorBucket3;
    form.BucketSortColorBucket4 = this.BucketSortColorBucket4;
    form.BucketSortColorBucket5 = this.BucketSortColorBucket5;


    form.Show();
}
```

```
private void Form_main_Load(object sender, EventArgs e)
{

}

private void вихідToolStripMenuItem1_Click(object sender, EventArgs e)
{
   this.DialogResult = DialogResult.OK;
}

private void тестуванняToolStripMenuItem_Click(object sender, EventArgs e)
{
   Form_test_start form = new Form_test_start(this.UserID);
   this.Hide();
   if(form.ShowDialog() == DialogResult.OK) this.Show();
   else this.Close();
}

private void результатиТестуванняToolStripMenuItem_Click(object sender, EventArgs e)
{
   Form_test_result form = new Form_test_result(this.UserID);
   this.Hide();
   if (form.ShowDialog() == DialogResult.OK) this.Show();
   else this.Close();
}

private void студентськіГрупиToolStripMenuItem_Click(object sender, EventArgs e)
{
   Form_groups form = new Form_groups();
   this.Hide();
   if (form.ShowDialog() == DialogResult.OK) this.Show();
   else this.Close();
}

private void викладачіToolStripMenuItem_Click(object sender, EventArgs e)
{
   Form_teachers form = new Form_teachers(this.UserID);
   this.Hide();
   if (form.ShowDialog() == DialogResult.OK) this.Show();
   else this.Close();
}

private void створенняТестівToolStripMenuItem_Click(object sender, EventArgs e)
{
   Form_tests form = new Form_tests(this.UserID);
   this.Hide();
   if (form.ShowDialog() == DialogResult.OK) this.Show();
   else this.Close();
}

private void вToolStripMenuItem_Click(object sender, EventArgs e)
{
   Form_questions form = new Form_questions(this.UserID);
   this.Hide();
   if (form.ShowDialog() == DialogResult.OK) this.Show();
   else this.Close();
}
```

```
        private void теоретичніВідомостіToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            Form_lectures_student form = new Form_lectures_student(this.UserID);
            this.Hide();
            if (form.ShowDialog() == DialogResult.OK) this.Show();
            else this.Close();
        }

        private void студентиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Form_students form = new Form_students(this.UserID);
            this.Hide();
            if (form.ShowDialog() == DialogResult.OK) this.Show();
            else this.Close();
        }

        private void теоретичніВідомостіToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Form_lectures form = new Form_lectures(this.UserID);
            this.Hide();
            if (form.ShowDialog() == DialogResult.OK) this.Show();
            else this.Close();
        }
    }
}
```

Form_questions.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace AlgoAnimation
{
    public partial class Form_questions : Form
    {
        int TeacherID;
        public Form_questions(int TeacherID)
        {
            InitializeComponent();
            this.TeacherID = TeacherID;
        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void Form_questions_Load(object sender, EventArgs e)
        {
```

```
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dataSetQuestions.answers". При необходимости она может быть перемещена или удалена.
            this.answersTableAdapter.Fill(this.dataSetQuestions.answers);
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dataSetQuestions.questions". При необходимости она может быть перемещена или удалена.
            this.questionsTableAdapter.Fill(this.dataSetQuestions.questions);
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dataSetQuestions.tests". При необходимости она может быть перемещена или удалена.
            this.testsTableAdapter.FillBy(this.dataSetQuestions.tests, this.TeacherID);

            //this.dataSetQuestions.tests.Columns[2].DefaultValue = 0;
            //MessageBox.Show(this.dataSetQuestions.tests.Columns[2].DefaultValue.ToString());
            this.bindingNavigatorAnswers.Items.Add(new ToolStripControlHost(this.checkBox1));
        }

        private void fKtestquestionsBindingSource_CurrentChanged(object sender, EventArgs e)
        {

        }

        private void зберегтиЗапитанняToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.questionsTableAdapter.Update(this.dataSetQuestions.questions);
        }

        private void зберегтиВідповідіToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.answersTableAdapter.Update(this.dataSetQuestions.answers);
        }

        private void зберегтиВсеToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.questionsTableAdapter.Update(this.dataSetQuestions.questions);
            this.answersTableAdapter.Update(this.dataSetQuestions.answers);
        }

        private void оновитиЗапитанняToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.questionsTableAdapter.Fill(this.dataSetQuestions.questions);
        }

        private void оновитиВідповідіToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.answersTableAdapter.Fill(this.dataSetQuestions.answers);
        }

        private void оновитиВсеToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.testsTableAdapter.FillBy(this.dataSetQuestions.tests, this.TeacherID);
            this.questionsTableAdapter.Fill(this.dataSetQuestions.questions);
            this.answersTableAdapter.Fill(this.dataSetQuestions.answers);
        }

        private void повернутисяДоМенюToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.DialogResult = DialogResult.OK;
        }
```

```
        private void fKtestquestionsBindingSource_CurrentChanged_1(object sender, EventArgs e)
        {
            BindingSource Tests = (BindingSource)sender;
            if (Tests.Current != null)
            {
                DataRow row = ((DataRowView)Tests.Current).Row;
                if (row.RowState == DataRowState.Unchanged || row.RowState == DataRowState.Modified)
                {
                    this.textBoxAnswers.Enabled = true;
                    this.bindingNavigatorAnswers.Enabled = true;
                    return;
                }
            }
            this.textBoxAnswers.Enabled = false;
            this.bindingNavigatorAnswers.Enabled = false;
        }
    }
}
```

Form_registration.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Configuration;

namespace AlgoAnimation
{
    public partial class Form_registration : Form
    {
        public Form_registration()
        {
            InitializeComponent();
        }

        private void Form_registration_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dataSetRegistration.groups". При необходимости она может быть перемещена или удалена.
            this.groupsTableAdapter.Fill(this.dataSetRegistration.groups);

        }

        private void buttonRegistration_Click(object sender, EventArgs e)
        {
            string diplomaConnectionString = ConfigurationManager.ConnectionStrings["diploma"].ConnectionString;
```

```
            string queryString = String.Format("select count(*) from students where userName = '{0}'", this.textBoxLogin.Text);

            using (SqlConnection connection = new SqlConnection(
              diplomaConnectionString))
            {
               SqlCommand command = new SqlCommand(queryString, connection);
               command.Connection.Open();

               int resalt = (int)command.ExecuteScalar();
               if (resalt > 0) MessageBox.Show("Такий логін вже існує", "Помилка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
               else
               {
                  string Surname = this.textBoxSurname.Text;
                  string name = this.textBoxName.Text;
                  string Patron = this.textBoxPatron.Text;
                  string Login = this.textBoxLogin.Text;
                  string Pass = this.textBoxPass.Text;
                  int groupID = (int)((DataRowView)this.comboBoxGroup.SelectedItem)[0];

                  string str = string.Format("insert into students(surname, name, patron, registered, userName, pass, groupID) values ('{0}','{1}','{2}',0,'{3}','{4}',{5})", Surname, name, Patron, Login, Pass, groupID);
                  command.CommandText = str;
                  if(command.ExecuteNonQuery()>0) MessageBox.Show("Вас зареєстровано", "Вітаю!", MessageBoxButtons.OK, MessageBoxIcon.Information);
                  else MessageBox.Show("Щось пішло не так, спробуйте ще раз!", "Помилка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                  this.DialogResult = DialogResult.OK;
                  this.Close();
               }
            }
         }

         private void buttonCancel_Click(object sender, EventArgs e)
         {
            this.DialogResult = DialogResult.Cancel;
            this.Close();
         }
      }
   }
```

Form_students.cs

```
   using System;
   using System.Collections.Generic;
   using System.ComponentModel;
   using System.Data;
   using System.Drawing;
   using System.Linq;
   using System.Text;
   using System.Threading.Tasks;
   using System.Windows.Forms;

   namespace AlgoAnimation
   {
```

```
public partial class Form_students : Form
{
   int TeacherID;
   public Form_students(int TeacherID)
   {
      InitializeComponent();
      this.TeacherID = TeacherID;

      DataTable TableSort = new DataTable("TableSearch");
      DataColumn value = new DataColumn("value", typeof(System.String));
      DataColumn text = new DataColumn("text", typeof(System.String));
      TableSort.Columns.AddRange(new DataColumn[] { value, text });
      TableSort.PrimaryKey = new DataColumn[] { value };

      DataRow up = TableSort.NewRow();
      up[0] = "or";
      up[1] = "Логічна сума";
      TableSort.Rows.Add(up);

      DataRow down = TableSort.NewRow();
      down[0] = "and";
      down[1] = "Логічне множення";
      TableSort.Rows.Add(down);

      this.comboBoxIf.DataSource = TableSort;
      this.comboBoxIf.ValueMember = "value";
      this.comboBoxIf.DisplayMember = "text";
      this.comboBoxIf.SelectedIndex = 0;



      DataTable TableRegistered = new DataTable("TableRegistered");
      DataColumn YesOrNo = new DataColumn("YesOrNo", typeof(System.Boolean));
      DataColumn heder = new DataColumn("heder", typeof(System.String));
      TableRegistered.Columns.AddRange(new DataColumn[] { YesOrNo, heder });
      TableRegistered.PrimaryKey = new DataColumn[] { YesOrNo };

      DataRow no = TableRegistered.NewRow();
      no[0] = false;
      no[1] = "Не підтверджена";
      TableRegistered.Rows.Add(no);

      DataRow yes = TableRegistered.NewRow();
      yes[0] = true;
      yes[1] = "Підтверджена";
      TableRegistered.Rows.Add(yes);


      this.comboBoxRegistered.DataSource = TableRegistered;
      this.comboBoxRegistered.ValueMember = "YesOrNo";
      this.comboBoxRegistered.DisplayMember = "heder";
      this.comboBoxRegistered.SelectedIndex = 0;
   }


   private void StudentAdd()
   {
```

```
            Form_teachers_add form = new Form_teachers_add("Додати студента", "Додати",
"Скасувати");
            if (form.ShowDialog() == DialogResult.OK)
            {
                DataSetStudents.studentsRow row =
(DataSetStudents.studentsRow)((DataRowView)this.fKgroupstudentsBindingSource.AddNew()).Row;

                row.surname = form.Surname;
                row.name = form.FirstName;
                row.patron = form.Patron;
                row.userName = form.UserName;
                row.pass = form.Password;
                row.registered = true;
                this.fKgroupstudentsBindingSource.EndEdit();
            }
        }
        private void StudentChenge()
        {
            if (this.fKgroupstudentsBindingSource.Current != null)
            {
                DataSetStudents.studentsRow row =
(DataSetStudents.studentsRow)((DataRowView)this.fKgroupstudentsBindingSource.Current).Row;
                Form_teachers_add form = new Form_teachers_add("Змінити дані студента", "Змінити",
"Скасувати");
                form.Surname = row.surname;
                form.FirstName = row.name;
                form.Patron = row.patron;
                form.UserName = row.userName;
                form.Password = row.pass;
                if (form.ShowDialog() == DialogResult.OK)
                {
                    row.surname = form.Surname;
                    row.name = form.FirstName;
                    row.patron = form.Patron;
                    row.userName = form.UserName;
                    row.pass = form.Password;
                    this.fKgroupstudentsBindingSource.EndEdit();
                }
            }
        }
        private void StudentDelete()
        {
            if (this.fKgroupstudentsBindingSource.Current != null)
            {
                ((DataRowView)this.fKgroupstudentsBindingSource.Current).Row.Delete();
                this.fKgroupstudentsBindingSource.EndEdit();
            }
        }



        private void Form_students_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"dataSetStudents.tests_students". При необходимости она может быть перемещена или удалена.
            this.tests_studentsTableAdapter.FillBy(this.dataSetStudents.tests_students);
```

```
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dataSetStudents.tests_students". При необходимости она может быть перемещена или удалена.
            this.testsTableAdapter.FillBy(this.dataSetStudents.tests, this.TeacherID);
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dataSetStudents.students". При необходимости она может быть перемещена или удалена.
            this.studentsTableAdapter.Fill(this.dataSetStudents.students);
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dataSetStudents.groups". При необходимости она может быть перемещена или удалена.
            this.groupsTableAdapter.FillBy(this.dataSetStudents.groups, this.TeacherID);


        }

        private void testsBindingSource_CurrentChanged(object sender, EventArgs e)
        {
            BindingSource bindingSource = (BindingSource)sender;
            DataSetStudents.testsRow row = (DataSetStudents.testsRow)((DataRowView)bindingSource.Current).Row;
            if (row != null)
            {
                this.fKstudenttestsstudentsBindingSource.Filter = String.Format("testID = '{0}'", row.testID);
            }

        }

        private void buttonSearch_Click(object sender, EventArgs e)
        {
            string strSeparator = string.Format(" {0} ", this.comboBoxIf.SelectedValue.ToString());
            bool[] ArrayChecked = new bool[] { this.checkBoxSurname.Checked, this.checkBoxName.Checked, this.checkBoxPatron.Checked, this.checkBoxLogin.Checked, this.checkBoxRegistered.Checked };
            string[] ArrayValues = new string[5];
            ArrayValues[0] = string.Format("surname='{0}'", this.textBoxSurname.Text);
            ArrayValues[1] = string.Format("name='{0}'", this.textBoxName.Text);
            ArrayValues[2] = string.Format("patron='{0}'", this.textBoxPatron.Text);
            ArrayValues[3] = string.Format("userName='{0}'", this.textBoxLogin.Text);
            ArrayValues[4] = string.Format("registered='{0}'", this.comboBoxRegistered.SelectedValue);

            List<string> ArrayFilters = new List<string>();

            for (int i = 0; i < ArrayChecked.Length; i++)
            {
                if (ArrayChecked[i]) ArrayFilters.Add(ArrayValues[i]);
            }

            StringBuilder sb = new StringBuilder();
            int Count = ArrayFilters.Count;
            for (int i = 0; i < Count; i++)
            {
                sb.Append(ArrayFilters[i]);
                if (i < Count - 1) sb.Append(strSeparator);
            }

            this.fKgroupstudentsBindingSource.Filter = sb.ToString();
        }
```

```
        private void buttonCancel_Click(object sender, EventArgs e)
        {
            this.fKgroupstudentsBindingSource.RemoveFilter();
        }

        private void видалитиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.StudentDelete();
        }

        private void додатиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.StudentAdd();
        }

        private void змінитиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.StudentChenge();
        }

        private void повернутисяДоМенюToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.DialogResult = DialogResult.OK;
        }

        private void панельІнструментівToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.splitContainer1.Panel1Collapsed = !this.splitContainer1.Panel1Collapsed;
        }

        private void зберегтиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.studentsTableAdapter.Update(this.dataSetStudents.students);
        }

        private void оновитиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.studentsTableAdapter.Fill(this.dataSetStudents.students);
        }
    }
}
```

Form_teachers.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace AlgoAnimation
```

```
{
    public partial class Form_teachers : Form
    {
        private int AdminID;
        public Form_teachers(int AdminID)
        {
            InitializeComponent();
            this.AdminID = AdminID;


            DataTable TableSort = new DataTable("TableSearch");
            DataColumn value = new DataColumn("value", typeof(System.String));
            DataColumn text = new DataColumn("text", typeof(System.String));
            TableSort.Columns.AddRange(new DataColumn[] { value, text });
            TableSort.PrimaryKey = new DataColumn[] { value };

            DataRow up = TableSort.NewRow();
            up[0] = "or";
            up[1] = "Логічна сума";
            TableSort.Rows.Add(up);

            DataRow down = TableSort.NewRow();
            down[0] = "and";
            down[1] = "Логічне множення";
            TableSort.Rows.Add(down);

            this.comboBoxIf.DataSource = TableSort;
            this.comboBoxIf.ValueMember = "value";
            this.comboBoxIf.DisplayMember = "text";
            this.comboBoxIf.SelectedIndex = 0;
        }


        private void TeacherAdd()
        {
            Form_teachers_add form = new Form_teachers_add("Додати викладача", "Додати",
"Скасувати");
            if (form.ShowDialog() == DialogResult.OK)
            {
                DataSetTeachers.teachersRow row =
(DataSetTeachers.teachersRow)((DataRowView)this.teachersBindingSource.AddNew()).Row;

                row.adminID = AdminID;
                row.surname = form.Surname;
                row.name = form.FirstName;
                row.patron = form.Patron;
                row.userName = form.UserName;
                row.pass = form.Password;
                this.teachersBindingSource.EndEdit();
            }
        }
        private void TeacherChenge()
        {
            if (this.groupsBindingSource.Current != null)
            {
                DataSetTeachers.teachersRow row =
(DataSetTeachers.teachersRow)((DataRowView)this.teachersBindingSource.Current).Row;
```

```
                Form_teachers_add form = new Form_teachers_add("Змінити дані викладача", "Змінити",
"Скасувати");
                form.Surname = row.surname;
                form.FirstName = row.name;
                form.Patron = row.patron;
                form.UserName = row.userName;
                form.Password = row.pass;
                if (form.ShowDialog() == DialogResult.OK)
                {
                   row.adminID = AdminID;
                   row.surname = form.Surname;
                   row.name = form.FirstName;
                   row.patron = form.Patron;
                   row.userName = form.UserName;
                   row.pass = form.Password;
                   this.teachersBindingSource.EndEdit();
                }
             }
          }
          private void TeacherDelete()
          {
             if (this.teachersBindingSource.Current != null)
             {
                ((DataRowView)this.teachersBindingSource.Current).Row.Delete();
                this.teachersBindingSource.EndEdit();
             }
          }


          private void Form_teachers_Load(object sender, EventArgs e)
          {
             // TODO: данная строка кода позволяет загрузить данные в таблицу
"dataSetTeachers.groups_teachers". При необходимости она может быть перемещена или удалена.
                this.groups_teachersTableAdapter.Fill(this.dataSetTeachers.groups_teachers);
                // TODO: данная строка кода позволяет загрузить данные в таблицу
"dataSetTeachers.groups". При необходимости она может быть перемещена или удалена.
                this.groupsTableAdapter.Fill(this.dataSetTeachers.groups);
                // TODO: данная строка кода позволяет загрузить данные в таблицу
"dataSetTeachers.teachers". При необходимости она может быть перемещена или удалена.
                this.teachersTableAdapter.FillBy(this.dataSetTeachers.teachers,this.AdminID);

          }

          private void buttonAdd_Click(object sender, EventArgs e)
          {
             DataSetTeachers.teachersRow rowTeacher =
(DataSetTeachers.teachersRow)((DataRowView)this.teachersBindingSource.Current).Row;
             if (rowTeacher != null)
             {
                DataSetTeachers.groups_teachersRow[] rows = rowTeacher.Getgroups_teachersRows();
                foreach (DataSetTeachers.groups_teachersRow item in rows) if (item.groupID ==
(int)this.comboBoxGroup.SelectedValue) return;

                DataSetTeachers.groups_teachersRow row =
(DataSetTeachers.groups_teachersRow)((DataRowView)this.fKteachergroupsteachersBindingSource.AddNe
w()).Row;
                row.groupID = (int)this.comboBoxGroup.SelectedValue;
```

```
            this.fKteachergroupsteachersBindingSource.EndEdit();
        }

    }

    private void додатиToolStripMenuItem_Click_1(object sender, EventArgs e)
    {
        this.TeacherAdd();
    }

    private void змінитиToolStripMenuItem_Click_1(object sender, EventArgs e)
    {
        this.TeacherChenge();
    }

    private void видалитиToolStripMenuItem_Click_1(object sender, EventArgs e)
    {
        this.TeacherDelete();
    }

    private void зберегтиToolStripMenuItem_Click_1(object sender, EventArgs e)
    {
        this.teachersTableAdapter.Update(this.dataSetTeachers.teachers);
        this.groups_teachersTableAdapter.Update(this.dataSetTeachers.groups_teachers);
    }

    private void оновитиToolStripMenuItem_Click_1(object sender, EventArgs e)
    {
        this.teachersTableAdapter.FillBy(this.dataSetTeachers.teachers, this.AdminID);
        this.groups_teachersTableAdapter.Fill(this.dataSetTeachers.groups_teachers);
        this.groupsTableAdapter.Fill(this.dataSetTeachers.groups);

    }

    private void панельІнструментівToolStripMenuItem_Click_1(object sender, EventArgs e)
    {
        this.splitContainer1.Panel1Collapsed = !this.splitContainer1.Panel1Collapsed;
    }

    private void buttonSearch_Click(object sender, EventArgs e)
    {
        string strSeparator = string.Format(" {0} ", this.comboBoxIf.SelectedValue.ToString());
        bool[] ArrayChecked = new bool[] { this.checkBoxSurname.Checked,
this.checkBoxName.Checked, this.checkBoxPatron.Checked, this.checkBoxLogin.Checked };
        string[] ArrayValues = new string[4];
        ArrayValues[0] = string.Format("surname='{0}'", this.textBoxSurname.Text);
        ArrayValues[1] = string.Format("name='{0}'", this.textBoxName.Text);
        ArrayValues[2] = string.Format("patron='{0}'", this.textBoxPatron.Text);
        ArrayValues[3] = string.Format("userName='{0}'", this.textBoxLogin.Text);

    }
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ДОКУМЕНТІВ НА МАГНІТНОМУ НОСІЇ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота_Дубовенко.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота_ Дубовенко.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Програма_ Дубовенко.rar	Архів. Містить коди програми і скомпільовану програму
Презентація	
Презентація_ Дубовенко.ppt	Презентація кваліфікаційної роботи