РЕФЕРАТ

Пояснювальна записка: ___ с., ___ рис., ___ табл., __ дод., ___ джерел.

Об'єкт розробки: програмний додаток для ведення документації.

Мета кваліфікаційної роботи: спроєктувати та розробити програмний додаток для автоматизації роботи кураторів, допомогти полегшити ведення та аналізу інформації про відповідні академічні групи.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано платформу для розробки, виконано проєктування і розробка програми, наведено опис алгоритму і структури функціонування програми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження програми, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенні кінцевого програмного додатка ведення роботи куратора, автоматизовано використання бази даних проекту для допомоги та полегшення обробки та аналізу інформації про відповідні академічні групи та студентів.

Актуальність даного програмного продукту визначається великим попитом на подібні розробки, тому що створення закладом освіти системи управління з використанням комп’ютерної техніки надасть змогу підвищити якість та оперативність вирішення завдань, які виникають у системі управління закладами освіти.

Список ключових слів: КУРАТОР, ГРУПА, СТУДЕНТ, АВТОМАТИЗОВАНА СИСТЕМА, ДОДАТОК, БАЗА ДАНИХ, ПРОЄКТУВАННЯ, ПРОГРАМУВАННЯ.

ABSTRACT

Explanatory note: ___ pp., ___ fig., ___ table, __ appendix, ___ sources.

Object of development: software application for maintaining documentation.

The purpose of the qualification work: to design and develop a software application to automate the work of curators, to help facilitate the maintenance and analysis of information about relevant academic groups.

The introduction considers the analysis and current state of the problem, specifies the purpose of the qualification work and its scope, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development are defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes existing solutions, selects a platform for development, designs and develops the program, describes the algorithm and structure of the program, determines the input and output data, provides characteristics of the parameters of hardware, describes the call and download of the program, describes the program.

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical significance lies in the creation of the final software application for the curator's work, the automated use of the project database to assist and facilitate the processing and analysis of information about relevant academic groups and students.

The relevance of this software product is determined by the high demand for such developments, because the creation of an educational institution management system using computer technology will improve the quality and efficiency of solving problems that arise in the management system of educational institutions.

List of keywords: CURATOR, GROUP, STUDENT, AUTOMATED SYSTEM, APPENDIX, DATABASE, DESIGN, PROGRAMMING.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних;

ІС – інформаційна система;

ЖЦПЗ – життєвий цикл програмного забезпечення;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

СКБД – система керування базами даних.

ЗМІСТ

ВСТУП

Сьогодні важко уявити собі життя без комп'ютерів. Темпи розвитку комп'ютерних технологій вражають.

Комп'ютерні технології, які з'явилися у середині XX-го століття, мали великий вплив на розвиток науки, техніки, бізнесу та багатьох інших сфер людської діяльності. Ці технології будуть і надалі користуватися великим попитом на ринку працевлаштування, а ті, хто працює в галузі інформаційних технологій, відіграватимуть вирішальну роль у формуванні майбутнього.

Тому не дивно, що для всіх людей в звичку ввійшло використання комп’ютерів і великої кількості комп’ютерних програм, які роблять роботу більш швидкою та ефективною. Також ці технології дозволяють швидко обробляти інформацію та тримати її в захищеному вигляді.

У багатьох країнах світу вільне володіння цими технологіями є складовою базової освіти. Через це майже всі установи використовують автоматизовані системи управління для ведення різного обліку. Всі управлінські завдання вирішуються шляхом складання зведеної і аналітичної звітності в офісних електронних документах (MSExcel, MSWord). Зараз для багатьох освітян ввійшло в звичку використовувати комп’ютерні програми, які можуть зробити розклад занять, розподіл аудиторного фонду навчального закладу або розрахувати навантаження викладачів. Тому актуальним є аналіз інформаційних освітніх систем.

Темою кваліфікаційної роботи бакалавра є «Розробка програмного додатку автоматизації ведення роботи куратора засобами мови C++ в середовищі Builder 10.3 Rio». Мета створення проєкту полягає у відображенні можливостей практичного використання програми, яка може зменшити роботу куратора групи та допомогти проводити моніторинг навчання студентів за кожним обраним семестром, складати необхідні звіти та вести інформацію стосовно студентів групи; систематизації тем для виховної роботі зі студентами.

Розроблений програмний додаток дає змогу користувачеві вирішувати великий обсяг завдань, оптимізуючи значну кількість обов’язків керівника групи, та відмовитися від писемної роботи, надати всю необхідну інформацію в швидкому режимі.

Отже, для забезпечення ефективного функціонування закладів освіти України в сучасних умовах необхідно реорганізувати систему управління навчальним процесом. У першу чергу, така реорганізація пов’язана з необхідністю реалізації «Болонського процесу» у системі освіти України та інтенсифікації діяльності навчальних закладів. Тому існує єдиний ефективний шлях - інформатизація системи управління закладами освіти на основі створення автоматизованих інформаційних систем та технологій, які забезпечують вирішення завдань збору, обробки, зберігання та ефективного використання інформації у процесі управління.

Створення закладом освіти системи управління з використанням комп’ютерної техніки надасть змогу підвищити якість та оперативність вирішення завдань, які виникають у системі управління закладами освіти.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Найпоширенішими автоматизованими системами керування навчальним закладом є пакети комп'ютерних систем «Деканат», «Бібліограф-2007», «Колоквіум», «ПС-персонал» приватного підприємства «Політек-Софт».

Серед усіх пакетів, які пропонує фірма «Політек-Софт», найбільш функціональними, з точки зору автоматизації сфер діяльності навчальних закладів, є пакети «Деканат» і «Колоквіум».

Пакет програм «Деканат» призначений для автоматизації планування та обліку навчального процесу в закладах освіти.

У даному пакеті створюється і підтримується база даних, в якій формується та реєструється:

– структура навчального процесу закладу (предмети, навчальні плани);

– дані про всіх викладачів закладу та їхнього планового навантаження, розклад роботи;

– дані щодо всіх студентів закладу та їхньої успішності за весь період навчання;

– дані про наявні корпуси та аудиторії навчального закладу,їхнього заповнення, розклад занять;

– щоденні дані про фактичну роботу кожного викладача з кожного предмету.

Пакет програм «Колоквіум» призначений для комп'ютерного тестування знань студентів навчальних закладів України.

Пакет має такі особливості:

– зручна оболонка для формування тестів;

– тестування студентів у різних режимах та на основі результатів тестування оцінювання знань;

– забезпечення доступу до бази даних з результатами тестування з метою всебічного аналізу;

– автоматична генерація і друк звітів за результатами тестування;

– інформаційна сумісність з пакетом програм «Деканат».

«Політек-Софт» пропонує ще допоміжні пакети, такі як «ПС-Персонал», «Бібліограф», «Деканат-Університет-Web», які використовуються для введення і збереження інформації, тобто використовуються як банки даних. Головним недоліком пакетів, які представляє ПП «Політек-Софт» є відсутність підтримки кредитно-модульної системи. Адже, в рамках інтеграції України в європейське освітнє товариство, проводиться поступова модернізація та удосконалення змісту освіти та організації навчального процесу.

Отже, автоматизація діяльності навчальних закладів, яка надає доступ до необхідної користувачам інформації. Інформаційне середовище дозволяє управляти процесами, даними і людьми [16].

1.2. Призначення розробки та область застосування

Автоматизовану систему створено для полегшення роботи керівників груп. Дана система поєднує функції для складання звітності, отримання інформації про студентів, їх навчальні досягнення, ведення інформації про зібрані папери у студентів та ін. За допомогою розробленої бази даних надано можливість автоматичного заповнення бази даних для подальшого використання.

Розроблений програмний додаток дає змогу користувачеві вирішувати великий обсяг завдань, оптимізуючи значну кількість обов’язків керівника групи, та відмовитися від писемної роботи, надати всю необхідну інформацію в швидкому режимі.

Результатом виконання кваліфікаційної роботи є створення кінцевого програмного додатка для ведення роботи куратора, автоматизовано

використання бази даних проекту для допомоги та полегшення обробки та аналізу інформації про відповідні академічні групи та студентів.

Створення закладом освіти системи управління з використанням комп'ютерної техніки надасть змогу підвищити якість та оперативність вирішення завдань, які виникають у системі управління закладами освіти.

1.3. Підстава для розробки

Підставою для розробки кваліфікаційної роботи бакалавра на тему «Розробка програмного додатку автоматизації ведення роботи куратора засобами мови C++ в середовищі Builder 10.3 Rio» є наказ по Національному технічному університету «Дніпровська політехніка» від __.__. 2021р. № ____-_.

1.4. Постановка завдання

Метою створення кваліфікаційної роботи є спроєктувати та розробити програмний додаток для автоматизації роботи кураторів, допомогти та полегшити ведення та аналізу інформації про відповідні академічні групи.

Дана робота полягає у відображенні можливостей практичного використання програми, яка може зменшити роботу куратора групи та допомогти проводити моніторинг навчання студентів за кожним обраним семестром, складати необхідні звіти та вести інформацію стосовно студентів групи; систематизації тем для виховної роботі зі студентами.

Перелік питань, які потрібно розробити під час роботи:

1. Описати процес проектування програмного додатку.

2. Обґрунтувати вибір мови програмування та результати його реалізації.

В додатку повинні бути реалізовані наступні функції:

– ведення персональних даних студентів та батьків, пільгових категорій студентів;

– ведення переліку дисциплін, обліку поточної та семестрової успішності;

– ведення тем та матеріалів виховних годин із завантаженням файлів *.doc, *.pdf;

– розмежування доступу користувачів до бази даних проєкту;

– пошук необхідної інформації, сортування, фільтрація даних;

– використання шаблонів, побудова графіків та звітів у форматах *.doc, *.xls

Розроблена програма повинна мати зручний та зрозумілий інтерфейс для роботи, який відповідає виконуваним функціям.

Розроблений додаток повинен забезпечувати:

– авторизація в базу даних, для використання певних функцій;

– швидке створення звітів, витрачення менше часу;

– створення графіків з даних бази;

– програмний додаток повинен бути багато профільним, тобто його можна було б використовувати для коледжу, університету та інших навчальних закладів зі схожою структурою організації навчального процесу.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

В рамках даної теми потрібно створити базу даних для ведення інформації про студентів групи та розробити програмний додаток для роботи куратора.

Програма повинна виконувати наступні функції:

– додавання інформації про студентів та їх батьків;

– надання можливість автоматичного заповнення необхідної інформації в базу даних;

– перегляд, редагування, видалення інформації про студентів;

– виконання пошуку інформації за прізвищем, групою;

– формування звітів за обраною умовою з можливістю друкування та перегляду в офісних електронних документах MSExcel, MSWord;

– додавання, редагування, видалення найменувань предмету, документу, пільг;

– ведення інформації про збір необхідних документів з можливістю перегляду, редагування, видалення інформації;

– формування шаблонів з можливістю гнучкого вибору необхідних даних та забезпечення можливості роздрукування;

– надання можливості побудови необхідних графіків на основі оцінок;

– забезпечення ведення та перегляду проміжних оцінок за семестр;

– складання звітності «Особова картка успішності студента».

Початкові дані:

1. Дані про студента:

- ПІБ студента;
- номери телефону;
- email;
- стать;
- дата народження;
- адреса прописки;
- адреса проживання;
- примітка;
- найменування групи;
- ознака відрахування;
- попередня освіта;
- фото.

2. Дані про найменування:

- предметів;
- документів;

- пільг.

3. Дані про пільговиків:
   - студент;
   - пільга;
   - примітка.
4. Дані про батьків:
   - студент;
   - ПІБ батьків;
   - номер телефону;
   - адреса проживання;
   - місце роботи;
   - посада;
   - стать;
   - примітка.
5. Дані про гуртожиток:
   - студент;
   - номер кімнати;
   - номер гуртожитку.
6. Дані про навчання протягом семестру:
   - студент;
   - предмет;
   - оцінка за предмет;
   - номер семестру.
7. Дані про зібрані документи:
   - студент;
   - документ;
   - дата;
   - здав;
   - примітка.

1.5.2. Вимоги до інформаційної безпеки

Для забезпечення цілісності роботи системи необхідно забезпечити перевірку виконання ситуацій, що можуть призвести до виникнення помилок. А саме:

- перевірка на підключення до локальної бази даних;
- перевірка на існування відкритого екземпляру програмного додатку;
- перевірка введення некоректних даних;
- перевірка при видаленні даних;
- перевірка на авторизацію в базі даних;
- блокування використання функцій бази даних без авторизації;
- перевірка при створенні звітів та шаблонів;
- перевірка на існування даних при введенні;
- запобігання помилок для компонентів;
- використання обробників подій;
- створення ini файлів для підключення локальної бази.

1.5.3. Вимоги до складу та параметрів технічних засобів

Рекомендовані вимоги до апаратного забезпечення:

- процесор Intel Pentium CPU N3540, 2.16GHz або краще;
- місце на жорсткому диску 2 Гб на початковому етапі без урахування операційної системи;
- обсяг оперативного запам’ятовуючого пристрою 1 Гб і більше.

1.5.4. Вимоги до інформаційної та програмної сумісності

Дана автоматизована система повинна бути розроблена засобами середовища C++ Builder 10.3 Rio.

Рекомендовані вимоги до програмного забезпечення:

– Microsoft Windows 7 та вищі версії;

– Microsoft Office 2007 та вищі версії;

– наявність додатків MSExcel, MSWord.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Мета створення проєкту полягає у відображенні можливостей практичного використання програми, яка може полегшити роботу куратора групи та допомогти проводити моніторинг навчання студентів за кожним обраним семестром, складати необхідні звіти та вести інформацію стосовно студентів групи; систематизації тем для виховної роботі зі студентами.

Програма виконує наступні функції:

– додавання інформації про студентів та їх батьків;

– надання можливість автоматичного заповнення необхідної інформації в базу даних;

– перегляд, редагування, видалення інформації про студентів;

– виконання пошуку інформації за прізвищем, групою;

– формування звітів за обраною умовою з можливістю друкування та перегляду в офісних електронних документах MSExcel, MSWord;

– додавання, редагування, видалення найменувань предмету, документу, пільг;

– ведення інформації про збір необхідних документів з можливістю перегляду, редагування, видалення інформації;

– формування шаблонів з можливістю гнучкого вибору необхідних даних та забезпечення можливості роздрукування;

– надання можливості побудови необхідних графіків на основі оцінок;

– забезпечення ведення та перегляду проміжних оцінок за семестр;

– складання звітності «Особова картка успішності студента».

2.2. Опис застосованих математичних методів

Математичні методи та моделі для проєктування додатку, окрім розрахунку середнього балу, не використовувалися.

2.3. Опис використаної архітектури та шаблонів проектування

Життєвий цикл програмного забезпечення – це безперервний процес, який починається з моменту прийняття рішення про необхідність створення ПЗ і закінчується в момент його повного вилучення з експлуатації.

Існує декілька підходів при визначенні фаз та робіт життєвого циклу програмного забезпечення (ЖЦПЗ), кроків процесу програмування, каскадна і спіральна моделі. Але всі вони містять загальні основні компоненти: постановка завдання, проектування рішення, реалізація, обслуговування.

В даному проекті використаю процес програмування по Боемі. Каскадна модель була введена в 70 - 80 рр.. Вона зручна для однорідних ПЗ, коли кожен додаток являло собою єдине ціле. Основні характеристики моделі:

– життєвий цикл розбивається на етапи (фази);

– перехід з етапу на етап – тільки після повного завершення поточного етапу;

– етап завершується випуском повного комплекту документації, достатньої для того, щоб робота могла бути виконана іншою командою розробників.

Головні характерні риси каскадної моделі наступні:

1. Завершення кожної фази верифікацією і підтвердженням, мета яких - усунути можливо більше число проблем, пов'язаних з розробкою вироби;

2. Циклічні повторення реалізованих фаз з можливо більш ранньої фази.

Основні етапи каскадної схеми ЖЦПЗ:

1. Аналіз здійсненності системи.
2. Підтвердження.

3. Планування і аналіз вимог до ПЗ.
4. Підтвердження.
5. Проектування виробу.
6. Верифікація.
7. Детальне проектування.
8. Верифікація.
9. Кодування.
10. Автономна налагодження.
11. Впровадження.
12. Системна налагодження.
13. Комплексування.
14. Верифікація виробу.
15. Функціонування і супровід.
16. Повторне підтвердження.

У каскадній моделі успішне закінчення однієї з фаз ЖЦПЗ означає досягнення відповідної мети інженерного програмування. До цих підцілей необхідно додати ще дві:

1. Детального проектування – отримання повних верифікованих специфікацій і структур управління і даних, інтерфейсних зв'язків, характеристик, основних алгоритмів та визначення умов роботи кожного програмного компонента.

2. Кодованого – отримання повного, верифікованого набору компонентів програми.

Основні переваги даної моделі:

1. Формування повного набору проектної документації в кінці роботи над етапом. Документація відповідає критеріям повноти і завершеності.

2. Можливість планування термінів і витрат. Для цілого ряду ПЗ ця модель реалізована - це для систем, для яких на етапі аналізу можна точно і повно сформувати всі вимоги. Наприклад, складні обчислювальні програми.

Основні недоліки:

1. Великі терміни від аналізу до завершення.

2. Вимоги до ПЗ «заморожені» у вигляді ТЗ до кінця розробки.

В економічному аналізі Боемі зазначає, що ця модель орієнтована на послідовне досягнення цілей та базується на двох головних передумовах:

1. Для отримання якісного програмного виробу (тобто такого, яке в повній мірі задовольняє всім цілям необхідного програмного виробу) необхідно в будь-якому випадку здійснити всі під-цілі на кожному етапі.

2. Будь-яке інше впорядкування підцілей призводить до створення менш якісного програмного виробу.

Приклад каскадної моделі ЖЦПЗ наведено на рис. 2.1.

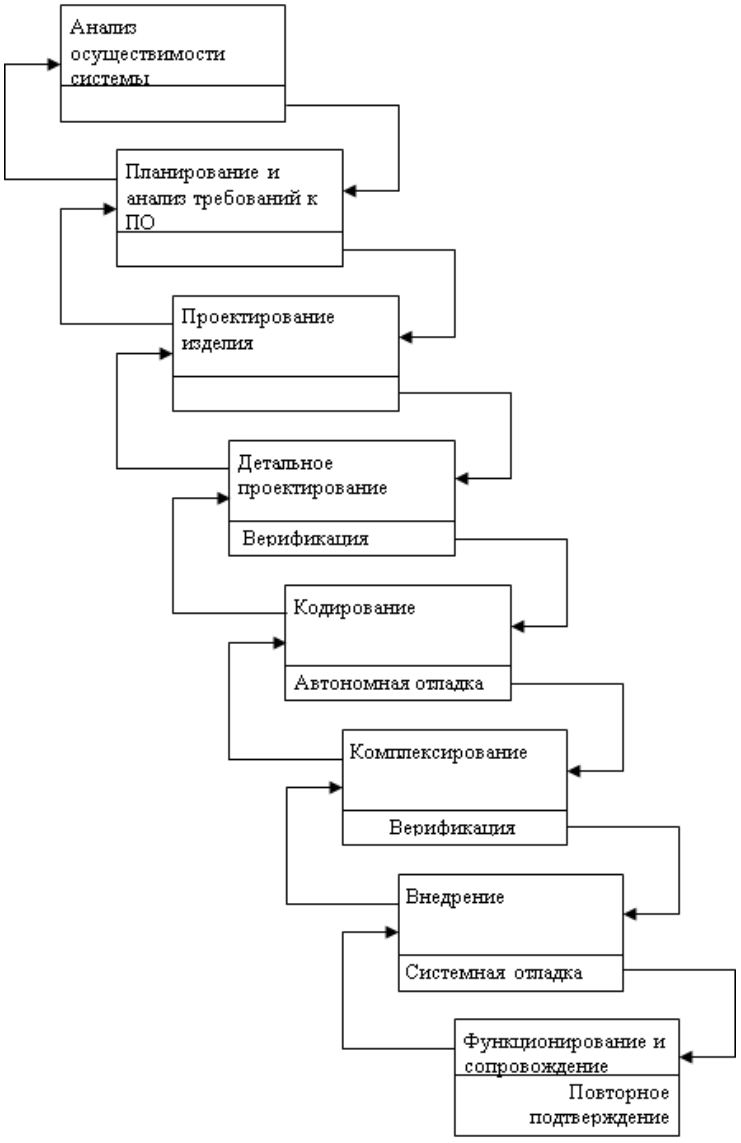


Рис. 2.1. Каскадна модель ЖЦПЗ

2.4. Опис використаних технологій та мов програмування

2.4.1. Опис середовища програмування

Кожна програма повинна мати зручний інтерфейс для спілкування з користувачем. Основним елементом інтерфейсу у Windows є вікна. Одним з різновидів вікон є форма, яка може містити кнопки, текстові поля тощо. Тому програми, написані для використання у Windows, зазвичай мають інтерфейс, подібний до вікон та форм. Для швидкого і зручного створювання програм з графічним інтерфейсом використовують спеціальні середовища візуального програмування. Майже кожна сучасна мова програмування має принаймні одне таке середовище: Object Pascal - Borland Delphi, C++ - Borland C++ Builder, Microsoft Visual C.

Переваги C++ Builder полягають в мові C++, яка лежить в основі. Це найпотужніша сучасна алгоритмічна мова загального призначення. Наряду із своєю простотою C++ Builder дає широкі можливості з розробки складних і ефективних програм.

Інструментальна система Builder, подібно до інших систем візуального програмування, насамперед є посередником поміж інтерфейсом прикладного програмування Windows та програмістом, надаючи змогу навіть програмістам початківцям оперативно створювати програмні проєкти, які матимуть графічний інтерфейс користувача найрізноманітнішої спрямованості, від суто обчислювальних і логічних до графічних і мультимедійних.

Оболонка C++ Builder надає змогу замість повного самостійного написання програми використовувати великий набір готових візуальних об’єктів, так званих компонентів, піктограми яких розміщені на відповідних вкладках палітри компонентів.

В C++ Builder існує понад 100 компонентів. Всі компоненти зібрано у бібліотеці візуальних компонентів VCL - Visual Class Library.

C++ Builder призначено для написання програм мовою програмування C++ і поєднує VCL та середовище програмування, написані на Delphi з

компілятором C++. Цикл створення програмних проєктів у C++ Builder є аналогічний до Delphi, але із суттєвими поліпшеннями. Більшість компонентів, розроблених у Delphi, можна використовувати і в C++ Builder без модифікації, але, зворотне твердження не слушне.

C++ Builder дозволяє методом «drag-and-drop» доволі просто розробляти інтерфейсні програми, що зумовлює підвищення ефективності та простоту програмування, оскільки програмістові не треба кожного разу створювати ті елементи власних програм, котрі може бути реалізовано за допомогою вже існуючих об’єктів.

Основним будівельним об’єктом візуального програмування є компонент. Компонентами в C++ Builder є об’єкти чи класи об’єктів, які є, у певному розумінні, об’єктами «реального світу». Їх безпосередньо видно на екрані (за винятком групи невидимих компонентів), їх можна пересувати мишею, вони можуть реагувати на клацання клавіш клавіатури і миші тощо. Своєю чергою, компонентам, на відміну від звичайних об’єктів C++, притаманна наявність властивостей, подій та методів, які дозволяють здійснювати різноманітні операції з цими компонентами. Властивості дозволяють легко встановлювати різні характеристики компонентів, такі як назва, розміри, контекстні підказки чи джерела даних. Методи виконують певні операції над компонентним об’єктом, у тому числі й такі складні, як відтворювання чи перемотування пристрою мультимедіа. Події пов’язують зовнішні впливи, на які реагують компоненти, такі як активізація, натиснення кнопок чи редаговане введення з кодами реакції на ці впливи. Вони також можуть виникати за таких специфічних змінювань стану компонента, як поновлення даних в інтерфейсних елементах доступу до баз даних [20].

Програмування в C++ Builder складається з двох етапів:

– конструювання візуального інтерфейсу за допомогою компонентів;

– написання програмного коду, виконання команд якого забезпечить розв’язок певної задачі.

Вікно середовища C++ Builder при завантаженні складається з таких

елементів:

– вікно форми;
– вікно коду програми;
– головне меню;
– «гарячі» кнопки інструментальних панелей;
– палітра компонентів;
– вікно інспектора об’єктів.

В плані роботи з документами, найбільш використовуваними в Windows для більшості користувачів є MS Word і MS Excel. Природно, результати вирішення прикладних завдань для них бажані у вигляді документів знайомих додатків.

Створення таких прикладних програм, де підсумковий документ формується не тільки в форматі, але і з використанням можливостей MS Word або MS Excel, отримало назву розробки контролерів автоматизації або управління серверами автоматизації. Прикладна програма розглядається як контролер, а Word і Excel - як сервер.

Контролери автоматизації здійснюють управління серверами автоматизації за допомогою викликів їх методів і з використанням змінних типу Variant - дозволяє працювати з усіма об'єктами сервера автоматизації за допомогою в основному чотирьох викликів:

– OlePropertyGet приймає рядок як параметр і повертає дані, що містяться в зазначеному властивості об'єкта;

– OlePropertySet приймає кілька параметрів, перший з яких - рядок, яка вказує на змінюване властивість, а наступні параметри - дані, які будуть записані в це властивість;

– OleProcedure і OleFunction виконують вказаний метод об'єкта.

Об'єкти до яких звертається сервер мають ієрархічну структуру (приклад для Excel):

– додаток (Excel.Application) містить одну або більше книг (Workbooks);
– книги, що містять одну або більше сторінок (Worksheets) або (і) діаграм

(Charts);

– сторінки, що містять комірки (Cells), рядки, стовпці, малюнки, діаграми.

Відповідно, всередині властивостями об'єктів Word можуть бути колекції: Paragraphs, Words, Tables [15].

2.4.2. Опис мови програмування

Найбільш поширеною мовою програмування протягом декількох останніх десятиліть, безперечно, є мова C ++, на підставі якої «виросло» багато сучасних мов програмування і програмних середовищ. Цьому сприяли такі її властивості, як лаконічність, потужність, гнучкість, мобільність, можливість доступу до всіх функціональних засобів системи. Програмувати на C ++ можна як для Windows, так і для Unix.

C++ - універсальна мова програмування високого рівня з підтримкою декількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної. Розроблена Б'ярном Страуструпом в AT&T Bell Laboratories у 1979 році та названа «Сі з класами». Страуструп перейменував мову у C++ у 1983 р. Базується на мові Сі. Визначена стандартом ISO/IEC 14882:2011 [19].

Мова програмування C++ є дуже зручною у розробці прикладних програм; драйверів пристроїв; розробці ОС; відеоігор. Програми, складені мовою C++, є мобільними, тобто можуть бути виконані на комп’ютерах різних виробників і в різних операційних системах, завдяки чому C++ є особливо популярною.

Реалізацією мови C++ займаються одночасно декілька проєктів як безкоштовних, так і комерційних, а саме: GNU, Microsoft і Embarcadero.

C++ - високорівнева мова програмування загального призначення, поєднує властивості як високорівневих, так і низькорівневих мов програмування. Від свого попередника, мови програмування C, C++ відрізняється тим, що найбільшу увагу при розробці цієї мови було приділено

підтримці об'єктно-орієнтованого та узагальненого програмування.

У 1990-х роках C++ стала однією з найуживаніших мов програмування загального призначення [5].

Стандартна бібліотека C++ включає стандартну бібліотеку Сі з невеликими змінами, які роблять її відповіднішою для мови C++. Інша велика частина бібліотеки C++ заснована на Стандартній Бібліотеці Шаблонів (STL).

Основним способом організації інформації в Сі++ є класи. На відміну від типу структура (struct) мови Сі, що складається тільки з полів, клас Сі++ складається з полів і функцій-членів або методів. Поля бувають публічними (public), захищеними (protected) і приватними (private). У Сі++ тип структура аналогічний типу клас, відмінність в тому, що за умовчанням поля і функції-члени у структури публічні, а у класу - приватні.

З публічними полями можна робити ззовні класу все. До захищених і приватних полів не можна звертатися ззовні класу, щоб не порушити цілісність даних класу. Спроба такого звернення викличе помилку компіляції. До таких полів можуть звертатися тільки функції-члени класу (а також так звані функції-друзі і функції-члени класів-друзів) Поза тілом функцій-членів (а також друзів) захищені і власні поля недоступні навіть для читання. Такий захист полів називається - інкапсуляцією.

Використовуючи інкапсуляцію, автор класу може захистити свої дані від некоректного використання. Малося на увазі, що зміна способу зберігання даних, оголошених як захищені або приватні не вимагає відповідних змін в класах, які використовують змінений клас. Наприклад, якщо в старій версії класу дані зберігалися у вигляді лінійного списку, а в новій версії - у вигляді дерева, ті класи, які були написані до зміни формату зберігання даних, переписувати не буде потрібно, якщо дані були приватними або захищеними (у останньому випадку - якщо використані класи не були класами-нащадками), оскільки жоден з класів не міг би безпосередньо звертатися до даних, а тільки через стандартні функції, які в новій версії мають вже коректно працювати з новим форматом даних. Навіть оператор доступу operator[] може бути

визначений як - стандартна функція.

Функції-члени, як і поля, можуть бути публічними, захищеними і приватними. Публічні функції може викликати будь-хто, а захищені і власні - тільки функції-члени і друзі [14].

При створенні С++ прагнули зберегти сумісність з мовою С. Більшість програм на С справно працюватимуть і з компілятором С++.

Нововведеннями С++ порівняно з С є:

– підтримка об'єктно-орієнтованого програмування через класи;
– підтримка узагальненого програмування через шаблони;
– доповнення до стандартної бібліотеки;
– додаткові типи даних;
– обробка винятків;
– простори імен;
– вбудовані функції;
– перевантаження операторів;
– перевантаження імен функцій;
– посилання і оператори управління вільно розподіленою пам'яттю [1].

2.4.3. Опис використаної СУБД

2.4.3.1. Microsoft Office Access

Було обрано середовище Microsoft Office Access для реалізації бази даних у кваліфікаційній роботі. Microsoft Access обрано, бо програма розрахована на одного користувача, тому не потрібні серверні бази даних. Система управління базами даних Microsoft Access входить до складу пакета Microsoft Office, програмний додаток передбачає роботу з Office, тому не має сенсу розділяти використання іншої бази даних. Також для роботи потрібно тільки підключити драйвер.

Microsoft Access - система управління базами даних, програма, що входить до складу пакету офісних програм Microsoft Office. Має широкий

спектр функцій, включаючи зв'язані запити, сортування по різних полях, зв'язок із зовнішніми таблицями і базами даних. Завдяки вбудованій мові VBA, в самому Access можна писати підпрограми, що працюють з старими версіями Microsoft Office Access. Він є додатком Windows, розроблений фірмою Microsoft. СУБД Access працює під управлінням Windows. Всі переваги Windows доступні в Access. Можна вирізати, копіювати і вставити дані з усіх програм Windows в додаток Access і навпаки [13].

Систему Access можна розглядати і як середовище для розробки додатків. Використовуючи макроси для автоматизації завдань, можна створювати такі ж потужні, орієнтовані на користувача додатки, як і додатки, створені за допомогою «повноцінних» мов програмування, доповнювати їх кнопками, меню і діалоговими вікнами. Програмуючи на VBA, можна створювати програми, за проєктною потужністю, які не поступаються самій Access. Потужність і гнучкість системи Access роблять її сьогодні однією з кращих програм для управління базами даних [18].

Основні компоненти MS Access:

– конструктор таблиць;

– конструктор екранних форм;

– конструктор SQL-запитів (мова SQL в MS Access не відповідає стандарту ANSI);

– конструктор звітів, що виводяться на друк.

Таблиця - це основний об'єкт бази даних, призначений для збереження даних, документів та інших облікових записів. Запит - вибирає дані з таблиць згідно з умовами, що задаються. Форма - відображає дані з таблиць або запитів відповідно до форматів, описаних користувачем. Форма дозволяє переглядати, редагувати та друкувати дані. Звіт - відображає і друкує дані з таблиць або запитів згідно з описаним користувачем форматом. У звіті дані редагувати не можна.

В Access нова редакція вмісту зміненої комірки таблиці записується на диск відразу, як тільки курсор клавіатури буде поміщений в іншу комірку (або

нова редакція зміненого запису записується на диск відразу, як тільки курсор клавіатури буде поставлений в іншу запис). Таким чином, якщо раптово відключать електрику, то пропаде тільки зміна того запису, якого не встигли покинути.

Цілісність даних в Access забезпечується також за рахунок механізму транзакцій.

Microsoft Access є пропрієтарним програмним забезпеченням, тобто для його використання необхідно придбати ліцензію. Однак для використання готових додатків, створених за допомогою Access, ліцензія не потрібна. Для роботи такого додатка необхідна runtime-версія Access, яка розповсюджується безкоштовно.

Корпорація Microsoft поширює повнофункціональну версію Access як окремо, так і спільно з іншими додатками у складі пакетів Microsoft Office Professional, Microsoft Office Professional Plus і Microsoft Office Enterprise [8].

2.4.3.2. Microsoft SQL Server

Microsoft SQL Server - це реляційна система управління базою даних (СКБД). У реляційних базах даних дані зберігаються в таблицях. Взаємопов'язані дані можуть групуватися в таблиці, крім того, можуть бути встановлені також і взаємини між таблицями. Звідси і пішла назва реляційні - від англійського слова «relational» (споріднений, пов'язаний відносинами, взаємозалежний). Користувачі отримують доступ до даних на сервері через програми, а адміністратори, виконуючи завдання конфігурування, адміністрування та підтримки бази даних, виробляють безпосередній доступ до сервера. SQL Server є масштабованої базою даних, це означає, що вона може зберігати значні обсяги даних і підтримувати роботу багатьох користувачів, які здійснюють одночасний доступ до бази даних [17].

СУБД настільки пов'язана з операційною системою Windows, що її надійність, масштабованість і продуктивність визначаються надійністю,

масштабованість і продуктивністю самої платформи, і положення SQL Server на ринку буде залежати від випуску нових версій Windows.

Однією з переваг SQL Server є простота його застосування, зокрема адміністрування. SQL Server Enterprise Manager, що входить до складу всіх редакцій Microsoft SQL Server (за винятком MSDE), являє собою повнофункціональний і досить простий засіб для адміністрування цієї СУБД. За даними Transaction Processing Performance Council (TPC), SQL Server зараз є рекордсменом по продуктивності.

Таким чином, головними перевагами SQL-Server є:

– високий ступінь захисту даних;

– потужні засоби роботи з даними;

– висока продуктивність;

– зберігання великих масивів даних;

– зберігання даних, що вимагають дотримання режиму секретності або при недопустимості їхньої втрати [17].

Основні функції в останніх версіях ще раз підтверджують той факт, що Microsoft продовжує розвивати свої продукти, намагаючись задовольнити зростаючі вимоги споживачів.

Головна перевага платного ПО в порівнянні з безкоштовним - це особлива підтримка, яку отримуєте. В даному випадку, перевага ще більш значуща, так як SQL сервер підтримується однією з найбільших компаній в світі. Microsoft створила додаткові інструменти для SQL сервера, які прив'язуються до реляційної СУБД, включаючи інструменти для аналізу даних. Система також має сервер звітів - Служба звітів SQL Сервера, так само як і інструмент ETL. Це робить SQL сервер швейцарським армійський ножем серед реляційних СУБД.

Модель HTAP дозволяє одночасно здійснювати операційні транзакції і аналітику на одних і тих же даних в одній і тій же пам'яті, також реалізуючи підхід in memory [21].

Розвиток SQL Server пішло по шляху інтеграції з іншими аналітичними

платформами, зокрема Spark, яка включена тепер в поставку SQL Server.

Spark є дуже популярним інструментом для машинного навчання, для просунутої аналітики, має ефективну in memory машину. І все це інтегровано з SQL, який дуже ефективний для візуалізації аналітики.

Правильний аналіз і ефективне представлення результатів безпосередньо впливає на ефективність аналізу даних і можливість приймати на їх основі управлінські рішення [21].

Захист конфіденційних даних за допомогою технології Always Encrypted з захищеними анклавами. Шифрування на місці дозволяє виконувати криптографічні операції з конфіденційними даними без їх переміщення за межі бази даних.

Криптографічні операції включають в себе шифрування стовпців, і ці операції тепер можна виконувати за допомогою Transact-SQL, вони не вимагають переміщення даних з бази даних. Усередині захищених анклавів підтримуються всі повнофункціональні обчислення, включаючи зіставлення і порівняння діапазонів, що значно розширює можливості їх застосування [21].

На рис. 2.2 зображено технологію «Always Encrypted», з захищеними анклавами доступна, в Windows Server 2019.

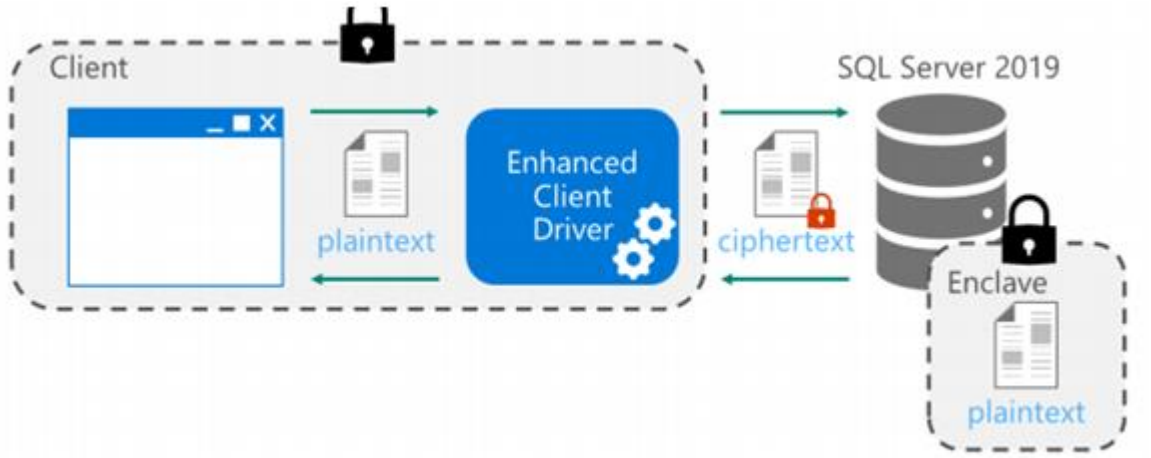


Рис. 2.2. Технологія «Always Encrypted» в SQL Server 2019

2.5. Опис структури програми та алгоритмів її функціонування

2.5.1. Опис логічної структури додатку

У даній роботі пропонується розробити програмний продукт, який дозволить автоматизувати задачу обліку поточної успішності студентів, оперативного контролю поточної успішності студентів і дозволить куратору працювати з електронним журналом в режимі реального часу.

Кожен семестр складається з розкладу і вноситься в базу даних електронного журналу. Крім того, в базі даних повинна бути актуальною інформація про студентів [12].

Загальна схема розроблювального програмного продукту наведена на рис. 2.3. Він буде складатись зі служби, що буде реалізована у вигляді консольного додатку, та клієнта – WinForms-додаток. Служба буде мати дві кінцеві точки (endpoint) для підключення клієнта (MEX та epLogin). Між службою і клієнтом буде встановлено дуплексний режим підключення.

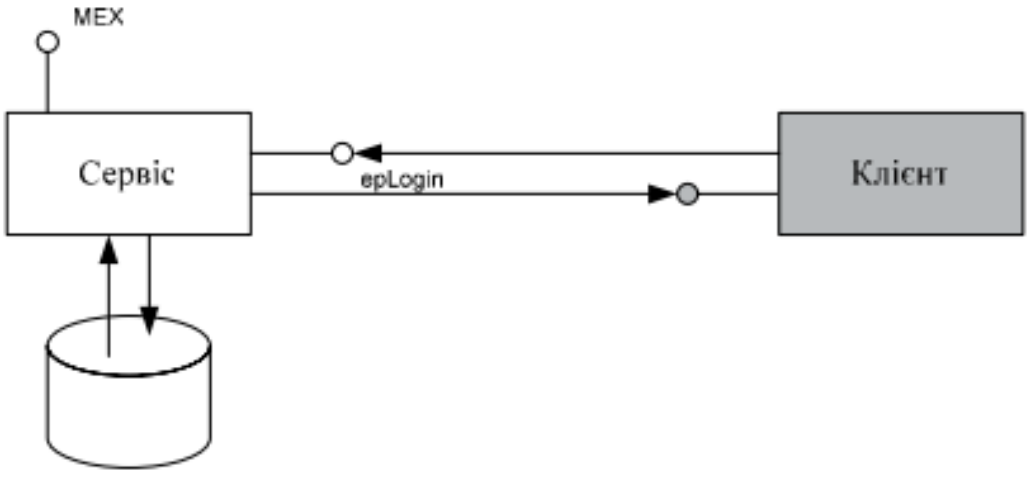


Рис. 2.3. Архітектура додатку

Крім того, служба буде підключатись до СУБД Access з використанням технології ADO.NET.

Клієнт (викладач) матиме наступні можливості:

– підключатись до служби;
– завершувати роботу зі службою;
– отримувати інформацію про студента;
– відмічати присутність студента на занятті;
– виставляти оцінки студентові.

Служба, завдяки дуплексному режиму підключення, буде періодично поновлювати інформацію у клієнта, зокрема сповіщати його, що почалося або закінчилося заняття.

Таким чином, в результаті аналізу вимог до програмного продукту виявлені наступні сутності:

– розклад занять;
– заняття;
– дисципліна;
– викладач;
– студенти;
– оцінки.

Між даними сутностями існують зв’язки, які можна розділити на дві групи: «один до багатьох» або «багато до одного» (викладач і розклад занять, дисципліна і розклад занять, розклад занять і заняття) та «багато до багатьох» (заняття і студенти, студенти і оцінки). Названі сутності та зв’язки між ними показані на рис. 2.4.

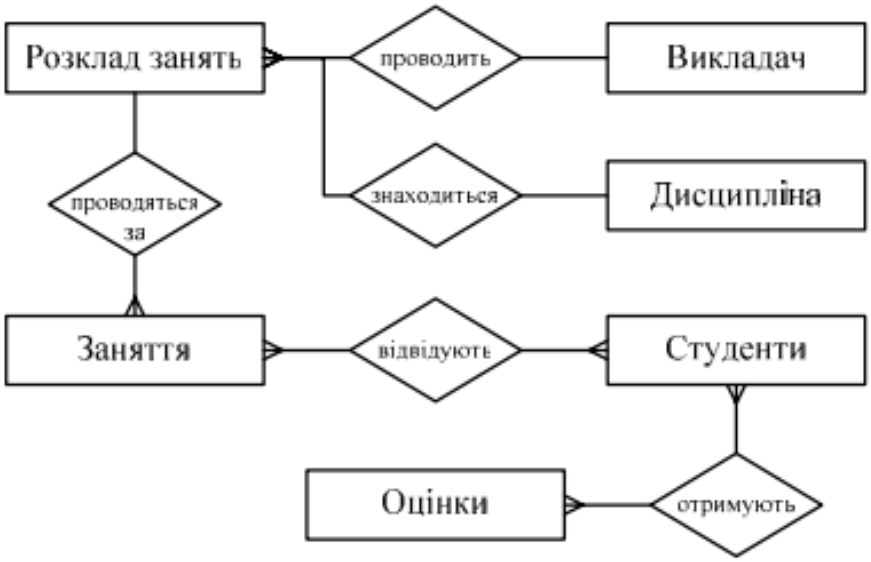


Рис. 2.4. Концептуальна модель даних

Виходячи із правил перетворення концептуальної моделі в логічну, включаючи перетворення зв’язків багато до багатьох, остання буде мати такий вигляд, як наведено на рис. 2.5.

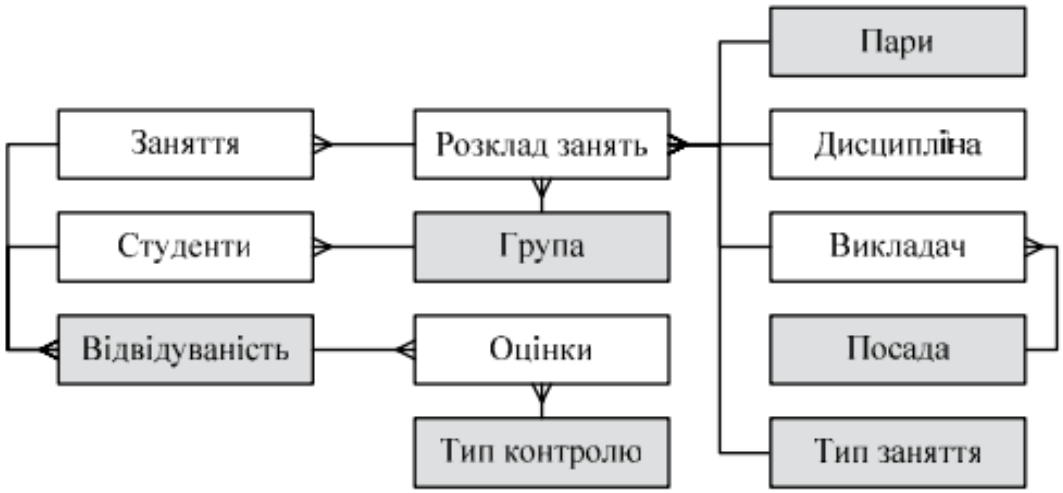


Рис. 2.5. Логічна модель даних

2.5.2. Опис бази даних

Для розробки програмного продукту було спроєктовано локальну базу даних, яка складається з одинадцяти таблиць.

Схему зв’язків між таблицями представлено на рис. 2.5.

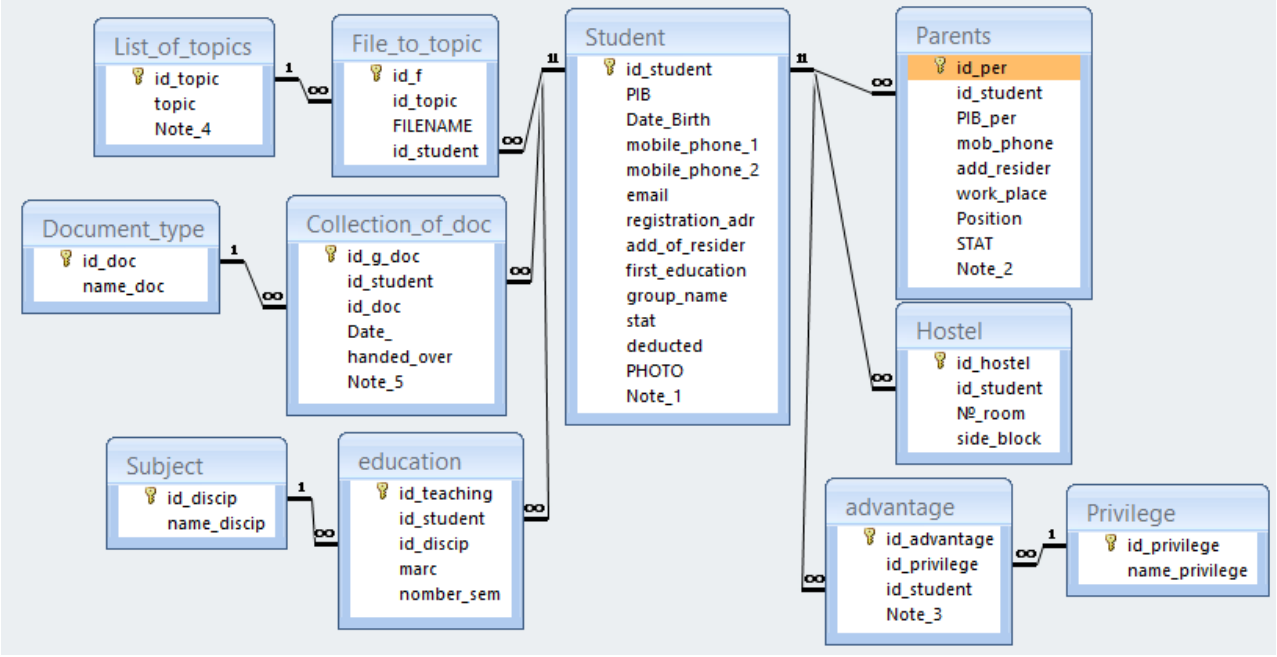


Рис. 2.5. Схема зв’язків між таблицями

В таблиці Student, яку наведено в таблиці 2.1, знаходиться інформація про студентів.

Таблиця 2.1

Дані про студента

Назва поля	Опис	Тип	Розмір	Ключ
id_student	Унікальний код студента	Лічильник		PK
PIB	ПІБ студента групи	Текстовий		
Date_Birth	Дата народження студента	Дата		
mobile_phone_1	Мобільний телефон студента	Чисельний		
mobile_phone_2	Додатковий мобільний телефон студента	Чисельний		
email	Електронна адреса студента	Текстовий		

Продовж. табл. 2.1

Назва поля	Опис	Тип	Розмір	Ключ
registration_adr	Адреса прописки студена	Текстовий		
add_of_resider	Адреса студента на даний час	Текстовий		
first_education	Інформація про освіту студента до вступу	Текстовий		
group_name	Назва групи студента	Текстовий		
stat	Стать чоловік/жінка	Текстовий	1	
deducted	Для позначення студентів, які були відраховані з групи	Логічний		
PHOTO	Зображення студента	Текстовий		
Note_1	Стисла, лаконічна довідка	Текстовий		

В таблиці Parents, яку наведено в таблиці 2.2, знаходиться інформація про батьків студентів.

Таблиця 2.2

Дані про батьків

Назва поля	Опис	Тип	Розмір	Ключ
id_per	Унікальний код батьків	Лічильник		PK
id_student	Зовнішній ключ - унікальний код студента	Чисельний		FK
PIB_per	ПІБ батьків студента групи	Текстовий		
mob_phone	Мобільний телефон батькамів	Чисельний		
add_resider	Адреса мешкання	Текстовий		
work_place	Місце роботи батьків	Текстовий		
Position	Посада	Текстовий		
STAT	Стать чоловік/жінка	Текстовий	1	
Note_2	Стисла, лаконічна довідка	Текстовий		

В таблиці Hostel, яку наведено в таблиці 2.3, знаходиться інформація про студентів які мешкають в гуртожитку.

Таблиця 2.3

Дані про гуртожиток

Назва поля	Опис	Тип	Розмір	Ключ
id_hostel	Унікальний код	Лічильник		PK
id_student	Зовнішній ключ - унікальний код студента	Чисельний		FK
№_room	№ кімнати	Чисельний		
side_block	Сторона блоку де знаходиться кімната студента	Текстовий		

В таблиці Privilege, яку наведено в таблиці 2.4, знаходиться інформація пільг, які доступні в закладі.

Таблиця 2.4

Дані про пільги

Назва поля	Опис	Тип	Розмір	Ключ
id_privilege	Унікальний код пільги	Лічильник		PK
name_privilege	Найменування пільги	Текстовий		

В таблиці advantage, яку наведено в таблиці 2.5, знаходиться інформація про студентів які мають пільги.

Таблиця 2.5

Дані про пільговиків

Назва поля	Опис	Тип	Розмір	Ключ
id_advantage	Унікальний код пільговика	Лічильник		PK
id_privilege	Зовнішній ключ - унікальний код пільги	Чисельний		FK
id_student	Зовнішній ключ - унікальний код студента	Чисельний		FK
Note_3	Стисла, лаконічна довідка, примітка	Текстовий		

В таблиці List_of_topics, яку наведено в таблиці 2.6, знаходиться інформація про теми для подальшого збереження матеріалів.

Таблиця 2.6

Дані про перелік тем

Назва поля	Опис	Тип	Розмір	Ключ
id_topic	Унікальний код теми	Лічильник		PK
topic	Тема матеріалів до виховної години	Текстовий		
Note_4	Стисла, лаконічна довідка, примітка	Текстовий		

В таблиці File_to_topic, яку наведено в таблиці 2.7, знаходиться інформація про додані матеріали.

Таблиця 2.7

Дані про файли до тем

Назва поля	Опис	Тип	Розмір	Ключ
id_f	Унікальний код файлу до певної теми	Лічильник		PK
id_topic	Зовнішній ключ - унікальний код теми	Чисельний		FK
FILENAME	Матеріали до теми	Текстовий		
id_student	Зовнішній ключ - унікальний код студента	Чисельний		FK

В таблиці Document_type, яку наведено в таблиці 2.8, знаходиться інформація про види документів, які збираються з групи.

Таблиця 2.8

Дані про вид документа

Назва поля	Опис	Тип	Розмір	Ключ
id_doc	Унікальний код документу	Лічильник		PK
name_doc	Найменування виду документа	Текстовий		

В таблиці Collection_of_doc, яку наведено в таблиці 2.9, знаходиться інформація про студентів які здали або не здали необхідні документи.

Таблиця 2.9

Дані про зібрані документи

Назва поля	Опис	Тип	Розмір	Ключ
id_g_doc	Унікальний код зібраного документа	Лічильник		PK
id_student	Зовнішній ключ - унікальний код студента	Чисельний		FK
id_doc	Зовнішній ключ - унікальний код виду документа	Чисельний		FK
Date_	Дата здачі документа	Дата		
handed_over	Здав всі документи	Логічний		
Note_5	Стисла, лаконічна довідка, примітка	Текстовий		

В таблиці Subject, яку наведено в таблиці 2.10, знаходиться інформація про дисципліни.

Таблиця 2.10

Дані про дисципліни

Назва поля	Опис	Тип	Розмір	Ключ
id_discip	Унікальний код предмету	Лічильник		PK
name_discip	Найменування (дисципліни)	Текстовий		

В таблиці education, яку наведено в таблиці 2.11, знаходиться інформація про студентів та їх оцінки.

Таблиця 2.11

Дані про навчання

Назва поля	Опис	Тип	Розмір	Ключ
id_teaching	Унікальний код навчання	Лічильник		PK
id_student	Зовнішній ключ - унікальний код студента	Чисельний		FK
id_discip	Зовнішній ключ - унікальний код дисципліни	Чисельний		FK
mark	Оцінка, яка була отримана студентом під час навчання	Чисельний		
nomber_sem	Номер семестру	Чисельний		

2.5.3. Опис розроблених методів

2.5.3.1. Створення шаблонів

При розробці програмного продукту були створені шаблони в офісних електронних документах (MSExcel, MSWord), в яких було встановлено «закладки».

В лістингу 2.1 наведено приклад підключення створеного шаблону до проєкту.

Лістинг 2.1. Фрагмент підключення шаблону MSExcel

```
App=CreateOleObject("Excel.Application");
App.OlePropertySet("Visible",false);
Bks=App.OlePropertyGet("Workbooks");
App.OlePropertySet("SheetsInNewWorkbook",1);
if (flag == true) {String pasES = ExtractFilePath (Application->ExeName) +
"\\model\\studentAutograph.xlsx" ;
Bks.OleProcedure("Add", WideString(pasES));}
```

При заповненні шаблонів «MSWord» потрібно знайти «закладку» та змінити її на необхідну інформацію. Для цього було розроблено функцію пошуку-заміни. В лістингу 2.2 наведено створену функцію.

Лістинг 2.2. Пошук-заміна «закладки» в електронній офісній системі «MSWord».

```
void classMy::changeWord(Variant vApp, Variant vDoc, String bookmarker,
String information)
{Variant vBookmarks, vSelection, vRange, vBookmark;
vBookmarks = vDoc.OlePropertyGet("Bookmarks");
vBookmark = vBookmarks.OleFunction("Item", WideString(bookmarker));
vBookmark.OleProcedure("Select");
vSelection = vApp.OlePropertyGet("Selection");
vSelection.OlePropertySet("Text", WideString(information));
vRange = vDoc.OleFunction("Range", 0, 0);
vRange.OleProcedure(_T("Select"));}
```

Для шаблонів створених в MSExcel створено функцію, яка допомагає створити «рамку» навколо необхідної інформації. В лістингу 2.3 наведено розроблену функцію.

Лістинг 2.3. Код для створення «рамки» навколо інформації

```
void classMy::borders(Variant Cll)
 {Cll.OlePropertySet("HorizontalAlignment",-4108);
  Cll.OlePropertySet("VerticalAlignment",-4108);
  Cll.OlePropertyGet("Borders",8).OlePropertySet("LineStyle",1);
  Cll.OlePropertyGet("Borders",9).OlePropertySet("LineStyle",1);
  Cll.OlePropertyGet("Borders",7).OlePropertySet("LineStyle",1);
  Cll.OlePropertyGet("Borders",10).OlePropertySet("LineStyle",1);}
```

2.5.3.2. Методи перевірки коректності роботи програми

Для коректної роботи програми, потрібно приділяти значну кількість часу, для усунення можливих помилок. Тому в коді програми прописані перевірки на коректність введеня даних, а саме перевірка на введеня будь-якого символа, превірка на введеня першого символа «Backspace», «Space»; перевірка введеня першим символом тире. Приклад представлено в лістингу 2.5.

Лістинг 2.5 – Унеможливлення введення даних при введенні або при пошуку

```
void MyClass::proverka_Key(TEdit *Edit1, System::WideChar &Key)
{if(iswalpha(Key)) return;
if(Key==8) return; if(Key==32
{if( Edit1->SelStart==0) Key=0; return;}if(Key=='-')
{ if( Edit1->SelStart==0)Key=0; return;}
if(input_1->Label1->Caption=="Введіть вид документа:"){if(isdigit(Key))
return;}
if(extra->Label1->Caption=="Пошук за назвою документа:")
{if(isdigit(Key)) return;} Key=0;}
```

Передбачено перевірку на перетворення введених слів. А саме перетворення першої літери у велику, а інші у маленьку. Приклад наведений в лістингу 2.6.

Лістинг 2.6. Перетворення у нормалізований вигляд введених слів

```
void  MyClass::proverka_priobraz(TEdit *Edit1){
AnsiString fam = Edit1->Text.SubString(1,1).UpperCase() + Edit1-
>Text.SubString(2,Edit1->Text.Length()).LowerCase();
Edit1->Text = fam;}
```

Також були застережені випадки видалення даних, які цьому не

підлягають, приклад представлений в лістингу 2.7. На рисунку 2.6 зображено попередження про заборону на видалення.

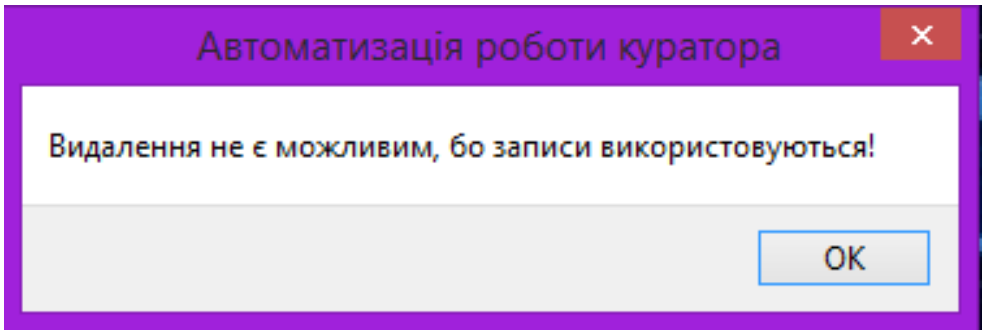


Рис. 2.6. Сповіщення про заборону

Лістинг 2.7 – Перевірка на заборону видалення даних, які використовуються

```
void MyClass::proverka_delet(int k, int k1, int volue1)
{
DM->AQ_dod->Close();
DM->AQ_dod->SQL->Clear();
switch (k)
{
case 1:{
DM->AQ_dod->SQL->Add("SELECT id_g_doc FROM Collection_of_doc where id_doc=:p");
DM->AQ_dod->Parameters->ParamByName("p")->Value=volue1;
break;}
case 2:{
DM->AQ_dod->SQL->Add("SELECT id_teaching FROM education where id_discip=:p");
DM->AQ_dod->Parameters->ParamByName("p")->Value=volue1;break;}
case 3:{
DM->AQ_dod->SQL->Add("SELECT id_advantage FROM advantage where
```

```
id_privilege=:p");DM->AQ_dod->Parameters->ParamByName("p")->Value=volue1;
break;}
        }
        DM->AQ_dod->Open();if(DM->AQ_dod->RecordCount)
        {
        ShowMessage("Видалення не є можливим, бо записи
використовуються!");}
        else{DM->AQ_dod->Close(); DM->AQ_dod->SQL->Clear();
        switch (k1){
        case 1:{
        DM->AQ_dod->SQL->Text="DELETE FROM Document_type where
id_doc=:p";
        DM->AQ_dod->Parameters->ParamByName("p")->Value=volue1;break;}
        case 2:{
        DM->AQ_dod->SQL->Text="DELETE FROM Subject where id_discip=:p";
        DM->AQ_dod->Parameters->ParamByName("p")->Value=volue1; break;}
        case 3:{
        DM->AQ_dod->SQL->Text="DELETE FROM Privilege where
id_privilege=:p";
        DM->AQ_dod->Parameters->ParamByName("p")->Value=volue1; break;}}
        DM->AQ_dod->ExecSQL();
        ShowMessage("Запис успішно видалено!");}}
```

Для використання всього функціоналу додатку необхідно мати доступ до бази даних, та авторизуватися в системі. На рисунку 2.7 зображено повідомлення про необхідність входу до бази даних.

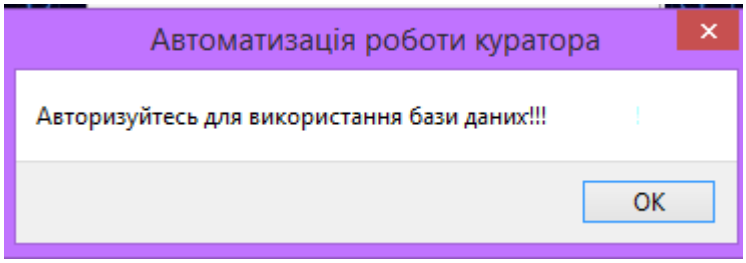


Рис. 2.7. Сповіщення про необхідність авторизації

На рис. 2.8. зображено спроєктовану форму для авторизації в базі даних

.

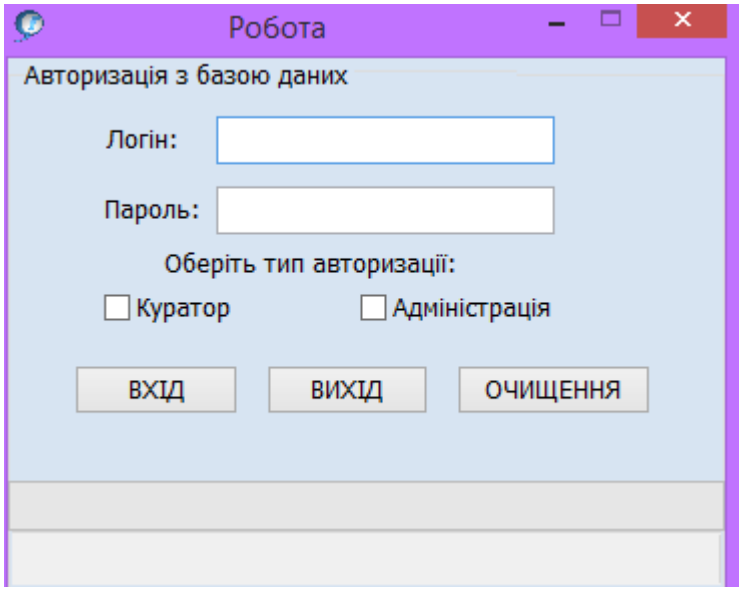


Рис. 2.8. Форма авторизації

Для авторизації з базою даних, було розроблено власний клас - «MyAccess».

Клас має наступні методи:

– MyAccess(){ access = false; idKurator = 0; PIB = ""; typeofAccess = "";} - конструктор за замовчуванням;

– void setAccess(TEdit *Edit1, TEdit *Edit2) - метод забезпечує перевірку логіну/пароля, визначає ПІБ користувача;

– bool getAccess() - повертає ознаку про доступ;

– String getPIB() - повертає ПІБ користувача;

– void denyAccess() - метод забезпечує вихід з бази даних;

– int getidKurator() - метод повертає ключове значення куратора групи;

– void setTypeofAccess(bool teacher, bool administration) - метод призначений для встановлення типу з’єднання, надає права доступу користувачу;

– String getTypeofAccess() - повертає тип з’єднання з базою даних.

В лістингу 2.8 наведено фрагмент коду-запиту - функції «setAccess», в якій отримаємо логін/пароль для того щоб забезпечити перевірку введених даних при авторизації в системі. Після отримання записів відбувається перевірка наявності пароля та логіна, якщо вони виявлені дозволяється вхід, якщо не виявлено користувач отримує повідомлення. На рис. 2.9 зображено приклад повідомлення про помилку при авторизації.

Лістинг 2.8 – Запит на отримання логінів/паролів

```
DM->ADOQueryAccess->Close();
DM->ADOQueryAccess->SQL->Clear();
DM->ADOQueryAccess->SQL->Add("select
[dekanat_new].[dbo].[security].*");
DM->ADOQueryAccess->SQL->Add("from [dekanat_new].[dbo].[security]");
DM->ADOQueryAccess->SQL->Add("order by
[dekanat_new].[dbo].[security].username");
DM->ADOQueryAccess->Open();
DM->ADOQueryAccess->First();
```

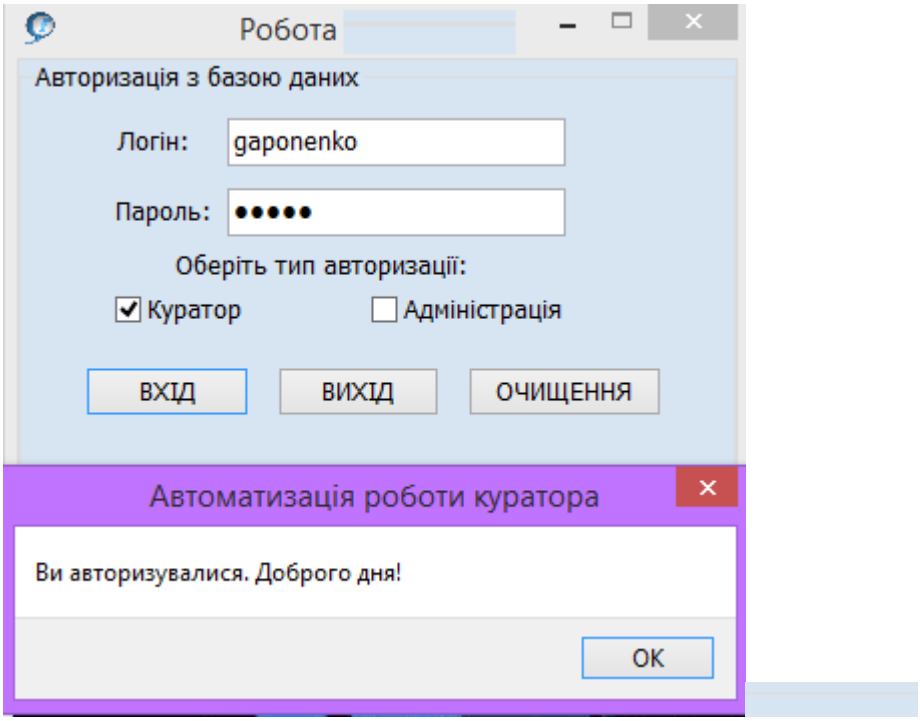


Рис. 2.9. Повідомлення при помилковій авторизації

Для з’єднання з локальною базою даних розроблено функцію, яка розташована у «Data Module». В лістингу 2.9 наведено фрагмент функції за допомогою якої відбувається встановлення з’єднання з базою даних.

Лістинг 2.9. Текст з’єднання з базою даних

```
if(!DM->ADOConnection1->Connected)
{DM->ADOConnection1-
>ConnectionString="Provider=Microsoft.Jet.OLEDB.4.0;Data
Source="+ExtractFilePath (Application->ExeName)+"BD_format.mdb"+";Persist
Security Info=False";
DM->ADOConnection1->LoginPrompt=false;
DM->ADOConnection1->Connected=true;
MainMENU->StatusBar1->Panels->Items[1]->Text="Успішне з'єднання з
базою даних.";}
else{MainMENU->StatusBar1->Panels->Items[1]->Text="З'єднання з базою
даних не є успішним.";}
```

2.5.4. Проєктування інтерфейсу користувача

Інтерфейс користувача є своєрідним комунікаційним каналом, по якому здійснюється взаємодія користувача і комп'ютера.

Кращий інтерфейс користувача - це такий інтерфейс, якому споживач не повинен приділяти багато уваги, майже не помічати його.

Кожна візуальна форма, яку використовують для створення інтерфейсу, має різні властивості та події.

Для вікна, що викликається з іншого, яке залишається відкритим, доцільно використовувати ShowModal, щоб користувач міг працювати тільки з новим відкритим вікном.

Візуальні форми передбачають наявність різних компонентів, які необхідні для взаємодії користувача з програмним додатком. Кожен компонент має властивості, які визначають поведінку в ході виконання програми. Також для компонентів можна створювати обробники подій, в яких доречно задавати подальші дії для обраного компонента.

На формі Input_1, що зображено на рис. 2.10 та 2.11 розміщено компоненти, які наведені нижче:

1. Компонент PageControl призначений для зберігання вкладок, на яких відображаються різні сценарії використання форми.
2. Компонент Edit1 призначений для введення інформації.
3. Компонент LabeledEdit1 призначений для введення примітки.
4. Компонент BitBtn1 призначений для збереження матеріалів.
5. Компонент BitBtn2 призначений для закриття(відміни) збереження.
6. Компонент DBLookupComboBox1 призначений для обирання теми виховної години.
7. Компонент Edit2 призначений для виведення шляху обраного файла.
8. Компонент OpenDialog1 призначений для відкриття шляху до необхідного файлу.
9. Компонент DBLookupComboBox2 призначений для обирання

студента, який створив файл.

10. Компоненти BitBtn3, BitBtn4, BitBtn5 призначені для збереження теми, обирання шляху, закриття або відміни збереження.

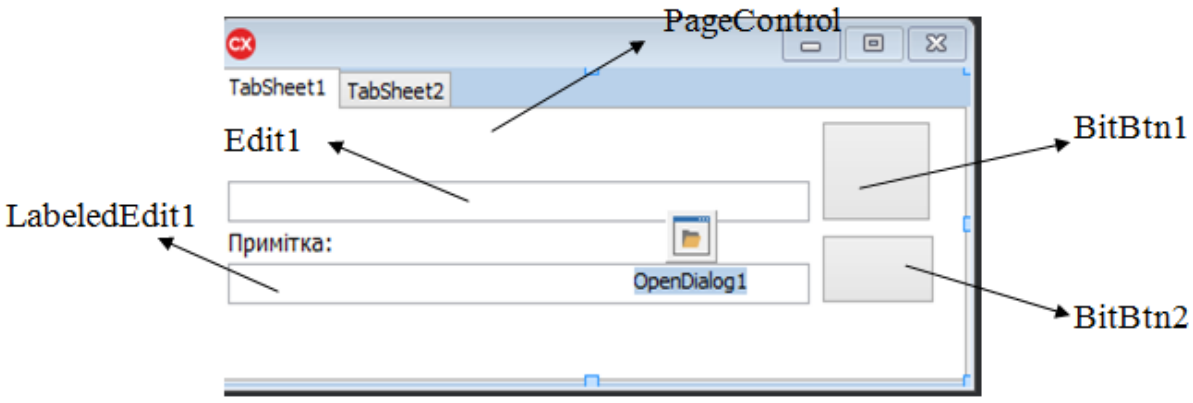


Рис. 2.10. Форма для введення даних Input_1

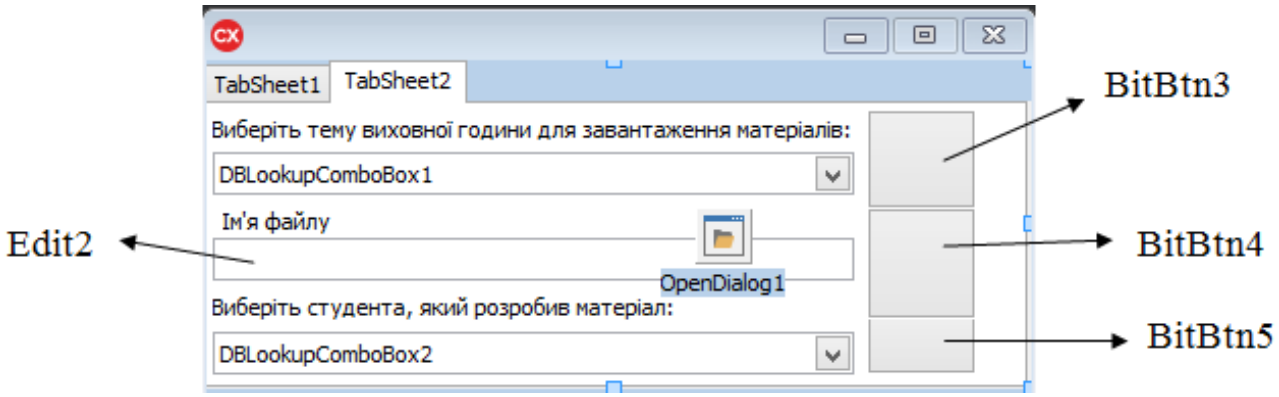


Рис. 2.11. Форма для введення матеріалу виховної години Input_1

Для форми deducted, що зображено на рис. 2.12- 2.14, використано компоненти, які наведені нижче:

1. Компонент MainMenu1 для відображення відповідного меню.
2. Компонент PageControl призначений для виконання різних напрямків використання форми: додавання та редагування записів про гуртожиток та пільги; для відображення зображення обраного студента.
3. Компонент StatusBar призначений для відображення результату роботи пошуку.
4. Компоненти BitBtn використано для швидкого виконання деяких дій

меню.

5. Компонент Edit1 призначений для виконання швидкого пошуку за ПІБ.

6. Компонент DBLookupComboBox2 призначений для обирання студента, виконання фільтрації.

7. Компоненти BitBtn1, BitBtn10 використано для скидання пошуку.

8. Компонент Image1 призначений для відображення зображень.

9. Компоненти BitBtn використано для збереження записів, відміни та збереження відредагованого запису.

10. Компонент DBLookupComboBox1 призначений для обирання пільги.

11. Компоненти Edit призначений для введеня інформації стосовно пільговиків.

12. Компонент Edit5 відобража ПІБ обраного студента, для додавання або редагування, що мешкає в гуртожитку.

13. Компоненти MaskEdit1 та UpDown2 призначені для введення номери кімнати в гуртожитку.

14. Компонент ListBox1 призначений для обирання сторони блоку в гуртожитку.

15. Компонент DBGrid1 призначений для обирання та відображення записів відповідно до обраного пункту меню.

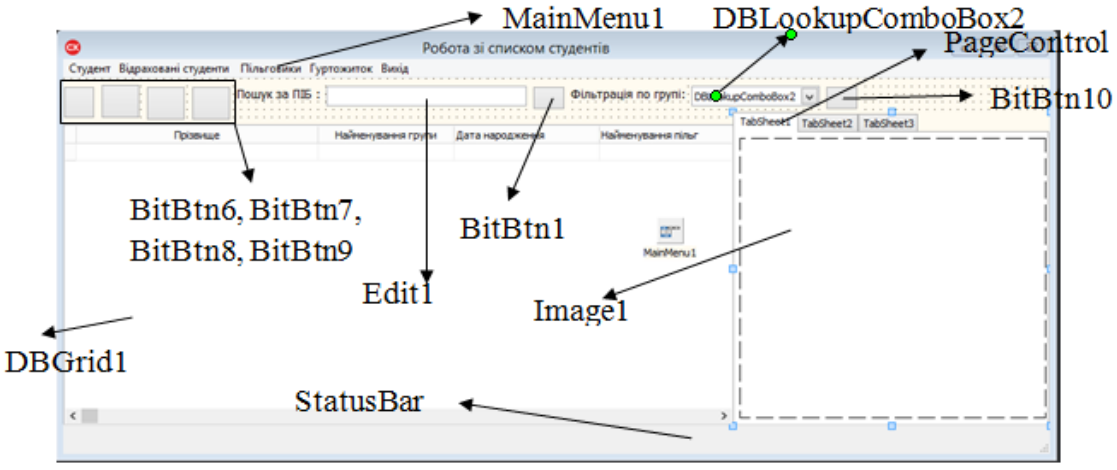


Рис. 2.12. Спроєктована форма deducted, відображення зображення

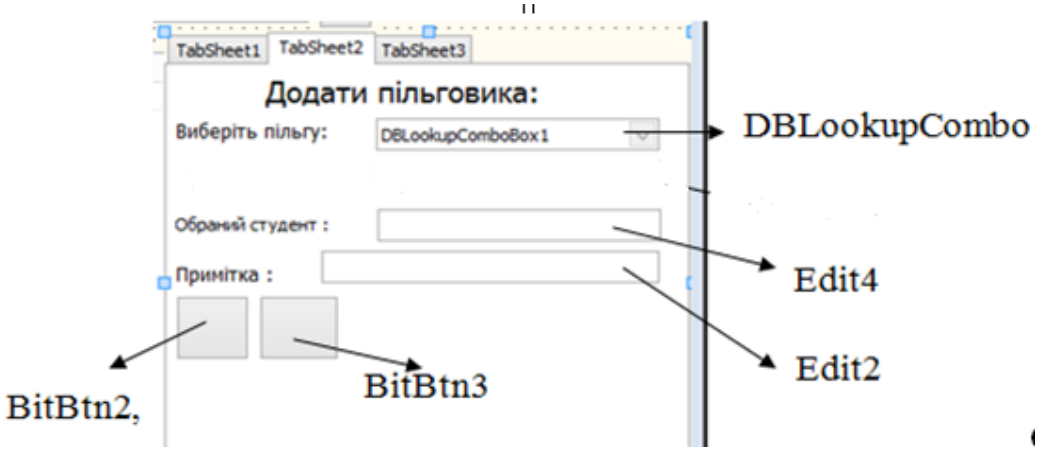


Рис. 2.13. Спроєктована форма deducted, для додавання пільговика

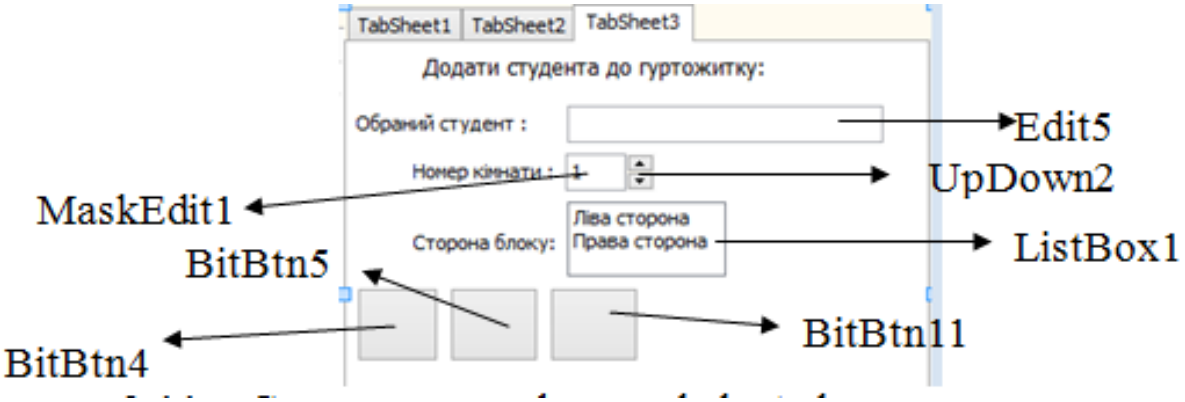


Рис. 2.14. Спроєктована форма для додавання запису про мешканців гуртожитку deducted.

Для форми smotr_student, використано наступні компоненти, які наведені нижче:

1. Компонети DBGrid1 використовую для відображення ПІБ та групи студента.
2. Компоненти BitBtn призначені для додавання інформації, редагування інформації про студента та батьків.
3. Компонент Image1 призначений для відображення зображень.
4. Компоненти Edit2 та Edit1 призначені для пошуку.
5. Компоненти BitBtn1, BitBtn2 призначені для скидання пошуку.
6. Компоненти DBLookupComboBox3, CheckBox використовують для

фільтрації даних.

7. Компоненти BitBtn використовують для скидання фільтрації, скидання відображеної інформації по студенту.

8. Компоненти Edit призначені для відображення детальної інформації по обраному студенту.

2.5.5. Структурна схема взаємодії складових програми

З головної форми користувач може перейти до іншої форми, з якої має можливість повернутися в головне меню. Також користувач може отримати доступ до інших форм та функціоналу програми не повертаючись в головне меню.

Схема взаємодії складових частин програми приведена на рис. 2.15.

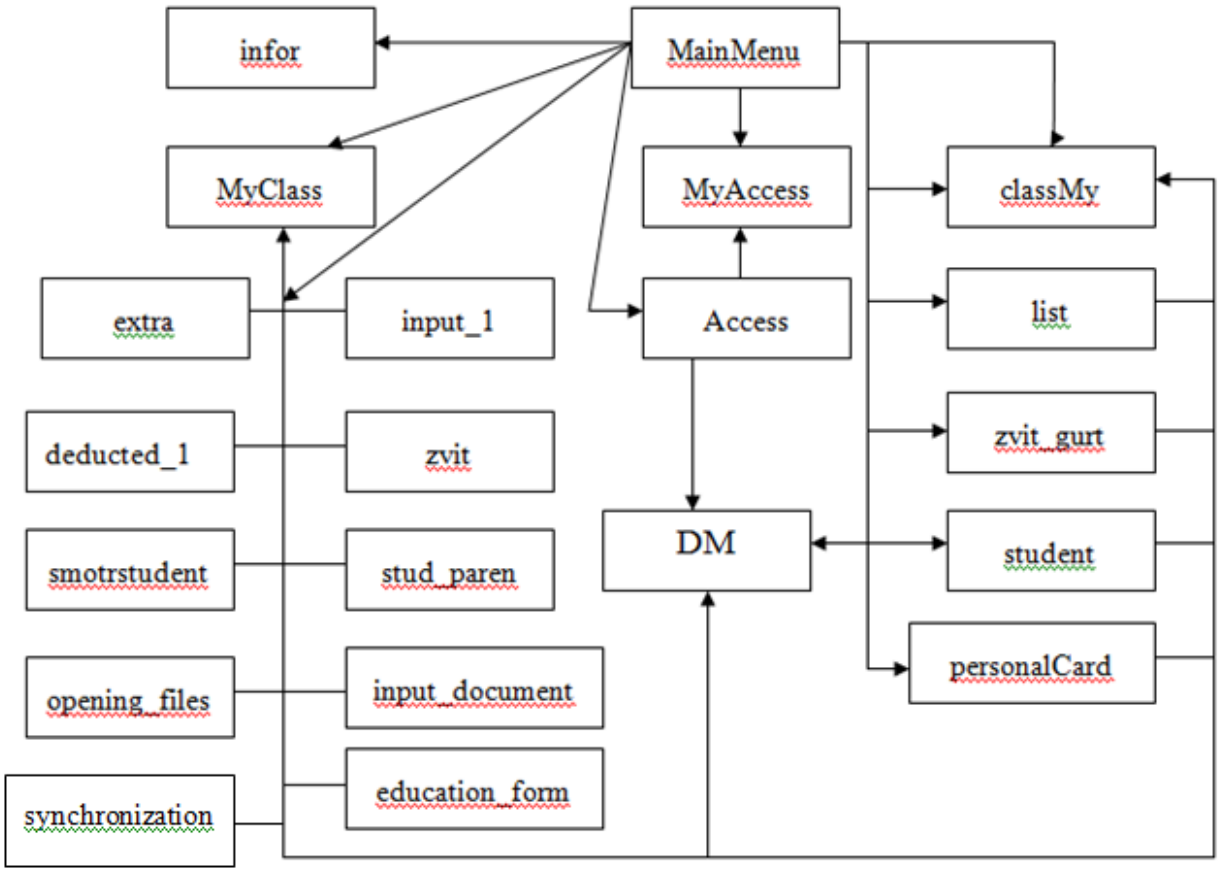


Рис. 2.15. Схема взаємодії складових частин програми

Програма містить двадцять один модуль. Всі компоненти, які пов’язані з базами даних винесені в «Data Module».

Спільний код при реалізації програми винесено в створені функції.

Форма «Access» має функцію для очищення полів - void clin().

Форма «deducted_1» має функції:

– pilg() - вивід(робота) зі списком студентів;

– prom_gur() - вивід списку студентів гуртожитку;

– ochis() - очищення полів;

– prosmotr_pil() - вивід списків - пільговиків;

– prosm_vidrah() - вивід відрахованих студентів;

– vidrah(int n) - відрахувати або поновити обраного студента.

– Форма «education_form» має наступні функції:

– vibor_student() - функція повертає «id» обраного студента, отримує ПІБ;

– clindv(),clindv2() - функції призначені для очищення полів;

– prosmotr_ocenok(int id_s) - вивід оцінки за обраним студентом.

Форма «input_document» має такі функції:

– clin() - очищення полів;

– vivod_doc(int nom) - вивід документів або вивід з пошуком.

Форма «opening_files» має функції, які наведені нижче:

– red_topic() - редагування обраної теми;

– delet_topic() - видалення обраної теми;

– delet_file() - видалення обраного файлу;

– open_file() - відкриття обраного файлу.

Форма «student» має функцію для побудови діаграми якості та успішності у навчання студентів обраної групи - void chart(bool flag).

Форма «stud_paren» має такі функції:

– proverka_vvoda() - функція забезпечує перевірку правильного введення даних;

– vvod_format(TLabeledEdit *LabeledEdit1) - перетворення першої літери

у велику, а інші у маленьку;

– ochistka(), ochistka_parent() - функція призначена для очищення полів даних;

– save_stud(int nom_m) - збереження інформації про студентів;

– save_mam(int nom_mam), save_pap(int nom_pap) збереження даних батьків.

Форма «extra» має наступні функції:

– delet_doc () - видалення найменування обраного документу;

– delet_sub() - видалення найменування обраної дисципліни;

– delet_priv() - видалення найменування обраної пільги;

– red_doc() - редагування найменування обраного документа;

– red_sub() - редагування найменуваня обраної дисципліни;

– red_priv() - редагування найменування обраної пільги.

Форма «zvit_f» має функції:

– chistka(), skid(), skid_spis(), ochis(), clin() - функції, які очищують поля після вибору;

– void gurtPilg(bool rez, bool printer) - функція, яка формує запит на формування звітів-списків для гуртожитку або пільговиків;

– void inforStud(int rez) - функція, яка формує звіт - інформація про студентів; друк звіту;

– zapr(int i) - функція, яка формує запрошення на батьківські збори; друк запрошення;

– smotr_print(int i) - функція, яка забезпечує перегляд та друк зібраних документів;

– void graf_2(int ocen_viv) - функція, яка формує запит для побудови діаграми.

Форма «timing» - синхронізація з базою даних «Деканат» має такі функції:

– lokalAdvantage() - функція, яка формує запит для локальної бази даних - студенти, які мають пільги;

– listStudent() - функція, яка формує запит для локальної бази даних - перелік студентів;

– poiskAdvantageDekanat(String fam, String im, String otch) - функція, яка формує запит для бази даних - студенти які мають пільги;

– lokalPrivilege() - функція, яка формує запит для локальної бази даних — список пільг;

– dekanatPrivilege() - функція, яка формує запит для бази даних - список пільг;

– lokalSubject() - функція, яка формує запит для локальної бази даних - список дисциплін;

– dekanatSubject() - функція, яка формує запит для бази даних - список дисциплін;

– dekanatGer(int kurs) - функція, яка формує запит для бази даних - список студентів з інформацією.

Програмний додаток має власні класи:

– class MyClass - клас призначений для роботи з розробленою власною базою даних та функціоналу деяких форм;

– class MyAccess - призначення для авторизації в базі даних;

– class classMy - клас має методи, які допомагають у роботі з базою даних; метод для створення ярлику програми, лістинг наведено в Додатку А.

Методи класу «MyClass»:

– bookmark_blocking - метод призначений для блокування вкладок компонента Pagecontrol;

– call_bookmark - метод призначений для активації необхідної вкладки;

– color_button - метод призначений для змінення розміру та кольору шрифту при активації кнопки;

– name_form - метод забезпечує виведення заголовку форми;

– proverka - метод призначений для перевірки наявності запису та забезпечення додавання, модифікації даних про дисципліну, пільгу та вид документа, теми виховної години;

– proverka_Key - перевірка правильності введення даних;

– proverka_priobraz - перетворення великої та малої літери;

– vivod - запит на вивід даних;

– poisk - запит для пошуку інформації;

– proverka_delet - видалення найменування дисципліни, документів, пільг;

– vivod_file - запит на вивід тем матеріалів;

– vivod_material - запит на перегляд матеріалів;

– vivod_spicok - запит на перегляд при роботі зі списком студентів;

– del_adv - запит на видалення пільговиків;

– del_student - запит на видалення інформації про студента: відрахованого та не відрахованого, студента з гуртожитку;

– zvit_host() - запит для звіту - гуртожиток;

– zvit_doc() - запит для звіту - документи, які здані чи не здані;

– zvit_advan() - запит для звіту - пільги;

– vibor_stud - запит на пошук студента за ПІБ, номером групи та приміткою.

Методи класу «class classMy»:

– CreateShortCut(const String &file) - метод призначений для створення ярлику програмного продукту;

– sqlGroup - метод відповідає за формування груп з бази даних «Деканат» враховуючи, що куратори дійсно були в цих групах;

– listStudent - формування списків студентів за обраною групою;

– studentInformation — формування повної інформації за обраним студентом;

– markNumber - переведення оцінок з стобальної системи у чотирьох бальну;

– borders(Variant Cll) - рамка для Excel звітів;

– markName - переведення оцінок з стобальної системи у «ECTS»;

– changeWord - метод за допомогою якого замінюються «закладки» в

офісних електронних системах у потрібний текст;

– disciplinesSearchByGroups - формування списку дисциплін за обраної групою та семестром;

– numberKurs(int semestr) - номер курсу за семестром;

– maxKurs(int idGroup) - максимальний курс (останній курс, який введено в базу даних) за обраною групою;

– subjectStudentBal - запит на отримання балів обраного студента за вказаною дисципліною.

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Для розробки програмного продукту було спроєктовано локальну базу даних, яка складається з одинадцяти таблиць та вміщує наступну вхідну інформацію для роботи системи:.

1. Дані про студента:
   - ПІБ студента;
   - номери телефону;
   - email;
   - стать;
   - дата народження;
   - адреса прописки;
   - адреса проживання;
   - примітка;
   - найменування групи;
   - ознака відрахування;
   - попередня освіта;
   - фото.
2. Дані про найменування:
   - предметів;

- документів;
- пільг.

3. Дані про пільговиків:
   - студент;
   - пільга;
   - примітка.
4. Дані про батьків:
   - студент;
   - ПІБ батьків;
   - номер телефону;
   - адреса проживання;
   - місце роботи;
   - посада;
   - стать;
   - примітка.
5. Дані про гуртожиток:
   - студент;
   - номер кімнати;
   - номер гуртожитку.
6. Дані про навчання протягом семестру:
   - студент;
   - предмет;
   - оцінка за предмет;
   - номер семестру.
7. Дані про зібрані документи:
   - студент;
   - документ;
   - дата;
   - здача;

– примітка.

В ході обробки цих даних, система у відповідь на запити з бази даних, формує вихідну інформацію, що відображається у відповідних полях інтерфейсу додатку та у звітних електронних документах додатків MSExcel та MSWord

2.7. Опис роботи розробленого програмного продукту

2.7.1. Використані технічні засоби

Для розробки даного ПЗ було використано ПК з наступними характеристиками:

- тип процесора: процесор з частотою 2.2 ГГц;
- ОЗУ об'ємом 4 Гб;
- 300 Мб доступного простору на жорсткому диску;
- жорсткий диск з частотою обертання 5400 об / хв.;
- дозвіл екрану 1024x768;
- клавіатура;
- маніпулятор «миша».

2.7.2. Використані програмні засоби

Дана автоматизована система розроблена засобами мови C++ середовища Builder 10.3 Rio з використанням СКБД Microsoft Office Access та офісних додатків MSExcel та MSWord.

2.7.3. Виклик та завантаження програми

Програма не потребує додаткового налаштування. Для запуску програмного додатку потрібно відкрити подвійним кліком файл - «Автоматизація роботи куратора.exe».

2.7.4. Опис інтерфейсу користувача

Після запуску додатку перед користувачем відкривається головна форма, яка дозволяє перейти в будь-який інший пункт або підпункт програми, отримати доступ до тек з фотографіями або файлами.

Головна форма додатку зображена на рис. 2.16.

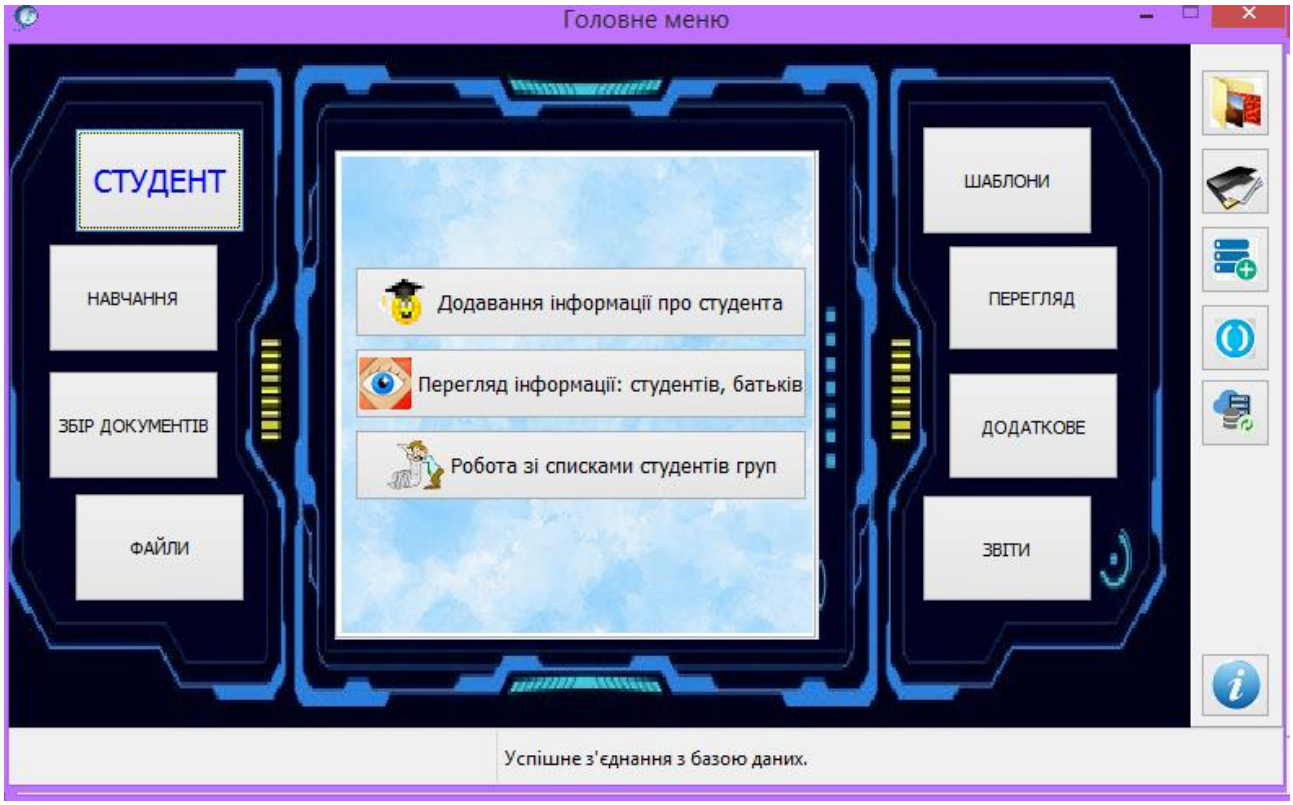


Рис. 2.16. Головне меню програми

Головна форма містить кнопку, яка дає можливість створити ярлик на робочому столі. На рис. 2.17 зображено повідомлення після натискання відповідної кнопки для створення ярлику.

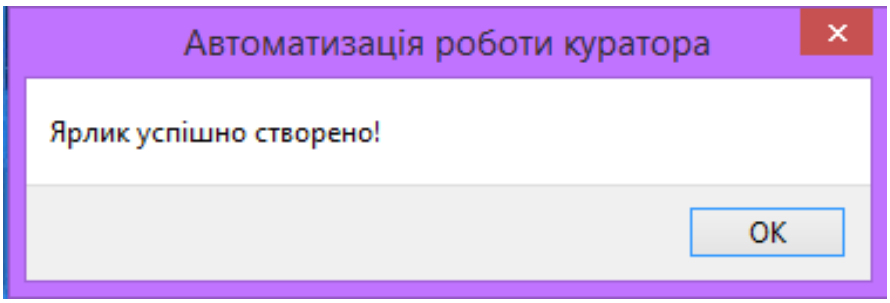


Рис. 2.17. Повідомлення після створення ярлику

Обравши кнопку у правому нижньому кутку головного меню - «Про програму» користувач отримує довідку про програмний додаток.

Обравши пункт «Додаткове», користувач може переглянути або додати інформацію стосовно: найменування пільг, найменування дисципліни, виду документа. На рис. 2. 18 зображено підпункти обраного пункту.

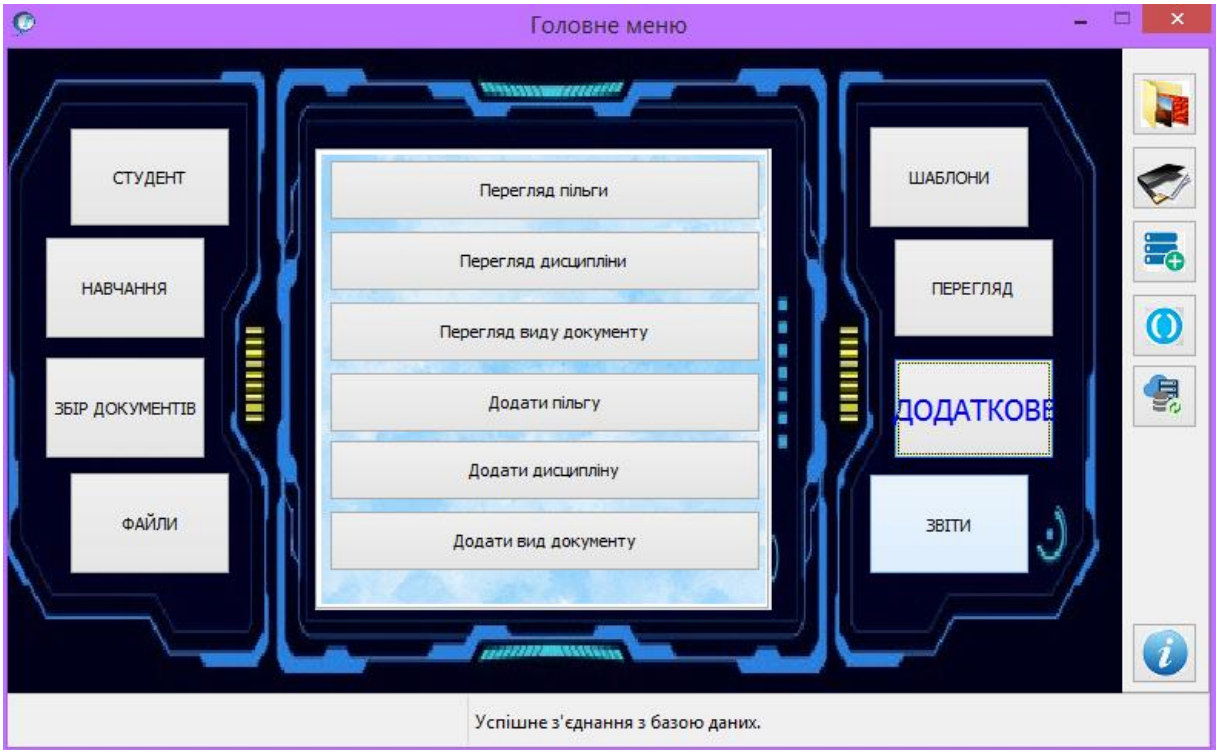


Рис. 2.19. Підпункти обраного пункту

Для додавання нової дисципліни, пільг та виду документа необхідно обрати відповідний підпункт.

Після введення назви дисципліни необхідно зберегти інформацію.

Якщо повторити введення дисципліни - Дискретна математика, то користувач отримає попередження про помилку. На рис. 2.20 зображено попередження про існування дисципліни. Для завершення роботи потрібно натиснути «ОК» та «Вихід», або змінити назву дисципліни.

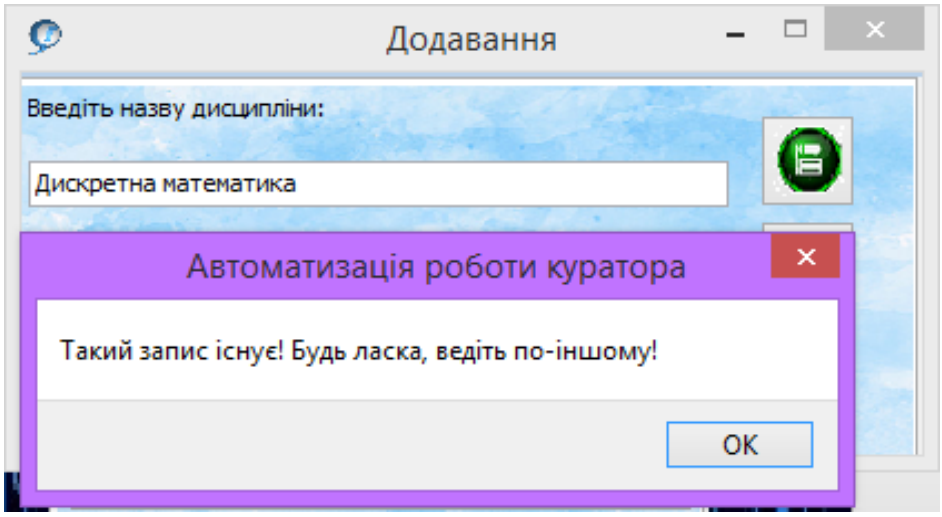


Рис. 2.20. Попередження про існування запису

Для перегляду записів дисциплін необхідно натиснути підпункт «Перегляд дисципліни». На рис. 2.21 зображено форму з переліком записів.

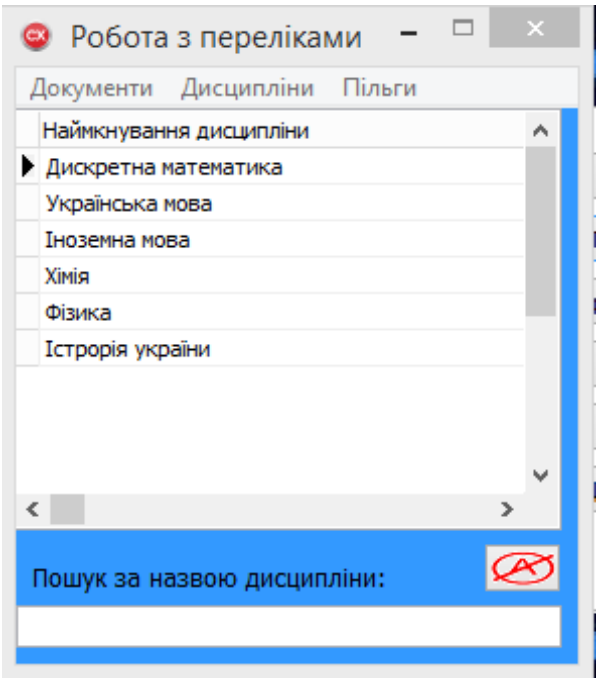


Рис. 2.21. Перелік введених дисциплін

Для перегляду пільг та видів документів необхідно повторити дії, які були описані про підпункт «Перегляд дисципліни».

При перегляді найменувань дисциплін користувач може переглянути і інші пункти. На рис. 2.23. зображено обирання пункту «Документи».

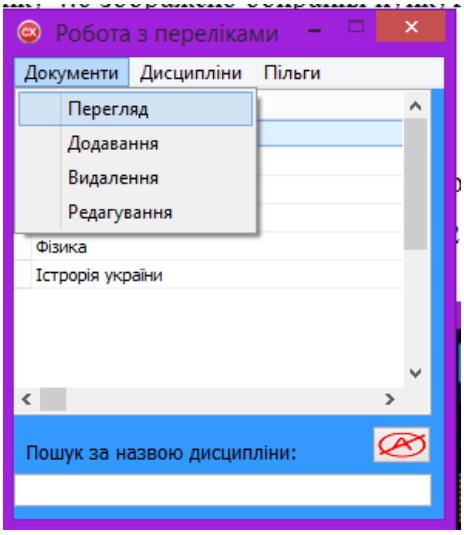


Рис. 2.22. Перегляд записів найменувань документів

Користувач має можливість видалити або відредагувати найменування документів, пільг та дисциплін, натиснувши праву клавішу миші. На рис. 2.23. зображено спливаюче меню. На формі «Робота з переліками» користувач може використати функцію пошуку відповідно до обраного пункту.

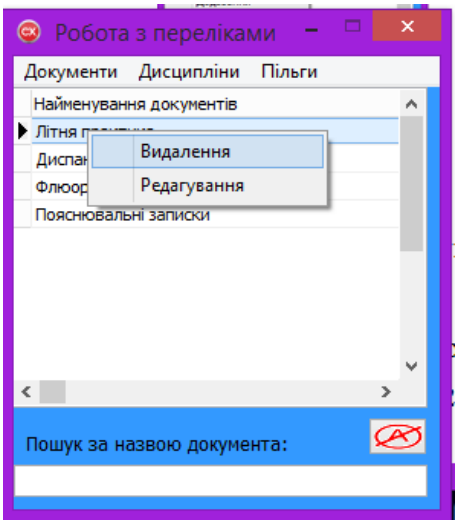


Рис. 2.23. Спливаюче меню

Обравши пункт меню «Перегляд» для користувача відкриваються підпункти цього меню. Користувач може швидко отримати доступ до даних стосовно студентів.

Користувач може обрати один з трьох варіантів перегляду даних. Обравши пункт меню «Перегляд списку гуртожитку», користувач отримує можливість додавати студентів, видаляти, переглядати, виконувати пошук; використовувати сортування та фільтрацію даних. Також надається можливість для редагування обраного запису. На рис. 2.24. зображено перегляд записів студентів, які мешкають в гуртожитку.

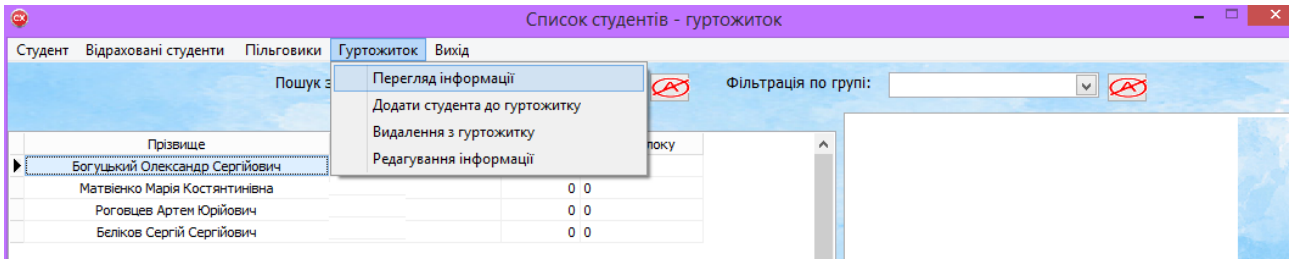


Рис. 2.24. Перегляд інформації про мешканців гуртожитку з використанням сортування за «Прізвищем»

Для додавання матеріалів, які потрібно зберегти для швидкого доступу, потрібно обрати відповідний підпункт меню. На рис. 2.25. зображено обирання користувачем пункту «Файли».

Рис. 2.25. Відкриття пункту меню «Виховна година»

Для додавання нової теми потрібно натиснути на кнопку «Додати тему для файлів». З’явиться вікно в якому потрібно заповнити дані про тему файлів та натиснути «Зберегти».

Для завантаження матеріалів для обраної теми потрібно натиснути кнопку «Додати матеріали». Але спочатку додати необхідні файли до відповідної теки, швидкий доступ до якої розташовано в меню програмного продукту, назва файлів - англійською (бажано). Після чого з’явиться вікно в якому потрібно обрати тему файлів, завантажити матеріал та по необхідності додати студента.

Для перегляду тем та для відкриття матеріалів потрібно натиснути кнопку «Відкриття матеріалів (файлів)». На рис. 2.26 зображено перегляд тем і матеріал. Для тем надано можливість пошуку за назвою.

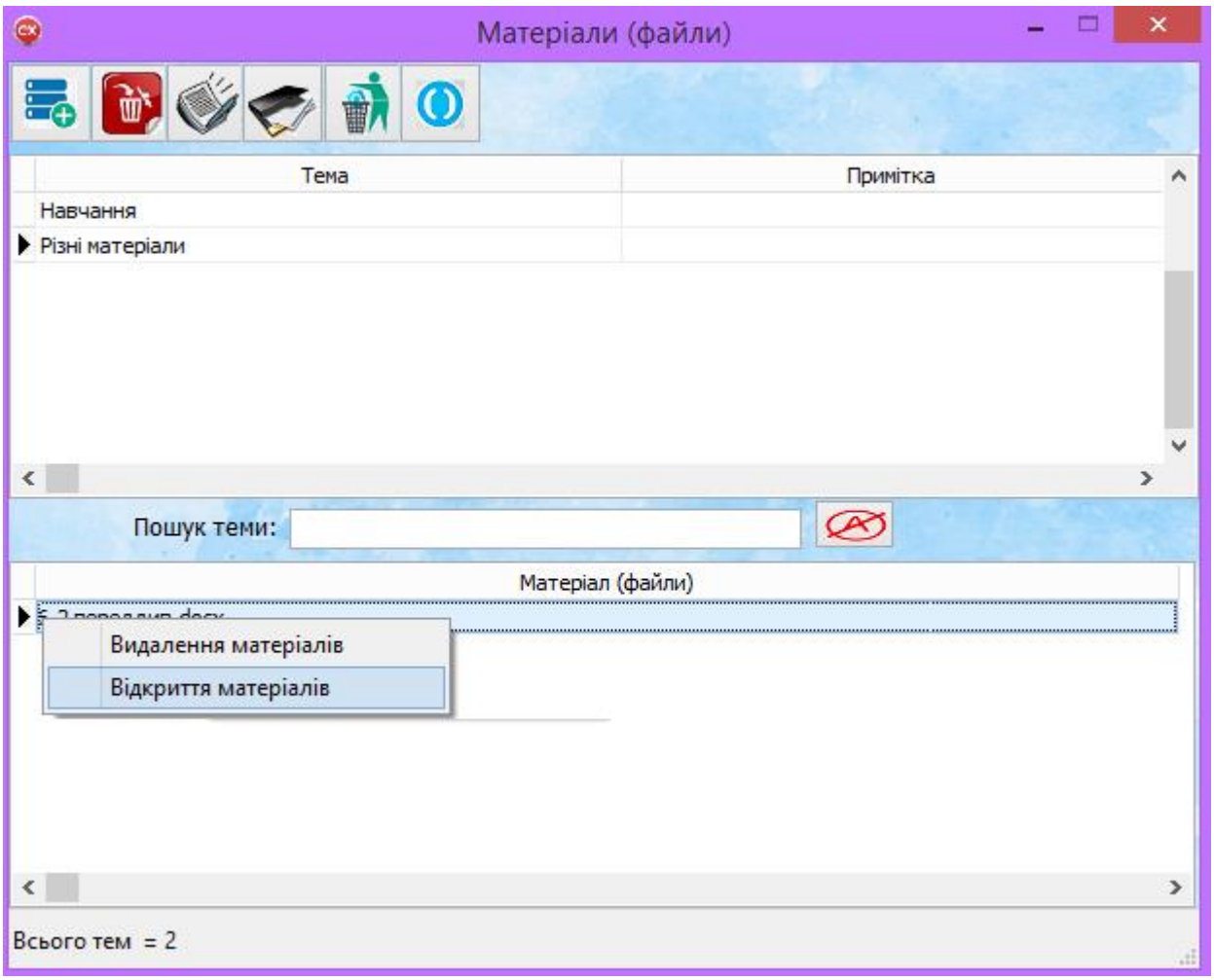


Рис. 2.26. Перегляд матеріалів

Основні елементи цієї форми - це можливість додати, видалити, відредагувати теми. А також відкрити потрібний матеріал (файл).

Для додавання зібраного документа потрібно перейти в пункт меню «Збір документів» та натиснути кнопку «Додати зібраний документ». На рис. 2.27 зображено вікно додавання зібраних документів з використанням фільтрації за обраною групою. Передбачено різну фільтрацію.

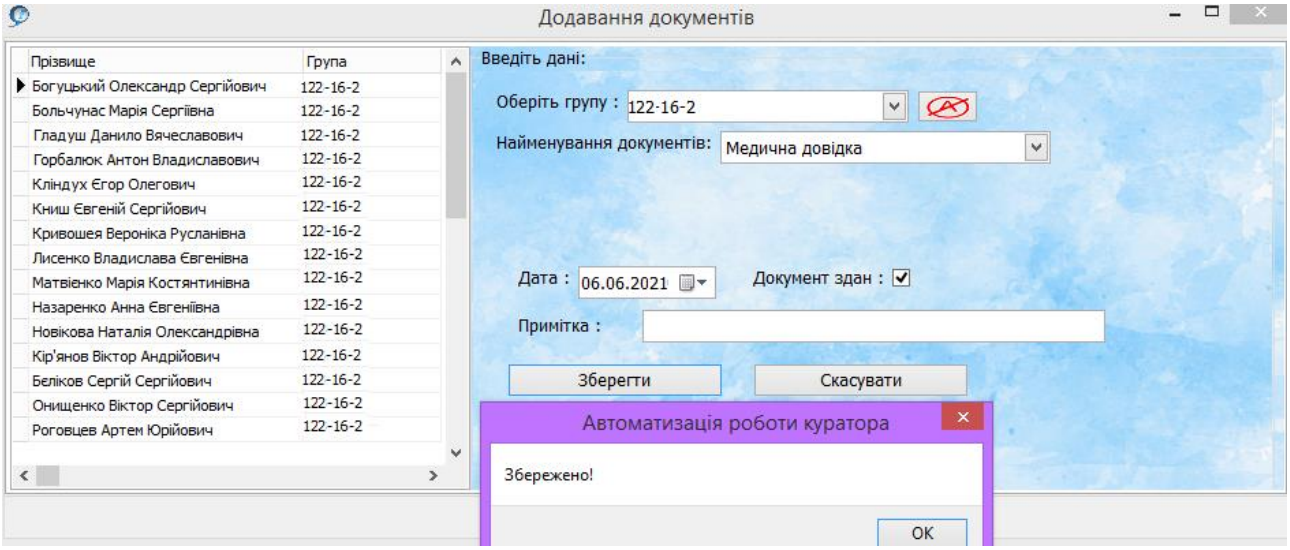


Рис. 2.27. Додавання зібраних документів з використанням фільтрації

Для перегляду зібраних документів необхідно натиснути кнопку «Перегляд зібраних документів». Для зручності користувач має можливість встановлювати фільтрацію для записів. На рис. 2.28 зображено перегляд зібраних документів.

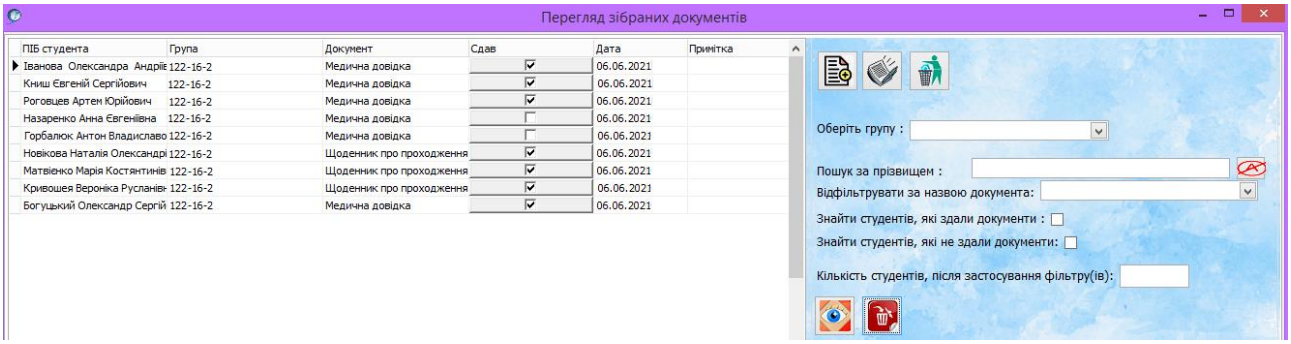


Рис. 2.28. Перегляд інформації про зібрані/не зібрані документи

Для видалення необхідного запису потрібно натиснути на нього та обрати кнопку «Видалення» - «чоловік біля кошику для сміття». Після видалення забезпечено автоматичне оновлення даних в таблиці.

На рис. 2.29 зображено повідомлення про успішне видалення запису.

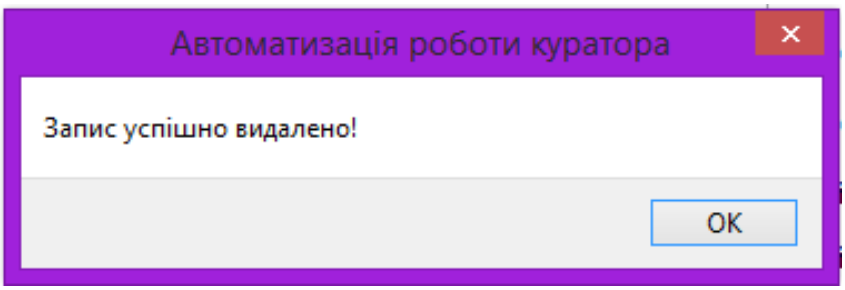


Рис. 2.29. Повідомлення про видалення запису

Для додавання поточних оцінок потрібно натиснути в меню на кнопку «Навчання» та обрати підпункт «Додавання поточних оцінок студента».

Для перегляду оцінок необхідно обрати підпункт «Перегляд оцінок». Користувач отримує можливість відсортувати дані, використати різну фільтрацію та пошук даних, розрахунком середнього балу за семестр використовуючи обрану дисципліну та семестр.

Також користувач має можливість побудувати графік кількість оцінок за певний семестр по кожній дисципліни. Для цього потрібно натиснути «Порівняння успішності навчання» рис. 2.30-2.31.

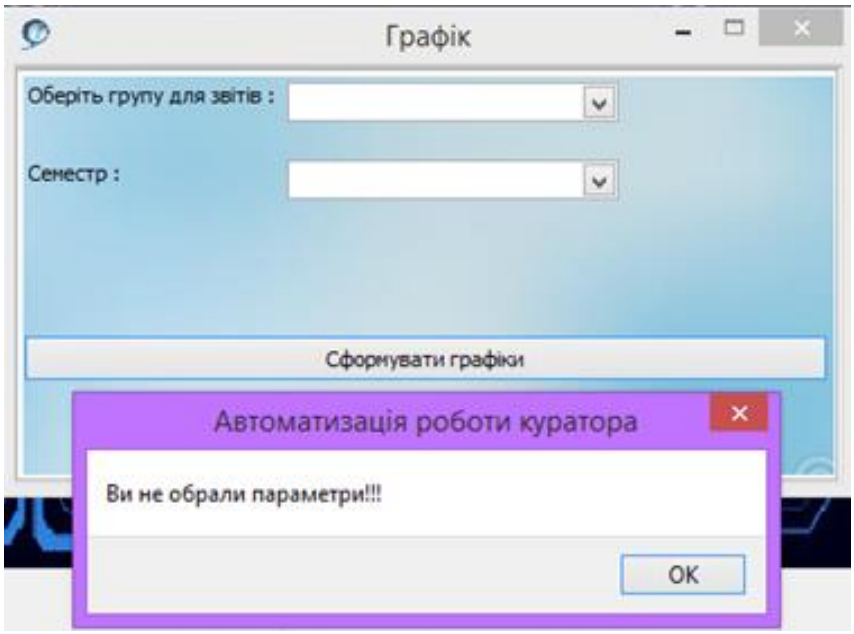


Рис. 2.30. Помилка при створені графіку

Аналіз успішності за побудованою	діаграмою
Дисципліна	Кількість
Безпека життєдіяльності	1
Біологія	11
Всесвітня історія	1
Захист Вітчизни	2
Іноземна мова	4
Інформатика	5
Історія України	4
Культурологія	3
Математика	1
Основи правознавства	2
Фізика	1
Хімія	2

Рис. 2.31. Діаграма для кількості задовільних оцінок

Для перегляду всіх оцінок певної групи необхідно натиснути «Особова картка успішності студента». На рис. 2.32 зображено обрану умову для побудови звіту.

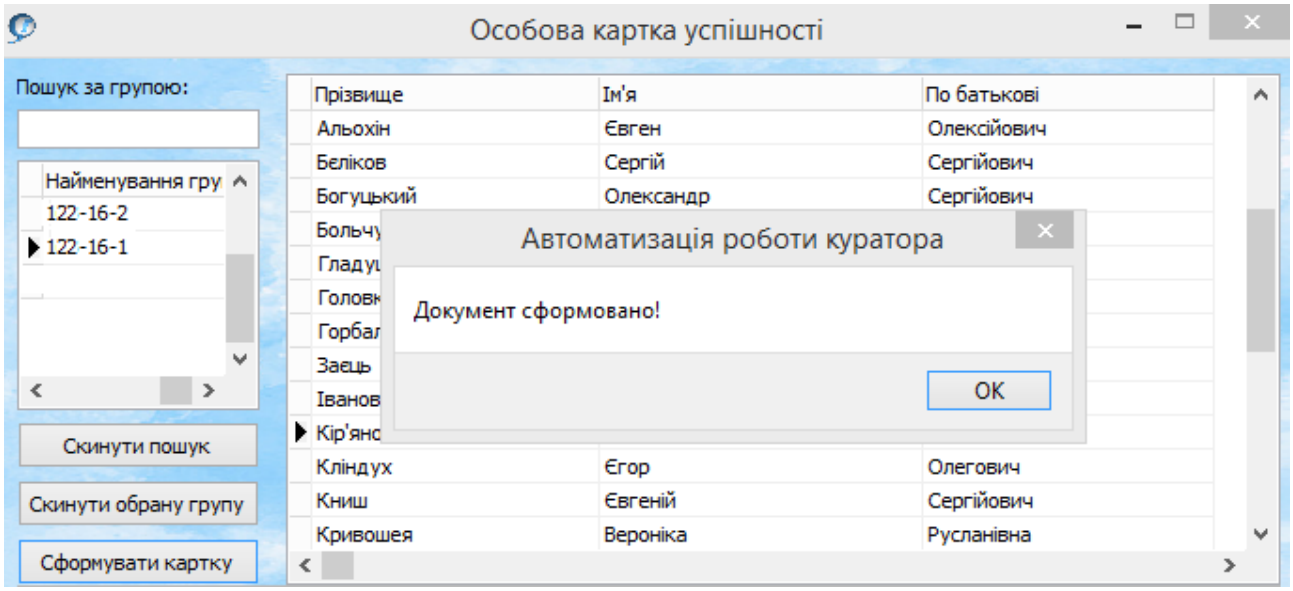
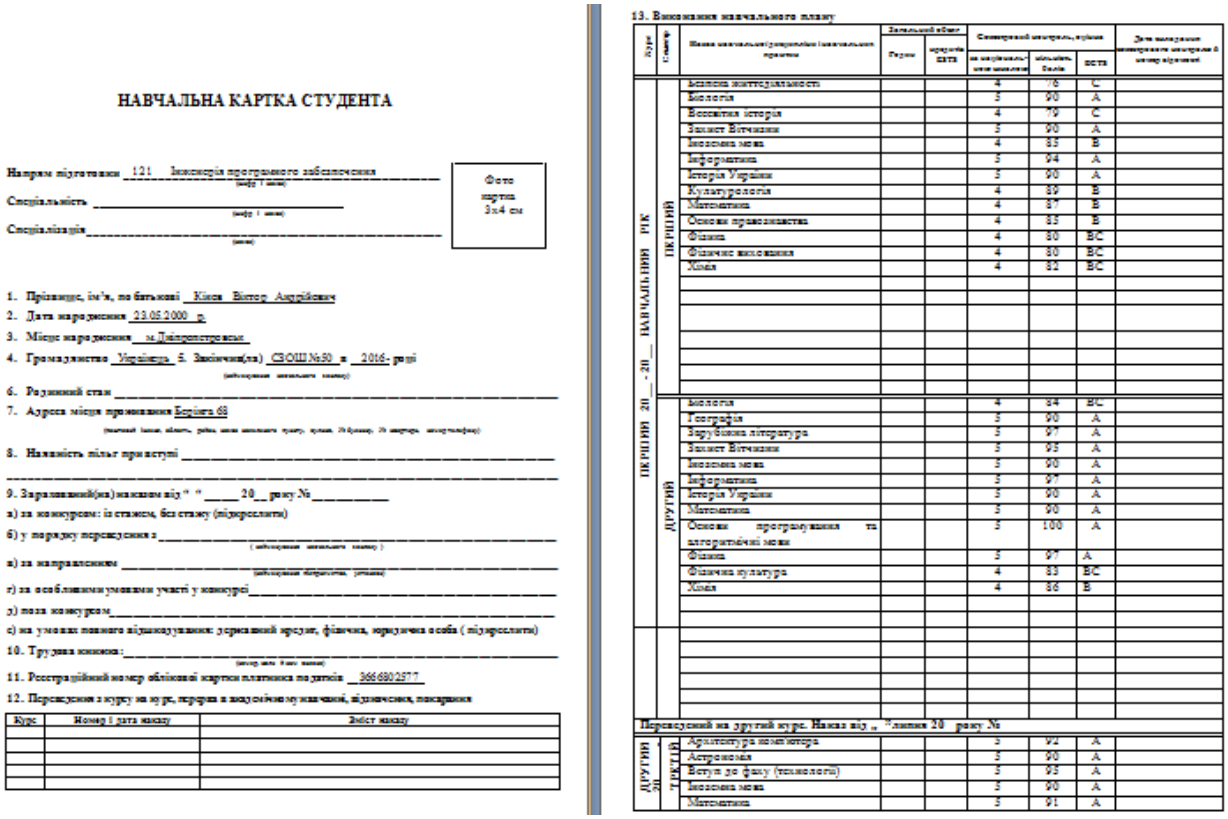


Рис. 2.32. Обрана умова для побудови картки успішності

На рис. 2.33 зображено частину побудованого звіту.



НАВЧАЛЬНА КАРТКА СТУДЕНТА

Напрям підготовки 121 Інженерія програмного забезпечення

Спеціальність

Спеціалізація

Фото картка 3х4 см

1. Прізвище, ім'я, по батькові Кіяк Віктор Андрійович
2. Дата народження 23.05.2000 р.
3. Місце народження м.Дніпропетровськ
4. Громадянство Україна 5. Закінчив(ла) СЗОШ №30 в 2016 році
6. Родинний стан
7. Адреса місця проживання Берлінга 68
8. Наявність пільг при вступі
9. Зарахований(на) наказом від " " 20 року №
а) за конкурсом: із стажем, без стажу (підкреслити)
б) у порядку переведення з
в) за направленням
г) за особливими умовами участі у конкурсі
д) поза конкурсом
е) на умовах повного відшкодування: державний кредит, фізична, юридична особа (підкреслити)
10. Трудова книжка:
11. Реєстраційний номер облікової картки платника податків 3666802577
12. Переведення з курсу на курс, перерви в академічному навчанні, відпустки, покарання

Курс	Номер і дата наказу	Зміст наказу

13. Виконання навчального плану

Курс	Семестр	Назва навчальної дисципліни (навчального проекту)	Загальний обсяг: Години	Загальний обсяг: кредити ЄКТС	Семестровий контроль, підсумковий контроль: вид (екзамен, диференційований залік, залік)	Семестровий контроль, підсумковий контроль: кількість балів	Семестровий контроль, підсумковий контроль: оцінка	Дата складання семестрового контролю та номер відомості
ПЕРШИЙ	ПЕРШИЙ	Безпека життєдіяльності			4	76	C	
		Біологія			5	90	A	
		Всесвітня історія			4	79	C	
		Захист Вітчизни			5	90	A	
		Іноземна мова			4	83	B	
		Інформатика			5	94	A	
		Історія України			5	90	A	
		Культурологія			4	89	B	
		Математика			4	87	B	
		Основи православ'я			4	83	B	
		Фізика			4	80	BC	
		Фізичне виховання			4	80	BC	
		Хімія			4	82	BC	
ПЕРШИЙ	ДРУГИЙ	Біологія			4	84	BC	
		Географія			5	90	A	
		Зарубіжна література			5	97	A	
		Захист Вітчизни			5	93	A	
		Іноземна мова			5	90	A	
		Інформатика			5	97	A	
		Історія України			5	90	A	
		Математика			5	90	A	
		Основи програмування та алгоритмічні мови			5	100	A	
		Фізика			5	97	A	
		Фізична культура			4	83	BC	
		Хімія			4	86	B	
Переведений на другий курс. Наказ від „ "липня 20 року №								
ДРУГИЙ	ТРЕТІЙ	Архітектура комп'ютера			5	92	A	
		Астрономія			5	90	A	
		Вступ до фаху (технології)			5	93	A	
		Іноземна мова			5	90	A	
		Математика			5	91	A	

Рис. 2.33. Картка успішності студента у навчанні

Для визначення якості та успішності у навчанні потрібно обрати підпункт «Якість навчання». Користувач може обрати перегляд або друк. На рис. 2.34 зображено умову для побудови діаграми.

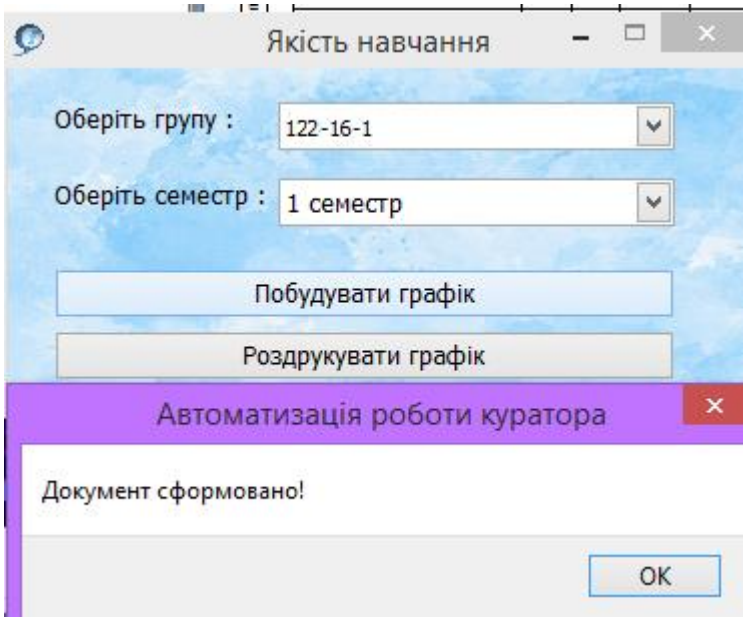


Рис. 2.34. Умова для діаграми

На рис. 2.35 зображено побудовану діаграму якості та успішності.

Рис. 2.35. Співвідношення успішності та якості

Для відрахування студента потрібно перейти в пункт меню «Студент» та обрати підпункт «Робота зі списками студентів груп». На рис. 2.36 зображено відрахування студентів.

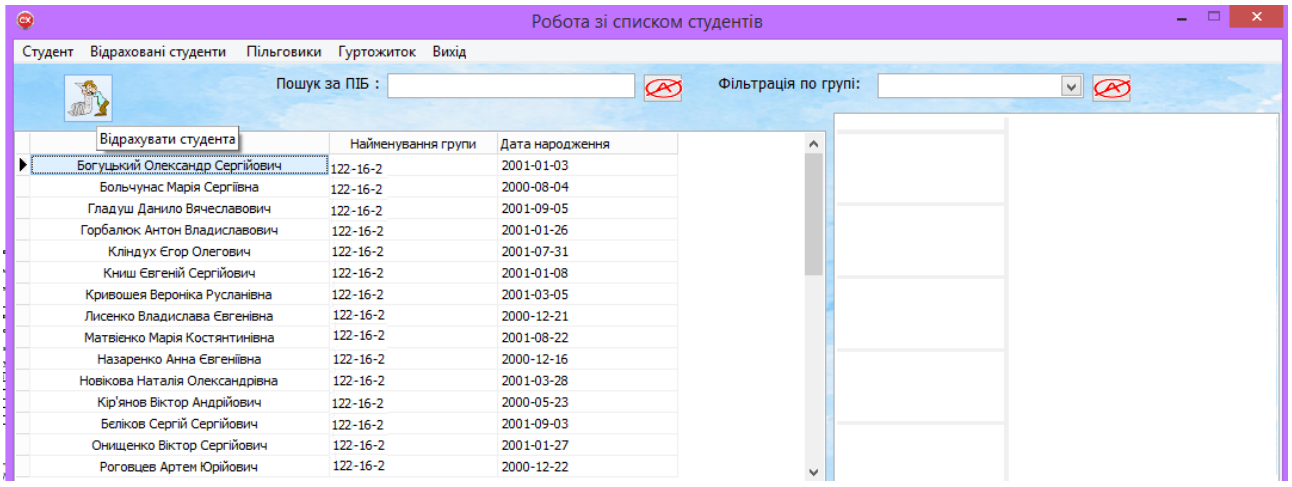


Рис. 2.36. Відрахування студента

Для додавання інформації про студента необхідно натиснути кнопку «Додавання інформації про студента».

Для перегляду даних стосовно студентів необхідно натиснути - «Перегляд

інформації: студентів, батьків». При перегляді інформації стосовно студента передбачено сортування та фільтрація. Також є можливість обрати пункт меню «Додавання студента», «Редагування студента», «Редагування батьків». Видалення студента можливе в пункті меню «Робота зі списками студентів груп».

Програмний додаток передбачає створення різних звітів та шаблонів.

Для створення звітів необхідно натиснути кнопку «Звіти» в меню та обрати необхідний підпункт меню для створення звіту.

На рис. 2.37 зображено всі можливі звіти пункту «Звіти».

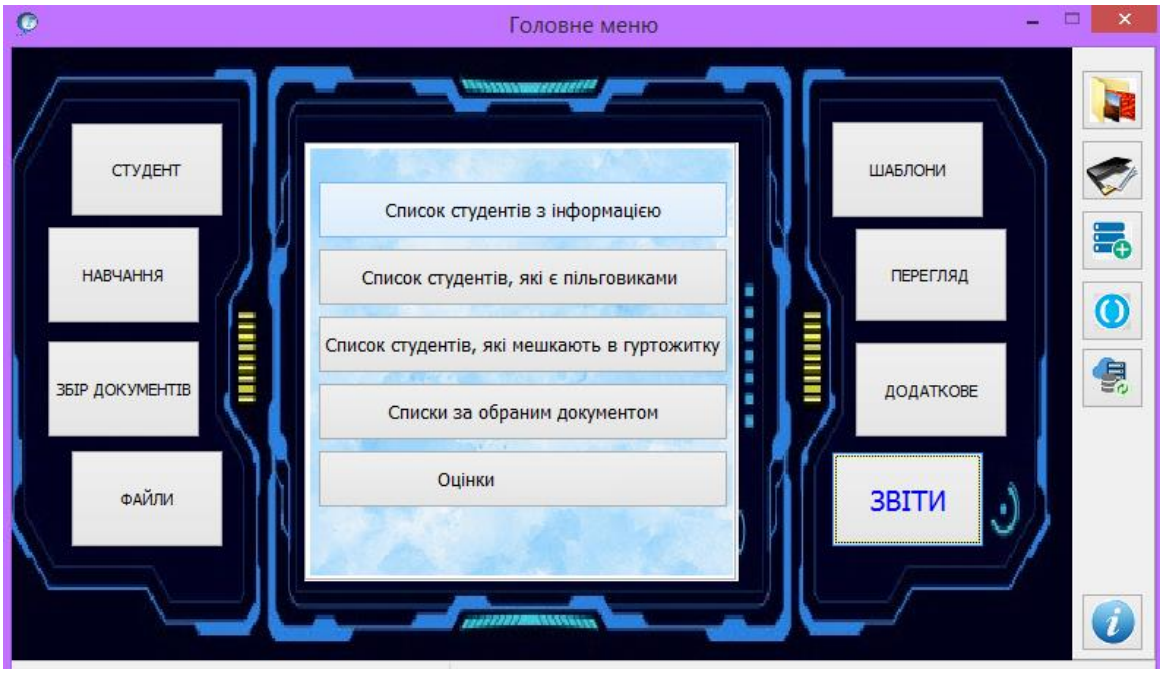


Рис. 2.37. Підпункти «Звіти»

Обравши підпункт «Оцінки» користувач має можливість побудувати таблицю оцінок для обраної групи та обраних дисциплін (враховуючи семестр). Результат використання цього підпункту дає можливість побудувати таблицю для різного призначення: для формування звітності - диплому, або для стипендіальної комісії та подібних заходів. На рис. 2.38 зображено обрану умову для побудови звітності. Для додавання дисциплін необхідно два рази «клацнути» на потрібну дисципліну. Для видалення обраного предмету

необхідно два рази «клацнути» у сформованому списку.

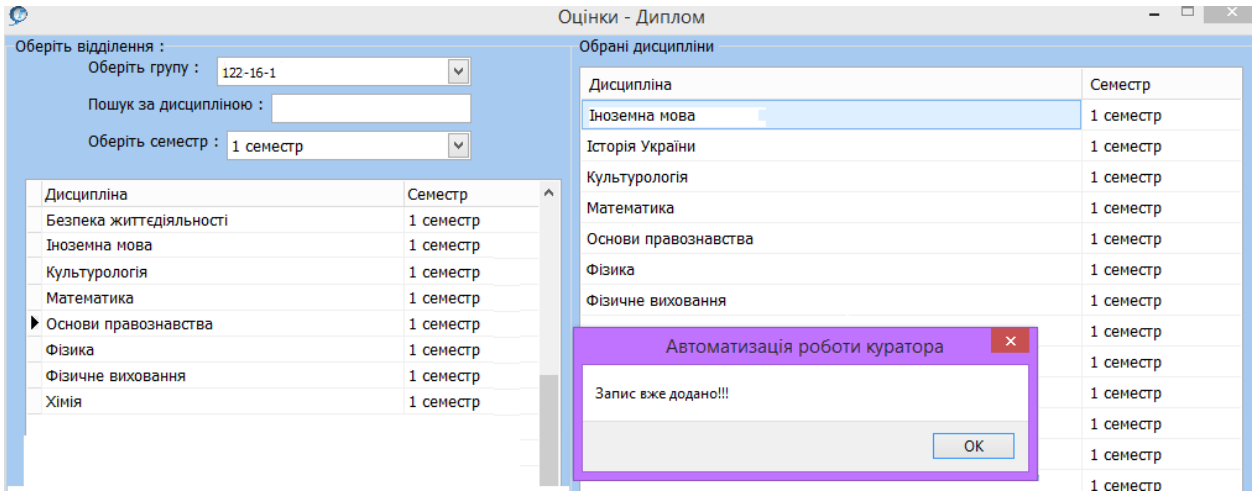


Рис. 2.38. Умова для звіту та повідомлення на зоборону додавання однакової дисципліни (враховується семестр)

На рис. 2.39 зображено побудовану таблицю оцінок для обраних дисциплін.

	ПВ-16-2	Безпека життєдіяльності	Біологія	Всесвітня історія	Захист Вітчизни	Іноземна мова	Інформатика	Історія України	Культурологія	Математика	Основи правознавства	Фізика	Фізичне виховання	Хімія
№	ПІБ студента / Семестр	1	1	1	1	1	1	1	1	1	1	1	1	1
1	Альохін Євген Олексійович	3	4	4	4	4	4	4	4	3	4	3	3	3
2	Беліков Сергій Сергійович	4	5	3	4	4	4	3	4	4	3	4	4	4
3	Богуцький Олександр Сергійович	4	3	3	4	3	3	3	3	3	4	3	4	3
4	Вольчунас Марія Сергіївна	4	5	4	4	5	4	4	4	4	4	4	4	4
5	Гладуш Данило Вячеславович	4	4	4	4	4	4	4	4	4	4	4	4	4
6	Горбалюк Антон Владиславович	4	5	4	5	5	4	5	5	5	5	4	4	5
7	Заець Олег Сергійович	4	5	4	4	4	5	4	4	4	4	4	4	4
8	Іванова Олександра Андріївна	3	4	4	3	4	3	4	4	3	3	4	4	4
9	Кір'янов Віктор Андрійович	4	5	4	5	4	5	5	4	4	4	4	4	4
10	Кліндух Єгор Олегович	4	4	4	4	4	3	4	3	4	4	4	4	4
11	Книш Євгеній Сергійович	3	3	3	3	3	3	3	3	3	3	3	3	3
12	Кривошея Вероніка Русланівна	3	3	3	2	2	3	3	3	3	3	3	3	3
13	Лисенко Владислава Євгенівна	4	5	4	4	5	4	4	4	4	3	4	4	3
14	Матвієнко Марія Костянтинівна	4	5	4	3	4	5	4	4	4	4	4	4	4
15	Назаренко Анна Євгенівна	3	4	4	3	4	4	4	4	4	4	4	4	3
16	Новікова Наталія Олександрівна	4	5	4	4	4	4	5	4	4	4	4	4	4
17	Онищенко Віктор Сергійович	4	5	3	4	4	5	4	4	4	4	4	4	4
18	Роговцев Артем Юрійович	4	3	4	3	4	4	3	4	3	3	3	3	3
19	Романчук Андрій Андрійович	4	4	4	4	4	4	4	4	4	4	4	4	4
20	Рябченко Максим Юрійович	5	5	5	4	5	5	5	5	4	5	5	4	5
21	Скороход Євген Геннадійович	4	4	4	4	4	4	4	4	4	4	4	4	4
22	Солоненко Денис Сергійович	4	5	4	4	4	4	4	4	4	4	4	4	4
23	Юдін Валентин Вячеславович	3	3	3	3	3	3	3	3	3	4	3	4	3
24	Яблонська Ксенія Олександрівна	4	4	4	4	4	4	4	5	4	4	4	4	4

Рис. 2.39. Побудована таблиця

Для використання порожніх звітів потрібно обрати - «Шаблони».

На рис. 2.40 зображено побудований шаблон - «Список - журнал групи для особистих нотаток».

Список — журнал групи для особистих нотаток

Група *122-16-2*

№ за поряд.	Місяць і число / Вид роботи / ФІО студента																								ПРИМІТКА
1	Беліков Сергій Сергійович																								
2	Больчунас Марія Сергіївна																								
3	Гладуш Данило Вячеславович																								
4	Головко Олександр Вікторович																								
5	Горбалюк Антон Владиславович																								
6	Заєць Олег Сергійович																								
7	Кір'янов Віктор Андрійович																								
8	Кліндух Єгор Олегович																								
9	Книш Євгеній Сергійович																								
10	Лисенко Владислава Євгенівна																								
11	Матвієнко Марія Костянтинівна																								
12	Назаренко Анна Євгеніївна																								
13	Новікова Наталія Олександрівна																								
14	Онищенко Віктор Сергійович																								
15	Романчук Андрій Андрійович																								
16	Рябченко Максим Юрійович																								
17	Скороход Євген Геннадійович																								
18	Солоненко Денис Сергійович																								
19	Юдін Валентин Вячеславович																								
20	Яблонська Ксенія Олександрівна																								
21																									
22																									
23																									
24																									
25																									
26																									
27																									
28																									
29																									
30																									

Рис. 2.40. Побудований шаблон - «Список журналу групи для особистих нотаток»

Таким чином, додаток має дружній інтерфейс і передбачає побудову багатьох документів, графіків.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

- передбачуване число операторів – 1000;
- коефіцієнт складності програми – 2;
- коефіцієнт корекції програми в ході її розробки – 0,08;
- годинна заробітна плата програміста, грн / год – 40.

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:

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

де to - витрати праці на підготовку й опис поставленої задачі (приймається 50),

ти - витрати праці на дослідження алгоритму рішення задачі,

та - витрати праці на розробку блок-схеми алгоритму,

тп - витрати праці на програмування по готовій блок-схемі,

тотл - витрати праці на налагодження програми на ЕОМ,

тд - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.:

- умовне число операторів (підпрограм):

$$Q = q \times C \times (1+ p) \text{ , людино-годин,} \qquad (3.2)$$

де q - передбачуване число операторів,

С - коефіцієнт складності програми,

р - коефіцієнт кореляції програми в ході її розробки.

$$Q = 1000 \cdot 2 \cdot (1 + 0{,}08) = 2808 \text{ людино-годин.}$$

Витрати праці на вивчення опису задачі ti визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{QB}{(75...85)K} \text{ , людино-годин,} \qquad (3.3)$$

де В - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі; В=1.2 ... 1.5,

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

Витрати праці на розробку алгоритму рішення задачі:

$$t_u = \frac{2808 \cdot 1{,}2}{80 \cdot 0{,}8} = 52 \text{ , людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t\alpha = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \qquad (3.4)$$

$$t_a = \frac{2808}{20 \cdot 0{,}8} = 175 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \tag{3.5}$$

$$t_n = \frac{2808}{25 \cdot 0{,}8} = 140 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

$$t_{отл} = \frac{Q}{(4...5)K} \quad \text{людино-годин.} \tag{3.6}$$

$$t_{отл} = \frac{2808}{4 \cdot 0{,}8} = 877 \text{ людино-годин.}$$

- за умови комплексного налагодження завдання:

$$t^{к}_{отл} = 1{,}2 \cdot t_{отл}; \tag{3.7}$$

$$t_{отл} = 877 * 1{,}2 = 1053$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}, \text{ людино-годин,} \tag{3.8}$$

де tдр - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{(15...20)K}, \text{ людино-годин.} \tag{3.9}$$

$$t_{др} = \frac{2808}{15 \cdot 0{,}8} = 234 \text{ людино-годин,}$$

tдо – трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{др}, \text{ людино-годин.} \quad (3.10)$$

$$tdo = 0{,}75 \cdot 234 = 175$$

$$t_{д} = 234 + 175 = 409 \text{ людино-годин.}$$

Отримуємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 52 + 175 + 140 + 1053 + 409 = 1861 \text{ людино-годин.}$$

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \quad (3.11)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,} \quad (3.12)$$

де t - загальна трудомісткість, людино-годин,

Спр - середня годинна заробітна плата програміста, грн/година

$$З_{ЗП} = 1861 \cdot 30 = 43755 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \times С_{М}, \text{ грн,} \quad (3.13)$$

де tотл - трудомісткість налагодження програми на ЕОМ, год.

Смч - вартість машино-години ЕОМ, 5 грн/год.

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

$$З_{МВ} = 810 \times 5 = 4050 \text{ грн.}$$

$$Кпо = 43755 + 4050 = 47805 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ міс.} \qquad (3.13)$$

де Bk - число виконавців,

Fp - місячний фонд робочого часу (при 40 годинному робочому тижні Fp=176 годин).

$$T = \frac{2160}{1 \cdot 176} = 8{,}3 \text{ міс.}$$

Висновок: були проведені обчислення трудомісткості та витрат для розробки програмного забезпечення. Для створення даної інформаційної системи знадобиться 8,3 місяця, трудомісткість розробки ПЗ – 1861 людино-годин, а витрати на її створення програмного забезпечення – 47805 грн.

ВИСНОВКИ

Автоматизовану систему створено для полегшення роботи керівників груп. Дана система поєднує функції для складання звітності, отримання інформації про студентів, їх навчальні досягнення, ведення інформації про зібрані папери у студентів та ін. За допомогою розробленої бази даних надано можливість автоматичного заповнення бази даних для подальшого використання.

Розроблений програмний додаток дає змогу користувачеві вирішувати великий обсяг завдань, оптимізуючи значну кількість обов'язків керівника групи, та відмовитися від писемної роботи, надати всю необхідну інформацію в швидкому режимі.

Дана автоматизована система розроблена засобами мови C++ середовища Builder 10.3 Rio з використанням СКБД Microsoft Office Access та офісних додатків MSExcel та MSWord.

Результатом виконання кваліфікаційної роботи є створення кінцевого програмного додатка для ведення роботи куратора, автоматизовано використання бази даних проекту для допомоги та полегшення обробки та аналізу інформації про відповідні академічні групи та студентів.

Розроблена програма має зручний та зрозумілий інтерфейс для роботи, який відповідає виконуваним функціям.

Розроблений додаток забезпечує:

- авторизація бази даних для використання певних функцій;
- швидке створення звітів, витрачення менше часу;
- створення графіків з даних бази.

Розроблена програма має зручний та зрозумілий інтерфейс для роботи, який відповідає виконуваним функціям.

Створення закладом освіти системи управління з використанням комп’ютерної техніки надасть змогу підвищити якість та оперативність вирішення завдань, які виникають у системі управління закладами освіти.

В економічному розділі були проведені обчислення трудомісткості та витрат для розробки програмного забезпечення. Для створення даної інформаційної системи знадобиться 8,3 місяця, трудомісткість розробки ПЗ – 1861 людино-годин, а витрати на її створення програмного забезпечення – 47805 грн.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бьёрн Страуструп. Язык программирования C++ = The C++ Programming Language / Пер. с англ. - 3-е изд. - СПб.; М.: Невский диалект - Бином, 1999. - 991 с. - ISBN 5-7940-0031-7 (Невский диалект), ISBN 5-7989-0127-0 (Бином), ISBN 0-201-88954-4 (англ.).

2. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання : (ГОСТ 7.1-2003, IDT) : ДСТУ ГОСТ 7.1:2006. – Чинний з 2007–07–01. – К. : Держспоживстандарт України, 2007. – 47 с. – (Система стандартів з інформації, бібліотечної та видавничої справи) (Національний стандарт України).

3. Бусигін Б.С., Коротенко Г.М., Коротенко Л.М. Прикладна інформатика. Підручник для студентів комп'ютерних спеціальностей. – Дніпропетровськ: Видавництво НГУ, 2004. – 559 с. URL: http://www.programmer.dp.ua/book-ua-k01.php. дата звернення: 15.03.2019.

4. Бусыгин Б.С., Дивизинюк М.М., Коротенко Г.М., Коротенко Л.М. Введение в современную информатику. Учебник. – Севастополь: Издательство СНУЯЭиП, 2005. – 644 с. / URL: http://www.programmer.dp.ua/book-ru-k02.php. дата звернення: 12.05.2021.

5. Вступ до програмування мовою С++. Організація обчислень : навч. посіб. / Ю. А. Бєлов, Т. О. Карнаух, Ю. В. Коваль, А. Б. Ставровський. – К. : Видавничо-поліграфічний центр "Київський університет", 2012. – 175 с. с.: іл. ISBN (укр.).

6. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.

7. ДСТУ 2394-94 Інформація та документація. Комплектування фонду, бібліографічний опис, аналіз документів. Терміни та визначення. – Чинний від 01.01.1995. - Київ : Держстандарт України, 1994. – 88 с.

8. Козырев А.А. Информационные технологии в экономике и

управлении: учебник. -4-е изд. -СПб: Высш. Проф. Образование. 2005. -444 с.

9. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи ” / Укладачі О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 23с.

10. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 121 «Програмна інженерія» галузі знань 12 Інформаційні технології/, Л.М. Коротенко , О.С. Шевцова; Нац. гірн. ун-т. – Д : ДВНЗ НГУ, 2019. – 65 с.

11. Методичні рекомендації щодо написання, оформлення та представлення учнівських науково-дослідницьких робіт учнів – членів Малої академії наук України / Г.Г. Півняк, Л.М. Коротенко, І.М. Удовик, Є.М. Головня – Д.: ДВНЗ «Національний гірничий університет», 2017. – 24 с.

12. Мирошниченко Г. А. Реляционные базы данных: практические приемы оптимальных решений / Г. А. Мирошниченко.- СПб. : БХВ-Петербург, 2005. - 400 с.Скорочення слів в українській мові у бібліографічному описі. Загальні правила та вимоги : ДСТУ 3582-97. – Чинний від 1998–07–01. – К. : Держстандарт України, 1998. – 24 с. – (Державний стандарт України).

13. Офіційний сайт Microsoft Access URL: https://products.office.com/uk-ua/access. дата звернення: 12.05.2021.

14. Огляд і основи мови програмування C++// URL: http://www.znannya.org/?view=Cpp_basics. дата звернення: 12.05.2021.

15. Робота з OLE-сервером URL: http://wladm.narod.ru/Borland/ olebegin.html. дата звернення: 12.05.2021.

16. Сидорова Н. М. Навчання інженерії програмного забезпечення - систематичний огляд літератури/ Н. М. Сидорова // Матеріали міжнародної науково- практичної конференції аспірантів і студентів «Інженерія програмного забезпечення 2011». URL: http://www.nbuv.gov.ua/portal/ natural/Ipz/2011_2 /Sidorova.pdf. дата звернення: 12.05.2021.

17. СУБД SQL Server основні особливості та її застосування// URL:

https://ua-referat.com/СУБД_SQL_Server. дата звернення: 12.05.2021.

18. Черкасов Ю.М. Информационные технологии управления: учеб. пособие/ под ред. Ю.М.Черкасова. -М : Инфра-М, 2001. -216 с.

19. ISO/IEC 14882:2014 - Information technology - Programming languages - C++.ISO. дата звернення: 12.05.2021.

20. С++. Теорія та практика : Навч. посібник / [О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката та ін.] ; за ред.О. Г.Трофименко. – 587 с.

21. Microsoft SQL Server 2019 //URL: https://softline.ru/about/blog/10-prichin-pereyti-na-microsoft-sql-server-2019. дата звернення: 12.05.2021.

ДОДАТОК А

КОД ПРОГРАМИ

Текст, модуля «MainMENU»

```
#include <vcl.h>
#include <windows.h>
#include <string.h>
#pragma hdrstop
#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
#include "Unit5.h"
#include "Unit6.h"
#include "openingfiles.h"
#include "deducted_1.h"
#include "inputdocument.h"
#include "educationform.h"
#include "smotrstudent.h"
#include "studparen.h"
#include "infor.h"
#include "zvit.h"
#include "personalCard.h"
#include "list.h"
#include "zvit_gurt.h"
#include "student.h"
#include "Access.h"
#include "MyAccess.h"
#include "synchronization.h"
TMainMENU *MainMENU;
MyClass my;
classMy newPoin;
bool accessDekanat;
int idKurator;
String typeAccess;
void __fastcall TMainMENU::FormCreate(TObject *Sender)
{
 my.bookmark_blocking(PageControl1);
 my.call_bookmark(PageControl1, TabSheet1);
 my.color_button(student, study, collect_doc, REPORTS, educt_work,PRINT,
 DORMITRY, supplementary);
 MainMENU->BitBtn5->Caption = "Додавання інформації про студента";
 MainMENU->BitBtn6->Caption = "Перегляд інформації: студентів, батьків";
}
__fastcall TMainMENU::TMainMENU(TComponent* Owner, bool accessDek, int idKur, String typeAcc):
TForm(Owner)
{accessDekanat = accessDek;idKurator = idKur;typeAccess = typeAcc;}
void __fastcall TMainMENU::studentClick(TObject *Sender)
{
 my.call_bookmark(PageControl1, TabSheet1);
 my.color_button(student, study, collect_doc, REPORTS, educt_work,PRINT,
 DORMITRY, supplementary);
}
void __fastcall TMainMENU::studyClick(TObject *Sender)
{
 my.call_bookmark(PageControl1, TabSheet2);
 my.color_button(study, student, collect_doc, REPORTS, educt_work,PRINT,
 DORMITRY, supplementary);
}
```

```
void __fastcall TMainMENU::collect_docClick(TObject *Sender)
{
my.call_bookmark(PageControl1, TabSheet3);
my.color_button(collect_doc, study, student,  REPORTS, educt_work,PRINT,
DORMITRY, supplementary);
}
void __fastcall TMainMENU::REPORTSClick(TObject *Sender)
{
my.call_bookmark(PageControl1, TabSheet4);
my.color_button(REPORTS, student, study, collect_doc, educt_work,PRINT,
DORMITRY, supplementary);
}
void __fastcall TMainMENU::educt_workClick(TObject *Sender)
{
my.call_bookmark(PageControl1, TabSheet5);
my.color_button(educt_work, student, study, collect_doc, REPORTS, PRINT,
DORMITRY, supplementary);
}
void __fastcall TMainMENU::PRINTClick(TObject *Sender)
{
my.call_bookmark(PageControl1, TabSheet6);
my.color_button(PRINT,student, study, collect_doc, REPORTS, educt_work,
DORMITRY, supplementary);
}
void __fastcall TMainMENU::DORMITRYClick(TObject *Sender)
{
my.call_bookmark(PageControl1, TabSheet7);
my.color_button(DORMITRY,student, study, collect_doc, REPORTS, educt_work,PRINT,
supplementary);
}
void __fastcall TMainMENU::supplementaryClick(TObject *Sender)
{
my.call_bookmark(PageControl1, TabSheet9);
my.color_button(supplementary, student, study, collect_doc, REPORTS, educt_work,PRINT,DORMITRY);
}
void __fastcall TMainMENU::Timer1Timer(TObject *Sender)
{
TDateTime NowTime = Now();
StatusBar1->Panels->Items[0]->Text=NowTime;
}
void __fastcall TMainMENU::BitBtn5Click(TObject *Sender)
{
stud_paren->Caption="Додавання інформації про студентів та батьків!";
stud_paren->ShowModal();
}
void __fastcall TMainMENU::BitBtn17Click(TObject *Sender)
{my.name_form(1,1);}
void __fastcall TMainMENU::Button5Click(TObject *Sender)
{my.name_form(1,2);}
void __fastcall TMainMENU::Button6Click(TObject *Sender)
{my.name_form(1,3);}
void __fastcall TMainMENU::Button7Click(TObject *Sender)
{my.name_form(1,4);}
void __fastcall TMainMENU::BitBtn18Click(TObject *Sender)
{
 input_1->Caption="Завантаження матеріалів";
 input_1->ShowModal();
}
void __fastcall TMainMENU::Button2Click(TObject *Sender)
{
 extra->Label1->Caption="Пошук за назвою пільги:";
 my.vivod(3);
```

```
 extra->ShowModal();
}
void __fastcall TMainMENU::Button3Click(TObject *Sender)
{
 extra-> Label1->Caption="Пошук за назвою дисципліни:";
 my.vivod(2);
 extra->ShowModal();
}
void __fastcall TMainMENU::Button4Click(TObject *Sender)
{
 extra->Label1->Caption="Пошук за назвою документа:";
 my.vivod(1);
 extra->ShowModal();
}
void __fastcall TMainMENU::BitBtn19Click(TObject *Sender)
{
my.vivod_file();
opening_files->ShowModal();
}
void __fastcall TMainMENU::BitBtn9Click(TObject *Sender)
{
deducted->Caption="Робота зі списком студентів";
my.vivod_spicok(1,0);
deducted->ShowModal();
}
void __fastcall TMainMENU::BitBtn8Click(TObject *Sender)
{
deducted->Caption="Список студентів - гуртожиток";
deducted->BitBtn6->Visible=false;
deducted->BitBtn8->Visible=false;
deducted->BitBtn9->Visible=false;
my.vivod_spicok(2,0);
deducted->ShowModal();
}
void __fastcall TMainMENU::BitBtn7Click(TObject *Sender)
{deducted->Caption="Список студентів - пільговики";
deducted->BitBtn6->Visible=true;
deducted->BitBtn8->Visible=false;
deducted->BitBtn9->Visible=false;
my.vivod_spicok(3,0);
deducted->ShowModal();}
void __fastcall TMainMENU::BitBtn3Click(TObject *Sender)
{
deducted->Caption="Список відрахованих студентів";
deducted->BitBtn6->Visible=false;
deducted->BitBtn8->Visible=true;
deducted->BitBtn9->Visible=true;
my.vivod_spicok(4,0);
deducted->ShowModal();}
void __fastcall TMainMENU::BitBtn16Click(TObject *Sender)
{input_document->Caption="Перегляд зібраних документів";
input_document->Height=558;
input_document->Width=1329;
input_document->Position=poDesktopCenter;
input_document->ShowModal();}
void __fastcall TMainMENU::BitBtn15Click(TObject *Sender)
{input_document->Caption="Додавання документів";
input_document->Height=395;
input_document->Width=947;
input_document->Position=poDesktopCenter;
input_document->ShowModal();}
void __fastcall TMainMENU::BitBtn12Click(TObject *Sender)
```

```
{education_form->Caption="Перегляд оцінок";
education_form->Height=515;
education_form->Width=1025;
education_form->Position=poDesktopCenter;
education_form->ShowModal();}
void __fastcall TMainMENU::BitBtn11Click(TObject *Sender)
{
education_form->Caption="Додавання оцінок";
education_form->Height=323;
education_form->Width=763;
education_form->Position=poDesktopCenter;
education_form->ShowModal();
}
void __fastcall TMainMENU::BitBtn6Click(TObject *Sender)
{smotr_student->ShowModal();}
void __fastcall TMainMENU::BitBtn1Click(TObject *Sender)
{
String path = ExtractFilePath (Application->ExeName) + "\\PHOTO";
 ShellExecute(
            NULL,
            _TEXT("open"),
            path.c_str(),
            NULL,
            NULL,
            SW_SHOWNORMAL);
}
void __fastcall TMainMENU::Button19Click(TObject *Sender)
{
zvit_f->Caption="Список студентів з розміткою для підпису";
zvit_f->ShowModal();
}
void __fastcall TMainMENU::Button18Click(TObject *Sender)
{
 zvit_f->Caption="Список студентів для внесення інформації";
 zvit_f->ShowModal();
}
void __fastcall TMainMENU::Button16Click(TObject *Sender)
{
 zvit_f->Caption="Список студентів з порожніми стовпцями";
 zvit_f->ShowModal();
}
void __fastcall TMainMENU::Button17Click(TObject *Sender)
{
 zvit_f->Caption="Поточні оцінки";
 zvit_f->ShowModal();
}
void __fastcall TMainMENU::Button11Click(TObject *Sender)
{
   zvit_f->Caption="Студенти які мешкають в гуртожитку";
   zvit_f->ShowModal();
}
void __fastcall TMainMENU::Button10Click(TObject *Sender)
{
 zvit_f->Caption="Студенти пільговики";
 zvit_f->ShowModal();
}
void __fastcall TMainMENU::Button12Click(TObject *Sender)
{
 zvit_f->Caption="Звіт за обраним документом";
 zvit_f->ShowModal();
}
void __fastcall TMainMENU::Button14Click(TObject *Sender)
```

```
{
 zvit_f->Caption="Запрошення на збори!";
 zvit_f->ShowModal();
}
void __fastcall TMainMENU::Button9Click(TObject *Sender)
{zvit_f->Caption="Список студентів з інформацією!";
 zvit_f->ShowModal();
}
void __fastcall TMainMENU::BitBtn14Click(TObject *Sender)
{
  if(DM->ADOConnection2->Connected)
   {
            if( accessDekanat == true)
             {
              card->Caption="Особова картка успішності студента";
      if (typeAccess == "Вид авторизації куратор.")
              {
               newPoin.sqlGroup(false, NULL, 0, true, idKurator);
               card = new Tcard(this, idKurator, typeAccess);
               card->ShowModal();
              }
              else
                        {
                         newPoin.sqlGroup(false, NULL, 0, false, 0);
        card = new Tcard(this, -1, typeAccess);
                         card->ShowModal();
                        }
             }
              else ShowMessage("Авторизуйтесь для використання бази даних Деканат!!!");
   }
            else ShowMessage("Помилка при підключені до бази даних Деканат!!!");
}
void __fastcall TMainMENU::BitBtn13Click(TObject *Sender)
{
  if(DM->ADOConnection2->Connected)
             {
              if( accessDekanat == true)
                        {
                                  zvit_f->Caption="Графік";
                                   if (typeAccess == "Вид авторизації куратор.")
                                             {
                                                       newPoin.sqlGroup(false, NULL, 0, true, idKurator);
                                                       zvit_f->ShowModal();
                                             }
                                               else
                                                         {
                                                          newPoin.sqlGroup(false, NULL, 0, false, 0);
                                                          zvit_f->ShowModal();
                                                         }
} else ShowMessage("Авторизуйтесь для використання бази даних Деканат!!!");
  } else ShowMessage("Помилка при підключені до бази даних Деканат!!!");
}
void __fastcall TMainMENU::BitBtn2Click(TObject *Sender)
{Form6->ShowModal();}
void __fastcall TMainMENU::BitBtn4Click(TObject *Sender)
{
   String path = ExtractFilePath (Application->ExeName) + "\\FILES";
 ShellExecute(
            NULL,
            _TEXT("open"),
            path.c_str(),
            NULL,
```

```
        NULL,
        SW_SHOWNORMAL);
}
void __fastcall TMainMENU::BitBtn10Click(TObject *Sender)
{String path = ExtractFilePath (Application->ExeName) + "\\Автоматизація роботи куратора.exe";
newPoin.CreateShortCut(path);}
void __fastcall TMainMENU::Button1Click(TObject *Sender)
{
 if( DM->ADOConnection2->Connected)
 {
        if(accessDekanat == true)
         {
          if (typeAccess == "Вид авторизації куратор.")
          {
           newPoin.sqlGroup(false, NULL, 0, true, idKurator);
           listdb = new Tlistdb(this, idKurator, typeAccess);
           listdb->ShowModal();
          }
          else
           {
                newPoin.sqlGroup(false, NULL, 0, false, 0);
      listdb = new Tlistdb(this, -1, typeAccess);
                listdb->ShowModal();
           }
         }
         else ShowMessage("Авторизуйтесь для використання бази даних Деканат!!!");
 } else ShowMessage("Помилка при підключені до бази даних Деканат!!!");
}
void __fastcall TMainMENU::Button8Click(TObject *Sender)
{
 if( DM->ADOConnection2->Connected)
 {
        if(accessDekanat == true)
         {
          if (typeAccess == "Вид авторизації куратор.")
          {
           zvit_gru= new Tzvit_gru(this, idKurator, typeAccess);
           zvit_gru->ShowModal();
          }
          else
           {
           zvit_gru = new Tzvit_gru(this, -1, typeAccess);
                zvit_gru->ShowModal();
           }
         }
         else ShowMessage("Авторизуйтесь для використання бази даних Деканат!!!");
 } else ShowMessage("Помилка при підключені до бази даних Деканат!!!");
}
void __fastcall TMainMENU::BitBtn20Click(TObject *Sender)
{
 if( DM->ADOConnection2->Connected)
 {
        if(accessDekanat == true)
         {
                if (typeAccess == "Вид авторизації куратор.")
          {
           Form7 = new TForm7(this, idKurator, typeAccess);
           Form7->ShowModal();
          }
                else
                 {
                  Form7 = new TForm7(this, -1, typeAccess);
```

```
                    Form7->ShowModal();
                   }
            }
            else ShowMessage("Авторизуйтесь для використання бази даних Деканат!!!");
  } else ShowMessage("Помилка при підключені до бази даних Деканат!!!");
}
void __fastcall TMainMENU::BitBtn21Click(TObject *Sender)
{
  if(DM->ADOConnection2->Connected)
            {
              Form8->ShowModal();
            }
 else ShowMessage("Помилка при підключені до бази даних Деканат!!!");}
void __fastcall TMainMENU::BitBtn22Click(TObject *Sender)
{if( DM->ADOConnection2->Connected)
  {
           if(accessDekanat == true)
            {
                    if (typeAccess == "Вид авторизації куратор.")
             {
              timing = new Ttiming(this, idKurator, typeAccess);
              timing->ShowModal();
             }
                    else
                     {
                      timing = new Ttiming(this, -1, typeAccess);
                      timing->ShowModal();
                     }
            }
            else ShowMessage("Авторизуйтесь для використання бази даних Деканат!!!");
  } else ShowMessage("Помилка при підключені до бази даних Деканат!!!");
}
```

Текст, файлу — h класу «MyClass»

```
#include <vcl.h>
#ifndef Unit2H
#define Unit2H
#include "Unit1.h"
#include "Unit3.h"
#include "Unit4.h"
#include "deducted_1.h"
#include "inputdocument.h"
#include "educationform.h"
#include <Vcl.Controls.hpp>
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>
#include <Vcl.ExtCtrls.hpp>
#include <Vcl.Imaging.jpeg.hpp>
#include <Vcl.Buttons.hpp>
#include <Vcl.ComCtrls.hpp>
class MyClass
{
 public:
                    int value1;
                    int value2;
                    int value3;
                    int value4;
                    int value5;
                    static int poisk1;
                    static int s_value;
                    static String s_sort;
```

```
                static int s_id_now;
                static int id_pap;
                static int id_mam;
                static int next_1;
                MyClass()
                {
                        value1=0;
                        value2=0;
                        value3=0;
                        value4=0;
                        value5=0;
                }
  void bookmark_blocking(TPageControl *page_con);
  void call_bookmark(TPageControl *page_con, TTabSheet *name_sheet);
  void color_button(TBitBtn *color1, TBitBtn *color2,
  TBitBtn *color3, TBitBtn *color4, TBitBtn *color5,
  TBitBtn *color6, TBitBtn *color7, TBitBtn *color8);
  void name_form(int n, int m);
  void proverka(int k, int k1);
  void proverka_Key(TEdit *Edit1, System::WideChar &Key);
  void proverka_priobraz(TEdit *Edit1);
  void  vivod(int nom);
  void  poisk(int poi);
  void proverka_delet(int k, int k1, int volue1);
  int zap_delet(int g);
  void proverka_material(int a, int b);
  void vivod_file();
  void vivod_material(int a);
  void vivod_spicok(int nom, int sort);
  void del_adv(int a);
  void del_student(int a, int ozn);
  void vibor_stud(int n, TEdit *Edit2, TDBLookupComboBox *DBLookupComboBox3);
  void zv_t();
  void zvit_host();
  void zvit_advan();
  void zvit_doc();
};
  int MyClass::s_value=1;
  int MyClass::poisk1=0;
  String MyClass::s_sort="";
  int MyClass::s_id_now=-1;
  int MyClass::id_pap=-1;
  int MyClass::id_mam=-1;
  int MyClass::next_1=1;
#endif
```

Текст, класу «MyClass»

```
#include <vcl.h>
#pragma hdrstop
#include "Unit2.h"
#pragma package(smart_init)
void MyClass::bookmark_blocking(TPageControl *page_con) //блокировка закладок
{for(int i(0); i<page_con->PageCount; i++)
page_con->Pages[i]->TabVisible=false;}
void MyClass::call_bookmark(TPageControl *page_con, TTabSheet *name_sheet)
{page_con->ActivePage=name_sheet;}
void MyClass::color_button(TBitBtn *color1, TBitBtn *color2,
TBitBtn *color3, TBitBtn *color4, TBitBtn *color5, TBitBtn *color6,
TBitBtn *color7, TBitBtn *color8)
{
  color1->Font->Size = 14;
  color1->Font->Color = clBlue;
```

```
 color2->Font->Size = 8;
 color2->Font->Color = clWindowText;
 color3->Font->Size = 8;
 color3->Font->Color = clWindowText;
 color4->Font->Size = 8;
 color4->Font->Color = clWindowText;
 color5->Font->Size = 8;
 color5->Font->Color = clWindowText;
 color6->Font->Size = 8;
 color6->Font->Color = clWindowText;
 color7->Font->Size = 8;
 color7->Font->Color = clWindowText;
 color8->Font->Size = 8;
 color8->Font->Color = clWindowText;
}
void MyClass::name_form(int n, int m)
{switch (n)
          {
           case 1: {input_1->Caption="Додавання"; break;}
           case 2: {input_1->Caption="Редагування";break;}
          }
          switch (m)
          {
           case 1:
                              {
                              input_1->Label1->Caption="Введіть тему :";
                              input_1->LabeledEdit1->Enabled=true;
                              break;
                              }
           case 2:
                              {
                               input_1->Label1->Caption="Введіть назву пільги:";
                               input_1->LabeledEdit1->Enabled=false;
                               break;
                              }
           case 3:
                              {
                              input_1->Label1->Caption="Введіть назву дисципліни:";
                              input_1->LabeledEdit1->Enabled=false;
                              break;
                              }
           case 4:
                              {
                               input_1->Label1->Caption="Введіть вид документа:";
                               input_1->LabeledEdit1->Enabled=false;
                               break;
                              }
          }
  input_1->ShowModal();
}
void MyClass::proverka(int k, int k1)
 {
 DM->AQ_List->Close();
 DM->AQ_List->SQL->Clear();
  switch (k)
          {
           case 1:
{DM->AQ_List->SQL->Add("SELECT id_topic, topic FROM List_of_topics where topic='"+input_1->Edit1-
>Text+"'");
                              break;
                              }
           case 2:
```

```
{DM->AQ_List->SQL->Add("SELECT id_privilege FROM Privilege where name_privilege='"+input_1->Edit1->Text+"'");
                                break;
                                }
 case 3:{
DM->AQ_List->SQL->Add("SELECT id_discip FROM Subject where name_discip='"+input_1->Edit1->Text+"');
break;}
case 4:
{
DM->AQ_List->SQL->Add("SELECT id_doc FROM Document_type where name_doc='"+input_1->Edit1->Text+"');
        break;}
        }
  DM->AQ_List->Open();
  if(k==1&&k1==5)
  {
        String p_nowb = DM->AQ_List->FieldByName("topic")->AsString;
        if(input_1->Edit1->Text==p_nowb && DM->AQ_List->RecordCount==1)
        {goto metka;}
  }
  if(DM->AQ_List->RecordCount)
  {ShowMessage("Такий запис існує! Будь ласка, ведіть по-іншому!");}
  else
  {
   metka:
        DM->AQ_List->Close();
        DM->AQ_List->SQL->Clear();
        switch (k1)
        {
         case 1:
                                {
DM->AQ_List->SQL->Text="INSERT INTO List_of_topics (topic, Note_4) VALUES('"+input_1->Edit1->Text+"', '"+input_1->LabeledEdit1->Text+"')";
                                break; }
         case 2:
                                {
                                DM->AQ_List->SQL->Text="INSERT INTO Privilege (name_privilege) VALUES('"+input_1->Edit1->Text+"')";
                                break; }
         case 3:
                                {
                                DM->AQ_List->SQL->Text="INSERT INTO Subject (name_discip) VALUES('"+input_1->Edit1->Text+"')";
                                break;
         case 4:
                                {
        DM->AQ_List->SQL->Text="INSERT INTO Document_type (name_doc) VALUES('"+input_1->Edit1->Text+"')";
                                break; }
         case 5:
                                {DM->AQ_List->SQL->Text="UPDATE List_of_topics SET topic=:n_topic, Note_4=:Nots where id_topic=:p";
                                DM->AQ_List->Parameters->ParamByName("n_topic")->Value=input_1->Edit1->Text;
                                DM->AQ_List->Parameters->ParamByName("Nots")->Value=input_1->LabeledEdit1->Text;
                                DM->AQ_List->Parameters->ParamByName("p")->Value=s_value;
                                break; }
         case 6:
                                {
                                DM->AQ_List->SQL->Text="UPDATE Privilege SET name_privilege=:name_priv where id_privilege=:p";
                                DM->AQ_List->Parameters->ParamByName("name_priv")->Value=input_1->Edit1->Text;
```

```
                                DM->AQ_List->Parameters->ParamByName("p")->Value=s_value;
                                break;
                                }
            case 7:
{DM->AQ_List->SQL->Text="UPDATE Subject SET name_discip=:name_dis where id_discip=:p";
DM->AQ_List->Parameters->ParamByName("name_dis")->Value=input_1->Edit1->Text;
                                DM->AQ_List->Parameters->ParamByName("p")->Value=s_value;
                                break;}
            case 8:
                        {DM->AQ_List->SQL->Text="UPDATE Document_type SET name_doc=:name_d where
id_doc=:p";
DM->AQ_List->Parameters->ParamByName("name_d")->Value=input_1->Edit1->Text;
                                DM->AQ_List->Parameters->ParamByName("p")->Value=s_value;
                                break;}
            }
            DM->AQ_List->ExecSQL();
            ShowMessage("Збережено!");
   input_1->Edit1->Text="";
            input_1->LabeledEdit1->Text="";
 }}
 void MyClass::proverka_Key(TEdit *Edit1, System::WideChar &Key)
 {
              if(iswalpha(Key)) return; //літери
  if(Key==8) return; //Backspace
            if(Key==32) // SPACE
            {
             if( Edit1->SelStart==0)
             Key=0;
             return;
            }
              if(Key=='-')
              {
              if( Edit1->SelStart==0)
              Key=0;
              return;
              }
            if(input_1->Label1->Caption=="Введіть вид документа:")
  {
            if(isdigit(Key)) return;
            }
            if(extra->Label1->Caption=="Пошук за назвою документа:")
  {
            if(isdigit(Key)) return;
            }
            Key=0;
 }
void  MyClass::proverka_priobraz(TEdit *Edit1)
{
AnsiString fam = Edit1->Text.SubString(1,1).UpperCase() + Edit1->Text.SubString(2,Edit1-
>Text.Length()).LowerCase();
Edit1->Text = fam;
}
void MyClass::vivod(int nom)
 {
 DM->AQ_dod->Close();
 DM->AQ_dod->SQL->Clear();
 switch(nom)
 {
             case 1 :
             {
               DM->AQ_dod->SQL->Add("SELECT * FROM Document_type");
      extra->DBGrid1->Columns->Items[1]->Visible=false;
```

```
            extra->DBGrid1->Columns->Items[3]->Visible=true;
            extra->DBGrid1->Columns->Items[5]->Visible=false;
            break;
           }
           case 2 :
           {
            DM->AQ_dod->SQL->Add("SELECT * FROM Subject");
            extra->DBGrid1->Columns->Items[1]->Visible=false;
            extra->DBGrid1->Columns->Items[3]->Visible=false;
            extra->DBGrid1->Columns->Items[5]->Visible=true;
           break;
           }
           case 3 :
           {
            DM->AQ_dod->SQL->Add("SELECT * FROM Privilege");
            extra->DBGrid1->Columns->Items[1]->Visible=true;
            extra->DBGrid1->Columns->Items[3]->Visible=false;
            extra->DBGrid1->Columns->Items[5]->Visible=false;
            break;
           }
}
 DM->AQ_dod->Open();
 }
void  MyClass::poisk(int poi)
 {DM->AQ_dod->Close();
  DM->AQ_dod->SQL->Clear();
  switch(poi)
 {
           case 1 :
           {
           DM->AQ_dod->SQL->Add("SELECT * FROM Document_type ");
           DM->AQ_dod->SQL->Add("where name_doc LIKE" " '%"+extra->Edit1->Text+"%' ");
           break;
           }
           case 2 :
           {
           DM->AQ_dod->SQL->Add("SELECT * FROM Subject ");
DM->AQ_dod->SQL->Add("where name_discip  LIKE" " '%"+extra->Edit1->Text+"%' ");
           break;
           }
           case 3 :
           {
           DM->AQ_dod->SQL->Add("SELECT * FROM Privilege ");
DM->AQ_dod->SQL->Add("where name_privilege LIKE" " '%"+extra->Edit1->Text+"%' ");
           break;
           }
}
 DM->AQ_dod->Open();
 }
void MyClass::proverka_delet(int k, int k1, int volue1)
  {
  DM->AQ_dod->Close();
  DM->AQ_dod->SQL->Clear();
          switch (k)
          {
           case 1:
                   {
DM->AQ_dod->SQL->Add("SELECT id_g_doc FROM Collection_of_doc where id_doc=:p");
                           DM->AQ_dod->Parameters->ParamByName("p")->Value=volue1;
                           break;
                           }
           case 2:
```

```
                        {
DM->AQ_dod->SQL->Add("SELECT id_teaching FROM education where id_discip=:p");
                        DM->AQ_dod->Parameters->ParamByName("p")->Value=volue1;
                        break;
                        }
          case 3:
                        {
DM->AQ_dod->SQL->Add("SELECT id_advantage FROM advantage where id_privilege=:p");
                        DM->AQ_dod->Parameters->ParamByName("p")->Value=volue1;
                        break;
                        }
          }
  DM->AQ_dod->Open();
  if(DM->AQ_dod->RecordCount)
  {
  ShowMessage("Видалення не є можливим, бо записи використовуються!");
  }
  else
  {
          DM->AQ_dod->Close();
          DM->AQ_dod->SQL->Clear();
          switch (k1)
          {
           case 1:
                        {
                DM->AQ_dod->SQL->Text="DELETE FROM Document_type where id_doc=:p";
                        DM->AQ_dod->Parameters->ParamByName("p")->Value=volue1;
                        break;
                        }
           case 2:
                        {
                        DM->AQ_dod->SQL->Text="DELETE FROM Subject where id_discip=:p";
                        DM->AQ_dod->Parameters->ParamByName("p")->Value=volue1;
                        break;
                        }
           case 3:
                        {
                DM->AQ_dod->SQL->Text="DELETE FROM Privilege where id_privilege=:p";
                        DM->AQ_dod->Parameters->ParamByName("p")->Value=volue1;
                        break;
                        }
          }
          DM->AQ_dod->ExecSQL();
          ShowMessage("Запис успішно видалено!");
  }
 }
int MyClass::zap_delet(int g)
{
  switch(g)
  {
          case 1:
          {
           value1=extra->DBGrid1->DataSource->DataSet->FieldByName("id_doc")->Value;
           return value1;
           break;
          }
          case 2:
          {
value2=extra->DBGrid1->DataSource->DataSet->FieldByName("id_discip")->Value;
           return value2;
           break;
          }
```

```
   case 3:
            {
value3=extra->DBGrid1->DataSource->DataSet->FieldByName("id_privilege")->Value;
    return value3;
            break;
            }
            case 4:
{value4=opening_files->DBGrid1->DataSource->DataSet->FieldByName("id_topic")->Value;
            return value4;
            break;
            }
            case 5:
            {
value5=opening_files->DBGrid2->DataSource->DataSet->FieldByName("id_f")->Value;
            return value5;
            break;
            }
  }
}
void MyClass::proverka_material(int a, int b)
{
 DM->AQ_List->Close();
 DM->AQ_List->SQL->Clear();
DM->AQ_List->SQL->Add("SELECT id_f FROM File_to_topic where FILENAME='"+input_1->Edit2->Text+"'");
 DM->AQ_List->Open();
  if(DM->AQ_List->RecordCount)
  {ShowMessage("Такий запис існує! Будь ласка, ведіть по-іншому!");
  }
  else
  {
            DM->AQ_List->Close();
            DM->AQ_List->SQL->Clear();
DM->AQ_List->SQL->Text="INSERT INTO File_to_topic (id_topic, FILENAME, id_student) VALUES( :id_t, :f_n,
:id_s)";
            DM->AQ_List->Parameters->ParamByName("id_t")->Value=a;
            DM->AQ_List->Parameters->ParamByName("f_n")->Value=input_1->Edit2->Text;
            DM->AQ_List->Parameters->ParamByName("id_s")->Value=b;
            DM->AQ_List->ExecSQL();
            ShowMessage("Збереженo!");
  }
   input_1->DBLookupComboBox1->KeyValue=-1;
            input_1->DBLookupComboBox2->KeyValue=-1;
            input_1->Edit2->Text="";
}
void MyClass::vivod_file()
{
 DM->AQ_dod->Close();
 DM->AQ_dod->SQL->Clear();
 DM->AQ_dod->SQL->Add("SELECT * FROM List_of_topics");
 DM->AQ_dod->Open();
 int i = opening_files->DBGrid1->DataSource->DataSet->RecordCount;
 opening_files->StatusBar1->Panels->Items[0]->Text="Всього тем  = " + IntToStr(i);
}
void MyClass::vivod_material(int a)
{
 DM->AQ1_pom->Close();
 DM->AQ1_pom->SQL->Clear();
 DM->AQ1_pom->SQL->Add("SELECT v.id_f, v.id_topic, v.FILENAME, v.id_student, w.PIB FROM File_to_topic
v, Student w where w.id_student=v.id_student and v.id_topic=:p");
 DM->AQ1_pom->Parameters->ParamByName("p")->Value=a;
 DM->AQ1_pom->Open();
}
```

```
void MyClass::vivod_spicok(int nom, int sort)
 {
 DM->AQ_dod->Close();
 DM->AQ_dod->SQL->Clear();
 switch(nom)
 { case 1 :
            {
             DM->AQ_dod->SQL->Add("SELECT v.id_student, v.PIB, v.group_name, v.Date_Birth, v.PHOTO  FROM
Student as v where v.deducted=false");
             if(sort==1)
             {
                    DM->AQ_dod->SQL->Add("and v.group_name=:p");
                    DM->AQ_dod->Parameters->ParamByName("p")->Value=s_sort;
             }
             deducted->DBGrid1->Columns->Items[1]->Visible=true;
             deducted->DBGrid1->Columns->Items[2]->Visible=true;
             deducted->DBGrid1->Columns->Items[3]->Visible=true;
             deducted->DBGrid1->Columns->Items[6]->Visible=false;
             deducted->DBGrid1->Columns->Items[7]->Visible=false;
             deducted->DBGrid1->Columns->Items[9]->Visible=false;
             deducted->DBGrid1->Columns->Items[10]->Visible=false;
             break;
            }
            case 2 :
            {
             DM->AQ_dod->SQL->Add("SELECT v.id_student, v.PIB, v.group_name, w.id_hostel, w.№_room,
w.side_block, v.PHOTO FROM Student as v, Hostel as w where v.id_student=w.id_student and v.deducted=false");
             if(sort==2)
             {
                    DM->AQ_dod->SQL->Add("and v.group_name=:p");
                    DM->AQ_dod->Parameters->ParamByName("p")->Value=s_sort;
             }
             deducted->DBGrid1->Columns->Items[1]->Visible=true;
             deducted->DBGrid1->Columns->Items[2]->Visible=true;
             deducted->DBGrid1->Columns->Items[3]->Visible=false;
             deducted->DBGrid1->Columns->Items[6]->Visible=false;
             deducted->DBGrid1->Columns->Items[7]->Visible=false;
             deducted->DBGrid1->Columns->Items[9]->Visible=true;
             deducted->DBGrid1->Columns->Items[10]->Visible=true;
            break;
            }
            case 3 :
            {
             DM->AQ_dod->SQL->Add("SELECT v.id_advantage, v.id_privilege, c.name_privilege, v.id_student,
v.Note_3, w.PIB, w.group_name, w.PHOTO  FROM advantage as v, Privilege as c, Student as w where
v.id_student=w.id_student and v.id_privilege=c.id_privilege and w.deducted=false");
             if(sort==3)
             {
                    DM->AQ_dod->SQL->Add("and w.group_name=:p");
                    DM->AQ_dod->Parameters->ParamByName("p")->Value=s_sort;
             }
             deducted->DBGrid1->Columns->Items[1]->Visible=true;
             deducted->DBGrid1->Columns->Items[2]->Visible=true;
             deducted->DBGrid1->Columns->Items[3]->Visible=false;
             deducted->DBGrid1->Columns->Items[6]->Visible=true;
             deducted->DBGrid1->Columns->Items[7]->Visible=true;
             deducted->DBGrid1->Columns->Items[9]->Visible=false;
             deducted->DBGrid1->Columns->Items[10]->Visible=false;
             break;
            }
             case 4 :
            {
```

```
DM->AQ_dod->SQL->Add("SELECT v.id_student, v.PIB, v.group_name, v.Date_Birth, v.PHOTO  FROM Student as
v where v.deducted=true");
            if(sort==4)
            {
                    DM->AQ_dod->SQL->Add("and v.group_name=:p");
                    DM->AQ_dod->Parameters->ParamByName("p")->Value=s_sort;
            }
            deducted->DBGrid1->Columns->Items[1]->Visible=true;
            deducted->DBGrid1->Columns->Items[2]->Visible=true;
            deducted->DBGrid1->Columns->Items[3]->Visible=true;
            deducted->DBGrid1->Columns->Items[6]->Visible=false;
            deducted->DBGrid1->Columns->Items[7]->Visible=false;
            deducted->DBGrid1->Columns->Items[9]->Visible=false;
            deducted->DBGrid1->Columns->Items[10]->Visible=false;
            break;
           }
 }
 DM->AQ_dod->Open();
 }
  void MyClass::del_adv(int a)
  {DM->AQ_dod->Close();
          DM->AQ_dod->SQL->Clear();
          DM->AQ_dod->SQL->Text="DELETE FROM advantage where id_advantage=:p";
          DM->AQ_dod->Parameters->ParamByName("p")->Value=a;
          DM->AQ_dod->ExecSQL();
          ShowMessage("Запис успішно видалено!");
  }
 void MyClass::del_student(int a, int ozn)
{
 DM->AQ_dod->Close();
 DM->AQ_dod->SQL->Clear();
 if (ozn==1)
 {   DM->AQ_dod->SQL->Text="DELETE FROM Student where id_student=:p and deducted=true";}
 if(ozn==2)
 {
  DM->AQ_dod->SQL->Text="DELETE FROM Student where id_student=:p and deducted=false";}
  if(ozn==3)
 {   DM->AQ_dod->SQL->Text="DELETE FROM Hostel where id_hostel=:p";}
 DM->AQ_dod->Parameters->ParamByName("p")->Value=a;
 DM->AQ_dod->ExecSQL();
 ShowMessage("Запис про студента успішно видалено!");
}
void MyClass::vibor_stud(int n, TEdit *Edit2, TDBLookupComboBox *DBLookupComboBox3)
{DM->AQ_red_doc->Close();
  DM->AQ_red_doc->SQL->Clear();
  DM->AQ_red_doc->SQL->Add("select Student.id_student, Student.PIB, Student.group_name, Student.deducted from
Student where Student.deducted=false");
  if(n==1)
  {
   DM->AQ_red_doc->SQL->Add("and Student.PIB LIKE" " '%"+Edit2->Text+"%' ");
  }
  if(n==2)
  {
   DM->AQ_red_doc->SQL->Add(" and Student.group_name=:p");
DM->AQ_red_doc->Parameters->ParamByName("p")->Value=DBLookupComboBox3->ListSource->DataSet-
>FieldByName("group_name")->AsString;
   DM->AQ_red_doc->SQL->Add(" order by Student.PIB");
  }
  if(n==3)
  { DM->AQ_red_doc->SQL->Add("and Student.Note_1 LIKE" " '%"+Edit2->Text+"%' ");}
  DM->AQ_red_doc->Open();}
void MyClass::zv_t()
```

```
{DM->ADOQ_zvit->Close();
  DM->ADOQ_zvit->SQL->Clear();
  DM->ADOQ_zvit->SQL->Add("select Student.* from Student where Student.deducted=false and
Student.group_name=:p5");                    //поиск по фио
  DM->ADOQ_zvit->Parameters->ParamByName("p5")->Value=zvit_f->Edit10->Text;
  DM->ADOQ_zvit->SQL->Add("order by Student.PIB");
  DM->ADOQ_zvit->Open();
}
  void MyClass::zvit_host()
  {
   DM->ADOQ_zvit->Close();
   DM->ADOQ_zvit->SQL->Clear();
   DM->ADOQ_zvit->SQL->Add("select Student.PIB, Student.group_name, Hostel.* from Student, Hostel where
Student.deducted=false and Student.id_student=Hostel.id_student and Student.group_name=:p");
   DM->ADOQ_zvit->Parameters->ParamByName("p")->Value=zvit_f->Edit1->Text;
   DM->ADOQ_zvit->SQL->Add("order by Student.PIB");
   DM->ADOQ_zvit->Open();
  }
  void MyClass::zvit_advan()
  {
   DM->ADOQ_zvit->Close();
   DM->ADOQ_zvit->SQL->Clear();
   DM->ADOQ_zvit->SQL->Add("select Student.PIB, Student.Date_Birth, Student.group_name,
Privilege.name_privilege from Student, advantage, Privilege where Student.deducted=false and
Student.id_student=advantage.id_student and Student.group_name=:p1 and
Privilege.id_privilege=advantage.id_privilege");
   DM->ADOQ_zvit->Parameters->ParamByName("p1")->Value=zvit_f->Edit1->Text;
   DM->ADOQ_zvit->SQL->Add("order by Student.PIB");
   DM->ADOQ_zvit->Open();}
  void MyClass::zvit_doc()
  {   DM->ADOQ_zvit->Close();
   DM->ADOQ_zvit->SQL->Clear();
   DM->ADOQ_zvit->SQL->Add("select Student.PIB, Student.group_name, Collection_of_doc.*,
Document_type.name_doc from Student, Collection_of_doc, Document_type where Student.deducted=false and
Student.id_student=Collection_of_doc.id_student and Student.group_name=:p2 and
Document_type.id_doc=Collection_of_doc.id_doc and Collection_of_doc.handed_over=:p3 and
Document_type.name_doc=:p4");
   DM->ADOQ_zvit->Parameters->ParamByName("p2")->Value=zvit_f->Edit5->Text;
   DM->ADOQ_zvit->Parameters->ParamByName("p4")->Value=zvit_f->Edit6->Text;
   if (zvit_f->CheckBox11->Checked==true)
   {DM->ADOQ_zvit->Parameters->ParamByName("p3")->Value=true;}
if (zvit_f->CheckBox12->Checked==true)
   {DM->ADOQ_zvit->Parameters->ParamByName("p3")->Value=false;}
   DM->ADOQ_zvit->Open();
 }
```

Текст, модуля «zvit_gurt»

```
#include <vcl.h>
#pragma hdrstop
#include "zvit_gurt.h"
#include "Unit3.h"
#include "zvit.h"
#include <Vcl.Controls.hpp>
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>
#include <Vcl.ExtCtrls.hpp>
#include <Vcl.Imaging.jpeg.hpp>
#include <Vcl.Buttons.hpp>
#include <Vcl.ComCtrls.hpp>
#include <OleServer.hpp>
#include "Unit1.h"
#include "Unit2.h"
```

```
#include "Unit4.h"
#include "Unit6.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
Tzvit_gru *zvit_gru;
classMy np;
int stroka, kolZap;
Variant bApp, bBks, bBk, bShts, bSht, bCll;
int kuratorId;
String accessType;
__fastcall Tzvit_gru::Tzvit_gru(TComponent* Owner, int idKur, String typeAc)
        : TForm(Owner)
{kuratorId =  idKur;accessType = typeAc;}
void clen()
{
 zvit_gru->DBLookupComboBox1->KeyValue=-1;
 zvit_gru->RadioButton1->Checked=false;
 zvit_gru->RadioButton2->Checked=false;
 zvit_gru->RadioButton3->Checked=false;
 zvit_gru->Edit1->Text = " ";
 zvit_gru->DBLookupComboBox2->KeyValue=-1;
 DM->ADOTableSemestr->Active=false;
 DM->AQsqlGroup->Close();
 DM->AQlistStudentNew->Close();
 stroka = 1;
 kolZap = 0;
 for (int h = 0; h <  zvit_gru->StringGrid1->RowCount; h++) {
        zvit_gru->StringGrid1->Rows[h]->Clear();
 }
  zvit_gru->StringGrid1->RowCount = 3;
 zvit_gru->StringGrid1->Cells[0][0]="Дисципліна";
 zvit_gru->StringGrid1->Cells[1][0]="Семестр";
 zvit_gru->StringGrid1->Cells[2][0]="id_дисципліни";
 zvit_gru->StringGrid1->Cells[3][0]="id_семестру";
}
void __fastcall Tzvit_gru::FormActivate(TObject *Sender)
{clen();}
void __fastcall Tzvit_gru::RadioButton1Click(TObject *Sender)
{
 if (accessType == "Вид авторизації куратор.")
 {
   np.sqlGroup(false, Edit1, 1, true, kuratorId);
 }
 else np.sqlGroup(false, Edit1, 1, false, 0);
 DM->AQlistStudentNew->Close();
}
void __fastcall Tzvit_gru::RadioButton2Click(TObject *Sender)
{
 if (accessType == "Вид авторизації куратор.")
 {
   np.sqlGroup(false, Edit1, 2, true, kuratorId);
 }
 else np.sqlGroup(false, Edit1, 2, false, 0);
 DM->AQlistStudentNew->Close();
}
void __fastcall Tzvit_gru::RadioButton3Click(TObject *Sender)
{
if (accessType == "Вид авторизації куратор.")
 {
   np.sqlGroup(false, Edit1, 3, true, kuratorId);
 }
 else np.sqlGroup(false, Edit1, 3, false, 0);
```

```
 DM->AQlistStudentNew->Close();
}
void poisk()
{
int idGroup = zvit_gru->DBLookupComboBox1->ListSource->DataSet->FieldByName("id")->Value;
 np.disciplinesSearchByGroups(idGroup, false, zvit_gru->Edit1, false, 0);
 DM->ADOTableSemestr->Active=true;
}
void __fastcall Tzvit_gru::DBLookupComboBox1Click(TObject *Sender)
{poisk();}
void __fastcall Tzvit_gru::DBGrid1DblClick(TObject *Sender)
{
  kolZap = 0;
  for (int i = 1; i < StringGrid1->RowCount; i++) {
            if (StringGrid1->Cells[0][i] == DBGrid1->DataSource->DataSet->FieldByName("name__subject")->AsString
             && StringGrid1->Cells[1][i] == DBGrid1->DataSource->DataSet->FieldByName("name_semestr")-
>AsString
             && StrToInt(StringGrid1->Cells[2][i]) == DBGrid1->DataSource->DataSet->FieldByName("id_subject")-
>Value)
            {
              ShowMessage("Запис вже додано!!!");
              kolZap ++;
              break;
            }
  }
  if (kolZap == 0)
   {
            StringGrid1->Cells[0][stroka] = DBGrid1->DataSource->DataSet->FieldByName("name__subject")-
>AsString;
            StringGrid1->Cells[1][stroka] = DBGrid1->DataSource->DataSet->FieldByName("name_semestr")-
>AsString;
            StringGrid1->Cells[2][stroka] = DBGrid1->DataSource->DataSet->FieldByName("id_subject")->Value;
            StringGrid1->Cells[3][stroka] = DBGrid1->DataSource->DataSet->FieldByName("id_semestr")->Value;
    stroka++;
            if (stroka == StringGrid1->RowCount)
             {
              StringGrid1->RowCount = StringGrid1->RowCount + 1;
             }
   }
}
void __fastcall Tzvit_gru::Button1Click(TObject *Sender)
{clen();}
void __fastcall Tzvit_gru::StringGrid1DblClick(TObject *Sender)
{
  int count = StringGrid1->RowCount;
  int row = StringGrid1->Row;
  if ( count - StringGrid1->FixedRows <= 1 ) return;
  for ( int i = row; i < count; i++ )
  {
   StringGrid1->Rows[i] = StringGrid1->Rows[i+1];
  }
   StringGrid1->RowCount--;
   StringGrid1->SetFocus();
   stroka--;
}
void __fastcall Tzvit_gru::Edit1Change(TObject *Sender)
{
 if ( zvit_gru->RadioButton1->Checked==true ||
             zvit_gru->RadioButton2->Checked==true ||
             zvit_gru->RadioButton3->Checked==true)
 {
   int idGroup = DBLookupComboBox1->ListSource->DataSet->FieldByName("id")->Value;
```

```
   np.disciplinesSearchByGroups(idGroup, true, Edit1, false, 0);
 }
}
void __fastcall Tzvit_gru::DBLookupComboBox2Click(TObject *Sender)
{
  if ( zvit_gru->RadioButton1->Checked==true ||
               zvit_gru->RadioButton2->Checked==true ||
               zvit_gru->RadioButton3->Checked==true)
  {
int semestr = DBLookupComboBox2->ListSource->DataSet->FieldByName("id")->Value;
int idGroup = DBLookupComboBox1->ListSource->DataSet->FieldByName("id")->Value;
            np.disciplinesSearchByGroups(idGroup, false, Edit1, true, semestr);
 }
}
void __fastcall Tzvit_gru::Button2Click(TObject *Sender)
{
 zvit_gru->Edit1->Text = " ";
 zvit_gru->DBLookupComboBox2->KeyValue=-1;
 DM->ADOTableSemestr->Active=false;
 if ( zvit_gru->RadioButton1->Checked==true ||zvit_gru->RadioButton2->Checked==true ||zvit_gru->RadioButton3-
>Checked==true)
               {poisk();}
}
void __fastcall Tzvit_gru::Button3Click(TObject *Sender)
{
   if (StringGrid1->Cells[0][1]!="")
   {
   String subject, semestr, fio, grup;
   int stolbec, idGroup, stroka, number, idStudent, idSubject, bal, max, kurs;
   double sum, sredBal, kolSum;
   bApp=CreateOleObject("Excel.Application");
   bApp.OlePropertySet("Visible",false);
   bBks=bApp.OlePropertyGet("Workbooks");
   bApp.OlePropertySet("SheetsInNewWorkbook",1);
   bBks.OleProcedure("Add");
   bBk=bBks.OlePropertyGet("Item",1);
   bShts=bBk.OlePropertyGet("Worksheets");
   bSht=bShts.OlePropertyGet("Item",1);
bSht.OlePropertyGet("Cells").OlePropertyGet("Item",2).OlePropertySet("ColumnWidth",35);
   bCll=bSht.OlePropertyGet("Cells").OlePropertyGet("Item",3,2);
   np.borders(bCll);
   bCll.OlePropertySet("Value",WideString("ПІБ студентів    /    Семестр"));
bSht.OlePropertyGet("Cells").OlePropertyGet("Item",1).OlePropertySet("ColumnWidth",3);
   bCll=bSht.OlePropertyGet("Cells").OlePropertyGet("Item",3,1);
   np.borders(bCll);
   bCll.OlePropertySet("Value",WideString("№"));
   bCll=bSht.OlePropertyGet("Cells").OlePropertyGet("Item",2,2);
   np.borders(bCll);
   grup = DBLookupComboBox1->ListSource->DataSet->FieldByName("name")->AsString;
   bCll.OlePropertySet("Value",WideString(grup));
   bCll=bSht.OlePropertyGet("Cells").OlePropertyGet("Item",2,1);
   np.borders(bCll);
   bCll.OlePropertySet("Value",WideString(""));
   stolbec = 3;
max = StrToInt(StringGrid1->Cells[3][1]);
   for (int i = 1; i < StringGrid1->RowCount-1; i++)
   {subject = StringGrid1->Cells[0][i];
bSht.OlePropertyGet("Cells").OlePropertyGet("Item",stolbec).OlePropertySet("ColumnWidth",3);
bCll=bSht.OlePropertyGet("Cells").OlePropertyGet("Item",2,stolbec);
            bCll.OlePropertySet("Orientation",90);
            np.borders(bCll);
            bCll.OlePropertySet("Value",WideString(subject));
```

```
            semestr = StringGrid1->Cells[3][i];
            bCll=bSht.OlePropertyGet("Cells").OlePropertyGet("Item",3,stolbec);
            np.borders(bCll);
            bCll.OlePropertySet("Value",WideString(semestr));
            stolbec++;
            if (max<StrToInt(semestr))
            {
              max = StrToInt(semestr);
            }
  }
   bCll=bSht.OlePropertyGet("Cells").OlePropertyGet("Item",3,stolbec);
   np.borders(bCll);
   bCll=bSht.OlePropertyGet("Cells").OlePropertyGet("Item",2,stolbec);
   bCll.OlePropertySet("Orientation",90);
   np.borders(bCll);
   bCll.OlePropertySet("Value",WideString("Середній бал"));
   idGroup = DBLookupComboBox1->ListSource->DataSet->FieldByName("id")->Value;
   kurs = np.numberKurs(max);
   np.listStudent(idGroup, kurs);
            DM->AQlistStudent->First();
            stroka = 4;
            number = 1;
            for (int h = 0; h < DM->AQlistStudent->RecordCount; h++)
            {
             fio = DM->AQlistStudent->FieldByName("familiya")->AsString;
             fio = fio + " " + DM->AQlistStudent->FieldByName("imya")->AsString;
             fio = fio + " " + DM->AQlistStudent->FieldByName("otchestvo")->AsString;
             bCll=bSht.OlePropertyGet("Cells").OlePropertyGet("Item",stroka,2);
             np.borders(bCll);
             bCll.OlePropertySet("Value",WideString(fio));
             bCll=bSht.OlePropertyGet("Cells").OlePropertyGet("Item",stroka,1);
             np.borders(bCll);
             bCll.OlePropertySet("Value",WideString(number));
             idStudent = DM->AQlistStudent->FieldByName("id")->Value;
             stolbec = 3;
             sum = 0;
             sredBal = 0;
             kolSum = 0;
                        for (int j = 1; j < StringGrid1->RowCount-1; j++)
                        {
                         idSubject = StrToInt(StringGrid1->Cells[2][j]);
                         np.subjectStudentBal(idStudent, idSubject);
                         if (DM->ADOQueryBal->RecordCount !=0 )
                         {
 bal = np.markNumber(DM->ADOQueryBal->FieldByName("kol_bal")->AsInteger);
bCll=bSht.OlePropertyGet("Cells").OlePropertyGet("Item",stroka,stolbec);
                          np.borders(bCll);
bCll=bSht.OlePropertyGet("Cells").OlePropertyGet("Item",stroka,stolbec);
                                  np.borders(bCll);
                         if (bal == 0)
                         {bCll.OlePropertySet("Value",WideString(" "));
                                  stolbec++;}
                         else
                                  {
                                   bCll.OlePropertySet("Value",WideString(bal));
                                   stolbec++;
                                   sum = sum + bal;
                                   kolSum++;}
                        }
                        else{
bCll=bSht.OlePropertyGet("Cells").OlePropertyGet("Item",stroka,stolbec);
                                   np.borders(bCll);
```

```
                                bCll.OlePropertySet("Value",WideString(" "));
                                stolbec++;
          }
                        }
                        if (kolSum != 0){
                          sredBal = sum / kolSum;
bCll=bSht.OlePropertyGet("Cells").OlePropertyGet("Item",stroka,stolbec);
                          np.borders(bCll);
                          bCll.OlePropertySet("Value",WideString(sredBal));
                        }
                        else{
bCll=bSht.OlePropertyGet("Cells").OlePropertyGet("Item",stroka,stolbec);
                         np.borders(bCll);
                         bCll.OlePropertySet("Value",WideString(" "));
                        }
            number++;stroka++;
            DM->AQlistStudent->Next();
           }
  ShowMessage("Документ сформовано!");
  bApp.OlePropertySet("Visible",true);
  }
   else ShowMessage("Ви не обрали параметри!!!");
}
```

Текст, модуля «student»

```
#include <vcl.h>
#pragma hdrstop
#include "student.h"
#include <windows.h>
#include <string.h>
#include "Unit3.h"
#include "zvit.h"
#include <Vcl.Controls.hpp>
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>
#include <Vcl.ExtCtrls.hpp>
#include <Vcl.Imaging.jpeg.hpp>
#include <Vcl.Buttons.hpp>
#include <Vcl.ComCtrls.hpp>
#include <OleServer.hpp>
#include "Unit1.h"
#include "Unit2.h"
#include "Unit4.h"
#include "Unit6.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm7 *Form7;
classMy newUk;
int idKurat;
String types;
__fastcall TForm7::TForm7(TComponent* Owner, int id, String ty)
          : TForm(Owner)
{idKurat = id; types = ty;}
void __fastcall TForm7::FormActivate(TObject *Sender)
{clens();}
void __fastcall TForm7::RadioButton1Click(TObject *Sender)
{
  if (types == "Вид авторизації куратор.")
 {
   newUk.sqlGroup(false, Edit1, 1, true, idKurat);
 }
 else newUk.sqlGroup(false, Edit1, 1, false, 0);
```

```
}
void __fastcall TForm7::RadioButton2Click(TObject *Sender)
{
  if (types == "Вид авторизації куратор.")
   {
            newUk.sqlGroup(false, Edit1, 2, true, idKurat);
   }
   else newUk.sqlGroup(false, Edit1, 2, false, 0);
}
void __fastcall TForm7::RadioButton3Click(TObject *Sender)
{
 if (types == "Вид авторизації куратор.")
 {
   newUk.sqlGroup(false, Edit1, 3, true, idKurat);
 }
  else newUk.sqlGroup(false, Edit1, 3, false, 0);
}
void __fastcall TForm7::DBLookupComboBox1Click(TObject *Sender)
{DM->ADOTableSemestr->Active=true;}
void chart(bool flag)
{
if (Form7->DBLookupComboBox1->KeyValue!=-1 && Form7->DBLookupComboBox2->KeyValue!=-1)
 {
  Variant zbApp, zbBks, zbBk, zbShts, zbSht, zbCll;
  int semestr, idGroup, idSubject, bal, kolSum, sredBal, sum, kurs;
  int idStudent, kolStud;
  double kol5, kol4, kol3, kol2, academicPerfomnce, qualityOfKnowledge;
  String pathExeApp;
semestr = Form7->DBLookupComboBox2->ListSource->DataSet->FieldByName("id")->Value;
idGroup = Form7->DBLookupComboBox1->ListSource->DataSet->FieldByName("id")->Value;
   kurs = newUk.numberKurs(semestr);
   newUk.listStudent(idGroup, kurs);
   if (DM->AQlistStudent->RecordCount == 0)
   {ShowMessage("Записів для обраного семестру не існує!!!");}
   else
  {
   newUk.disciplinesSearchByGroups(idGroup, false, NULL, true, semestr);
   DM->AQlistStudent->First();
   kol5 = 0;
   kol4 = 0;
   kol3 = 0;
   kol2 = 0;
   kolStud = 0;
   academicPerfomnce = 0;
   qualityOfKnowledge = 0;
   for (int h = 0; h < DM->AQlistStudent->RecordCount; h++)
   {
            idStudent = DM->AQlistStudent->FieldByName("id")->Value;
            sum = 0;
            sredBal = 0;
            kolSum = 0;
             DM->AQlistStudentNew->First();
             for (int j = 0; j < DM->AQlistStudentNew->RecordCount; j++)
             {
               idSubject = DM->AQlistStudentNew->FieldByName("id_subject")->AsInteger;
               newUk.subjectStudentBal(idStudent, idSubject);
             bal = newUk.markNumber(DM->ADOQueryBal->FieldByName("kol_bal")->AsInteger);
               sum = sum + bal;
               kolSum++;
               DM->AQlistStudentNew->Next();
             }
             sredBal = sum / kolSum;
```

```
            if (sredBal == 5) {kol5++; }
            if (sredBal == 4) {kol4++;}
            if (sredBal == 3) {kol3++;}
            if (sredBal == 2) {kol2++; }
            kolStud++;
            DM->AQlistStudent->Next(); }
           academicPerfomnce = (kol5 + kol4 + kol3) / kolStud;
           qualityOfKnowledge = (kol5 + kol4) / kolStud;
 pathExeApp = ExtractFilePath (Application->ExeName) + "\\model\\chart.xlsx" ;
           zbApp=CreateOleObject("Excel.Application");
           zbApp.OlePropertySet("Visible",false);
           zbBks=zbApp.OlePropertyGet("Workbooks");
           zbApp.OlePropertySet("SheetsInNewWorkbook",1);
           zbBks.OleProcedure("Add", WideString(pathExeApp));
           zbBk=zbBks.OlePropertyGet("Item",1);
           zbShts=zbBk.OlePropertyGet("Worksheets");
           zbSht=zbShts.OlePropertyGet("Item",1);
           zbCll=zbSht.OlePropertyGet("Cells").OlePropertyGet("Item",7,2);
           zbCll.OlePropertySet("Value",WideString(academicPerfomnce));
           zbCll=zbSht.OlePropertyGet("Cells").OlePropertyGet("Item",7,3);
           zbCll.OlePropertySet("Value",WideString(qualityOfKnowledge));
String grupp = "Успішність та якість навчання групи " + Form7->DBLookupComboBox1->ListSource->DataSet-
>FieldByName("name")->AsString;
           zbCll=zbSht.OlePropertyGet("Cells").OlePropertyGet("Item",2,2);
           zbCll.OlePropertySet("Value",WideString(grupp));
           String studKol = IntToStr(kolStud);
           zbCll=zbSht.OlePropertyGet("Cells").OlePropertyGet("Item",12,2);
           zbCll.OlePropertySet("Value",WideString(studKol));
           if (flag == true) {
             ShowMessage("Документ сформовано!");
             zbApp.OlePropertySet("Visible",true);
           }
           if (flag == false) {
            ShowMessage("Документ сформовано! Відправлено на друк!");
             try{
                    zbApp.OlePropertyGet("Workbooks",1).OleProcedure("PrintOut");
             }
             catch(...)
             {
                    ShowMessage("Друк не можливий!!!");
             }
            zbApp.OlePropertyGet("Workbooks", 1).OleProcedure("Close", false);
            zbApp.OleProcedure("Quit");
           }
           clens();}
 }else ShowMessage("Ви не обрали параметри!!!");
}
void __fastcall TForm7::Button1Click(TObject *Sender)
{chart(true);}
void __fastcall TForm7::Button2Click(TObject *Sender)
{chart(false);}
```

Лістинг Б.10 – Текст, модуля «listdb»

```
#include <vcl.h>
#pragma hdrstop
#include "list.h"
#include <OleServer.hpp>
#include <windows.h>
#include <string.h>
#include <array>
#include <iostream>
#include "Unit3.h"
```

```
#include "Unit6.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
Tlistdb *listdb;
classMy newP;
int kurId;
String aType;
__fastcall Tlistdb::Tlistdb(TComponent* Owner, int idKurs, String typeAcs)
        : TForm(Owner)
{kurId = idKurs;aType = typeAcs;}
void __fastcall Tlistdb::FormActivate(TObject *Sender)
{
 Button2->Visible=false;
 CheckBox1->Checked=false;
 CheckBox2->Checked=false;
}
void __fastcall Tlistdb::Edit1Change(TObject *Sender)
{
 if (aType == "Вид авторизації куратор.")
 {newP.sqlGroup(true, Edit1, 0, true, kurId);}
 else newP.sqlGroup(true, Edit1, 0, false, 0);
}
void __fastcall Tlistdb::DBGrid1CellClick(TColumn *Column)
{
 int kurs;
 int idGroup = DBGrid1->DataSource->DataSet->FieldByName("id")->Value;
 kurs = newP.maxKurs(idGroup);
 newP.listStudent(idGroup, kurs);
}
void __fastcall Tlistdb::Button1Click(TObject *Sender)
{Edit1->Text="";}
void __fastcall Tlistdb::Button2Click(TObject *Sender)
{
  if (CheckBox1->Checked==true||CheckBox2->Checked==true)
  {
  Variant vApp, vDocs, vDoc, vBookmarks, vSelection, vRange, vBookmark;
  int i1;
  double pos = 0, kol;
  ProgressBar1->Position = 0;
  String path1 = ExtractFilePath (Application->ExeName);
  String path = path1 + "\\model\\shList.doc" ;
  vApp = CreateOleObject("Word.Application");
  vApp.OlePropertySet("Visible", false);
  vDocs = vApp.OlePropertyGet("Documents");
  vDocs.OleFunction("Add", WideString(path));
  vDoc=vDocs.OleFunction("Item",1);
  vDoc.OleProcedure("Activate");
newP.changeWord(vApp, vDoc,"grup",DM->AQlistStudent->FieldByName("name")->AsString);
  pos = pos + 1;
  ProgressBar1->Position = pos;
  DM->AQlistStudent->First();
  i1=1;
   for(int p = 0; p < DM->AQlistStudent->RecordCount; p++)
  {
newP.changeWord(vApp, vDoc,"fio_"+IntToStr(i1),DM->AQlistStudent->FieldByName("familiya")->AsString);
newP.changeWord(vApp, vDoc,"name_"+IntToStr(i1),DM->AQlistStudent->FieldByName("imya")->AsString);
newP.changeWord(vApp, vDoc,"otch_"+IntToStr(i1),DM->AQlistStudent->FieldByName("otchestvo")->AsString);
            DM->AQlistStudent->Next();
            i1++;
            if (pos>60) {
            kol=0.2;
             }
```

```
            else kol = 1;
             pos = pos + kol;
             ProgressBar1->Position = pos;
  }
            if (pos>80) { kol=0.2;}
             else kol= 1;
            for (int count = i1; count <= 30; count++)
            {
            newP.changeWord(vApp, vDoc,"fio_"+IntToStr(count),"");
            newP.changeWord(vApp, vDoc,"name_"+IntToStr(count),"");
            newP.changeWord(vApp, vDoc,"otch_"+IntToStr(count),"");
                        pos = pos + kol;
             ProgressBar1->Position = pos;
            }
            if (pos>90) {
                        kol=0.2;
            }
             else kol= 1;
  if (CheckBox1->Checked==true&&CheckBox2->Checked==false) {
  vApp.OlePropertySet("Visible", true);
  }
  if (CheckBox2->Checked==true&&CheckBox1->Checked==false) {
            pos = pos + kol;
            ProgressBar1->Position = pos;
              try{
                        vApp.OlePropertyGet("Documents",1).OleProcedure("PrintOut");
              }
              catch(...)
              { ShowMessage("Друк не можливий!!!");}
             vApp.OlePropertyGet("Workbooks", 1).OleProcedure("Close", false);
             vApp.OleProcedure("Quit");
  }
  if (CheckBox1->Checked==true&&CheckBox2->Checked==true) {
            pos = pos + kol;
            ProgressBar1->Position = pos;
     try{ vApp.OleProcedure("PrintOut");
        ShowMessage("Документ друкується!");
              }
              catch(...)
              {
                        ShowMessage("Друк не можливий!!!");
              }
             vApp.OlePropertySet("Visible", true);
             vApp.OlePropertyGet("Documents").OleProcedure("Close", false);
             vApp.OleProcedure("Quit");
            }
  if(pos<200)
   ProgressBar1->Position = 200;
             ShowMessage("Документ сформовано!");
    vApp=NULL;
  }
  else ShowMessage("Оберіть пункт!");
  ProgressBar1->Position = 0;
  CheckBox1->Checked=false;
  CheckBox2->Checked=false;
  Button2->Visible=false;
}
void __fastcall Tlistdb::CheckBox1Click(TObject *Sender)
{
  if (Button2->Visible==false) {
              Button2->Visible=true;
  }
```

```
}
void __fastcall Tlistdb::CheckBox2Click(TObject *Sender)
{
  if (Button2->Visible==false) { Button2->Visible=true;}
}
```

Текст, модуля «input_document»

```
#include <vcl.h>
#pragma hdrstop
#include "Unit2.h"
#include "Unit3.h"
#include "inputdocument.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
Tinput_document *input_document;
MyClass newCl;
void vivod_doc(int nom)
{
 DM->AQ__document->Close();
 DM->AQ__document->SQL->Clear();
 DM->AQ__document->SQL->Add("Select Collection_of_doc.*,  Student.PIB, Student.group_name,
Student.deducted, Document_type.name_doc from Collection_of_doc, Student, Document_type where
Collection_of_doc.id_student=Student.id_student and Collection_of_doc.id_doc=Document_type.id_doc");
 if(nom==1)
 {
  DM->AQ__document->SQL->Add("and Student.PIB LIKE" " '%"+input_document->Edit1->Text+"%' ");
 }
 DM->AQ__document->Open();
}
void __fastcall Tinput_document::DBGrid1TitleClick(TColumn *Column)
{   if (DM->AQ__document->Active)
if ((DM->AQ__document->Sort.Pos(Column->FieldName) > 0) && (DM->AQ__document->Sort.Pos("ASC") > 0))
                                {DM->AQ__document->Sort = Column->FieldName + " DESC";}
                        else {DM->AQ__document->Sort = Column->FieldName + " ASC";}
}
void __fastcall Tinput_document::FormActivate(TObject *Sender)
{
 DM->AQ_grup_stud->Active=true;
 if(input_document->Caption=="Перегляд зібраних документів")
 {
  newCl.call_bookmark(PageControl1, TabSheet1);
  DM->AQ__document->Active=true;
  DM->AQ_v_doc->Active=true;
 }
 if(input_document->Caption=="Додавання документів")
 {
  Edit7->Visible=false;
  Label15->Visible=false;
  newCl.call_bookmark(PageControl1, TabSheet2);
  DM->AQ_red_doc->Active=true;
  DM->AQ_v_doc->Active=true;
  DBGrid2->Enabled=true;
 }
}
void __fastcall Tinput_document::Edit1KeyPress(TObject *Sender, System::WideChar &Key)
{newCl.proverka_Key(Edit1,Key);}
void __fastcall Tinput_document::BitBtn1Click(TObject *Sender)
{
 if(Edit1->Text!="")
 {
 vivod_doc(0);
 Edit1->Text="";
```

```
 StatusBar1->Panels->Items[0]->Text="";
 }
}
void __fastcall Tinput_document::Edit1Change(TObject *Sender)
{
 vivod_doc(1);
 int i = DBGrid1->DataSource->DataSet->RecordCount;
 if(i!=0)
 StatusBar1->Panels->Items[0]->Text="Всього студентів = " + IntToStr(i);
 else StatusBar1->Panels->Items[0]->Text="Не знайдено студентів за прізвищем!";
}
void __fastcall Tinput_document::BitBtn2Click(TObject *Sender)
{
  vivod_doc(0);
  DBLookupComboBox1->KeyValue=-1;
  DBLookupComboBox2->KeyValue=-1;
  Edit2->Text="";
  Edit3->Text="";
  Edit4->Text="";
  CheckBox1->Checked=false;
  CheckBox2->Checked=false;
}
void __fastcall Tinput_document::BitBtn3Click(TObject *Sender)
{
 DM->AQ__document->Close();
 DM->AQ__document->SQL->Clear();
 DM->AQ__document->SQL->Add("Select Collection_of_doc.*,  Student.PIB, Student.group_name,
Student.deducted, Document_type.name_doc from Collection_of_doc, Student, Document_type where
Collection_of_doc.id_student=Student.id_student and Collection_of_doc.id_doc=Document_type.id_doc");
  if(Edit2->Text!="")
 {
  DM->AQ__document->SQL->Add(" and Document_type.name_doc=:l");
  DM->AQ__document->Parameters->ParamByName("l")->Value=Edit2->Text;
 }
  if(Edit3->Text!="")
 {
  DM->AQ__document->SQL->Add(" and Student.group_name=:p");
  DM->AQ__document->Parameters->ParamByName("p")->Value=Edit3->Text;
 }
   if(CheckBox1->Checked==true && CheckBox2->Checked==false)
  {
   DM->AQ__document->SQL->Add(" and Collection_of_doc.handed_over=true");
  }
   if(CheckBox2->Checked==true && CheckBox1->Checked==false)
  {
   DM->AQ__document->SQL->Add(" and Collection_of_doc.handed_over=false");
  }
  if(CheckBox1->Checked==true && CheckBox2->Checked==true)
  {
   ShowMessage("Фільтр не доречний!");
   CheckBox1->Checked=false;
   CheckBox2->Checked=false;
  }
 DM->AQ__document->Open();
  int i = DBGrid1->DataSource->DataSet->RecordCount;
  if(i!=0)
  Edit4->Text=i;
  else Edit4->Text=0;
}
void __fastcall Tinput_document::BitBtn6Click(TObject *Sender)
{
   int a = input_document->DBGrid1->DataSource->DataSet->FieldByName("id_g_doc")->Value;
```

```
  DM->AQ__document->Close();
  DM->AQ__document->SQL->Clear();
  DM->AQ__document->SQL->Text="DELETE FROM Collection_of_doc where id_g_doc=:p";
  DM->AQ__document->Parameters->ParamByName("p")->Value=a;
  DM->AQ__document->ExecSQL();
  ShowMessage("Запис успішно видалено!");
  vivod_doc(0);
}
void __fastcall Tinput_document::DBLookupComboBox1Click(TObject *Sender)
{Edit2->Text = DBLookupComboBox1->ListSource->DataSet->FieldByName("name_doc")->AsString;}
void __fastcall Tinput_document::DBLookupComboBox2Click(TObject *Sender)
{
 Edit3->Text = DBLookupComboBox2->ListSource->DataSet->FieldByName("group_name")->AsString;
}
void __fastcall Tinput_document::BitBtn7Click(TObject *Sender)
{
 DBLookupComboBox3->KeyValue=-1;
  DM->AQ_red_doc->Close();
 DM->AQ_red_doc->SQL->Clear();
 DM->AQ_red_doc->SQL->Add("select Student.id_student, Student.PIB, Student.group_name, Student.deducted from
Student where Student.deducted=false");
 DM->AQ_red_doc->Open();
}
void __fastcall Tinput_document::DBLookupComboBox3Click(TObject *Sender)
{
DM->AQ_red_doc->Close();
DM->AQ_red_doc->SQL->Clear();
DM->AQ_red_doc->SQL->Add("select Student.id_student, Student.PIB, Student.group_name, Student.deducted from
Student where Student.deducted=false");
DM->AQ_red_doc->SQL->Add(" and Student.group_name=:p");
DM->AQ_red_doc->Parameters->ParamByName("p")->Value=DBLookupComboBox3->ListSource->DataSet-
>FieldByName("group_name")->AsString;
DM->AQ_red_doc->Open();
}
void __fastcall Tinput_document::BitBtn4Click(TObject *Sender)
{
 input_document->Caption="Додавання документів";
 Edit7->Visible=false;
 Label15->Visible=false;
 DM->AQ_red_doc->Active=true;
 DM->AQ_v_doc->Active=true;
 DBGrid2->Enabled=true;
 newCl.call_bookmark(PageControl1, TabSheet2);
}
void __fastcall Tinput_document::Button2Click(TObject *Sender)
{
clin();
input_document->Close();
}
void __fastcall Tinput_document::DBGrid2CellClick(TColumn *Column)
{Edit5->Text=DBGrid2->DataSource->DataSet->FieldByName("PIB")->Value;}
void __fastcall Tinput_document::Button1Click(TObject *Sender)
{
String perem;
if(CheckBox3->Checked==true)  perem=true;
else perem=false;
 DM->AQ_List->Close();
 DM->AQ_List->SQL->Clear();
 DM->AQ_List->SQL->Add("SELECT id_g_doc FROM Collection_of_doc where id_student=:p and id_doc=:k and
handed_over=:n");
 DM->AQ_List->Parameters->ParamByName("p")->Value=DBGrid2->DataSource->DataSet-
>FieldByName("id_student")->AsString;
```

```
 DM->AQ_List->Parameters->ParamByName("k")->Value=DBLookupComboBox4->ListSource->DataSet-
>FieldByName(DBLookupComboBox4->KeyField)->AsString;
 DM->AQ_List->Parameters->ParamByName("n")->Value=CheckBox3->Checked;
 DM->AQ_List->Open();
  if(DM->AQ_List->RecordCount>0)
  {ShowMessage("Такий запис існує! Будь ласка, ведіть по-іншому!");}
  else
  {
        DM->AQ_List->Close();
        DM->AQ_List->SQL->Clear();
        if(input_document->Caption=="Додавання документів")
        {
         DM->AQ_List->SQL->Text="INSERT INTO Collection_of_doc (id_student, id_doc, Date_, handed_over,
Note_5) VALUES( :id_s, :id_d, :Dat, :han, :not)";
         DM->AQ_List->Parameters->ParamByName("id_s")->Value=DBGrid2->DataSource->DataSet-
>FieldByName("id_student")->Value;
        }
        if(input_document->Caption=="Редагування документів")
        {
         DM->AQ_List->SQL->Text="UPDATE Collection_of_doc SET id_student=:id_s, id_doc=:id_d, Date_=:Dat,
handed_over=:han, Note_5=:not where id_g_doc=:p";
         DM->AQ_List->Parameters->ParamByName("id_s")->Value=nomer;
         DM->AQ_List->Parameters->ParamByName("p")->Value=nomer2;
        }
        DM->AQ_List->Parameters->ParamByName("id_d")->Value=DBLookupComboBox4->ListSource-
>DataSet->FieldByName(DBLookupComboBox4->KeyField)->AsString;
        DM->AQ_List->Parameters->ParamByName("Dat")->Value=DateToStr(DateTimePicker1->Date);
        DM->AQ_List->Parameters->ParamByName("han")->Value=perem;
        DM->AQ_List->Parameters->ParamByName("not")->Value=Edit6->Text;
        DM->AQ_List->ExecSQL();
        ShowMessage("Збережено!");
  }
  clin();
  perem=false;
  if(input_document->Caption=="Редагування документів")
  {
        input_document->Close();
  }
   vivod_doc(0);
}
void __fastcall Tinput_document::BitBtn5Click(TObject *Sender)
{
 input_document->Caption="Редагування документів";
 Edit5->Text=DBGrid1->DataSource->DataSet->FieldByName("PIB")->Value;
 nomer = DBGrid1->DataSource->DataSet->FieldByName("id_student")->Value;
 nomer2 = DBGrid1->DataSource->DataSet->FieldByName("id_g_doc")->Value;
 Edit6->Text=DBGrid1->DataSource->DataSet->FieldByName("Note_5")->Value;
 Edit7->Visible=true;
 Edit7->Enabled=false;
 Label15->Visible=true;
 Edit7->Text=DBGrid1->DataSource->DataSet->FieldByName("name_doc")->Value;
 DBGrid2->Enabled=false;
 DM->AQ_red_doc->Active=true;
 DM->AQ_v_doc->Active=true;
 newCl.call_bookmark(PageControl1, TabSheet2);
}
void __fastcall Tinput_document::DBGrid2TitleClick(TColumn *Column)
{
  if (DM->AQ_red_doc->Active)
                  if ((DM->AQ_red_doc->Sort.Pos(Column->FieldName) > 0) && (DM->AQ_red_doc-
>Sort.Pos("ASC") > 0))
                        {DM->AQ_red_doc->Sort = Column->FieldName + " DESC";}
```

```
                    else {DM->AQ_red_doc->Sort = Column->FieldName + " ASC";}
  else ShowMessage("Помилка при сортуванні!");
}
void DrawGridCheckBox(TCanvas * Canvas,TRect Rect,bool Checked)
{
          int DrawFlags;
  Canvas->TextRect(Rect, Rect.Left + 1, Rect.Top + 1, " ");
  DrawFrameControl(Canvas->Handle, &Rect, DFC_BUTTON, DFCS_BUTTONPUSH | DFCS_ADJUSTRECT);
  DrawFlags = DFCS_BUTTONCHECK | DFCS_ADJUSTRECT;// DFCS_BUTTONCHECK
  if (Checked)
  {
   DrawFlags = DrawFlags | DFCS_CHECKED;
  }
  DrawFrameControl(Canvas->Handle, &Rect, DFC_BUTTON, DrawFlags);
}
void __fastcall Tinput_document::DBGrid1DrawColumnCell(TObject *Sender, const TRect &Rect,
      int DataCol, TColumn *Column, TGridDrawState State)
{
 if (Column->FieldName == "handed_over")
 {
         if (Column->Field->AsString == "True")
  {
           DrawGridCheckBox(DBGrid1->Canvas, Rect, true);
  }
  else
         {
           DrawGridCheckBox(DBGrid1->Canvas, Rect, false);
  }
 }
}
```

Текст, модуля «opening_files»

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
#include "Unit5.h"
#include "openingfiles.h"
#include <windows.h>
#include <Shellapi.h>
#pragma package(smart_init)
#pragma resource "*.dfm"
Topening_files *opening_files;
MyClass my5;
void red_topic()
 {
  input_1->Edit1->Text=opening_files->DBGrid1->DataSource->DataSet->FieldByName("topic")->Value;
  input_1->LabeledEdit1->Text=opening_files->DBGrid1->DataSource->DataSet->FieldByName("Note_4")->Value;
  int a = my5.zap_delet(4);
  my5.s_value=a;
  my5.name_form(2,1);
  my5.vivod_file();
 }
 void delet_topic()
 {
  int value4=my5.zap_delet(4);
  DM->AQ_dod->Close();
  DM->AQ_dod->SQL->Clear();
  DM->AQ_dod->SQL->Text="DELETE FROM List_of_topics where id_topic=:p";
```

```
   DM->AQ_dod->Parameters->ParamByName("p")->Value=value4;
   DM->AQ_dod->ExecSQL();
   ShowMessage("Запис успішно видалено!");
   my5.vivod_file();
  }
void delet_file()
  {
   int value5=my5.zap_delet(5);
   DM->AQ1_pom->Close();
   DM->AQ1_pom->SQL->Clear();
   DM->AQ1_pom->SQL->Text="DELETE FROM File_to_topic where id_f=:p";
   DM->AQ1_pom->Parameters->ParamByName("p")->Value=value5;
   DM->AQ1_pom->ExecSQL();
   ShowMessage("Запис успішно видалено!");
   my5.vivod_file();
   int value4=my5.zap_delet(4);
   my5.vivod_material(value4);
  }
void open_file()
{
String fileName = opening_files->DBGrid2->DataSource->DataSet->FieldByName("FILENAME")->AsString;
  String path = ExtractFilePath (Application->ExeName) + "\\FILES"+"\\" + fileName;
 if(path=="")
            {
             if (MessageDlg("Матеріал не обрано. Закрити додаток?", mtError, TMsgDlgButtons() << mbYes <<
mbNo,0) == mrYes)
      opening_files->Close();
            }
 else
 {
  ShellExecute(
            NULL,
            _TEXT("open"),
            path.c_str(),
            NULL,
            NULL,
            SW_SHOWNORMAL);
 }
}
__fastcall Topening_files::Topening_files(TComponent* Owner)
            : TForm(Owner){}
void __fastcall Topening_files::Edit1KeyPress(TObject *Sender, System::WideChar &Key)
{my5.proverka_Key(Edit1,Key);}
void __fastcall Topening_files::BitBtn1Click(TObject *Sender)
{
if(Edit1->Text!="")
Edit1->Text="";
}
void __fastcall Topening_files::Edit1Change(TObject *Sender)
{
  DM->AQ_dod->Close();
  DM->AQ_dod->SQL->Clear();
  DM->AQ_dod->SQL->Add("SELECT * FROM List_of_topics ");
  DM->AQ_dod->SQL->Add("where topic LIKE" " '%"+Edit1->Text+"%' ");
  DM->AQ_dod->Open();
}
void __fastcall Topening_files::BitBtn2Click(TObject *Sender)
{my5.name_form(1,1);
my5.vivod_file();}
void __fastcall Topening_files::BitBtn3Click(TObject *Sender)
{delet_topic();}
void __fastcall Topening_files::BitBtn4Click(TObject *Sender)
```

```
{red_topic();}
void __fastcall Topening_files::DBGrid1CellClick(TColumn *Column)
{int value4=my5.zap_delet(4);
 my5.vivod_material(value4);}
void __fastcall Topening_files::BitBtn5Click(TObject *Sender)
{
 input_1->Caption="Завантаження матеріалів";
 input_1->ShowModal();
 my5.vivod_file();
}
void __fastcall Topening_files::BitBtn6Click(TObject *Sender)
{delet_file();}
void __fastcall Topening_files::BitBtn7Click(TObject *Sender)
{open_file();}
void __fastcall Topening_files::DBGrid1TitleClick(TColumn *Column)
{
  if (DM->AQ_dod->Active)
if((DM->AQ_dod->Sort.Pos(Column->FieldName)>0)&& (DM->AQ_dod->Sort.Pos("ASC") >0))
                                {DM->AQ_dod->Sort = Column->FieldName + " DESC";}
                        else
                                {DM->AQ_dod->Sort = Column->FieldName + " ASC";}
}
void __fastcall Topening_files::N1Click(TObject *Sender)
{delet_topic();}
void __fastcall Topening_files::N2Click(TObject *Sender)
{red_topic();}
void __fastcall Topening_files::N3Click(TObject *Sender)
{delet_file();}
void __fastcall Topening_files::N4Click(TObject *Sender)
{open_file();}
```

Текст, модуля «personalCard»

```
#include <vcl.h>
#pragma hdrstop
#include "personalCard.h"
#include "Unit3.h"
#include <OleServer.hpp>
#include <windows.h>
#include <string.h>
#include <array>
#include <iostream>
#include "Unit6.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
classMy newPointer;
Tcard *card;
int kuId;
String acType;
using namespace std;
__fastcall Tcard::Tcard(TComponent* Owner, int idKu, String tyAcs)
            : TForm(Owner)
{kuId = idKu;acType = tyAcs;}
void __fastcall Tcard::FormActivate(TObject *Sender)
{Button3->Visible=false;
DM->AQlistStudent->Close();}
void __fastcall Tcard::Edit1Change(TObject *Sender)
{
if (acType == "Вид авторизації куратор.")
 {
   newPointer.sqlGroup(true, Edit1, 0, true, kuId);
 }
 else newPointer.sqlGroup(true, Edit1, 0, false, 0);
```

```
}
void __fastcall Tcard::Button1Click(TObject *Sender)
{
 if (acType == "Вид авторизації куратор.")
 {
   newPointer.sqlGroup(false, Edit1, 0, true, kuId);
 }
 else newPointer.sqlGroup(false, Edit1, 0, false, 0);
  Edit1->Text="";
}
void __fastcall Tcard::DBGrid1CellClick(TColumn *Column)
{int idGroup = DBGrid1->DataSource->DataSet->FieldByName("id")->Value;
 newPointer.listStudent(idGroup, 0);
 Button3->Visible=true;
}
void __fastcall Tcard::Button2Click(TObject *Sender)
{
 Button3->Visible=false;
 Edit1->Text="";
 DM->AQlistStudent->Close();
}
void __fastcall Tcard::DBGrid2CellClick(TColumn *Column)
{
  int idStudent = DBGrid2->DataSource->DataSet->FieldByName("id")->Value;
  newPointer.studentInformation(idStudent);
}
void __fastcall Tcard::Button3Click(TObject *Sender)
{
  Variant vApp, vDocs, vDoc, vBookmarks, vSelection, vRange, vBookmark;
  Integer semestr, count, mark, sumBal, m_e, m_g, m_a, numberMark, idGroup, idSubject;
  double pos = 0, kol;
  bool logic;
  String s1, s2, i1, i2, day, month, year;
  array <Integer, 9> i {1,1,1,1,1,1,1,1,1};
  String paths = ExtractFilePath (Application->ExeName) + "\\filesModel";
  ProgressBar1->Position = 0;
  int idStudent = DBGrid2->DataSource->DataSet->FieldByName("id")->Value;
  pos = pos + 1;
  ProgressBar1->Position = pos;
  String path = ExtractFilePath (Application->ExeName) + "\\model\\shDoc.doc" ;
  vApp = CreateOleObject("Word.Application");
  vApp.OlePropertySet("Visible", false);
  vDocs = vApp.OlePropertyGet("Documents");
  vDocs.OleFunction("Add", WideString(path));
  vDoc=vDocs.OleFunction("Item",1);
  vDoc.OleProcedure("Activate");
  DM->AQstudentInfo->First();
  pos = pos + 1;
  ProgressBar1->Position = pos;
  newPointer.changeWord(vApp, vDoc, "familiya", DM->AQstudentInfo->FieldByName("familiya")->AsString);
  newPointer.changeWord(vApp, vDoc, "imya", DM->AQstudentInfo->FieldByName("imya")->AsString);
  newPointer.changeWord(vApp, vDoc, "otchestvo", DM->AQstudentInfo->FieldByName("otchestvo")->AsString);
  s1= DM->AQstudentInfo->FieldByName("data_rojden")->AsString;
  TStringList *stringList = new TStringList;
  stringList->Delimiter = '-';
  stringList->Clear();
  stringList->DelimitedText=s1;
  day = stringList->Strings[2];
  month = stringList->Strings[1];
  year = stringList->Strings[0];
  if (StrToInt(year)>1990)
  {
```

```
   s2 = day + '.' + month + '.' + year;
   newPointer.changeWord(vApp, vDoc, "data_rojden", s2);
  }
  else newPointer.changeWord(vApp, vDoc, "data_rojden", "_____");
newPointer.changeWord(vApp, vDoc, "mesto_rojd",DM->AQstudentInfo->FieldByName("mesto_rojd")->AsString);
  pos = pos + 1;
  ProgressBar1->Position = pos;
  logic = DM->AQstudentInfo->FieldByName("ukranian")->AsBoolean;
  if (logic == true) {
          newPointer.changeWord(vApp, vDoc, "ukranian","Українець");
  }
else newPointer.changeWord(vApp, vDoc, "ukranian","_____");
  s1 = DM->AQstudentInfo->FieldByName("adres_post")->AsString;
  if (s1.Trim().Length() != 0) {
   newPointer.changeWord(vApp, vDoc, "adres_post",DM->AQstudentInfo->FieldByName("adres_post")->AsString);
   newPointer.changeWord(vApp, vDoc, "adres_gorod",DM->AQstudentInfo->FieldByName("adres_gorod")-
>AsString);
   newPointer.changeWord(vApp, vDoc, "adres_adr",DM->AQstudentInfo->FieldByName("adres_adr")->AsString);
  }
  else
  {newPointer.changeWord(vApp, vDoc, "adres_post", "_____");
   newPointer.changeWord(vApp, vDoc, "adres_gorod", "_____");
newPointer.changeWord(vApp, vDoc, "adres_adr", "_____");
  }
   newPointer.changeWord(vApp, vDoc, "inn",DM->AQstudentInfo->FieldByName("inn")->AsString);
   newPointer.changeWord(vApp, vDoc, "shifr",DM->AQstudentInfo->FieldByName("shifr")->AsString);
   newPointer.changeWord(vApp, vDoc, "name",DM->AQstudentInfo->FieldByName("name")->AsString);
   newPointer.changeWord(vApp, vDoc, "otdel",DM->AQstudentInfo->FieldByName("nazvanie")->AsString);
  pos = pos + 1;
  ProgressBar1->Position = pos;
  s1= DM->AQstudentInfo->FieldByName("data")->AsString;
  TStringList *stringDate = new TStringList;
  stringDate->Delimiter = '-';
  stringDate->Clear();
  stringDate->DelimitedText=s1;
  s2 = stringDate->Strings[0];
  if (StrToInt(s2)>2000) {
   newPointer.changeWord(vApp, vDoc, "obrazovanie",DM->AQstudentInfo->FieldByName("vidan")->AsString);
   newPointer.changeWord(vApp, vDoc, "data",s2);
  }
  else{
newPointer.changeWord(vApp, vDoc, "obrazovanie", "_____");
   newPointer.changeWord(vApp, vDoc, "data", "_____");
  }
  pos = pos + 3;
  ProgressBar1->Position = pos;
  sumBal = 0;
  m_e = 0;
  m_g = 0;
  m_a = 0;
  idGroup = DBGrid1->DataSource->DataSet->FieldByName("id")->Value;
   for (int p = 1; p < 9; p++)
   {
           newPointer.disciplinesSearchByGroups(idGroup, false, Edit1, true, p);
                   DM->AQlistStudentNew->First();
                   int  RedCount = DM->AQlistStudentNew->RecordCount;
                   for (int g = 0; g < RedCount; g++)
                   {
                     semestr = p;
                     do
                     {
                   idSubject = DM->AQlistStudentNew->FieldByName("id_subject")->Value;
```

```
                        newPointer.subjectStudentBal(idStudent, idSubject);
                        mark = DM->ADOQueryBal->FieldByName("kol_bal")->AsInteger;
                    if (mark == 0)
                        {
                         DM->AQlistStudentNew->Next();
        RedCount--;
                        }
                    }while(mark == 0);
                        mark = DM->ADOQueryBal->FieldByName("kol_bal")->AsInteger;
                        sumBal = sumBal + 1;
                        if (mark >= 90) { m_e = m_e + 1; }
                        if (mark >= 80) { m_g = m_g + 1; }
                        if (mark >= 70) { m_a = m_a + 1; }
                    i1 = IntToStr(i[semestr]);
                    i2 = IntToStr(semestr);
                    newPointer.changeWord(vApp, vDoc, "subj_"+i1+"_"+i2,DM->AQlistStudentNew-
>FieldByName("name__subject")->AsString);
                    newPointer.changeWord(vApp, vDoc, "h"+i1+"_"+i2,DM->ADOQueryBal-
>FieldByName("kol_bal")->AsString);
                    mark = DM->ADOQueryBal->FieldByName("kol_bal")->AsInteger;
                    numberMark = newPointer.markNumber(mark);
                    newPointer.changeWord(vApp, vDoc, "n"+i1+"_"+i2, IntToStr(numberMark));
                    newPointer.changeWord(vApp, vDoc, "e"+i1+"_"+i2, newPointer.markName(mark));
                    i[semestr] = i[semestr] + 1;
                    DM->AQlistStudentNew->Next();
                        if (pos>60) {
                         kol = 0.2;
                        }
                        else kol = 1;
                     pos = pos + kol;
                     ProgressBar1->Position = pos;
                   }
                        if (pos>60) {
                         kol = 0.5;
                        }
                        else kol = 1;
                     pos = pos + kol;
                     ProgressBar1->Position = pos;
            }
  newPointer.changeWord(vApp, vDoc, "sum_bal", IntToStr(sumBal));
  newPointer.changeWord(vApp, vDoc, "m_e", IntToStr(m_e));
  newPointer.changeWord(vApp, vDoc, "m_g", IntToStr(m_g));
  newPointer.changeWord(vApp, vDoc, "m_a", IntToStr(m_a));
   for (semestr = 1; semestr <= 9; semestr++)
  {for (count = i[semestr]; count <= 20; count++) {
          newPointer.changeWord(vApp, vDoc, "subj_"+IntToStr(count)+"_"+IntToStr(semestr), " ");
          newPointer.changeWord(vApp, vDoc, "h"+IntToStr(count)+"_"+IntToStr(semestr), " ");
          newPointer.changeWord(vApp, vDoc, "n"+IntToStr(count)+"_"+IntToStr(semestr), " ");
          newPointer.changeWord(vApp, vDoc, "e"+IntToStr(count)+"_"+IntToStr(semestr), " ");
         }
  }
         if(pos<200)
           ProgressBar1->Position = 200;
  ShowMessage("Документ сформовано!");
  vApp.OlePropertySet("Visible", true);
  delete stringList;
  delete stringDate;
}
```

Текст, модуля «smotr_student»

```
#include <vcl.h>
#pragma hdrstop
```

```
#include "Unit2.h"
#include "Unit3.h"
#include "smotrstudent.h"
#include "studparen.h"
#include "StrUtils.hpp"
#pragma package(smart_init)
#pragma resource "*.dfm"
Tsmotr_student *smotr_student;
 MyClass mu;
__fastcall Tsmotr_student::Tsmotr_student(TComponent* Owner)
        : TForm(Owner){}
void __fastcall Tsmotr_student::FormActivate(TObject *Sender)
{
  DM->AQ_red_doc->Active=true;
  DM->AQ_grup_stud->Active=true;
  mu.vibor_stud(0,NULL,NULL);
}
void __fastcall Tsmotr_student::BitBtn1Click(TObject *Sender)
{
 mu.vibor_stud(0,NULL,NULL);
 Edit2->Text="";
}
void __fastcall Tsmotr_student::Edit2Change(TObject *Sender)
{ mu.vibor_stud(1,Edit2,NULL);}
void __fastcall Tsmotr_student::BitBtn7Click(TObject *Sender)
{
 DBLookupComboBox3->KeyValue=-1;
 CheckBox2->Checked=false;
 CheckBox3->Checked=false;
 CheckBox4->Checked=false;
 grup="";
 mu.vibor_stud(0,NULL,NULL);
}
void __fastcall Tsmotr_student::DBGrid1TitleClick(TColumn *Column)
{
     //сортування по стовпцю
  if (DM->AQ_red_doc->Active)
                  if ((DM->AQ_red_doc->Sort.Pos(Column->FieldName) > 0) && (DM->AQ_red_doc-
>Sort.Pos("ASC") > 0))
                        {DM->AQ_red_doc->Sort = Column->FieldName + " DESC";}
                  else
                        {DM->AQ_red_doc->Sort = Column->FieldName + " ASC";}
}
void __fastcall Tsmotr_student::BitBtn2Click(TObject *Sender)
{
 mu.vibor_stud(0,NULL,NULL);
 Edit1->Text="";
}
void __fastcall Tsmotr_student::Edit1Change(TObject *Sender)
{mu.vibor_stud(3,Edit1,NULL);}
void __fastcall Tsmotr_student::Edit2KeyPress(TObject *Sender, System::WideChar &Key)
{mu.proverka_Key(Edit2,Key);}
void __fastcall Tsmotr_student::BitBtn3Click(TObject *Sender)
{
if(grup==""&& CheckBox3->Checked==false && CheckBox4->Checked==false && CheckBox2->Checked==false)
 {
          if (MessageDlg("Фільтр не обрано.", mtError, TMsgDlgButtons() << mbYes<<mbNo,0) == mrYes)
           {grup="";}
 }
 else
 {
  if(CheckBox3->Checked==true && CheckBox4->Checked==true)
```

```
   {
   if (MessageDlg("Фільтр не є доречним. Очистити фільтр", mtError, TMsgDlgButtons() << mbYes << mbNo,0) ==
mrYes)
              {CheckBox3->Checked=false;
               CheckBox4->Checked=false;}
   }
  else
  {
  int n;
  n=0;
  DM->AQ_red_doc->Close();
  DM->AQ_red_doc->SQL->Clear();
  DM->AQ_red_doc->SQL->Add("select Student.id_student, Student.PIB, Student.group_name, Student.deducted from
Student where ");
   if(grup!="")
  {
   if(n!=0) DM->AQ_red_doc->SQL->Add("and");
   DM->AQ_red_doc->SQL->Add("Student.group_name=:p");
   DM->AQ_red_doc->Parameters->ParamByName("p")->Value=grup;
   n++;
  }
   if(CheckBox2->Checked==true)
  {
   if(n!=0) DM->AQ_red_doc->SQL->Add("and");
   DM->AQ_red_doc->SQL->Add("Student.deducted=true");
   n++;
  }
            if(CheckBox3->Checked==true&&CheckBox4->Checked==false)
  {
   if(n!=0) DM->AQ_red_doc->SQL->Add("and");
   DM->AQ_red_doc->SQL->Add("Student.stat='ч'");
   n++;
  }
   if(CheckBox4->Checked==true&&CheckBox3->Checked==false)
  {
   if(n!=0) DM->AQ_red_doc->SQL->Add("and");
   DM->AQ_red_doc->SQL->Add("Student.stat='ж'");
   n++;
  }
  DM->AQ_red_doc->Open();
  }
  }
}
void __fastcall Tsmotr_student::DBLookupComboBox3Click(TObject *Sender)
{grup=DBLookupComboBox3->ListSource->DataSet->FieldByName("group_name")->AsString;}
void __fastcall Tsmotr_student::DBGrid1CellClick(TColumn *Column)
{
  // перегляд інформації студента
  DM->AQ_dod->Close();
  DM->AQ_dod->SQL->Clear();
  DM->AQ_dod->SQL->Add("SELECT Student.PIB, Student.Date_Birth, Student.mobile_phone_1,
Student.mobile_phone_2, Student.email, Student.registration_adr, Student.add_of_resider, Student.first_education,
Student.group_name, Student.stat, Student.PHOTO, Student.Note_1 FROM Student where id_student=:p");
  DM->AQ_dod->Parameters->ParamByName("p")->Value=DBGrid1->DataSource->DataSet-
>FieldByName("id_student")->Value;
  DM->AQ_dod->Open();
   Edit3->Text=DM->AQ_dod->FieldByName("PIB")->AsString;
   Edit4->Text=DM->AQ_dod->FieldByName("Date_Birth")->AsString;
   Edit5->Text=DM->AQ_dod->FieldByName("mobile_phone_1")->AsString;
   Edit6->Text=DM->AQ_dod->FieldByName("mobile_phone_2")->AsString;
   Edit7->Text=DM->AQ_dod->FieldByName("email")->AsString;
   Edit8->Text=DM->AQ_dod->FieldByName("registration_adr")->AsString;
```

```
  Edit9->Text=DM->AQ_dod->FieldByName("add_of_resider")->AsString;
  Edit10->Text=DM->AQ_dod->FieldByName("first_education")->AsString;
  Edit11->Text=DM->AQ_dod->FieldByName("group_name")->AsString;
  Edit12->Text=DM->AQ_dod->FieldByName("stat")->AsString;
  Edit13->Text=DM->AQ_dod->FieldByName("Note_1")->AsString;
  String s=DM->AQ_dod->FieldByName("PHOTO")->AsString;
 if(s!="")
  {Image1->Picture->LoadFromFile(ExtractFilePath (Application->ExeName) + "\\PHOTO"+"\\"+ s );
bn=ExtractFilePath (Application->ExeName) + "\\PHOTO"+"\\"+ s;}
 else Image1->Picture=NULL;
  //перегляд інформації батьків
 DM->AQ_List->Close();
 DM->AQ_List->SQL->Clear();
 DM->AQ_List->SQL->Add("SELECT Parents.* FROM Parents where Parents.id_student=:p");
 DM->AQ_List->Parameters->ParamByName("p")->Value=DBGrid1->DataSource->DataSet-
>FieldByName("id_student")->Value;
 DM->AQ_List->Open();
}
void __fastcall Tsmotr_student::BitBtn6Click(TObject *Sender)
{
 stud_paren->Caption="Додавання інформації про студентів та батьків!";
 stud_paren->ShowModal();
}
void __fastcall Tsmotr_student::BitBtn5Click(TObject *Sender)
{
 stud_paren->Caption="Редагування інформації про студентів!";
 String pib_x = Edit3->Text;
 TStringList *stringList = new TStringList;
 stringList->Delimiter = ' ';
 stringList->Clear();
 stringList->DelimitedText=pib_x;
 stud_paren->Edit2->Text=stringList->Strings[0];
 stud_paren->Edit3->Text=stringList->Strings[1];
 stud_paren->Edit4->Text=stringList->Strings[2];
 stud_paren->LabeledEdit4->Text=Edit4->Text;
 stud_paren->MaskEdit1->Text=Edit5->Text;
 stud_paren->MaskEdit2->Text=Edit6->Text;
 stud_paren->Edit1->Text=Edit7->Text;
 stud_paren->LabeledEdit5->Text=Edit8->Text;
 stud_paren->LabeledEdit6->Text=Edit9->Text;
 stud_paren->LabeledEdit7->Text=Edit10->Text;
 stud_paren->LabeledEdit8->Text=Edit11->Text;
 stud_paren->LabeledEdit9->Text=Edit13->Text;
 if(bn!="")
  {stud_paren->DBImage1->Picture->LoadFromFile(bn);}
 else stud_paren->DBImage1->Picture = NULL;
 mu.s_id_now = DBGrid1->DataSource->DataSet->FieldByName("id_student")->Value;
 stud_paren->ShowModal();
}
void __fastcall Tsmotr_student::BitBtn8Click(TObject *Sender)
{
  stud_paren->Caption="Редагування інформації про батьків!";
  DM->AQ_List->First();
  for (int i=0; i < DM->AQ_List->RecordCount; i++)
  {
            if(DBGrid2->DataSource->DataSet->FieldByName("STAT")->Value=="ж")
            {
 String pib_mam = DBGrid2->DataSource->DataSet->FieldByName("PIB_per")->Value;
              TStringList *stringList = new TStringList;
              stringList->Delimiter = ' ';
              stringList->Clear();
              stringList->DelimitedText=pib_mam;
```

```
            stud_paren->Edit5->Text=stringList->Strings[0];
            stud_paren->Edit6->Text=stringList->Strings[1];
            stud_paren->Edit7->Text=stringList->Strings[2];
            stud_paren->MaskEdit3->Text=DBGrid2->DataSource->DataSet->FieldByName("mob_phone")->Value;
            stud_paren->LabeledEdit13->Text=DBGrid2->DataSource->DataSet->FieldByName("add_resider")->Value;
            stud_paren->LabeledEdit14->Text=DBGrid2->DataSource->DataSet->FieldByName("work_place")->Value;
            stud_paren->LabeledEdit15->Text=DBGrid2->DataSource->DataSet->FieldByName("Positions")->Value;
            stud_paren->LabeledEdit16->Text=DBGrid2->DataSource->DataSet->FieldByName("Note_2")->Value;
            mu.id_mam = DBGrid2->DataSource->DataSet->FieldByName("id_per")->Value;
    DM->AQ_List->Next();
          }
          else
          {
String pib_pap = DBGrid2->DataSource->DataSet->FieldByName("PIB_per")->Value;
            TStringList *stringList = new TStringList;
            stringList->Delimiter = ' ';
            stringList->Clear();
            stringList->DelimitedText=pib_pap;
            stud_paren->Edit8->Text=stringList->Strings[0];
            stud_paren->Edit9->Text=stringList->Strings[1];
            stud_paren->Edit10->Text=stringList->Strings[2];
            stud_paren->MaskEdit4->Text=DBGrid2->DataSource->DataSet->FieldByName("mob_phone")->Value;
            stud_paren->LabeledEdit20->Text=DBGrid2->DataSource->DataSet->FieldByName("add_resider")->Value;
            stud_paren->LabeledEdit21->Text=DBGrid2->DataSource->DataSet->FieldByName("work_place")->Value;
            stud_paren->LabeledEdit22->Text=DBGrid2->DataSource->DataSet->FieldByName("Positions")->Value;
stud_paren->LabeledEdit23->Text=DBGrid2->DataSource->DataSet->FieldByName("Note_2")->Value;
            mu.id_pap = DBGrid2->DataSource->DataSet->FieldByName("id_per")->Value;
    DM->AQ_List->Next();
          }
 }
  stud_paren->ShowModal();
}
```

Текст, модуля «synchronization»

```
#include <vcl.h>
#pragma hdrstop
#include "synchronization.h"
#include "Unit3.h"
#include "Unit1.h"
#include "Unit2.h"
#include "Unit4.h"
#include "Unit6.h"
#include <algorithm>
#include <iostream>
#include <windows.h>
#include <string.h>
#pragma package(smart_init)
#pragma resource "*.dfm"
Ttiming *timing;
classMy ukaz;
int kuratorsId;
String accessTypes;
__fastcall Ttiming::Ttiming(TComponent* Owner, int idKurs, String typeAcs)
          : TForm(Owner)
{kuratorsId =  idKurs;accessTypes = typeAcs;}
void __fastcall Ttiming::RadioButton1Click(TObject *Sender)
{
if (accessTypes == "Вид авторизації куратор.")
 {
  ukaz.sqlGroup(false, NULL, 1, true, kuratorsId);
 }
```

```
 else ukaz.sqlGroup(false, NULL, 1, false, 0);
}
void __fastcall Ttiming::RadioButton2Click(TObject *Sender)
{
if (accessTypes == "Вид авторизації куратор.")
 {
   ukaz.sqlGroup(false, NULL, 2, true, kuratorsId);
 }
 else ukaz.sqlGroup(false, NULL, 2, false, 0);
}
void __fastcall Ttiming::RadioButton3Click(TObject *Sender)
{
if (accessTypes == "Вид авторизації куратор.")
 {
   ukaz.sqlGroup(false, NULL, 3, true, kuratorsId);
 }
 else ukaz.sqlGroup(false, NULL, 3, false, 0);
}
void lokalAdvantage()
{
 DM->ADOQuerySynchronization1->Close();
 DM->ADOQuerySynchronization1->SQL->Clear();
 DM->ADOQuerySynchronization1->SQL->Add("SELECT Student.PIB, Privilege.name_privilege");
 DM->ADOQuerySynchronization1->SQL->Add("FROM Student, Privilege, advantage");
 DM->ADOQuerySynchronization1->SQL->Add("WHERE Student.id_student = advantage.id_student");
 DM->ADOQuerySynchronization1->SQL->Add("and Privilege.id_privilege = advantage.id_privilege");
 DM->ADOQuerySynchronization1->SQL->Add("and Student.group_name =:namesGroup");
 DM->ADOQuerySynchronization1->Parameters->ParamByName("namesGroup")->Value = timing-
>DBLookupComboBox1->ListSource->DataSet->FieldByName("name")->AsString;
 DM->ADOQuerySynchronization1->Open();
}
void listStudent()
{
  DM->ADOQueryPosrednik->Close();
  DM->ADOQueryPosrednik->SQL->Clear();
  DM->ADOQueryPosrednik->SQL->Add("SELECT Student.*");
  DM->ADOQueryPosrednik->SQL->Add("FROM Student");
  DM->ADOQueryPosrednik->SQL->Add("where Student.group_name =:nameGroup order by PIB");
  DM->ADOQueryPosrednik->Parameters->ParamByName("nameGroup")->Value = timing->DBLookupComboBox1-
>ListSource->DataSet->FieldByName("name")->AsString;
  DM->ADOQueryPosrednik->Open();
}
void poiskAdvantageDekanat(String fam, String im, String otch)
{
 DM->ADOQuerySynchronization2->Close();
 DM->ADOQuerySynchronization2->SQL->Clear();
 DM->ADOQuerySynchronization2->SQL->Add("select [dekanat_new].[dbo].[ins_abit].familiya,");
 DM->ADOQuerySynchronization2->SQL-
>Add("[dekanat_new].[dbo].[ins_abit].imya,[dekanat_new].[dbo].[ins_abit].otchestvo, ");
 DM->ADOQuerySynchronization2->SQL->Add("[dekanat_new].[dbo].[ins_info_lgoty].name");
 DM->ADOQuerySynchronization2->SQL->Add("from [dekanat_new].[dbo].[ins_info_lgoty],
[dekanat_new].[dbo].[ins_abit], [dekanat_new].[dbo].[ins_lgota]");
 DM->ADOQuerySynchronization2->SQL->Add("where [dekanat_new].[dbo].[ins_lgota].id_abit =
[dekanat_new].[dbo].[ins_abit].id");
 DM->ADOQuerySynchronization2->SQL->Add("and [dekanat_new].[dbo].[ins_lgota].id_typelgota =
[dekanat_new].[dbo].[ins_info_lgoty].id");
 DM->ADOQuerySynchronization2->SQL->Add("and [dekanat_new].[dbo].[ins_abit].familiya =:fams");
 DM->ADOQuerySynchronization2->SQL->Add("and [dekanat_new].[dbo].[ins_abit].imya =:ims");
 DM->ADOQuerySynchronization2->SQL->Add("and [dekanat_new].[dbo].[ins_abit].otchestvo =:otchs");
 DM->ADOQuerySynchronization2->Parameters->ParamByName("fams")->Value = fam;
 DM->ADOQuerySynchronization2->Parameters->ParamByName("ims")->Value =  im;
 DM->ADOQuerySynchronization2->Parameters->ParamByName("otchs")->Value = otch;
```

```
 DM->ADOQuerySynchronization2->Open();
}
void lokalPrivilege()
{
 DM->ADOQuerySynchronization1->Close();
 DM->ADOQuerySynchronization1->SQL->Clear();
 DM->ADOQuerySynchronization1->SQL->Add("SELECT Privilege.*");
 DM->ADOQuerySynchronization1->SQL->Add("FROM Privilege");
 DM->ADOQuerySynchronization1->Open();
}
void dekanatPrivilege()
{
 DM->ADOQuerySynchronization2->Close();
 DM->ADOQuerySynchronization2->SQL->Clear();
 DM->ADOQuerySynchronization2->SQL->Add("select [dekanat_new].[dbo].[ins_info_lgoty].*");
 DM->ADOQuerySynchronization2->SQL->Add("from [dekanat_new].[dbo].[ins_info_lgoty]");
 DM->ADOQuerySynchronization2->Open();
}
void lokalSubject()
{
 DM->ADOQuerySynchronization1->Close();
 DM->ADOQuerySynchronization1->SQL->Clear();
 DM->ADOQuerySynchronization1->SQL->Add("SELECT Subject.*");
 DM->ADOQuerySynchronization1->SQL->Add("FROM Subject");
 DM->ADOQuerySynchronization1->Open();
}
void dekanatSubject()
{
 DM->ADOQuerySynchronization2->Close();
 DM->ADOQuerySynchronization2->SQL->Clear();
 DM->ADOQuerySynchronization2->SQL->Add("SELECT [dekanat_new].[dbo].[dek_employ].*");
 DM->ADOQuerySynchronization2->SQL->Add("FROM [dekanat_new].[dbo].[dek_employ]");
 DM->ADOQuerySynchronization2->SQL->Add("where [dekanat_new].[dbo].[dek_employ].id_group =:grup");
 DM->ADOQuerySynchronization2->Parameters->ParamByName("grup")->Value = timing->DBLookupComboBox1-
>ListSource->DataSet->FieldByName("id")->Value;
 DM->ADOQuerySynchronization2->Open();
}
void lokalGer()
{
  DM->ADOQuerySynchronization1->Close();
  DM->ADOQuerySynchronization1->SQL->Clear();
  DM->ADOQuerySynchronization1->SQL->Add("SELECT Student.*");
  DM->ADOQuerySynchronization1->SQL->Add("FROM Student");
  DM->ADOQuerySynchronization1->SQL->Add("where Student.group_name =:nameGroup order by PIB");
  DM->ADOQuerySynchronization1->Parameters->ParamByName("nameGroup")->Value = timing-
>DBLookupComboBox1->ListSource->DataSet->FieldByName("name")->AsString;
  DM->ADOQuerySynchronization1->Open();
}
void dekanatGer(int kurs)
 {
  DM->ADOQuerySynchronization2->Close();
  DM->ADOQuerySynchronization2->SQL->Clear();
  DM->ADOQuerySynchronization2->SQL->Add("SELECT nameGroup.name, a.familiya, a.imya, a.otchestvo,
a.data_rojden, a.stat,a.adres_gorod,a.adres_adr,");
  DM->ADOQuerySynchronization2->SQL->Add("a.mesto_rojd, educ.vidan, s.gurtoj,");
  DM->ADOQuerySynchronization2->SQL->Add("a.contact_tel, a.contact_telm, a.contact_email, a.roditel_otec,
a.roditel_otect, a.roditel_mat,");
  DM->ADOQuerySynchronization2->SQL->Add("a.roditel_mett");
  DM->ADOQuerySynchronization2->SQL->Add("FROM [dekanat_new].[dbo].[ins_abit] a");
  DM->ADOQuerySynchronization2->SQL->Add("left join [dekanat_new].[dbo].[ins_educ] educ");
  DM->ADOQuerySynchronization2->SQL->Add("on (educ.id_abit=a.id)");
  DM->ADOQuerySynchronization2->SQL->Add("left join [dekanat_new].[dbo].[dek_student] s");
```

```
 DM->ADOQuerySynchronization2->SQL->Add("on (s.id_abit=educ.id_abit)");
 DM->ADOQuerySynchronization2->SQL->Add("left join [dekanat_new].[dbo].[dek_student_stay] ss");
 DM->ADOQuerySynchronization2->SQL->Add("on (ss.id_student=s.id)");
 DM->ADOQuerySynchronization2->SQL->Add("left join [dekanat_new].[dbo].[dek_group] nameGroup");
 DM->ADOQuerySynchronization2->SQL->Add("on(ss.id_group = nameGroup.id)");
 if (kurs == 1)
 {
        DM->ADOQuerySynchronization2->SQL->Add("where (ss.id_group=:idGroup) and (ss.id_kurs = 1) and
(ss.id_stay = 6) order by familiya");
  }
  if (kurs == 2)
 {
        DM->ADOQuerySynchronization2->SQL->Add("where (ss.id_group=:idGroup) and (ss.id_kurs = 2) and
(ss.id_stay = 6) order by familiya");
  }
  if (kurs == 3)
 {
        DM->ADOQuerySynchronization2->SQL->Add("where (ss.id_group=:idGroup) and (ss.id_kurs = 3) and
(ss.id_stay = 6) order by familiya");
  }
  if (kurs == 4)
 {
        DM->ADOQuerySynchronization2->SQL->Add("where (ss.id_group=:idGroup) and (ss.id_kurs = 4) and
(ss.id_stay = 6) order by familiya");
  }
 DM->ADOQuerySynchronization2->Parameters->ParamByName("idGroup")->Value = timing-
>DBLookupComboBox1->ListSource->DataSet->FieldByName("id")->Value;
 DM->ADOQuerySynchronization2->Open();
}
void __fastcall Ttiming::DBLookupComboBox1Click(TObject *Sender)
{
 dekanatGer(1);
 lokalGer();
}
void __fastcall Ttiming::BitBtn1Click(TObject *Sender)
{
  String fio, resider, statn, statnM, statnP, famil, imy, otches;
  bool flag, choice, otv, metr;
  int raz;
if (timing->DBLookupComboBox1->KeyValue!=-1 && timing->RadioButton5->Checked==true)
        {choice = true;
                otv = false; }
        else{
if(timing->DBLookupComboBox1->KeyValue!=-1 && timing->RadioButton4->Checked==true)
                {
                        choice = false;
          otv = false;
            if (DM->ADOQuerySynchronization1->RecordCount==0)
                                {
                                 ShowMessage("Локальна база порожня!!!");
                                }
                }
                else otv = true;
         }
        if (otv == true)
        {
         ShowMessage("Оберіть всі параметри!!!");
        }
        else{
                        raz = 1;
                                for (int h = 0; h < 4; h++)
                                {
```

```
                                                DM->ADOQuerySynchronization2->First();
for (int i = 0; i < DM->ADOQuerySynchronization2->RecordCount; i++)
                                                {
                                                  lokalGer();
fio = DM->ADOQuerySynchronization2->FieldByName("familiya")->AsString;
fio = fio + " " + DM->ADOQuerySynchronization2->FieldByName("imya")->AsString;
   fio = fio + " " + DM->ADOQuerySynchronization2->FieldByName("otchestvo")->AsString;
   resider = DM->ADOQuerySynchronization2->FieldByName("adres_gorod")->AsString;
resider = resider + " " + DM->ADOQuerySynchronization2->FieldByName("adres_adr")->AsString;

if(DM->ADOQuerySynchronization2->FieldByName("stat")->AsString == "True")
                                                {
                                                 statn="ч";
                                                }
                                                 else statn="ж";
if (DM->ADOQuerySynchronization1->RecordCount==0)
                                                 {
                                                        flag=false;
                                                 }
                                                 else
                                                        {
                                                          DM->ADOQuerySynchronization1->First();
for(int gh = 0; gh < DM->ADOQuerySynchronization1->RecordCount; gh++)
                                                                {
if (CompareText(DM->ADOQuerySynchronization1->FieldByName("group_name")->AsString,DM-
>ADOQuerySynchronization2->FieldByName("name")->AsString)==0 && CompareText(DM-
>ADOQuerySynchronization1->FieldByName("PIB")->AsString,fio)== 0)
{flag = true; break; } else flag = false;
                                        DM->ADOQuerySynchronization1->Next();
                                                                }
                                                                if (flag == true)
                                                                {
        DM->AQ_dod->Close();
        DM->AQ_dod->SQL->Clear();
DM->AQ_dod->SQL->Add("SELECT Student.id_student FROM Student where PIB=:pz");
DM->AQ_dod->Parameters->ParamByName("pz")->Value=fio;
DM->AQ_dod->Open();}
}if(flag == true && choice == false) metr = true;
                        else metr = false;
if (metr == false) {
String stroka = DM->ADOQuerySynchronization2->FieldByName("contact_tel")->AsString;
                                                                String newPho = "";
                                                                if (stroka == "" || stroka == "-")
                                                                { newPho = " ";}
else{
for (int a = 1; a < stroka.Length()+1; a++)
                                                                {
                if(a == 5) newPho = newPho + "-";
                if(a == 7) newPho = newPho + "-";
                                                                        if (a > 1)
                                                                        {
                                                                         newPho = newPho + stroka[a];
                                                                        }
                                                                }
}String strokas = DM->ADOQuerySynchronization2->FieldByName("contact_telm")->AsString;
                                                                String newPh = "";
                                                                if (strokas == ""||strokas == "-")
                                                                {newPh = " ";}
                                                                else
                                {
                                                                        for (int ad = 1; ad < strokas.Length()+1; ad++)
                                                                        {
```

```
        if(ad == 5) newPh = newPh + "-";
        if(ad == 7) newPh = newPh + "-";
                                                                    if (ad > 1)
                                                                    {
                                                                     newPh = newPh + strokas[ad];
                                                                    }
                                                            }
                                                        }
                                                DM->ADOQueryInput->Close();
                                                        DM->ADOQueryInput->SQL->Clear();
                                                        if (flag == false)
                                                        {
DM->ADOQueryInput->SQL->Text="INSERT INTO Student (PIB, Date_Birth, mobile_phone_1, mobile_phone_2,
email, registration_adr, add_of_resider, first_education, group_name, stat) VALUES(:PIB1, :Date_Birth1,
:mobile_phone_1_1, :mobile_phone_2_2, :email1, :registration_adr1, :add_of_resider1, :first_education1,
:group_name1, :stat1)";
                                                        }
                                                        if(flag == true && choice == true)
                                                        {
DM->ADOQueryInput->SQL->Text="UPDATE Student SET PIB=:PIB1, Date_Birth=:Date_Birth1,
mobile_phone_1=:mobile_phone_1_1, mobile_phone_2=:mobile_phone_2_2, email=:email1,
registration_adr=:registration_adr1, add_of_resider=:add_of_resider1, first_education=:first_education1,
group_name=:group_name1, stat=:stat1 where id_student=:idStud";
DM->ADOQueryInput->Parameters->ParamByName("idStud")->Value=DM->AQ_dod->FieldByName("id_student")-
>Value;
                                                        }
 DM->ADOQueryInput->Parameters->ParamByName("PIB1")->Value=fio;
 DM->ADOQueryInput->Parameters->ParamByName("Date_Birth1")->Value=DM->ADOQuerySynchronization2-
>FieldByName("data_rojden")->AsString;
 DM->ADOQueryInput->Parameters->ParamByName("mobile_phone_1_1")->Value=newPho;
 DM->ADOQueryInput->Parameters->ParamByName("mobile_phone_2_2")->Value=newPh;
 DM->ADOQueryInput->Parameters->ParamByName("email1")->Value=DM->ADOQuerySynchronization2-
>FieldByName("contact_email")->AsString;
 DM->ADOQueryInput->Parameters->ParamByName("registration_adr1")->Value=DM-
>ADOQuerySynchronization2->FieldByName("mesto_rojd")->AsString;
DM->ADOQueryInput->Parameters->ParamByName("add_of_resider1")->Value=resider;
DM->ADOQueryInput->Parameters->ParamByName("first_education1")->Value=DM->ADOQuerySynchronization2-
>FieldByName("vidan")->AsString;
DM->ADOQueryInput->Parameters->ParamByName("group_name1")->Value=DM->ADOQuerySynchronization2-
>FieldByName("name")->AsString;
DM->ADOQueryInput->Parameters->ParamByName("stat1")->Value=statn;
                                                        DM->ADOQueryInput->ExecSQL();
 if (flag == false) {
DM->AQ_dod->Close();
DM->AQ_dod->SQL->Clear();
DM->AQ_dod->SQL->Add("SELECT id_student FROM Student where PIB=:pz");
DM->AQ_dod->Parameters->ParamByName("pz")->Value=fio;
                                        DM->AQ_dod->Open();}
 String str = DM->ADOQuerySynchronization2->FieldByName("roditel_mett")->AsString;
                                                        String newPhone = "";
                                                        if (str == ""||str == "-")
                                                        {newPhone = " ";}
                                                        else
                                  for (int as = 1; as < str.Length()+1; as++)
                                                                {
                                 if(as == 5) newPhone = newPhone + "-";
                                 if(as == 7) newPhone = newPhone + "-";
                                                                if (as > 1)
                                                                {
                                                                 newPhone = newPhone + str[as];
                                                                }
                                                         }
```

```
                                                        }
                        DM->ADOQueryInput->Close();
                                        DM->ADOQueryInput->SQL->Clear();
                                                        statnM="ж";
                                                                if (flag == false)
                                                                {
          DM->ADOQueryInput->SQL->Text="INSERT INTO Parents (id_student, PIB_per, mob_phone, STAT,
add_resider, work_place, Positions, Note_2) VALUES(:idd, :PIB_per1, :mob_phone1, :STAT1, :addResiders,
:workPlaces, :Poss, :Note2s)";
   DM->ADOQueryInput->Parameters->ParamByName("addResiders")->Value = " ";
DM->ADOQueryInput->Parameters->ParamByName("workPlaces")->Value = " ";
DM->ADOQueryInput->Parameters->ParamByName("Poss")->Value = " ";
->ADOQueryInput->Parameters->ParamByName("Note2s")->Value = " ";
                                                                }
                                                                if(flag == true && choice == true)
                                                                {
 DM->ADOQueryInput->SQL->Text="UPDATE Parents SET PIB_per=:PIB_per1, mob_phone=:mob_phone1,
STAT=:STAT1  where id_student=:idd and STAT=:STAT1";
                                                                }
            DM->ADOQueryInput->Parameters->ParamByName("idd")->Value = DM->AQ_dod-
>FieldByName("id_student")->Value;
            DM->ADOQueryInput->Parameters->ParamByName("PIB_per1")->Value = DM-
>ADOQuerySynchronization2->FieldByName("roditel_mat")->AsString;
  DM->ADOQueryInput->Parameters->ParamByName("mob_phone1")->Value = newPhone;
  DM->ADOQueryInput->Parameters->ParamByName("STAT1")->Value = statnM;
  DM->ADOQueryInput->ExecSQL();
String strok = DM->ADOQuerySynchronization2->FieldByName("roditel_otect")->AsString;
                                                        String newPhons = "";
                                                        if (strok == "" || strok == "-")
                                                        {
                                                                newPhons = " ";
                                                        }
                                                        else
                                                                {
                                                for (int asa = 1; asa < strok.Length()+1; asa++)
                                                                {
                                                        if(asa == 5) newPhons = newPhons + "-";
                                        if(asa == 7) newPhons = newPhons + "-";
                                                                        if (asa > 1)
                                                                        {
                                         newPhons = newPhons + strok[asa];
                                                                         }
                                                                }
                                                        }
                        DM->ADOQueryInput->Close();
                                                        DM->ADOQueryInput->SQL->Clear();
                                                        statnP="ч";
                                                                if (flag == false)
                                                                 {
                                                                   DM->ADOQueryInput->SQL-
>Text="INSERT INTO Parents (id_student, PIB_per, mob_phone, STAT, add_resider, work_place, Positions, Note_2)
VALUES(:papid, :PIB_per2, :mob_phone2, :STAT2, :addResider, :workPlace, :Pos, :Note2)";
            DM->ADOQueryInput->Parameters->ParamByName("addResider")->Value = " ";
   DM->ADOQueryInput->Parameters->ParamByName("workPlace")->Value = " ";
            DM->ADOQueryInput->Parameters->ParamByName("Pos")->Value = " ";
            DM->ADOQueryInput->Parameters->ParamByName("Note2")->Value = " ";
                                                                }
                                                                if(flag == true && choice == true)
                                                                {
                                                        DM->ADOQueryInput->SQL->Text="UPDATE
Parents SET PIB_per=:PIB_per2, mob_phone=:mob_phone2, STAT=:STAT2 where id_student=:papid and
STAT=:STAT2";
```

```
                                                                }
  DM->ADOQueryInput->Parameters->ParamByName("papid")->Value =  DM->AQ_dod-
>FieldByName("id_student")->Value;
  DM->ADOQueryInput->Parameters->ParamByName("PIB_per2")->Value = DM->ADOQuerySynchronization2-
>FieldByName("roditel_otec")->AsString;
  DM->ADOQueryInput->Parameters->ParamByName("mob_phone2")->Value = newPhons;
DM->ADOQueryInput->Parameters->ParamByName("STAT2")->Value = statnP;
  DM->ADOQueryInput->ExecSQL();
if(DM->ADOQuerySynchronization2->FieldByName("gurtoj")->AsString == "True" && flag == false)
                                                                {
                                                                 DM->AQ_List->Close();
                                                                 DM->AQ_List->SQL->Clear();
                                                                 DM->AQ_List->SQL->Text="INSERT INTO
Hostel (id_student, №_room, side_block) VALUES(:p, 0, 0)";
          DM->AQ_List->Parameters->ParamByName("p")->Value = DM->AQ_dod->FieldByName("id_student")-
>Value;
                                                                 DM->AQ_List->ExecSQL();
                                                                }
                                                  DM->ADOQuerySynchronization2->Next();
                                                  }
                                             else DM->ADOQuerySynchronization2->Next();
                                             }
                                             raz++;
                                              if (raz == 2)
                                                        {
                                                         dekanatGer(2);
                                                         if (DM->ADOQuerySynchronization2->RecordCount==0) break;
                                                         else continue;
                                                        }
                                         if (raz == 3)
                                                         {
                                                        dekanatGer(3);
if (DM->ADOQuerySynchronization2->RecordCount==0) break;
                                                        else continue;
                                                         }
                                                         if (raz == 4)
                                                         {
                                                        dekanatGer(4);
if (DM->ADOQuerySynchronization2->RecordCount==0) break;
                                                        else continue;
                                                         }
                                        }
                                        dekanatSubject();
for (int i = 0; i < DM->ADOQuerySynchronization2->RecordCount; i++)
                                        {
                                          lokalSubject();
                                          if (DM->ADOQuerySynchronization1->RecordCount==0)
                                                        {
                                                                flag = false;
                                                        }
                                                        else
                                                                {
                                                                  DM->ADOQuerySynchronization1->First();
for(int gh = 0; gh < DM->ADOQuerySynchronization1->RecordCount; gh++)
                                                                                {
if (CompareText(DM->ADOQuerySynchronization1->FieldByName("name_discip")->AsString,DM-
>ADOQuerySynchronization2->FieldByName("name")->AsString)==0)
                                                                                                {
                                                                                flag = true;
                                                                                                break;
                                                                                }
                                                                        else flag = false;
```

```
                                                DM->ADOQuerySynchronization1->Next();
                                                        }
                                                if (flag == true)
                                                        {
                                                                DM->AQ_dod->Close();
                                                                DM->AQ_dod->SQL->Clear();
DM->AQ_dod->SQL->Add("SELECT Subject.id_discip FROM Subject where name_discip=:nameN");
DM->AQ_dod->Parameters->ParamByName("nameN")->Value=DM->ADOQuerySynchronization2->FieldByName("name")->AsString;
                                                                DM->AQ_dod->Open();
                                                        }
                                                }
                                                if(flag == true && choice == false) goto metkas;
                DM->ADOQueryInput->Close();
                                                DM->ADOQueryInput->SQL->Clear();
                if (flag == false)
                  {
DM->ADOQueryInput->SQL->Text="INSERT INTO Subject (name_discip) VALUES(:name_disp)";
                                                }
                                                if(flag == true && choice == true)
                                                        {
                                                                DM->ADOQueryInput->SQL->Text="UPDATE Subject SET name_discip=:name_disp where id_discip=:p";
                                                                DM->ADOQueryInput->Parameters->ParamByName("p")->Value=DM->AQ_dod->FieldByName("id_discip")->Value;
                                                }
                                        DM->ADOQueryInput->Parameters->ParamByName("name_disp")->Value=DM->ADOQuerySynchronization2->FieldByName("name")->AsString;
                        DM->ADOQueryInput->ExecSQL();
                                                metkas:
                                                DM->ADOQuerySynchronization2->Next();
                                        }
                                        dekanatPrivilege();
                                for (int i = 0; i < DM->ADOQuerySynchronization2->RecordCount; i++)
                                        {
                                         lokalPrivilege();
                                         if (DM->ADOQuerySynchronization1->RecordCount==0)
                                                {
                                                flag = false;
                                                 }
                                                 else
                                                        {
                                                DM->ADOQuerySynchronization1->First();
                                                         for(int gh = 0; gh < DM->ADOQuerySynchronization1->RecordCount; gh++)
                                                                {
                                                if (CompareText(DM->ADOQuerySynchronization1->FieldByName("name_privilege")->AsString,DM->ADOQuerySynchronization2->FieldByName("name")->AsString)==0)
                                                                                                {
                                                                flag = true;
                                                                                                        break;
                                                                                                }
                                                                                        else flag = false;
                                                DM->ADOQuerySynchronization1->Next();
                                                                }
                                                                 if (flag == true)
                                                                 {
        DM->AQ_dod->Close();
                        DM->AQ_dod->SQL->Clear();
DM->AQ_dod->SQL->Add("SELECT Privilege.id_privilege FROM Privilege where name_privilege=:nameV");
DM->AQ_dod->Parameters->ParamByName("nameV")->Value=DM->ADOQuerySynchronization2-
```

```
>FieldByName("name")->AsString;
                                                DM->AQ_dod->Open();
                                                                        }
                                                                }
if(flag == true && choice == false) goto metks;
                        DM->ADOQueryInput->Close();
                                                                 DM->ADOQueryInput->SQL->Clear();
                                                                if (flag == false)
                                                                        {
DM->ADOQueryInput->SQL->Text="INSERT INTO Privilege (name_privilege) VALUES(:name_priv)";
                                                                        }
                                                                if(flag == true && choice == true)
                                                                        {
DM->ADOQueryInput->SQL->Text="UPDATE Privilege SET name_privilege=:name_priv where id_privilege=:isp";
DM->ADOQueryInput->Parameters->ParamByName("isp")->Value=DM->AQ_dod->FieldByName("id_privilege")-
>Value;
                                                                        }
                                DM->ADOQueryInput->Parameters->ParamByName("name_priv")->Value=DM-
>ADOQuerySynchronization2->FieldByName("name")->AsString;
                                DM->ADOQueryInput->ExecSQL();
                                                        metks:
                                                        DM->ADOQuerySynchronization2->Next();
                                                }
                                                        listStudent();
                                                for (int i = 0; i < DM->ADOQueryPosrednik->RecordCount; i++)
                                                {
                                                 lokalAdvantage();
                        String pib_x = DM->ADOQueryPosrednik->FieldByName("PIB")->AsString;
                                                 TStringList *stringList = new TStringList;
                                                 stringList->Delimiter = ' ';
                                                 stringList->Clear();
                                                 stringList->DelimitedText=pib_x;
                                                 famil=stringList->Strings[0];
                                                 imy=stringList->Strings[1];
                                                 otches=stringList->Strings[2];
                                                 if (DM->ADOQuerySynchronization1->RecordCount==0)
                                                                {
                                                                        flag = false;
poiskAdvantageDekanat(famil, imy, otches);
  if (DM->ADOQuerySynchronization2->RecordCount==0)
                                                                        {goto metksin; }
                                                                }
                                                                else
                                                                {
                        poiskAdvantageDekanat(famil, imy, otches);
if (DM->ADOQuerySynchronization2->RecordCount == 0)
{ goto metksin;
                                }
                                                                 else
                                                                 {
fio = DM->ADOQuerySynchronization2->FieldByName("familiya")->AsString;
fio = fio + " " + DM->ADOQuerySynchronization2->FieldByName("imya")->AsString;
fio = fio + " " + DM->ADOQuerySynchronization2->FieldByName("otchestvo")->AsString;
DM->ADOQuerySynchronization1->First();
for(int gh = 0; gh < DM->ADOQuerySynchronization1->RecordCount; gh++)
                                {
if (CompareText(DM->ADOQuerySynchronization1->FieldByName("name_privilege")->AsString,DM-
>ADOQuerySynchronization2->FieldByName("name")->AsString)==0&& CompareText(DM-
>ADOQuerySynchronization1->FieldByName("PIB")->AsString,fio)== 0)
                                                                                        {
                                                                                flag = true;
                                                                                                break;
```

```
                                                                    }
                                                                else flag = false;
                                                        DM->ADOQuerySynchronization1->Next();
                                                                }
                                                        }
                                                }
                                        if(flag == true && choice == false) goto metksin;
                                                        DM->AQ_dod->Close();
                                                        DM->AQ_dod->SQL->Clear();
                                                        DM->AQ_dod->SQL->Add("SELECT
Privilege.id_privilege FROM Privilege where name_privilege=:nameV");
                                DM->AQ_dod->Parameters->ParamByName("nameV")->Value=DM-
>ADOQuerySynchronization2->FieldByName("name")->AsString;
                                                        DM->AQ_dod->Open();
                                                        DM->ADOQueryInput->Close();
                                                        DM->ADOQueryInput->SQL->Clear();
                                                if (flag == false)
                                                        {
                                                        DM->ADOQueryInput->SQL->Text="INSERT INTO
advantage (id_privilege, id_student) VALUES(:id_priv, :id_stud)";
                                                        }
                                                if(flag == true && choice == true)
                                                        {
                                                                DM->ADOQueryInput->SQL->Text="UPDATE
advantage SET id_privilege=:id_priv where id_student=:id_stud";
                                                        }
                                                        DM->ADOQueryInput->Parameters-
>ParamByName("id_priv")->Value = DM->AQ_dod->FieldByName("id_privilege")->Value;
                                                        DM->ADOQueryInput->Parameters-
>ParamByName("id_stud")->Value = DM->ADOQueryPosrednik->FieldByName("id_student")->Value;
                                                        DM->ADOQueryInput->ExecSQL();
                                                metksin:
                                                DM->ADOQueryPosrednik->Next();
                                        }
                                        ShowMessage("Синхронізацію закінчено!!!");
                }
}
```

Текст, модуля «Access»

```
#include <vcl.h>
#pragma hdrstop
#include "Access.h"
#include "MyAccess.h"
#include "Unit2.h"
#include "Unit1.h"
#include "Unit3.h"
#include "Unit4.h"
#include "Unit6.h"
#include "deducted_1.h"
#include "inputdocument.h"
#include "educationform.h"
#include <Vcl.Controls.hpp>
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>
#include <Vcl.ExtCtrls.hpp>
#include <Vcl.Imaging.jpeg.hpp>
#include <Vcl.Buttons.hpp>
#include <Vcl.ComCtrls.hpp>
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm8 *Form8;
MyAccess access;
```

```
__fastcall TForm8::TForm8(TComponent* Owner)
        : TForm(Owner){}
void clin()
{
  Form8->Edit1->Text = "";
  Form8->Edit2->Text = "";
  Form8->CheckBox1->Checked = false;
  Form8->CheckBox2->Checked = false;
}
void __fastcall TForm8::Button2Click(TObject *Sender)
{  clin();}
void __fastcall TForm8::Button1Click(TObject *Sender)
{
 if(ProgressBar1->Position == 0)
 {
  if (Edit1->Text != "" && Edit2->Text != "" && (CheckBox1->Checked == true || CheckBox2->Checked == true))
  {
   if (CheckBox1->Checked == true && CheckBox2->Checked == true)
   {
            ShowMessage("Оберіть один вид доступу!!!");
   }
   else
   {
          access.setAccess(Edit1,Edit2);
          if (access.getAccess() == true)
          {if (CheckBox1->Checked == true)
            {access.setTypeofAccess(true, false); }
            else access.setTypeofAccess(false, true);
          }
          else ShowMessage("Помилка при авторизації!!!");
          if (access.getAccess() == true)
          {ProgressBar1->Position = 100;
            String status = access.getTypeofAccess() + " " + access.getPIB();
            Form8->StatusBar1->Panels->Items[0]->Text = status;
            MainMENU = new TMainMENU(this, true, access.getidKurator(), access.getTypeofAccess());
            delete MainMENU;
          }
          }
  }
  else ShowMessage("Введіть всі дані!!!");
 }else ShowMessage("Ви вже авторизовані в базі даних Деканат!!!");
}
void __fastcall TForm8::FormActivate(TObject *Sender)
{
  if (access.getAccess() == true)
  {ProgressBar1->Position = 100;}
}
void __fastcall TForm8::Button3Click(TObject *Sender)
{
  if (access.getAccess() == true)
  {ProgressBar1->Position = 0;access.denyAccess();
    String status = access.getTypeofAccess() + " " + access.getPIB();
          Form8->StatusBar1->Panels->Items[0]->Text = status;
          MainMENU = new TMainMENU(this, false, 0, "");
          clin();
          delete MainMENU;
  }
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_ doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_.ppt	Презентація кваліфікаційної роботи