МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КОЛЕДЖ РАКЕТНОКО-СМІЧНОГО МАШИНОБУДУВАННЯ

ДНІПРОВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ім. О. ГОНЧАРА

Предметна комісія програмної інженерії

**КУРСОВИЙ ПРОЕКТ**

з навчальної дисципліни

"ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ"

на тему: «Програмна реалізація механізму логічного висновку експертної системи»

(вказати тему курсового проекту)

Студента \_\_\_IV\_\_ курсу ПЗ-16-1 групи

спеціальності 121 Інженерія

програмного забезпечення

Вайчекаускас С.К.

(прізвище та ініціали студента)

Керівник викладач Н.В Гапоненко

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_ Оцінка ECTS: \_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_ \_Н.П. Дено

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_ С.С.Ланська

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_ Н.В.Гапоненко

(підпис) (прізвище та ініціали)

Дніпро

2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КОЛЕДЖ РАКЕТНО-КОСМІЧНОГО МАШИНОБУДУВАННЯ

ДНІПРОВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ім. О. ГОНЧАРА

Предметна комісія програмної інженерії

|  |  |
| --- | --- |
|  | **ЗАТВЕРДЖУЮ**  Голова комісії ПІ  \_\_\_\_\_\_\_\_\_\_\_\_ С.С.Ланська  "\_\_\_\_"\_\_\_\_\_\_\_\_\_\_\_ 2019 р. |

**ЗАВДАННЯ**

**на виконання курсового проекту**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| з дисципліни | | | Об’єктно-орієнтоване програмування | | | | | | | |
| студенту | Вайчекаускасу Станіславу Костянтиновичу | | | | | | | | | |
|  | (прізвище, ім'я та по батькові) | | | | | | | | | |
| Відділення | | Комп'ютерної та програмної інженерії | | | | | | | | |
| Спеціальність | | | | 121 Інженерія програмного забезпечення | | | | | | |
| Курс | IV | | | | | |  | Група (шифр) | | ПЗ-16-1 |
| 1 Тема проекту | | | | | «Програмна реалізація механізму логічного висновку | | | | | |
| експертної системи» | | | | | | | | | | |
|  | | | | | | | | | | |
| 2 Початкові дані | | | | | | Перелік видів завдань, які підлягають вирішенню, та | | | | |
| цілей розробки, перелік типів користувачів та інформації про експерта, дані | | | | | | | | | | |
| про експертну систему (завдання, ціль, експерт, послідовність кроків | | | | | | | | | | |
| вирішення), алгоритм реалізації механізму логічного висновку експертної | | | | | | | | | | |
| системи | | | | | | | | | | |
| Розглянуто і ухвалено на засіданні предметної комісії | | | | | | | | | програмної інженерії | |
| Протокол № 2 від 10.09.2019 р. | | | | | | | | | | |

Керівник КП \_\_\_\_\_\_\_\_\_\_\_\_\_ Н.В.Гапоненко

(підпис) (ініціали та прізвище)

Завдання до виконання

одержав студент \_\_\_\_\_\_\_\_\_\_\_\_\_ Вайчекаускас С.К.

(підпис) (ініціали та прізвище)

Дата видачі 10 вересня 2019 р.

Термін виконання 21 листопада 2019 р.

ЗМІСТ

[ВСТУП 4](#_Toc26262016)

[1 ПОСТАНОВКА ЗАДАЧІ 6](#_Toc26262017)

[2 ХАРАКТЕРИСТИКА ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ ПРОЕКТУ 8](#_Toc26262018)

[2.1 Опис середовища програмування 8](#_Toc26262019)

[2.2 Опис мови програмування 9](#_Toc26262020)

[2.3 Опис СУБД MySQL 11](#_Toc26262021)

[2.4 Опис мови стилів CSS 12](#_Toc26262022)

[2.5 Опис основних принципів ООП 13](#_Toc26262023)

[2.7 Огляд існуючих рішень 17](#_Toc26262024)

[2.8 Опис реалізації алгоритму нечіткої логіки 21](#_Toc26262025)

[3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ 24](#_Toc26262026)

[3.1 Опис структури бази даних 24](#_Toc26262027)

[3.2 Проектування користувацького інтерфейсу програми 28](#_Toc26262028)

[4 ІНСТРУКЦІЯ З КОРИСТУВАННЯ ПРОГРАМНОЮ СИСТЕМОЮ 33](#_Toc26262029)

[ВИСНОВКИ 38](#_Toc26262030)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 40](#_Toc26262031)

[Додаток А 41](#_Toc26262032)

[Додаток Б 46](#_Toc26262033)

# ВСТУП

В умовах сучасного інформаційного суспільства все актуальнішою стає потреба використання новітніх технологій у навчанні, як для покращення методів та технології навчання, так і для підвищення його ефективності. Також використання новітніх інформаційних технологій дає можливість набагато збільшити стандартизацію знань та методи покращення сприймання їх студентами навчального курсу, та значно зменшити часові та фінансові затрати у різних галузях.

Одним з шляхів розв’язання цієї проблеми є система безперервної освіти, яка складається з базової і подальшої освіти та передбачає на другому етапі послідовне чергування навчання в системі спеціально створених освітніх закладів з професійною діяльністю.

Завдяки своїй змістовій наповненості й необмеженості в часі безперервна професійна освіта має можливості для виконання важливих функцій, а саме:

* загальноосвітньої, компенсуючої (усунення недоліків у базовій освіті, її доповнення новою інформацією, що з'являється в умовах інформаційно-технологічної революції);
* адаптивної (гнучка професійна підготовка, перепідготовка й підвищення кваліфікації з метою оновлення професійного досвіду, здобуття іншого фаху в умовах постійних змін на виробництві, розвитку теле- та радіокомунікацій, комп'ютерного доступу до інформаційних банків даних тощо);
* економічної (задоволення потреб держави, регіонів, різних галузей промисловості, сільського господарства і сфери послуг у конкурентоспроможних фахівцях, підготовлених до впровадження новітніх технологій, техніки та ін.).

Сьогодні процес складання інтегрованих навчальних планів, заснований на досвіді і інтуїції працівників вищої школи потребує серйозного удосконалення та наукового підґрунтя прийнятих рішень. Процес конструювання індивідуального навчального плану студента або групи студентів може являти собою педагогічну технологію, орієнтовану на реалізацію інформаційних технологій за допомогою експертної системи. В умовах скорочених строків навчання застосування експертних систем забезпечує можливість побудови індивідуального навчального плану, що підвищує ефективність процесів навчання, викладання і самоосвіти.

Програмна оболонка для створення індивідуального навчального плану — це складова експертної системи яка забезпечує формування індивідуальних навчальних планів здобувачів в системі «неперервної освіти» та перелік питань для організації вступного фахового випробування.

Використання експертної системи при складанні інтегрованих індивідуальних навчальних планів має ряд переваг. По-перше, з часом система буде розширюватися і накопичувати відповідності компетенцій як з суміжних спеціальностей, так і з інших галузей знань. По-друге, можна складати робочі навчальні плани напрямів підготовки без повторного залучення експертів з різних предметних областей. По-третє, систему легко можна буде перенавчати відповідно до нових вимог складання інтегрованих планів.

Виходячи з вище написаного, можна зробити висновок, що тема курсового проекту є актуальною, а поставлене завдання — своєчасним.

# 1 ПОСТАНОВКА ЗАДАЧІ

Темою курсового проекту є «Програмна реалізація механізму логічного висновку експертної системи».

Мета проекту: реалізувати механізм створення інтегрованих індивідуальних навчальних планів з автоматичним формуванням вступного фахового іспиту.

Програмний додаток повинен:

1. Визначати, згідно матриці порівнянь, перелік питань, з яких можуть складатися фахові вступні випробування на наступному освітньому рівні за обраною спеціальністю та формувати з них вступний фаховий іспит.
2. Визначати індивідуальний інтегрований навчальний план здобувача або групи здобувачів з урахуванням компетентностей, які здобувач успішно отримав на попередньому освітньому рівні та у визначеній кількості кредитів ЄКТС.
3. Для досягнення поставленої мети експертна система повинна використати математичну модель оптимального розподілу кредитів на базі нечіткої логіки.

Початкові дані містять:

1. Дані про дисципліни:

* назва дисципліни;
* вид контролю;

1. Дані про модулі:

* назва модуля;
* кількість годин для модуля;

1. Дані про компетенції:

* назва компетенції;
* значення компетенції;

1. Дані про відповідності:

* значення відповідності двох між двома модулями;

1. Дані про питання для іспитів:

* значення питання;
* варіант відповіді А;
* варіант відповіді Б;
* варіант відповіді В;
* варіант відповіді Г;
* правильний варіант;

1. Дані про типи компетенцій:

* Найменування типу;

Мінімальні вимоги до конфігурації ПК для користування цією програмою:

* центральний процесор Pentium IV або відповідний AMD;
* оперативна пам’ять 512 МБ;
* об’єм пам’яті жорсткого диску 256мб для клієнта і 5гб для сервера;
* операційні системи Windows 7, 8, 10 або Linux.

# 2 ХАРАКТЕРИСТИКА ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ ПРОЕКТУ

## 2.1 Опис середовища програмування

Програма курсового проекту написана в середовищі Qt Creator Community Edition.

 Qt Creator — кросплатформенний інструментарій розробки програмного забезпечення (ПЗ) мовою програмування C++. Дозволяє запускати написане за його допомогою ПЗ на більшості сучасних операційних систем (ОС), просто компілюючи текст програми для кожної операційної системи без зміни серцевого коду. Містить всі основні класи, які можуть бути потрібні для розробки прикладного програмного забезпечення, починаючи з елементів графічного інтерфейсу й закінчуючи класами для роботи з мережею, базами даних, OpenGL, SVG і XML. Бібліотека дозволяє керувати потоками, працювати з мережею та забезпечує кросплатформенний доступ до файлів.

Qt Creator має вбудований редактор форм, що дає змогу власноруч прописувати стиль форми можливостями мови CSS. Підтримує CMake, що дає змогу писати кросплатформенні додатки. З відомих додатків Qt використовує Google Earth, завдяки зручності використання цієї IDE .

Була обрана за можливість розробляти кросплатформенні проекти, легке перенесення проекту з однієї ОС у іншу, можливість написання мобільних додатків мовою С++, широкий вибір інструментів для розробки програмного забезпечення та зручний інтерфейс.

## 2.2 Опис мови програмування

Програма курсового проекту написана мовою програмування С++. Текст програми знаходиться у додатку Д.

C++, яка спочатку проектувалася, як розширення для С, а також на Java та C#, які запозичили у С синтаксис.

Мова програмування C++ найбільш поширена серед розробників програмного забезпечення. Вона є дуже зручною у розробці прикладних програм; драйверів пристроїв; розробка ОС; відео ігор. Реалізацією мови С++ займаються одночасно декілька проектів як безкоштовних, так і комерційних, а саме: GNU, Microsoft і Embarcadero (Borland).

C++ －високорівнева мова програмування загального призначення, поєднує властивості як високорівневих, так і низькорівневих мов програмування. Здатна до підтримки об'єктно-орієнтованого, узагальненого та процедурного програмування. Від свого попередника, мову програмування С, С++ відрізняється тим, що найбільшу увагу при розробці цієї мови було приділено підтримці об'єктно-орієнтованого та узагальненого програмування. Назва «мова програмування C++» походить від мови програмування C, в якому унарний оператор ++ позначає інкремент змінної.

На початку 1980-х років мову програмування С++ розробив співробітник фірми Bell Laboratories － Бьерн Страуструп. Створення мови С++ не планувалось, для власних потреб Бьерн розробив ряд удосконалень мови програмування С. Ранні версії мови С++, називались «Cі з класами» та почали з'являтися з 1980 року. Страуструп додав до мови програмування С можливість роботи з класами та об'єктами, поклавши фундамент нової, заснованої на синтаксисі С, мови програмування. Синтаксис C++ був заснований на синтаксисі C, так як Бьерн Страуструп прагнув зберегти сумісність з мовою C.

Нововведеннями С++ порівняно з С є:

* простий і практичний інтерфейс;
* 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 імен функцій;
* посилання і оператори управління вільно розподіленою пам'яттю.

Переваги С++:

* 1. Швидкість роботи програм на С++ майже не змінилась порівняно із С, хоча розробники отримали нові можливості і засоби.
  2. Мова С++ є кросплатформенною.
  3. Можливість роботи на низькому рівні з пам'яттю, адресами, портами.
  4. Можливість створення узагальнених алгоритмів для різних типів даних, їхня спеціалізація і обчислення на етапі компіляції з використанням шаблонів.
  5. Підтримуються різні стилі та технології програмування, включаючи традиційне директивне програмування, ООП, узагальнене програмування, мета програмування (шаблони, макроси).

Недоліки С++:

* погана підтримка модульності. Підключення інтерфейсу зовнішнього модуля через процесорну вставку заголовного файлу (#include) серйозно уповільнює компіляцію при підключенні великої кількості модулів. Для усунення цього недоліку багато компіляторів реалізують механізм компіляції заголовних файлів;
* недостача інформації про типи даних під час компіляції;
* мова C++ є складною для вивчення і для компіляції;
* деякі перетворення типів не інтуїтивні. Зокрема, операція над беззнаковим і знаковим числами видає беззнаковий результат.

## 2.3 Опис СУБД MySQL

MySQL — це система управління базами даних.

База даних являє собою структуровану сукупність даних. Ці дані можуть бути будь-якими — від простого списку майбутніх покупок до переліку експонатів картинної галереї або величезної кількості інформації в корпоративній мережі. Для запису, вибірки й обробки даних, що зберігаються в комп'ютерній базі даних, необхідна система управління базою даних, якою і є ПО MySQL. Оскільки комп'ютери чудово справляються з обробкою великих обсягів даних, управління базами даних відіграє центральну роль в обчисленнях. Реалізовано таке управління може бути по-різному - як у вигляді окремих утиліт, так і у вигляді коду, що входить до складу інших додатків.

MySQL — це система управління реляційними базами даних.

В реляційній базі даних дані зберігаються не всі скопом, а в окремих таблицях, завдяки чому досягається виграш в швидкості і гнучкості. Таблиці зв'язуються між собою за допомогою відносин, завдяки чому забезпечується можливість об'єднувати при виконанні запиту дані з декількох таблиць. SQL як частина системи MySQL можна охарактеризувати як мову структурованих запитів й одну з найвикористовуваних мов для доступу до баз даних.

Програмне забезпечення MySQL — це ПЗ з відкритим кодом.

ПЗ з відкритим кодом означає, що застосовувати і модифікувати його може будь-хто. Таке ПЗ можна отримувати по Internet і використовувати безкоштовно. При цьому кожен користувач може вивчити вихідний код і змінити його відповідно до своїх потреб. Використання програмного забезпечення MySQL регламентується ліцензією GPL (GNU General Public License), в якій зазначено, що можна і чого не можна робити з цим програмним забезпеченням в різних ситуаціях. Якщо робота у рамках GPL вас не влаштовує або планується вбудовування MySQL-коду в комерційний додаток, є можливість купити комерційну ліцензовану версію у компанії MySQL AB. See section 1.6.3 ліцензії на ПЗ MySQL.

MySQL є дуже швидким, надійним і легким у використанні. MySQL має також ряд зручних можливостей, розроблених в тісному контакті з користувачами. Спочатку сервер MySQL розроблявся для управління великими базами даних з метою забезпечити більш високу швидкість роботи в порівнянні з існуючими на той момент аналогами. І ось вже протягом декількох років даний сервер успішно використовується в умовах промислової експлуатації з високими вимогами. Незважаючи на те що MySQL постійно вдосконалюється, він вже сьогодні забезпечує широкий спектр корисних функцій. Завдяки своїй доступності, швидкості і безпеки MySQL дуже добре підходить для доступу до баз даних по Internet.

Технічні можливості СУБД MySQL.

MySQL є системою клієнт-сервер, яка містить багатопоточний SQL-сервер, що забезпечує підтримку різних обчислювальних машин баз даних, а також кілька різних клієнтських програм і бібліотек, засоби адміністрування і широкий спектр програмних інтерфейсів (API). Також поставляється у вигляді багатопоточної бібліотеки, яку можна підключити до користувача додатком і отримати компактний, швидкий і легкий в управлінні продукт.

## 2.4 Опис мови стилів CSS

CSS — спеціальна мова, що використовується для опису зовнішнього вигляду сторінок, написаних мовами розмітки даних.

Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів.

Специфікації CSS були створені та розвиваються Консорціумом Всесвітньої мережі.

CSS має різні рівні та профілі. Наступний рівень CSS створюється на основі попередніх, додаючи нову функціональність або розширюючи вже наявні функції. Рівні позначаються як CSS1, CSS2 та CSS3. Профілі — сукупність правил CSS одного або більше рівнів, створені для окремих типів пристроїв або інтерфейсів. Наприклад, існують профілі CSS для принтерів, мобільних пристроїв тощо.

CSS (каскадна або блочна верстка) прийшла на заміну табличній верстці веб-сторінок. Головна перевага блочної верстки — розділення змісту сторінки (даних) та їхньої візуальної презентації.

CSS використовується авторами та відвідувачами веб-сторінок, щоб визначити кольори, шрифти, верстку та інші аспекти вигляду сторінки. Одна з головних переваг — можливість розділити зміст сторінки (або контент, наповнення, зазвичай HTML, XML або подібна мова розмітки) від вигляду документу (що описується в CSS).

Таке розділення може покращити сприйняття та доступність контенту, забезпечити більшу гнучкість та контроль за відображенням контенту в різних умовах, зробити контент більш структурованим та простим, прибрати повтори тощо. CSS також дозволяє адаптувати контент до різних умов відображення (на екрані монітора, мобільного пристрою (КПК), у роздрукованому вигляді, на екрані телевізора, пристроях з підтримкою шрифту Брайля або голосових браузерах та ін.).

Один і той самий HTML або XML документ може бути відображений по-різному залежно від використаного CSS.

## 2.5 Опис основних принципів ООП

Об'єктно-орієнтоване програмування або ООП (object — oriented programming) — методологія програмування, заснована на представленні програми у вигляді сукупності об'єктів кожен з яких являється реалізацією визначеного типу, що використовує механізм пересилки повідомлень і класи організовані в ієрархію наслідування.

Центральний елемент ООП — абстракція. Дані за допомогою абстракції перетворяться в об'єкти, а послідовність обробки цих даних перетворюється на набір повідомлень, що передаються між цими об'єктами. Кожен з об'єктів має свою власну унікальну поведінку. До об'єктів можна звертатися як до окремих сутностей, які реагують на повідомлення, що наказують їм виконати якісь дії.

ООП характеризується наступними принципами (по Алану Кею) :

1. Усе являється об'єктом;
2. Обчислення здійснюються шляхом взаємодії (обміну даними) між об'єктами, при якому один об'єкт вимагає, щоб інший об'єкт виконав деяку дію; об'єкти взаємодіють, посилаючи і отримуючи повідомлення; повідомлення - це запит на виконання дії, доповнений набором аргументів, які можуть знадобитися при виконанні дії;
3. Кожен об'єкт має незалежну пам'ять яка складається з інших об'єктів
4. Кожен об'єкт є представником класуякий виражає загальні властивості об'єктів даного типу
5. У класі задається функціональність (поведінка об'єкту); тим самим усе об'єкти, які являються екземплярами одного класу, можуть виконувати одні і ті ж дії;
6. Класи організовані в єдину деревовидну структуру із загальним коренем, звану ієрархією спадкоємства; пам'ять і поведінка, пов'язана з екземплярами певного класу, автоматично доступні будь-якому класу, розташованому нижче в ієрархічному дереві.

Абстрагування (abstraction) — метод рішення задачі, при якому об'єкти різного роду об'єднуються загальним поняттям (концепцією), а потім згруповані суті розглядаються як елементи єдиної категорії. Абстрагування дозволяє відокремити логічний сенс фрагмента програми від проблеми його реалізації, розділивши зовнішній опис (інтерфейс) об'єкту і його внутрішню організацію (реалізацію).

Інкапсуляція (encapsulation) — техніка, при якій несуттєва з точки зору інтерфейсу об'єкту інформація ховається усередині нього.

Спадкоємство (inheritance) — властивість об'єктів, за допомогою якої екземпляри класу дістають доступ до даних і методів класів-предків без їх повторного визначення. Спадкоємство дозволяє різним типам даних спільно використовувати один і той же код, призводячи до зменшення його розміру і підвищення функціональності.

Поліморфізм (polymorphism) —- властивість, що дозволяє використовувати один і той же інтерфейс для різних дій; поліморфній змінній, наприклад, може відповідати декілька різних методів.

Поліморфізм перекроює загальний код, що реалізовує деякий інтерфейс, так, щоб задовольнити конкретним особливостям окремих типів даних. Клас (class) - безліч об'єктів, пов'язаних спільністю структури і поведінка; абстрактний опис даних і поведінки (методів) для сукупності схожих об'єктів, представники якої називаються екземплярами класу.

Об'єкт (object) — конкретна реалізація класу, що володіє характеристиками стану, поведінки і індивідуальності, синонім екземпляра.

2.6 Розкриття поняття експертної системи

Експе́ртна систе́ма — це методологія адаптації алгоритму успішних рішень однієї сфери науково-практичної діяльності в іншу. З поширенням комп'ютерних технологій це тотожна (подібна, заснована на оптимізуючому алгоритмі чи евристиках) інтелектуальна комп'ютерна програма, що містить знання та аналітичні здібності одного або кількох експертів щодо деякої галузі застосування, і здатна робити логічні висновки на основі цих знань, тим самим забезпечуючи вирішення специфічних завдань (консультування, навчання, діагностування, тестування, проектування тощо) без участі експерта (фахівця в конкретній проблемній галузі). Також визначається як система, яка використовує базу знань для вирішення завдань (видачі рекомендацій) у деякій предметній галузі.

Експертна система відрізняється від інших прикладних програм наявністю таких ознак:

* моделює механізм [мислення](https://uk.wikipedia.org/wiki/%D0%9C%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F) людини під час [розв'язання задач](https://uk.wikipedia.org/wiki/%D0%A0%D0%BE%D0%B7%D0%B2%27%D1%8F%D0%B7%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B7%D0%B0%D0%B4%D0%B0%D1%87) в цій предметній галузі. Це істотно відрізняє експертні системи від систем математичного моделювання або комп'ютерної анімації. Однак, ЕС не повинні повністю відтворювати психологічну модель фахівця в цій області, а повинні лише відтворювати за допомогою комп'ютера деякі методики розв'язання проблем, що використовуються експертом.
* мистема, окрім виконання обчислювальних операцій, формує певні висновки, базуючись на тих знаннях, якими вона володіє. Знання в системі, зазвичай, описані деякою спеціалізованою мовою і зберігаються окремо від програмного коду, що формує висновки. Компонент збереження знань прийнято називати [базою знань](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B7%D0%BD%D0%B0%D0%BD%D1%8C).
* під час розв'язання задач основну роль відіграють евристичні і наближені методи, що, на відміну від алгоритмічних, не завжди гарантують успіх. Евристика, в принципі, є правилом впливу ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) rule of thumb), що в машинному вигляді відображає деяке знання, набуте людиною разом із накопичуванням практичного досвіду розв'язання аналогічних проблем. Такі методи є наближеними у тому сенсі, що, по-перше, вони не потребують вичерпної вихідної інформації, а, по-друге, існує певний ступінь впевненості (або невпевненості) в тому, що запропонований розв'язок є правильним.

Експертні системи відрізняються і від інших видів програм із галузі штучного [інтелекту](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D1%82).

* Експертні системи застосовуються для предметів реального світу, операції з якими зазвичай вимагають великого досвіду, накопиченого людиною. Експертні системи мають яскраво виражену практичну направленість для застосування в науковій або комерційній сфері.
* Однією з основних характеристик експертної системи є її швидкодія, тобто швидкість отримання результату та його достовірність (надійність). Дослідницькі програми штучного інтелекту можуть бути і не дуже швидкими, натомість, експертна система повинна за прийнятний час знайти розв'язок, що був би не гіршим за розв'язок, що може запропонувати фахівець в цій предметній області.
* Експертна система повинна мати можливість пояснити, чому запропоновано саме цей розв'язок і довести його обґрунтованість. Користувач повинен отримати всю інформацію, необхідну йому для того, аби переконатись в обґрунтованості запропонованого розв'язку.

## 2.7 Огляд існуючих рішень

У зв’язку з розробкою курсового проекту було прийнято рішення аналізувати існуючі програмні оболонки схожого призначення для співставлення перспектив проекту та для кращого розуміння того, які можливості та функціонал повинен бути втілений у програмній оболонці дипломної роботи.

Головним додатком, який було обрано як аналог і як приклад для аналізу оболонка «Мала Експертна Система 2.0» для створення системи медичної діагностики. Головне вікно програми представлено на рисунку 2.1.

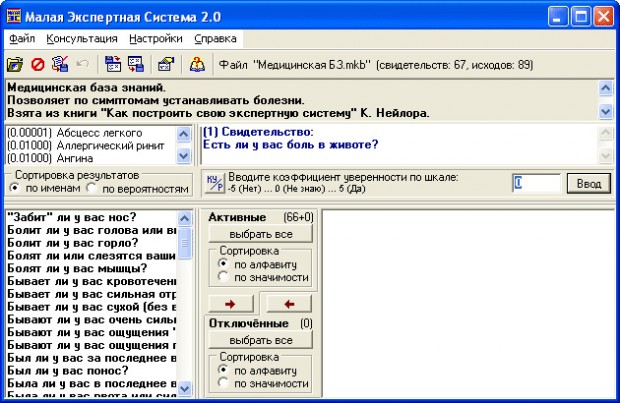


Рисунок 2.1 – Головне вікно оболонки «Мала Експертна Система 2.0»

Програма «Мала Експертна Система 2.0» представляє з себе просту оболонку експертної системи, на основі байєсівської системи логічного висновку. Оболонка призначена для проведення консультації з користувачем в будь-якої прикладної області з метою визначення ймовірностей можливих результатів і використовує для цих цілей оцінку правдоподібності деяких передумов, які система отримує від користувача.

Одним з основних переваг цієї програми є можливість створення, редагування і зміни бази знань, створеної самим користувачем. Для цього можна використовувати «Редактор баз знань», що поставляється разом з «Малої Експертної Системою». Вікно редактора представлено на рисунку 2.2.

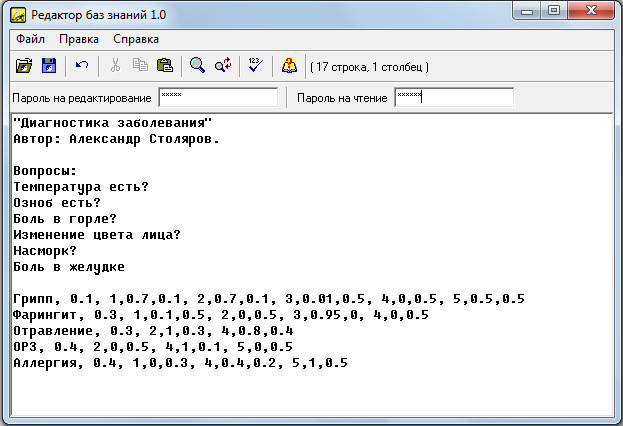


Рисунок 2.2 – Вікно редактора баз знань

Але ознайомившись краще з розробкою, інтерфейс виявився недостатньо інтуітивно-зрозумілим для посереднього користувача, тому в своїй розробці упор було зроблено саме на цю складову, щоб, навіть, користувач який має мінімальний досвід в роботі з комп’ютером, міг з лекістью користуватися програмою.

2.8 Опис методу реалізації експертної системи

Сьогодні процес складання інтегрованих навчальних планів, заснований на досвіді і інтуїції працівників вищої школи потребує серйозного удосконалення та наукового підґрунтя прийнятих рішень. Процес конструювання індивідуального навчального плану студента або групи студентів може являти собою педагогічну технологію, орієнтовану на реалізацію інформаційних технологій за допомогою експертної системи. В умовах скорочених строків навчання застосування експертних систем забезпечує можливість побудови індивідуального навчального плану, що підвищує ефективність процесів навчання, викладання і самоосвіти.

Використання експертної системи для узгодження освітніх програм між певними освітніми рівнями та формування переліку питань для перевірки рівня знань по зарахованим дисциплінам або модулям дисциплін представлені в даному дослідженні. Експертна система, представлена на рисунку 1, включає в себе наступні підсистеми: бази знань аналізованих освітніх рівнів; експертний рівень, на якому уточнюється формулювання компетентностей, їх приналежність до того чи іншого модулю та необхідний рівень їх сформованості для випускника; механізму логічного висновку, який забезпечує формування індивідуальних навчальних планів здобувачів в системі «неперервної освіти» та перелік питань для організації вступного фахового випробування.

База компетентностей освітнього рівня випускника

База компетентностей наступного освітнього рівня

Експертний рівень

Механізм логічного висновку

Варіант індивід. навчального плану

Перелік питань для вступного випробування

Варіант індивід. навчального плану

Перелік питань для вступного випробування

Варіант індивід. навчального плану

Перелік питань для вступного випробування

**…**

Варіант індивід. навчального плану

Перелік питань для вступного випробування

Рисунок 2.3 – Модель формування індивідуального навчального плану безперервної підготовки фахівців

Експертна система (ЕС) може функціонувати в 2-х режимах:

1. Режим введення знань – в цьому режимі експерт за допомогою редактора бази знань вводить відомості про предметну область в базу знань ЕС.

2. Режим консультації – користувач веде діалог з ЕС, повідомляючи їй відомості про поточне завдання і отримуючи рекомендації ЕС.

При роботі в режимі введення знань експертна система повинна мати такі можливості:

1.Складати бази компетентностей перехідних освітніх рівнів, які формуються на основі освітньо-професійних програм кожного з рівнів.

2. Поставлення у відповідність кожній компетентності однієї або декілька навчальних дисциплін із зазначенням модуля, який "перекриває" дану компетентність.

3. Складати матриці парних порівнянь компетентностей різних освітніх рівнів.

Для роботи в цьому режимі можуть бути залучені науково-педагогічні працівники ВНЗ.

У режимі консультації можна буде використовувати систему при вирішенні наступних завдань:

1. Визначати, згідно матриці порівнянь, перелік питань, з яких можуть складатися фахові вступні випробування на наступному освітньому рівні за обраною спеціальністю.

2. Визначати індивідуальний інтегрований навчальний план здобувача або групи здобувачів з урахуванням компетентностей, які здобувач успішно отримав на попередньому освітньому рівні та у визначеній кількості кредитів ЄКТС.

Для досягнення поставленої мети в експертній системі пропонується використати математичну модель оптимального розподілу кредитів на базі нечіткої логіки.

## 2.8 Опис реалізації алгоритму нечіткої логіки

Постановка задачі полягає в зарахуванні  кредитів на наступному навчальному рівні при складанні індивідуального навчального плану студента або групи студентів. Сутність методики формування таких планів полягає в порівняння пар об’єктів (компетентності різних освітніх рівнів) експертом. Результатом такого порівняння може бути таблиця, лівий стовбець та верхній рядок якої формують компетентності відповідних освітніх рівнів. На перетині стовпця та рядка експерт визначає ступінь відповідності  між компетентностями (значення ступеню розподілені на відрізку [0,1]). Введемо наступні нечіткі значення відповідності показника : 0-0,3 – не відповідає; 0,4-0,5 – не визначено; 0,6-0,7 – більш-менш відповідає; 0,8-0,9 – майже відповідає; 1- відповідає.

Таблиця 2.1－Таблиця порівняння компетентностей

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Освітній рівень наступний (ІІ рівень) | | | | | |
| Освітній рівень попередній (І рівень | Компетент-ності |  |  |  |  |  |
|  |  |  |  | … |  |
|  |  |  |  | … |  |
|  |  |  |  | … |  |
|  | … | … | … | … | … |
|  |  |  |  | … |  |
|  | |  |  |  | … |  |
|  | |  |  |  | … |  |

У матриці , представленій в таблиці 1, індекс  () – це номер компетентності попереднього освітнього рівня, (умовно позначимо його як І рівень), а   – номер компетентності наступного освітнього рівня (ІІ рівень). Знаходимо максимальне значення в кожному стовпці.

Також компетентностям ІІ-го рівня задамо ваговий коефіцієнт , який буде дорівнюватися значенню кількості відведених на нього кредитів.

Будемо вважати, що відповідність компетентностей освітніх рівнів було встановлено, якщо . Класифікуємо все компетентності ІІ рівня за наступними правилами:

ІF (=’ЗК’ AND THEN = класу 1;

ІF (=’ФК’ AND THEN = класу 2;

ІF (=’ЗК’ AND THEN = класу 3;

ІF (=’ФК’ AND THEN = класу 4;

ІF (=’ЗК’ AND THEN = класу 5;

ІF (=’ФК’ AND THEN = класу 6.

Підсумуємо вагові коефіцієнти компетентностей 1-го класута порівняємо зі значенням вхідного показника , якщо значення  більше, то додамо ще суму вагових коефіцієнтів компетентностей 2-го класу і так далі, доки значення суми не стане рівним або не перевищить показник . Якщо ж було накопичено суму вагових коефіцієнтів до 6-го класу і  , це означає, що в даному випадку задача не має рішення, тобто сформувати інтегрований навчальний при таких початкових даних немає можливості. Сформувати індивідуальний план вдасться, якщо на якомусь з етапів підсумовування вагових коефіцієнтів виконається нерівність , де, а  – номер етапу. Тоді всі компетентності , де будуть перезараховані, а модулі дисциплін, які відповідають цим компетентностям виключені з індивідуального навчального плану.

# 3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

## 3.1 Опис структури бази даних

Програма має можливість обробляти дані різних проектів.

База даних всіх проектів має однакову структуру ER-діаграма якої представлена на рисунку 3.1.

База даних програми складається з 10 пов’язаних між собою таблиць-моделей, які умовно поділені на два рівня, перелік основних таблиць: Discipline — «Дисципліна», Module— «Модуль», Mod\_comp— проміжна таблиця, Competence — «Компетенція», Conformity — «Відповідності», Question — «Питання для тесу», Type\_compet ­ «Типи компетенцій». Перелік таблиць БД, їх поля та призначення полів наведено у таблицях3.1 – 3.7.

Таблиця 3.1 – Таблиця Discipline — «Дисципліна»

|  |  |
| --- | --- |
| Поле | Призначення |
| id\_discipline | Ідентифікатор дисципліни |
| name\_discipline | Назва дисципліни |
| control\_discipline | Вид контролю |

Таблиця 3.2 – Таблиця Module— «Модуль»

|  |  |
| --- | --- |
| Поле | Призначення |
| id\_module | Ідентифікатор модуля |
| name\_module | Назва модуля |
| hours\_module | Кількість годин для модуля |
| id\_discipline | Ідентифікатор дисципліни, до якої відноситься модуль |

Таблиця 3.3 – Таблиця Mod\_comp — проміжна таблиця

|  |  |
| --- | --- |
| Поле | Призначення |
| id\_module | Ідентифікатор модуля |
| id\_compet | Ідентифікатор компетенції |

Таблиця 3.4 – Таблиця Competence — «Компетенція»

|  |  |
| --- | --- |
| Поле | Призначення |
| id\_compet | Ідентифікатор компетенції |
| name\_compe | Назва компетенції |
| kind\_compet | Значення компетенції |

Таблиця 3.5 – Таблиця Conformity — «Відповідності»

|  |  |
| --- | --- |
| Поле | Призначення |
| id\_coeff | Ідентифікатор коефіцієнту відповідності |
| id\_compet | Ідентифікатор компетенції першого рівня |
| id\_compet\_2 | Ідентифікатор компетенції другого рівня |
| value\_coeff | Значення відповідності |

Таблиця 3.6 – Таблиця Question — «Питання для тесу»

|  |  |
| --- | --- |
| Поле | Призначення |
| id\_question | Ідентифікатор питання |
| id\_compet\_2 | Ідентифікатор компетенції другого рівня |
| question | Значення питання |
| answer\_a | Варіант відповіді А |

Продовження таблиці 3.6

|  |  |
| --- | --- |
| Поле | Призначення |
| answer\_b | Варіант відповіді Б |
| answer\_c | Варіант відповіді В |
| answer\_d | Варіант відповіді Г |
| answer\_tru | Значення правильної відповіді |

Таблиця 3.7 – Таблиця Type\_compet — типи компетенцій

|  |  |
| --- | --- |
| Поле | Призначення |
| id\_type | Ідентифікатор типу |
| name\_type | Найменування типу |

Між таблицями встановлено зв’язок «один-до-багатьох». Також є реалізованими функції каскадного оновлення та видалення даних, та забезпечення цілісності даних.

name\_discipline

control\_discipline

**discipline**

id\_discipline PK

name\_discipline\_2

control\_discipline\_2

**discipline\_2**

id\_discipline\_2 PK

name\_module

hours\_module

**module**

id\_module PK

id\_discipline FK

name\_module\_2

hours\_module\_2

**module\_2**

id\_module\_2 PK

id\_discipline\_2 FK

info\_compet

type\_compet FK

**competence**

id\_compet PK

**mod\_comp**

id\_module FK

id\_compet FK

**mod\_comp\_2**

id\_module FK

id\_compet FK

info\_compet\_2

type\_compet FK

**competence\_2**

id\_compet\_2 PK

id\_compet FK

id\_compet\_2 FK

**conformity**

id\_coeff PK

value\_coeff

id\_compet\_2 FK

question

**quastion**

id\_question PK

answer\_a

answer\_c

answer\_d

answer\_b

answer\_true

**type\_compet**

id\_type PK

name\_type

Рисунок 3.1 — ER-діаграма проектів бази даних

3.2 Проектування користувацького інтерфейсу програми

Задачею графічного інтерфейсу програми є забезпечення комфортної взаємодії користувача з програмним додатком. Графічний інтерфейс складається з форм та елементів, які розташовуються на формах. Розробка інтерфейсу програми починається зі створення нової, стартової форми, шляхом внесення змін до властивостей автоматично згенерованої форми (Form1), а також шляхом додавання на форму нових необхідних елементів. На форму можуть бути додані різноманітні компоненти для взаємодії з текстом, числовими даними, базами даних тощо. Для перегляду і зміни значень властивостей форми і її компонентів використовується вікно Object Inspector, у якому знаходиться більшість даних про об’єкт, який є активним на даний момент (ім’я об’єкту, шрифт, ширина, висота тощо). Під час проектування дуже важливо цілеспрямовано обрати компоненти.

Інтерфейс програми складається з двох форм та декількох слайдів, що змінюють один одного.

Інтерфейс програми можна описати декількома модулями:

* Authorization — форма для входу користувача у свій обліковий запис у базі даних;
* MainWindow — форма для доступу користувачем до бази даних та обробки інформації.

Форма Authorization містить напис привітання на об’єкти для введення логіну і паролю, кнопки для підключення бази даних, закриття програми, та напис-повідомлення, якщо до бази даних підключитися не вдалося. Форму можна побачити на рисунку 3.2.

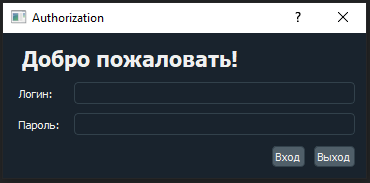


Рисунок 3.2 — Форма Authorization

Форма MainForm складається з трьох слайдів, які поступово змінюють один одного. Події на формі складаються з трьох етапів, спочатку обирання проекту, далі меню налаштування виводу інформації, і третій слайд — виведення розрахунків. Форму можна побачити на рисунках 3.3-3.5.

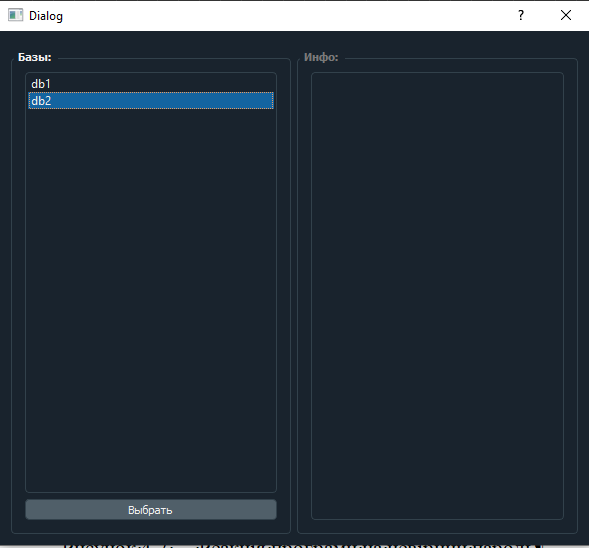


Рисунок 3.3— Перший слайд форми MainFrom

Форма має об’єкт QListWidget для виведення та надання можливості вибору користувачем проекту, який він хоче переглянути. З правої сторони знаходиться QLabel, об’єкт який буде виводити інформацію у проекті, наприклад кількість предметів, для яких курсів, вузів і спеціальностей був створений проект, або будь-які підказки для користувачів. при натисканні на кнупку «Выбрать» із QListWidget буде обрану назву проекту для відкриття і программа покаже наступний слайд на якому будуть показані параметри перегляду вихідної інформації, які доступні для налаштування користувачем.

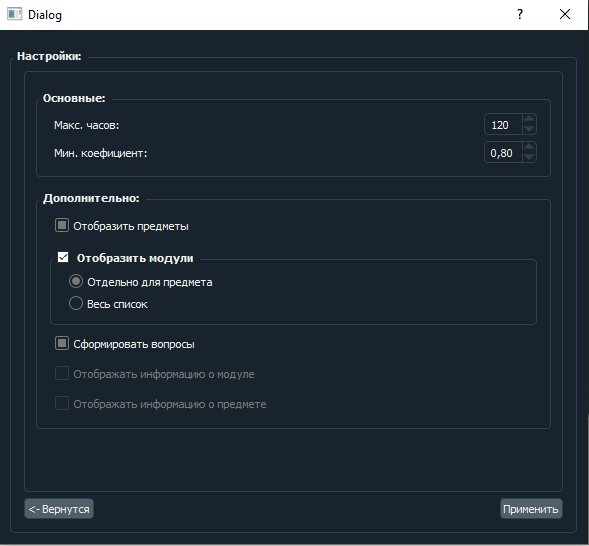


Рисунок 3.4 — Другий слайд форми MainFrom

На другому слайді важливі для корегування параметри знаходяться у GroupBox`і — «Основные», а саме кількість годин доступних для перезарахування, та мінімальний коефіцієнт відповідності між компетенція ми доступних для пере зарахування. У GroupBox`і — «Дополнительно» доступні параметри, які впливають лише на інформацію, що буде виведена користувачу та її зміст.

Пункт «Отобразить предметы» відповідає за те, чи будуть виводитися перезараховані предмети.

Пункт «Отобразить модули» відповідає за те, чи будуть виводитися модулі, які були перезараховані. Також цей пункт додатково дає можливість обрати користувачеві в якому порядку будуть виводитися модулі: одразу всі, чи для обраного предмету.

Пункт «Сформировать вопросы» дає можливість обрати користувачеві, потрібно йому бачити на екрані питання на екзамени чи ні.

Кнопка «<-Вернуться» повертає користувача на минулий слайд для вибору БД, а кнопка «Применить» зберігає налаштування, запускає алгоритм розрахунку та переходить на наступний слайд.

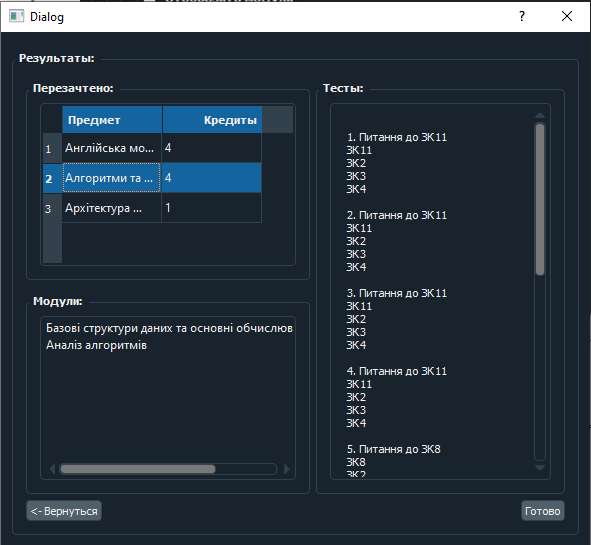


Рисунок 3.5 — Третій слайд форми MainFrom

На третьому слайді буде зображено, в залежності від вибору користувача на минулому, інформацію про перезараховані предмети, модулі, та сформованих на їх основі тестів до вступного екзамену.

Представлення програмного інтерфейсу у вигляді слайдів робить користування програмою простіше, через те що програма в такий спосіб буде працювати швидше і витрачатиме менше системних ресурсів, а також не буде відволікати увагу користувача на перемикання форм.

Також для створення більш комфортного і приємного користування програмой було використано CSS стилі. Тепер програма знаходиться у м’яких темно-синіх відтінках, кожний об’єкт програми має округлі кути, а також з’явилося більш чітке і контрастне підсвічування обраних и ключових об’єктів. Контрастне виділення ключових елементів буде допомагати користувачу швидше орієнтуватися у программі, а м’які темні кольори зменшать напругу на очі.

4 ІНСТРУКЦІЯ З КОРИСТУВАННЯ ПРОГРАМНОЮ СИСТЕМОЮ

Після запуску програми відкриється вікно авторизації. Воно матиме поле для введення логіну и поле для паролю. Після цього треба натиснути кнопку «Вход» і очікувати підключення, якщо невірно було введено логін або пароль — програма повідомить про це. Якщо всі дані введені вірно і є доступ до бази даних вікно авторизації закриється і з’явиться головне вікно програми.

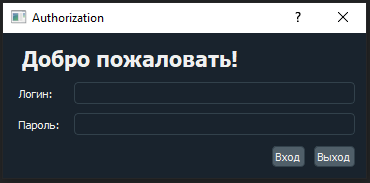


Рисунок 4.1 — Вікно авторизації

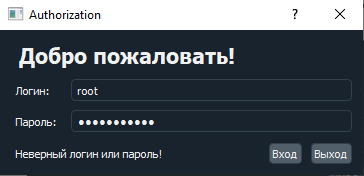


Рисунок 4.2 — Реакція програми на невірний пароль

Після успішної авторизації відкривається головне вікно програми, на якому по ліву сторону зображено усі доступні для перегляду проекти, а справа їх опис. Для обрання проекту необхідно натиснути на його назву у лівому стовпчику і натиснути кнопку «Выбрать».

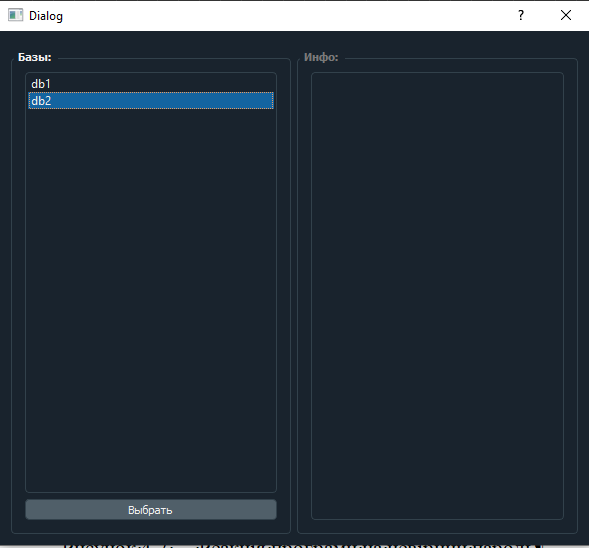


Рисунок 4.3 — Головне вікно програми

Далі на вікні з’являться налаштування користувачу інформації, що відображається. Обов’язкові для налаштування поля знаходять у самому початку під заголовком «Основные». Першим важливим параметром є кількість годин які готов пере зарахувати вищий навчальний заклад, а другим — коефіцієнт відповідності між компетенціями навчальних закладів. Ці параметри є важливими, оскільки саме вони впливають на роботу логіки програми та побудови результатів.

Також є додаткові параметри, на логіку роботи програми вони не впливають, а дають можливість користувачу обрати лише ту інформацію, яка його цікавить, і отримати її в такому виді, в якому йому буде зручніше.

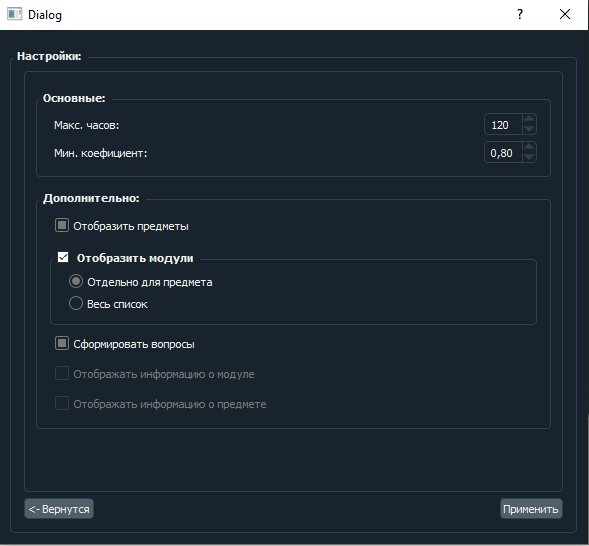


Рисунок 4.4 — Вікно налаштування

На нижній частині вікна знаходяться дві кнопки, кнопка «<-Вернутся» — поверне нас до меню вибору проекту, а кнопка «Применить» зберігає налаштування, запускає механізм розрахунку та переходить у наступне меню.

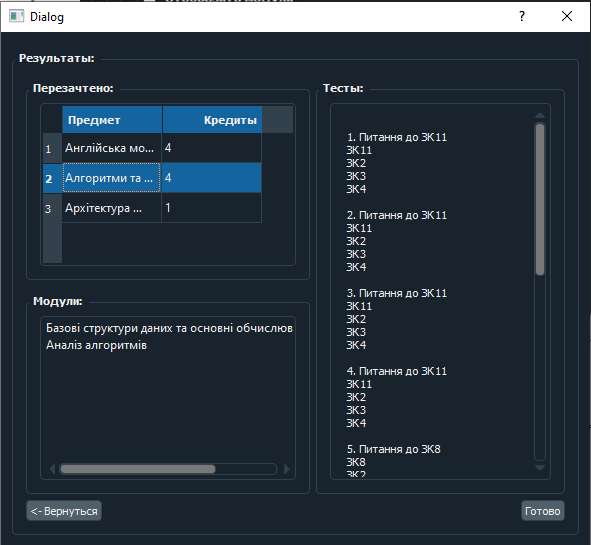


Рисунок 4.5 — Вікно перегляду інформації за стандартних налаштувань

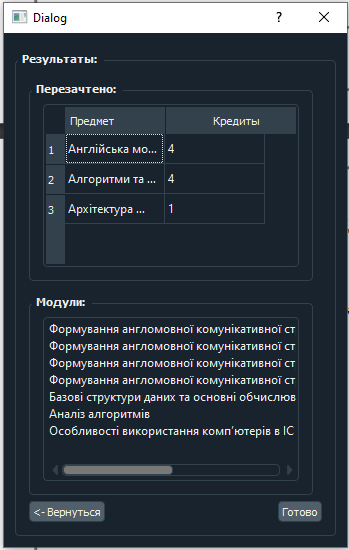


Рисунок 4.6 — Вікно перегляду інформації за без формування питань і виведенням усіх модулів одразу

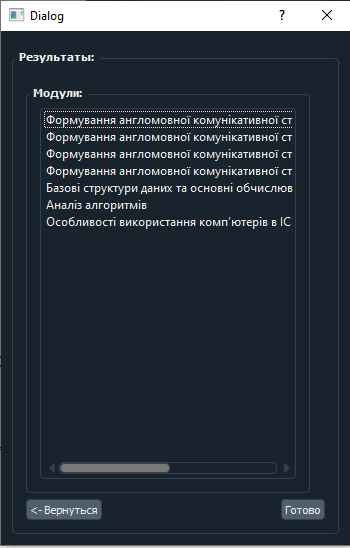


Рисунок 4.7 — Вікно перегляду інформації лише з виведенням пере зарахованих модулів

Вікно перегляду інформації містить таблицю із перезарахованими предметами та кількість. перезарахованих кредитів, список перезарахованих модулів, який в залежності від налаштування буде виводити модулі для обраного у минулій таблиці предмета, або одразу всі пере зараховані модулі, а також міститиме сформовані тести для вступного екзамену.

Кнопка «<-Вернуться» поверне нас до попереднього меню налаштувань, що дасть змогу знову налаштувати виведення інформації, а кнопка «Готово» поверне нас до меню вибору проектів.

Для закриття програми необхідно натиснути на хрести у правому верхньому куті програми.

ВИСНОВКИ

Під час роботи над курсовим проектом «Програмна реалізація механізму логічного висновку експертної системи» засобами середовища Qt Creator було створено механізм, що забезпечує формування індивідуальних навчальних планів в системі «неперервної освіти» та перелік питань для організації вступного фахового випробування.

У ході виконання курсового проекту було створено програмний додаток з реалізованим механізмом логічного висновку експертної системс, у якому було реалізовано увесь функціонал, який було задано у постановці задачі:

1. Можливість визначення, згідно матриці порівнянь, перелік питань, з яких можуть складатися фахові вступні випробування на наступному освітньому рівні за обраною спеціальністю та формувати з них вступний фаховий іспит.
2. Можливість визначення індивідуального інтегрованого навчального плану здобувача або групи здобувачів з урахуванням компетентностей, які здобувач успішно отримав на попередньому освітньому рівні та у визначеній кількості кредитів ЄКТС.
3. Механізм логічного висновку експертної системи використовує математичну модель оптимального розподілу кредитів на базі нечіткої логіки

Програма забезпечує роботу з проектами, розрахунками та їх результатами.

Розроблена програмна оболонка має інтуїтивно-зрозумілий графічний інтерфейс та приємне кольорове оформлення, а користувач потребує лише базових знань роботи з комп’ютером.

Програма може бути корисною та актуальною для використання у вищих навчальних закладах, оскільки може в декілька раз пришвидшити та спростити побудову індивідуальних навчальних планів та питань для складання вступних фахових екзаменів.

Програма є актуальною, оскільки спрощує і пришвидшує складання індивідуальних інтегрованих планів, має можливість накопичення компетенцій із сусідніх спеціальностей, дає змогу формувати інтегровані плани без повторного залучення експертів та легко перенавчається відповідно до нових вимог складання інтегрованих планів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Відео та лекції по виченню Qt Creatorehttps://resources.qt.io/resources-by-content-type-videos-demos-tutorials.
2. Батиршин І.З. Основні операції нечіткої логіки та їх узагальнення. Казань: Отечество, 2001. ― 102 c.
3. Алтунін А.Е., Семухін М.В. Моделі та алгоритми прийняття рішень в нечітких умовах. Тюмень: Видав-во Тюменського державного університету, 2000.– 352 с.
4. Нейлор, К. Как построить свою экспертную систему / К. Нейлор. - М.: Энергоатомиздат, 2013. ― 286 c.
5. Нильсон, Н. Принципы искусственного интеллекта / Н. Нильсон. - М.: Радио и связь, 2014. ― 373 c.
6. Електронний научно-практический журнал «Современная техника и технологии» [Електронний ресурс] // technology.snauka.ru. Режим доступу: <http://technology.snauka.ru/2016/12/11465>
7. М. Саммерфілд «Qt. Профессиональное программирование. Разработка кроссплатформенных приложений на С++», Символ-Плюс, 2011
8. Ж. Бланшет, М. Саммерфілд «QT 4: программирование GUI на С++», КУДИЦ-Пресс, 2008
9. М. Шлеє «Qt 5.3. Профессиональное программирование на C++», БХВ-Петербург, 2015
10. Офіційна документація — doc.qt.io

Додаток А

Лістинг програми

Лістинг А.1 – Клас роботи з базою даних

#ifndef DATABASE\_H

#define DATABASE\_H

#include <vector>

#include <QString>

#include <QSqlDatabase>

#include <QtSql>

*using* *namespace* std;

*struct* Module

{

QString id;

QString discipline;

QString name;

int hours;

};

*struct* Conformity

{

QString compet\_1;

QString compet\_2;

float value\_coef;

};

*struct* Mod\_Comp

{

QString id\_module;

QString id\_competence;

};

*struct* Question

{

QString id\_compet;

QString text;

QString a, b, c, d;

QString answer\_true;

};

*class* Database

{

*public*:

vector <Module> mod\_1;

vector <Module> mod\_2;

vector <Conformity> conf;

vector <Mod\_Comp> mod\_comp\_1;

vector <Mod\_Comp> mod\_comp\_2;

vector <Question> question;

vector <Module> result\_mod;

vector <Question> result\_question;

void LoadData(QSqlDatabase \*db);

void Calculate(int hourse, float coef);

Database();

};

Database::Database()

{

}

void Database::LoadData(QSqlDatabase \*db)

{

db->open();

QSqlQuery query;

query.exec("select module.id\_module, module.name\_module, module.hours\_module, discipline.name\_discipline from module, discipline where discipline.id\_discipline = module.id\_discipline group by module.id\_module;");

Module temp;

*while* (query.next())

{

temp.id = query.value(0).toString();

temp.name = query.value(1).toString();

temp.hours = query.value(2).toInt();

temp.discipline = query.value(3).toString();

mod\_1.push\_back(temp);

qDebug() << query.value(0).toString() << query.value(1).toString() << query.value(2).toInt() << query.value(3).toString();

}

query.exec("select module\_2.id\_module\_2, module\_2.name\_module\_2, module\_2.hours\_module\_2, discipline\_2.name\_discipline\_2 from module\_2, discipline\_2 where discipline\_2.id\_discipline\_2 = module\_2.id\_discipline\_2 group by module\_2.id\_module\_2;");

*while* (query.next())

{

temp.id = query.value(0).toString();

temp.name = query.value(1).toString();

temp.hours = query.value(2).toInt();

temp.discipline = query.value(3).toString();

mod\_2.push\_back(temp);

qDebug() << query.value(0).toString() << query.value(1).toString() << query.value(2).toInt() << query.value(3).toString();

}

Mod\_Comp temp\_MC;

query.exec("select \* from mod\_comp;");

*while* (query.next())

{

temp\_MC.id\_module = query.value(0).toString();

temp\_MC.id\_competence = query.value(1).toString();

mod\_comp\_1.push\_back(temp\_MC);

qDebug() << query.value(0).toString() << query.value(1).toString();

}

query.exec("select \* from mod\_comp\_2;");

*while* (query.next())

{

temp\_MC.id\_module = query.value(0).toString();

temp\_MC.id\_competence = query.value(1).toString();

mod\_comp\_2.push\_back(temp\_MC);

qDebug() << query.value(0).toString() << query.value(1).toString();

}

Conformity temp\_conf;

query.exec("select id\_compet, id\_compet\_2, value\_coeff from conformity;");

*while* (query.next())

{

temp\_conf.compet\_1 = query.value(0).toString();

temp\_conf.compet\_2 = query.value(1).toString();

temp\_conf.value\_coef = query.value(2).toFloat();

conf.push\_back(temp\_conf);

qDebug() << query.value(0).toString() << query.value(1).toString() << query.value(2).toFloat();

}

Question temp\_quest;

query.exec("select \* from question");

*while* (query.next())

{

temp\_quest.id\_compet = query.value(1).toString();

temp\_quest.text = query.value(2).toString();

temp\_quest.a = query.value(3).toString();

temp\_quest.b = query.value(4).toString();

temp\_quest.c = query.value(5).toString();

temp\_quest.d = query.value(6).toString();

temp\_quest.answer\_true = query.value(7).toString();

question.push\_back(temp\_quest);

qDebug() << query.value(0).toString() << query.value(1).toString() << query.value(2).toString() << query.value(3).toString() << query.value(4).toString() << query.value(5).toString() << query.value(6).toString() << query.value(7).toString();

}

}

void Database::Calculate(int hours\_full, float coef)

{

*for*(int i = 0; i < conf.size()-1; i++)

*for*(int j = i; j < conf.size(); j++)

*if*(conf[i].value\_coef < conf[j].value\_coef)

swap(conf[i], conf[j]);

qDebug() << "First sort complete!";

*for*(int i = 0; i < conf.size(); i++)

qDebug() << conf[i].compet\_1 << conf[i].value\_coef;

*for*(int i = 0; i < mod\_1.size()-1; i++)

*for*(int j = i+1; j < mod\_1.size(); j++)

{

int number\_i;

int number\_j;

*for*(int z = 0; z < mod\_comp\_1.size(); z++)

{

*if*(mod\_1[i].id == mod\_comp\_1[z].id\_module)

{

*for*(int x = 0; x < conf.size(); x++)

{

*if*(mod\_comp\_1[z].id\_competence == conf[x].compet\_1)

number\_i = x;

}

}

*if*(mod\_1[j].id == mod\_comp\_1[z].id\_module)

{

*for*(int x = 0; x < conf.size(); x++)

{

*if*(mod\_comp\_1[z].id\_competence == conf[x].compet\_1)

number\_j = x;

}

}

}

*if*(number\_i > number\_j)

swap(mod\_1[i], mod\_1[j]);

}

qDebug() << "Second sort complete!";

*for*(int j = 0; j < mod\_1.size(); j++)

{

*for*(int z = 0; z < mod\_comp\_1.size(); z++)

{

*if*(mod\_1[j].id == mod\_comp\_1[z].id\_module)

{

qDebug() << mod\_1[j].id << mod\_comp\_1[z].id\_competence;

}

}

}

*for*(int i = 0; i < mod\_1.size(); i++)

{

float max\_coef = 0;

QString competence;

int number\_z1;

int number\_j1;

*for*(int z = 0; z < mod\_comp\_1.size(); z++)

{

*if*(mod\_1[i].id == mod\_comp\_1[z].id\_module)

{

*for*(int j = 0; j < conf.size(); j++)

{

*if*(mod\_comp\_1[z].id\_competence == conf[j].compet\_1)

{

*if*(conf[j].value\_coef > max\_coef)

{

max\_coef = conf[j].value\_coef;

competence = conf[j].compet\_2;

number\_z1 = z;

}

}

}

}

}

qDebug() << "First half calculate complete!" << max\_coef << coef;

*if*(max\_coef >= coef)

{

int number\_j;

int number\_z;

int hours = 0;

*for*(int j = 0; j < mod\_comp\_2.size(); j++)

{

*if*(mod\_comp\_2[j].id\_competence == competence)

{

*for*(int z = 0; z < mod\_2.size(); z++)

{

*if*(mod\_2[z].id == mod\_comp\_2[j].id\_module)

{

*if*(mod\_2[z].hours > hours)

{

hours = mod\_2[z].hours;

number\_z = z;

number\_j = j;

}

}

}

}

}

result\_mod.push\_back(mod\_2[number\_z]);

*for*(int z = 0; z < mod\_comp\_2.size(); z++)

{

*if*(mod\_2[number\_z].id == mod\_comp\_2[z].id\_module)

{

*for*(int j = 0; j < question.size(); j++)

{

*if*(mod\_comp\_2[z].id\_competence == question[j].id\_compet)

result\_question.push\_back(question[j]);

}

}

}

mod\_2.erase(mod\_2.begin() + number\_z);

mod\_comp\_2.erase(mod\_comp\_2.begin() + number\_j);

mod\_comp\_1.erase(mod\_comp\_1.begin() + number\_z1);

mod\_1.erase(mod\_1.begin() + number\_j1);

hours\_full -= hours;

}

qDebug() << "Second half calculate complete!";

*if*(hours\_full <= 0 || max\_coef < coef)

*break*;

}

qDebug() << "Calculate complete!";

*for*(int i = 0; i < result\_mod.size(); i++)

qDebug() << result\_mod[i].id << result\_mod[i].name << result\_mod[i].hours << result\_mod[i].discipline;

}

#endif *//* *DATABASE\_H*

Додаток Б

Лістинг бази даних

Лістинг Б.1 – База даних проекту

DROP TABLE IF EXISTS `competence`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `competence` (

`id\_compet` char(7) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`info\_compet` text CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`type\_compet` int(2) NOT NULL,

PRIMARY KEY (`id\_compet`),

KEY `type\_compet` (`type\_compet`),

CONSTRAINT `competence\_ibfk\_1` FOREIGN KEY (`type\_compet`) REFERENCES `type\_compet` (`id\_type`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `competence\_2`

--

DROP TABLE IF EXISTS `competence\_2`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `competence\_2` (

`id\_compet\_2` char(7) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`info\_compet\_2` text CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`type\_compet\_2` int(2) NOT NULL,

PRIMARY KEY (`id\_compet\_2`),

KEY `type\_compet\_2` (`type\_compet\_2`),

CONSTRAINT `competence\_2\_ibfk\_1` FOREIGN KEY (`type\_compet\_2`) REFERENCES `type\_compet` (`id\_type`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `conformity`

--

DROP TABLE IF EXISTS `conformity`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `conformity` (

`id\_coeff` varchar(13) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`id\_compet` char(7) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`id\_compet\_2` char(7) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`value\_coeff` float(3,2) NOT NULL,

PRIMARY KEY (`id\_coeff`),

KEY `id\_compet` (`id\_compet`),

KEY `id\_compet\_2` (`id\_compet\_2`),

CONSTRAINT `conformity\_ibfk\_1` FOREIGN KEY (`id\_compet`) REFERENCES `competence` (`id\_compet`),

CONSTRAINT `conformity\_ibfk\_2` FOREIGN KEY (`id\_compet\_2`) REFERENCES `competence\_2` (`id\_compet\_2`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `discipline`

--

DROP TABLE IF EXISTS `discipline`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `discipline` (

`id\_discipline` char(7) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`name\_discipline` varchar(50) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`control\_discipline` char(8) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

PRIMARY KEY (`id\_discipline`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `discipline\_2`

--

DROP TABLE IF EXISTS `discipline\_2`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `discipline\_2` (

`id\_discipline\_2` char(7) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`name\_discipline\_2` varchar(50) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`control\_discipline\_2` char(7) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

PRIMARY KEY (`id\_discipline\_2`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `mod\_comp`

--

DROP TABLE IF EXISTS `mod\_comp`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `mod\_comp` (

`id\_module` char(7) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`id\_compet` char(7) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

PRIMARY KEY (`id\_module`),

KEY `id\_compet` (`id\_compet`),

CONSTRAINT `mod\_comp\_ibfk\_1` FOREIGN KEY (`id\_compet`) REFERENCES `competence` (`id\_compet`),

CONSTRAINT `mod\_comp\_ibfk\_2` FOREIGN KEY (`id\_module`) REFERENCES `module` (`id\_module`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `mod\_comp\_2`

--

DROP TABLE IF EXISTS `mod\_comp\_2`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `mod\_comp\_2` (

`id\_module\_2` char(7) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`id\_compet\_2` char(7) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

KEY `id\_compet\_2` (`id\_compet\_2`),

KEY `id\_module\_2` (`id\_module\_2`),

CONSTRAINT `mod\_comp\_2\_ibfk\_1` FOREIGN KEY (`id\_compet\_2`) REFERENCES `competence\_2` (`id\_compet\_2`),

CONSTRAINT `mod\_comp\_2\_ibfk\_2` FOREIGN KEY (`id\_module\_2`) REFERENCES `module\_2` (`id\_module\_2`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `module`

--

DROP TABLE IF EXISTS `module`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `module` (

`id\_module` char(7) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`name\_module` text CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`hours\_module` int(4) NOT NULL,

`id\_discipline` char(7) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

PRIMARY KEY (`id\_module`),

KEY `id\_discipline` (`id\_discipline`),

CONSTRAINT `module\_ibfk\_1` FOREIGN KEY (`id\_discipline`) REFERENCES `discipline` (`id\_discipline`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `module\_2`

--

DROP TABLE IF EXISTS `module\_2`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `module\_2` (

`id\_module\_2` char(7) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`name\_module\_2` text CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`hours\_module\_2` int(4) NOT NULL,

`id\_discipline\_2` char(7) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

PRIMARY KEY (`id\_module\_2`),

KEY `id\_discipline\_2` (`id\_discipline\_2`),

CONSTRAINT `module\_2\_ibfk\_1` FOREIGN KEY (`id\_discipline\_2`) REFERENCES `discipline\_2` (`id\_discipline\_2`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `question`

--

DROP TABLE IF EXISTS `question`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `question` (

`id\_question` int(4) NOT NULL,

`id\_compet\_2` char(7) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`question` text CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`answer\_a` varchar(21) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`answer\_b` varchar(21) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`answer\_c` varchar(21) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`answer\_d` varchar(21) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`answer\_true` char(9) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

PRIMARY KEY (`id\_question`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `type\_compet`

--

DROP TABLE IF EXISTS `type\_compet`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `type\_compet` (

`id\_type` int(2) NOT NULL,

`name\_type` varchar(2) NOT NULL,

PRIMARY KEY (`id\_type`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;