ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ДЕННОЇ ОСВІТИ
ФОРМА НАВЧАННЯ ДЕННА
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕНХОЛОГІЙ

Допускається до захисту

Завідувач кафедри
_____Олена ОЛЬХОВСЬКА
(підпис)

«_____»_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему
«АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ ТРЕНАЖЕРУ «ФУНКЦІОНАЛЬНЕ ПРОГРАМУВАННЯ» ДИСТАНЦІЙНОГО НАВЧАЛЬНОГО КУРСУ «ТЕОРІЯ ПРОГРАМУВАННЯ»

зі спеціальності 122 «Комп'ютерні науки»
освітня програма «Комп'ютерні науки»
ступеня бакалавра

Виконавець роботи Дзюба Олександр Петрович

_____
«___»_____2023 р.
(підпис)

Науковий керівник к., ф.-м. н., доц., Черненко Оксана Олексіївна

_____ «___»_____2023 р.
(підпис)

ПОЛТАВА 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних позначень, скорочень, символів
ЕОМ	Електронно-обчислювальна машина
ЕП	Електронний підручник
ІСР	Інтегроване середовище розробки
ПЗ	Програмне забезпечення
IDE	Integrated development environment

ВСТУП

Використання веб-технології помітно зростає в усіх сферах людського життя, зокрема це стосується сфери освіти. Однією з головних причин посиленої уваги педагогів до проблеми упровадження веб-технологій є зручність та простота використання наявних інструментів для пошуку, створення та використання веб-ресурсів. Використовуючи веб-ресурси, можна суттєво підвищити ефективність навчального процесу, активізувати навчально-пізнавальну та самостійну діяльність.

Візуалізація навчальної інформації у різних своїх формах дозволяє вирішити цілий ряд педагогічних завдань: забезпечення інтенсифікації навчання, активізації навчальної та пізнавальної діяльності, формування і розвиток критичного і візуального мислення, зорового сприйняття, образного представлення знань і навчальних дій, передачі знань і розпізнавання образів, підвищення візуальної грамотності та візуальної культури.

Метою роботи є алгоритмізація та програмування тренажеру «Функціональне програмування» дистанційного навчального курсу «Теорія програмування».

Об'єктом розробки є програмна реалізація тренажера для дистанційного курсу «Теорія програмування».

Предметом розробки є програмно реалізований навчальний тренажер з теми «Функціональне програмування».

Методи, які були використанні при розробці. Для розроблення алгоритму використано функціональне програмування. Для програмної реалізації тренажера використано мову програмування C#.

Новизною даної роботи є створення раніше не існуючого навчального тренажера з теми «Функціональне програмування» дистанційного курсу «Теорія програмування».

Практична значимість роботи полягає в створенні тренажера з теми «Функціональне програмування» для навчання студентів. Розроблений

навчальний тренажер рекомендовано використовувати в дистанційному навчальному курсу «Теорія програмування» студентами спеціальності 122 «Комп’ютерні науки» в ПУЕТ. Навчальний тренажер впроваджений в дистанційний навчальний курс з дисципліни «Теорія програмування».

Пояснювальна записка до дипломної роботи складається з чотирьох розділів. Перший розділ містить постановку задачі. У другому розділі описано аналіз комп’ютерних тренажерів та їх видів, огляд навчальних тренажерів розроблених в університеті, описані їх переваги та недоліки, вибір середовища для розробки програмного забезпечення та програмування, а також проаналізовано необхідність та актуальність теми. Третій розділ містить огляд матеріалу з теми, розроблений алгоритм та блок-схему алгоритму начального тренажера. У четвертому розділі описано процес програмної реалізації тренажера, перевірено правильність роботи на кожному кроці навчального тренажера.

Обсяг пояснювальної записки до диплому складає: 82 сторінки, з яких основна частина 53 сторінки та 29 сторінок додатку, літературних джерел – 14 назв, рисунків – 33, блок-схем – 1 сторінка.

1. ПОСТАНОВКА ЗАДАЧІ

1.1.Постановка задачі

Процес розв’язку більшості задач з використанням ЕОМ можна розбити на декілька характерних етапів: постановка задачі, описання алгоритму, запис і трансляція програми, налагодження програми, експлуатація програми.

І етап. Постановка задачі

Для того, щоб розв’язати задачу, пов’язану з дослідженням об’єкту, необхідно спочатку описати цей об’єкт. Задачею кваліфікаційної роботи є алгоритмізація та програмування тренажера «Функціональне програмування» дистанційного навчального курсу «Теорія програмування».

ІІ етап. Описання алгоритму

Після того, як описаний об’єкт дослідження, необхідно описати алгоритм роботи. До кваліфікаційної роботи з теми «Функціональне програмування», необхідно розробити питання/завдання, які проведуть перевірку/вдосконалять знання користувача з даної теми.

Побудова алгоритму носить творчий характер – кожна нова задача вимагає нових підходів і нових способів розв’язку, але для того, щоб побудувати алгоритм, який можна легко зрозуміти, модифікувати й удосконалити, необхідно дотримуватися певної дисципліни та технології його конструювання.

ІІІ етап. Запис і трансляція програми

Коли алгоритм складено, залишається більш технічне завдання, реалізувати тренажер. Враховуючи характер задачі необхідно вибрати певну мову програмування для запису програми та записати її текст.

Кожна мова програмування має свій транслятор, що дозволяє автоматично перекласти текст програми на машину мову. При цьому одержується текст програми на машинній мові, який еквівалентний тексту

програми на мові програмування, тому що ці тексті реалізують один і той же алгоритм.

Так як алгоритм містить вхідні дані та у якому порядку вони будуть показуватися користувачу, то можна з легкістю і точністю реалізувати перевірку правильності отриманої відповіді та вивести результат.

IV етап. Налагодження програми

На цьому етапі виявляються можливі помилки, які допущені на попередніх етапах. Синтаксичні помилки у тексті програми автоматично виявляються ще на етапі трансляції і вносяться відповідні зміни до тексту. Може статися так, що текст програми записано вірно, а помилка допущена при складанні алгоритму – програма працює, але видає неправильні результати. Якщо ж помилку допущено на етапі постановки задачі, то програма працює правильно, але розв’язує іншу задачу.

Суть налагодження полягає у тому, що користувач розробляє систему тестів, за допомогою якої перевіряється робота програми у різних можливих режимах. Кожен тест має набір вхідних даних, для яких відомий результат. Тест намагаються вибрати так, щоб не тільки встановити сам факт помилки, але й локалізувати її, тобто виявити частину програми, що містить помилку.

V етап. Експлуатація програми

Якщо розроблена програма розрахована на тривалу експлуатацію, то необхідно на етапі реалізації прорахувати всі можливі сумісництва з дистанційною платформою навчання [1].

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1.Комп’ютерні тренажери та їх види

У багатьох комп’ютерних іграх гравець виступає водієм перегонового автомобіля або мотоцикла і керує ним, намагаючись першим дістатися фінішу всупереч усім перешкодам. Такі ігри є добрим способом тренування реакції водія, і здобуті навички стануть у нагоді, коли гравець і насправді візьметься за кермо автомобіля.

Комп’ютерні тренажери є незамінними при підготовці майбутніх космонавтів і пілотів надшвидкісних літаків. Працюючи з таким тренажером, пілот відчуває себе нібито в кабіні справжнього літака. Комп’ютер відтворює різні кризові ситуації: літак потрапляє в грозові розряди, виходить із ладу те чи інше обладнання, виникають пожежі тощо. Такі тренування є незамінними, тому що пілот має бути готовим до дій у подібних ситуаціях, а не тільки вивчати комплексом програмного й технічного обладнання.

Вивчення студентських або шкільних предметів невід’ємне від тренування. Адже студент/учень має впевнено розв’язувати задачі, набути різні навички і т. ін. Для тренування навичок з розв’язання типових задач використовуються програми-тренажери.

Програми-тренажери забезпечує:

- послідовне виведення на екран завдань заданої складності з вибраної теми;
- контроль за діями користувача з розв’язання запропонованого завдання;
- миттєву реакцію на неправильні дії;
- виправлення помилок користувача;
- демонстрацію правильного розв’язання завдання;
- виведення підсумкового повідомлення про результати роботи користувача (можливо, з рекомендаціями чи порадами).

При роботі з програмою-тренажером кожний студент/учень підпадає під пильне «око» комп’ютера, який терпляче виправляє його помилки і не виводить оцінку в журнал, а надає можливість вдосконалювати навички до бажаного рівня.

Види віртуальних тренажерів.

Згідно виконуваним функціям віртуальні тренажери можна розділити на групи:

1. Навчаючі знань тренажери – Електронні підручники (ЕП). За рахунок широкого використання засобів мультимедіа (графіки, анімації, звуку) істотно підвищується ефективність навчання. Сучасні технології дозволяють легко доповнювати присутні в ЕП математичні формули «спливаючими» підказками, а графічні ілюстрації – контекстними пояснення.

2. Контролюючі тренажери – програми тестування, призначені для перевірки знань студента по темах лабораторних робіт. Вони можуть застосовуватися для самопідготовки або для отримання теоретичного допуску до роботи. До складу тестів можна включати питання, що дозволяють встановити ступінь готовності студента до осмисленої роботи з тренажером. Для посилення контролюючого ефекту результати проходження тестів оцінюються в балах, студентові повідомляється кількість пройдених тестів і сума штрафних балів. Для мінімізації вгадування відповідей в програмі блокується вивід на екран інформації з результатом кожного окремого тесту.

3. Навчаючі вміння тренажери – мультимедійні анімаційні імітатори, призначені для імітації зміни станів фізичного устаткування (приладів, пристроїв) за різних умов, створюючи ілюзію дій з фізичною апаратурою. Основною їх особливістю є максимально повне відтворення зовнішнього вигляду фізичних пристроїв (передніх панелей, шкал, стрілок і інших елементів показуючих і реєструючих приладів) і елементів управління ними (кнопок, тумблерів, перемикачів), а також рухи окремих елементів відповідно до дій користувача на основі створення анімаційних об’єктів і складних сцен. Студент дістає можливість детально розглянути технічний пристрій, ознайомитися з

його деталями, а також виконати обмежений набір дій, пов’язаних з розбиранням або настройкою приладу.

2.2.Огляд навчальних тренажерів розроблених в університеті

Для розгляду взяті навчальні тренажери з навчального дистанційного курсу «Теорія програмування» та «Теорія алгоритмів».

Першим для розгляду взято навчальний тренажер студента Алексова С. В. з теми «Алгоритмізація та програмування тренажера «Дерева розбору» дистанційного навчального курсу «Теорія програмування» [2].

Переваги навчального тренажеру:

- Є кнопка «Інформаційна сторінка» для ознайомлення студентом теоретичного матеріалу.
- Два розділи для тестування (теоретичний та практичний).
- Присутні підказки для студента.
- Підсумок правильних відповідей.
- Вибір складності для практичних завдань.

Наступним для розгляду взято тренажер студента Цехмейстера В. О. з теми «Тренажер з теми «Обрізання основ» дистанційного навчального курсу «Теорії програмування» та розробка його програмного забезпечення [3].

Переваги навчального тренажеру:

- Є можливість перед початком роботи переглянути теоретичний матеріал.
- Два розділи: теоретичні питання та практичні завдання.

Третім навчальним тренажером було розглянуто Закірова Р. Р. на тему «Програмне забезпечення для тренажера з теми «Лямбда-вирахування і типізація» дистанційного навчального курсу «Теорія програмування» [4].

Наступні переваги має цей навчальний тренажер:

- Теоретичний матеріал можна відкрити або в новому вікні або зберегти на персональний комп’ютер.

- При помилці відбувається показ підказки.

- Після закінчення, показує на які питання були дані правильні відповіді, а на які неправильні.

Четвертим навчальним тренажером було розглянуто Волосевича М. В. на тему «Тренажер з теми «Марківські підстановки» дистанційного навчального курсу «Теорія алгоритмів» та розробка його програмного забезпечення» [5].

Наступні переваги, які має навчальний тренажер:

- Перед початком можна ознайомитись з теоретичною інформацією. Після чого можна повернутись до початку або перейти вже до завдань.

- Має два види завдань: тести (тестові запитання по теоретичній інформації та практичні завдання – розв'язок прикладів з теми).

П'ятим навчальним тренажером було розглянуто Будяй В. О. на тему «Тренажер з теми «Математичні основи теорії алгоритмів» дистанційного навчального курсу «Теорія програмування» та його програмного забезпечення» [6].

Даний тренажер має ряд наступних переваг:

- Різноманітність завдань (тестові, на введення відповіді).
- Виведення результатів проходження тренажеру.

Отже, в ході розгляду навчальних тренажерів розроблених в університеті було з'ясовано, що однією з головних вимог є різноманітність завдань (теоретичних та практичних питань), а також будь-яка допомога студентові згадати або вивчити матеріал по темі, яка реалізована у вигляді підказок/теоретичному матеріалі.

2.3.Необхідність та актуальність теми роботи

Освітні трансформації на сучасному етапі розвитку суспільства базуються на зміні вектору організації навчального процесу – від традиційного, орієнтованого на засвоєння певної сукупності знань, на процес підготовки людини інноваційного типу, яку може сформувати лише інноваційна за своєю

сутністю освіта [1], базована на технологізації навчально-виховного середовища.

Таким чином, унаслідок використання інтерактивних технологій у навчальному процесі відбувається підвищення рівня різних видів активності студентів. Результатом прогресивної динаміки розумової активності є інтенсивність мислення, генерування ідей, висловлювання припущень, проектування, конструювання, моделювання, дослідження, виявлення творчої уяви, зосередженості, уваги, спостережливості, здійснення аналітико-синтетичних операцій [1].

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Опис матеріалу з теми роботи

Функціональне програмування використовує парадигму, яка повністю відмінна від імперативної. Функціональна програма являє собою деякий вираз, виконання програми означає обчислення цього виразу.

При порівнянні функціонального й імперативного підходів до програмування можна помітити такі властивості програм.

Наведене стосується чистих функціональних мов, прикладом яких є ранні версії Lisp.

1. Функціональні програми не використовують змінних у тому значенні, в якому вони використовуються в імперативному програмуванні. (На рівні машинної мови проміжні змінні існують, однак деталі сховані від програміста). Зокрема у функціональних програмах не використовуються оператори присвоювання. Тому функціональна програма не може мати побічних ефектів.

2. Як наслідок з попереднього пункту, у функціональних програмах немає циклів. Замість функціональні програми широко використовують рекурсію.

3. Функціональні програми використовують функції складним способом. Їх можна передавати в інші функції як аргумент і повертати як результат.

4. Єдиним способом поділу програми на частини є введення імені для функції і задання виразу для цього імені.

Сучасні функціональні мови, як правило, мають імперативні властивості. Так можуть бути і змінні, і присвоювання, і цикли.

Найбільш відомим серед сучасних функціональних мов є Common Lisp, Scheme, Haskell.

Якщо програма інтенсивно використовує функції, у тому числі логічні, наприклад, в задачах штучного інтелекту, функціональний підхід має ряд переваг перед імперативним саме:

- Функціональні програми більш безпосередньо відповідають математичним об’єктам, а отже, дозволяють проводити строгі доведення. Установити значення імперативної програми, тобто тієї, яку вона реалізує, може виявитися більш складно, ніж функціональної.

Наприклад, розглянемо таку програму на Haskell:

*factorial n = if n == 0 then 1 else n * factorial(n-1)*

Практично відразу видно, що ця програма відповідає функції *f(n) = n!* для $n \geq 0$ і невизначена для *n < 0*.

Для програми на Сі ця відповідь неочевидна:

```
inf f(int n)
{
int x = 1;
while (n > 0)
{
x = x * n;
n = n – 1;
}
return x;
}
```

- При виконанні функціональних програм можна виконувати паралельні обчислення, оскільки компоненти виразів можуть виконуватися одночасно.

Слід зазначити, що функції мови Сі та інших імперативних мов не є строго математичними функціями. Їх значення може залежати від аргументів. Результат виконання можуть бути різні побічні ефекти (наприклад, зміна значення глобальної змінної). Два виклики однієї і тієї самої функції і тими ж аргументами може привести до різних результатів. Функції у функціональних

програмах не мають цих недоліків. Порядок обчислення підвиразів не впливає на результат.

- Математичною моделлю для функціонального програмування є лямбда-вирахування Черча.

Як правило, функція представляється як відображення однієї множини в іншу. У λ-вирахуванні функції у першу чергу розглядаються як математичні вирази щодо деяких параметрів. Λ-функції є анонімними функціями.

У чистому λ-вирахуванні функції мають тільки один аргумент, у розширеному λ-вирахуванні функції можуть мати кілька аргументів.

Мова функціонального програмування Lisp.

Розглянемо основи найбільш відомої мови функціонального програмування Lisp.

Мова програмування Lisp оперує з двома основними структурами даних – атомами і списками. Атоми і списки не є типами даних у тому значенні, яке надається типам в процедурних мовах програмування. Структури даних у мові Lisp – спосіб представлення одних об’єктів через сукупність інших, більш простих.

Атом – елементарний об’єкт, який не може бути розбитий на складові. У програмі на Lisp атоми можуть позначити константи і функції, що мають фіксований зміст; визначені користувачем константи, змінні і функції; і, нарешті, атоми можуть позначати самих себе, тобто ті послідовності символів, якими атоми зображуються. Остання обставина дозволяє використовувати мову Lisp як базову мову під час організації символьних і аналітичних обчислень, тому що дає можливість працювати з об’єктами, яким не присвоєно ніякого конкретного значення. На відміну від змінної у процедурних мовах програмування атом являє собою спосіб маркування будь-якого об’єкта, припустимого в Lisp.

Приклад атомів: 3.14, *, nil, book.

Атоми в Lisp можуть бути вільними або зв’язаними. Вільний атом позначає сам себе, тобто послідовність символів, що утворюють його ім’я.

Зв’язаний атом може позначати будь-який припустимий Lisp-вираз, атом або список. Установити деяке значення атома або змінити його можна функцією setq, наприклад (setq A 125).

Особливу роль у мові відіграють атоми T і NIL. У логічних виразах вони мають значення true і false відповідно. Крім того, атом NIL еквівалентний порожньому спискові.

Списки є основною структурою даних мови Lisp. За допомогою списків можна представити практично будь-який математичний об’єкт: множину, тензор, граф, і т. д. Функції також можуть бути подані у вигляді списку. Останнє твердження має важливі наслідки, тому що будь-яка програма мовою Lisp є якою-небудь функцією, що може мати своїм значенням список, який представляє іншу функцію. Таким чином, виконання програм на Lisp може продовжувати нову програму.

Списком називається послідовність елементів, взятих у дужки, елементом списку може бути або атом, або список.

Приклад списків: (1 2 3), (name price), (a(b c)), ().

У вигляді списків записується виклики функцій, наприклад: (sum 1 2 3), (+ 5 7).

У вигляді списків можна подати предикати.

Розглянемо основні функції мови Lisp. Оскільки основними структурами є списки, основними функціями мови є функції роботи зі списками.

Функція *list* повертає список, складений із значень своїх аргументів. Наприклад: (*list* 1 2 3) → (1 2 3).

Функція *nth* вибирає 1-й елемент списку. Наприклад: (*nth* 0’ (a b c)) → a. Апостроф перед дужками запобігає оцінюванню виразу (a b c).

Функція *length* повертає довжину списку, наприклад: (*length* (a b c d)) → 4.

Функція *member* перевіряє, чи є елемент членом списку, наприклад: (*member* 7’ (1 2 3)) → *nil*.

Функція *car* повертає перший елемент списку, наприклад: (*car* ’(a b c)) → a.

Функція *cdr* повертає цей самий список після видалення 1-го елемента, наприклад: (*cdr* ’(a b c)) → (b c).

Функція *cons* додає до списку новий елемент. Другий аргумент функції – список, перший аргумент додається до початку списку, наприклад: (*cons* 1’(a b c)) → (1 a b c), (*cons* (*list* ’a) ’(a b c)) → ((a) a b c).

Функція *append* об’єднує два списки, наприклад: (*append* (*list* 1 2) (*list* 3 4)) → (1 2 3 4).

3.2. Алгоритм навчального тренажера

Початковий крок. На панелі виводиться інформація:

- тема;
- назва дистанційного курсу;
- ПІБ автора;
- поточний рік;
- кнопка для переходу до початку тестування.

При натисканні на кнопку відображається перший крок алгоритму.

1 крок. Оберіть відповідь. На панелі виводиться питання та варіанти відповідей:

Що являє собою функціональна програма?

- Деяку константу
- Деяке значення
- Деякий вираз
- Деякий масив
- Деякий набір

Якщо обрано неправильні варіанти вказується помилка, інакше відбувається перехід далі на крок.

2 крок. *Оберіть чи вірне твердження, яке представлене в таблиці.* На панелі виводиться питання та варіанти відповідей:

При порівнянні функціонального й імперативного підходів до програмування які властивості чистих функціональних програм можна помітити?

Властивості	Оберіть відповідь
У функціональних програмах немає циклів.	+
Функціональні програми використовують функції більш легким способом.	-
У функціональних програмах не використовують оператори присвоєння.	+
У функціональних програмах використовують лише цикли.	-
Замість циклів функціональні програми широко використовують рекурсію.	+
Функціональні програми використовують функції для передачі в інші функції як аргумент так і повернення як результат.	+
У функціональних програмах використовують оператори присвоєння.	-
Функціональні мови використовують функції більш складним способом.	+
Єдиним способом поділу програми на частини є введення імені для функції і задання виразу для цього імені.	+

Якщо обрано неправильні варіанти вказується помилка, інакше відбувається перехід далі на крок.

3 крок. *Оберіть відповіді.* На панелі виводиться питання та варіанти відповідей:

Найбільш відомими серед сучасних функціональних мов є

- Scheme

- Common Lisp
- C++
- Haskell
- Python
- C#
- PHP

Якщо обрано неправильні варіанти вказується помилка, інакше відбувається перехід далі на крок.

4 крок. *Оберіть відповідь.* На панелі виводиться питання та варіанти відповідей:

Чи мають сучасні функціональні мови імперативні властивості (змінні, присвоєння, цикли)?

- Так
- Ні

Якщо обрано неправильні варіанти вказується помилка, інакше відбувається перехід далі на крок.

5 крок. *Оберіть відповіді.* На панелі виводиться питання та варіанти відповідей:

Якщо програма інтенсивно використовує функції, у тому числі логічні, то функціональний підхід має ряд переваг перед імперативним, а саме:

- Функціональні програми менш безпосередньо відповідають математичним об’єктам.
- Математичною моделлю для функціонального програмування є лінійна задача.
- Функціональні програми більш безпосередньо відповідають математичним об’єктам.
- При виконанні функціональних програм можна використовувати паралельні обчислення.

•Математичною моделлю для функціонального програмування є лямда-вирахування Черча.

Якщо обрано неправильні варіанти вказується помилка, інакше відбувається перехід далі на крок.

6 крок. *Оберіть відповіді.* На панелі виводиться питання та варіанти відповідей:

Якими структурами даних оперує мова програмування Lisp?

•Масивами

•Списками

•Константами

•Аргументами

•Атомами

Якщо обрано неправильні варіанти вказується помилка, інакше відбувається перехід далі на крок.

7 крок. Оберіть відповідь. На панелі виводиться питання та варіанти відповідей:

Що являється основною структурою даних мови Lisp?

•Атом

•Список

Якщо обрано неправильні варіанти вказується помилка, інакше відбувається перехід далі на крок.

8 крок. Оберіть відповідь. На панелі виводиться питання та варіанти відповідей:

Що являє собою атом в мові Lisp?

•Складний об’єкт, який може бути розбитий на складові.

•Елементарний об’єкт, який не може бути розбитий на складові.

Якщо обрано неправильні варіанти вказується помилка, інакше відбувається перехід далі на крок.

9 крок. *Оберіть відповіді.* На панелі виводиться питання та варіанти відповідей:

У програмі на Lisp атоми можуть позначати:

- Цикли.
- Константи і функції, що мають фіксований зміст.
- Позначати самих себе.
- Визначені користувачем константи, змінні і функції.
- Умовні оператори.

Якщо обрано неправильні варіанти вказується помилка, інакше відбувається перехід далі на крок.

10 крок. Оберіть відповіді. На панелі виводиться питання та варіанти відповідей:

Які можуть бути атоми в Lisp?

- Вільні
- Залежні
- Зв’язані
- Присвоєнні

Якщо обрано неправильні варіанти вказується помилка, інакше відбувається перехід далі на крок.

11 крок. *Оберіть відповідь.* На панелі виводиться питання та варіанти відповідей:

Що позначає вільний атом?

- Позначає будь-який припустимий Lisp-вираз, атом або список.
- Позначає сам себе, тобто послідовність символів, що утворюють його ім’я.

Якщо обрано неправильні варіанти вказується помилка, інакше відбувається перехід далі на крок.

12 крок. *Оберіть відповідь.* На панелі виводиться питання та варіанти відповідей:

Що позначає зв’язаний атом?

- Позначає будь-який припустимий Lisp-вираз, атом або список.

•Позначає сам себе, тобто послідовність символів, що утворюють його ім’я.

Якщо обрано неправильні варіанти вказується помилка, інакше відбувається перехід далі на крок.

13 крок. *Оберіть відповідь.* На панелі виводиться питання та варіанти відповідей:

Чи можна за допомогою списків представити практично будь-який математичний об’єкт в Lisp?

•Так

•Ні

Якщо обрано неправильні варіанти вказується помилка, інакше відбувається перехід далі на крок.

14 крок. *Оберіть відповідь.* На панелі виводиться питання та варіанти відповідей:

Чи можуть функції бути подані у вигляді списку в мові Lisp?

•Так

•Ні

Якщо обрано неправильні варіанти вказується помилка, інакше відбувається перехід далі на крок.

15 крок. *Оберіть відповідь.* На панелі виводиться питання та варіанти відповідей:

Що називається списком в мові Lisp?

•Списком називається послідовність елементів, взятих у дужки, елементом списку може бути або атом, або список.

•Списком називається функція, яка містить атоми.

•Списком називається послідовність елементів, яку не треба брати в дужки, а елементами списку може бути атом, або список.

Якщо обрано неправильні варіанти вказується помилка, інакше відбувається перехід далі на крок.

16 крок. Введіть відповідь. На панелі виводиться питання та поле для введення:

Напишіть результат виконання функції

(*list* 1 2 3) → (?)

(*list* 1 2 3) → (1 2 3)

Якщо введено невірну відповідь, з'являється підказка «Функція *list* повертає список, складений із значень своїх аргументів.», інакше відбувається перехід далі на крок.

17 крок. Введіть відповідь. На панелі виводиться питання та поле для введення:

Напишіть результат виконання функції

(*list* a c b) → (?)

(*list* a c b) → (a c b)

Якщо введено невірну відповідь, з'являється підказка «Функція *list* повертає список, складений із значень своїх аргументів.», інакше відбувається перехід далі на крок.

18 крок. Введіть відповідь. На панелі виводиться питання та поле для введення:

Напишіть результат виконання функції

(*nth 0'* (a b c)) → (?)

(*nth 0'* (a b c)) → *a*

Якщо введено невірну відповідь, з'являється підказка «Функція *nth* вибирає 1-й елемент списку.», інакше відбувається перехід далі на крок.

19 крок. Введіть відповіддю. На панелі виводиться питання та поле для введення:

Напишіть результат виконання функції

(*nth 0'* (3 5 6)) → (?)

(*nth 0'* (3 5 6)) → *3*

Якщо введено невірну відповідь, з'являється підказка «Функція *nth* вибирає 1-й елемент списку.», інакше відбувається перехід далі на крок.

20 крок. Введіть відповідь. На панелі виводиться питання та поле для введення:

Напишіть результат виконання функції

(*length* (a b c d)) → (?)

(*length* (a b c d)) → *4*

Якщо введено невірну відповідь, з'являється підказка «Функція *length* повертає довжину списку.», інакше відбувається перехід далі на крок.

21 крок. Введіть відповідь. На панелі виводиться питання та поле для введення:

Напишіть результат виконання функції

(*length* (11 1 3 0 7 8)) → (?)

(*length* (11 1 3 0 7 8)) → *6*

Якщо введено невірну відповідь, з'являється підказка «Функція *length* повертає довжину списку.», інакше відбувається перехід далі на крок.

22 крок. Оберіть відповідь із випадаючого списку. На панелі виводиться питання та поле із випадаючим списком:

Оберіть результат виконання функції

(member 7' (1 2 3)) → (?)

- 7 1 2 3
- Nil
- 1 2 3 7

Якщо обрано невірну відповідь, з'являється підказка «Функція *member* перевіряє, чи є елемент членом списку.», інакше відбувається перехід далі на крок.

23 крок. Оберіть відповідь із випадаючого списку. На панелі виводиться питання та поле із випадаючим списком:

Оберіть результат виконання функції

(*member* b' (a c b d)) → (?)

- a c b d
- b d

- nil
- a c b d b
- b a c b d

Якщо обрано невірну відповідь, з’являється підказка «Функція *member* перевіряє, чи є елемент членом списку.», інакше відбувається перехід далі на крок.

24 крок. Введіть відповідь. На панелі виводиться питання та поле для введення:

Напишіть результат виконання функції

(*car* ’ (a b c)) → (?)

(*car* ’ (a b c)) → *a*

Якщо введено невірну відповідь, з’являється підказка «Функція *car* повертає перший елемент списку.», інакше відбувається перехід далі на крок.

25 крок. Введіть відповідь. На панелі виводиться питання та поле для введення:

Напишіть результат виконання функції

(*cdr* ’ (a b c)) → (?)

(*cdr* ’ (a b c)) → *(b c)*

Якщо введено невірну відповідь, з’являється підказка «Функція *cdr* повертає частину списку після першого елементу.», інакше відбувається перехід далі на крок.

26 крок. Введіть відповідь. На панелі виводиться питання та поле для введення:

Напишіть результат виконання функції

(*cons* 1’ (a b c)) → (?)

(*cons* 1’ (a b c)) → *(1 a b c)*

Якщо введено невірну відповідь, з’являється підказка «Функція *cons* додає до списку новий елемент.», інакше відбувається перехід далі на крок.

27 крок. Введіть відповідь. На панелі виводиться питання та поле для введення:

Напишіть результат виконання функції

(*append* (*list* 1 2) (*list* 3 4)) → (?)

(*append* (*list* 1 2) (*list* 3 4)) → *(1 2 3 4)*

Якщо введено невірну відповідь, з’являється підказка «Функція *append* об’єднує два списки.», інакше відбувається перехід далі на крок.

28 крок. Розставити відповідність. На панелі виводяться назви функцій та варіанти виконання, які потрібно розставити відповідно.

4	(*cons* (*list* ’a) (*list* ’a b c))
1	(*car* ’ (15 3 6 8 9))
2	(*append* (*list* a b) (*list* 3 8))
3	(*cdr* ’ (15 3 6 8 9))

1	15
2	(a b 3 8)
3	(3 6 8 9)
4	((a) a b c)

Якщо обрано невірні відповіді, з’являється підказка «Функція *cdr* повертає частину списку після першого елементу. Функція *car* повертає перший елемент списку. Функція *cons* додає до списку новий елемент. Функція *append* об’єднує два списки.» інакше відбувається перехід далі на крок.

Кінцевий крок. На панелі виводиться інформація:

- повідомлення про завершення тестування;
- кнопка для повторного тестування;
- кнопка для завершення роботи з тренажером.

Якщо натиснути на кнопку для завершення, то тренажер закривається. Якщо натиснути на продовження, то відображається початковий крок.

3.3. Блок-схема навчального тренажера по алгоритму

Блок-схема алгоритму зображає послідовність блоків, з'єднаних між собою стрілками, які вказують послідовність виконання і зв'язок між блоками. Всередині блоків записується їх короткий зміст.

Блок-схема - це спосіб представлення алгоритму в графічній формі, у вигляді геометричних фігур, сполучених між собою лініями (стрілками). Форма блока визначає тип дії, а текст всередині блоку дає детальне пояснення

конкретної дії. Стрілки на лініях, що сполучають блоки схеми, вказують послідовність виконання команд, передбачених алгоритмом. На рисунку 3.1 зображено блок-схему алгоритму роботи навчального тренажера з теми «Функціональне програмування».

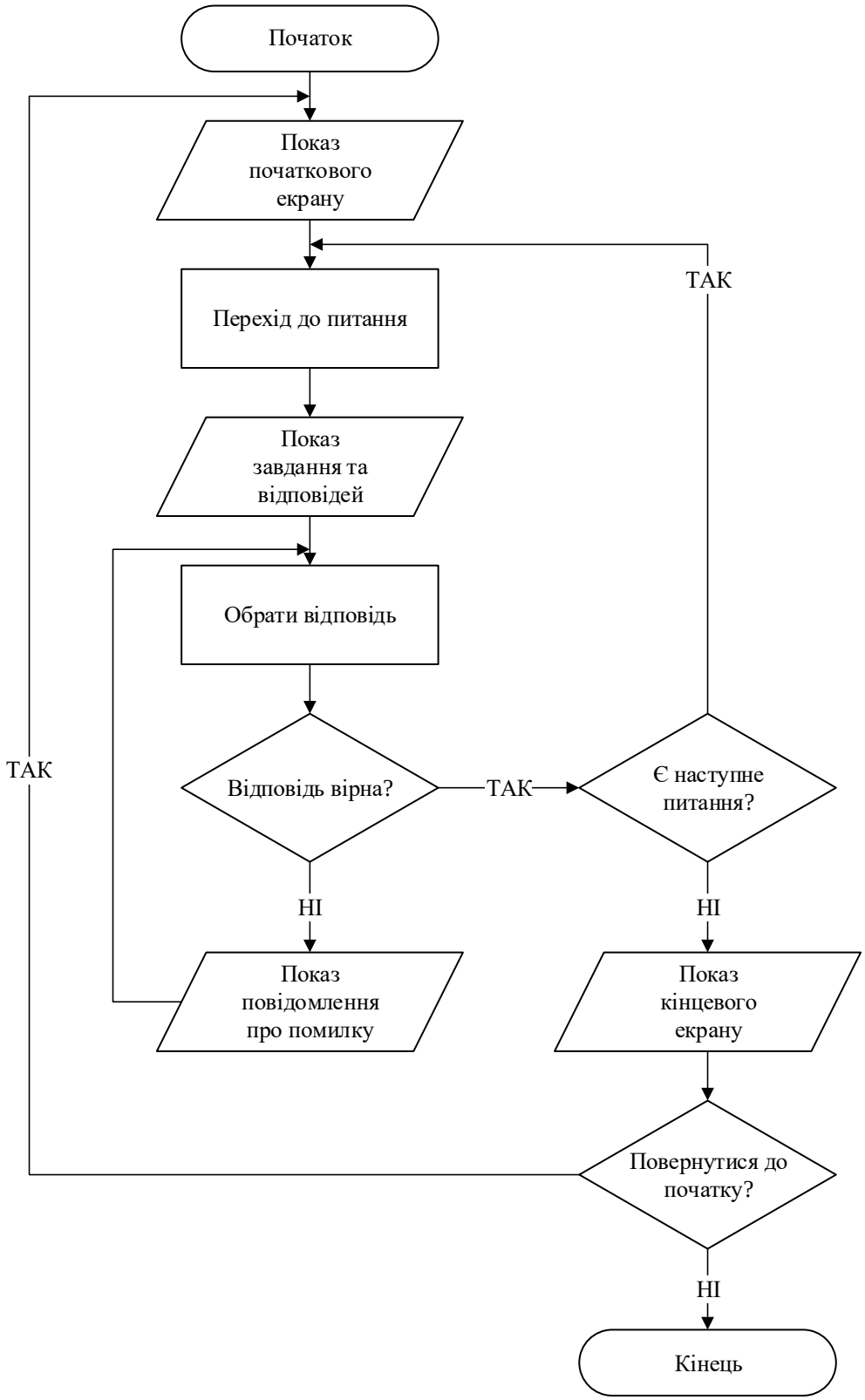

Рисунок 3.1 – Блок-схема алгоритму роботи навчального тренажера

4. ПРАКТИЧНА ЧАСТИНА

4.1. Середовище для розробки програмного забезпечення та програмування

Важливою складовою в процесі розробки програмного забезпечення вважається вибір вірною IDE, що залежить не лише тільки від платформи, але і значення особистої підготовки.

Інтегроване середовище розробки, ICP (Integrated development environment-IDE) – це комплекс програмних засобів, який використовується програмістами для розробки програмного забезпечення [8].

Впровадження ICP для розробки програмного забезпечення (ПЗ) вважається прямою протилежністю методиці, в якій застосовуються незв’язані інструменти, такі як текстовий редактор, компілятор, і т. п.

Інтегровані середовища розробки були створені для того, щоб максимізувати продуктивність програміста завдяки тісно пов’язаних компонентів з простими користувацькими інтерфейсами. Це дозволяє розробнику зробити менше дій для перемикання різних режимів, на відміну від дискретних програм розробки. Однак так як ICP є складним програмним комплексом, то середовище зможе якісно прискорити процес розробки ПЗ лише після спеціального навчання.

Для зменшення бар’єру входження багато досить інтерактивні, а для полегшення переходу з однієї на іншу інтерфейс у одного виробника максимально близький, аж до використання однієї ICP.

ICP зазвичай являє собою єдину програму, в якій проводиться вся розробка. Вона як правило, містить багато функцій для створення, зміни, компілювання, розгортання і налагодження програмного забезпечення. Мета інтегрованого середовища полягає в тому, щоб об’єднати різні утиліти в одному модулі, який дозволить абстрагуватися від виконання допоміжних завдань, тим самим дозволяючи програмісту зосередитися на вирішенні власне

алгоритмічної задачі і уникнути втрат часу при виконанні типових технічних дій (наприклад, виклику компілятора). Таким чином, підвищується продуктивність праці розробника. Також вважається, що тісна інтеграція завдань розробки може далі підвищити продуктивність за рахунок можливості додаткових функцій на проміжних етапах роботи. Наприклад, ICP дозволяє проаналізувати код і тим самим забезпечити миттєвий зворотний зв'язок і повідомити про синтаксичні помилки.

Більшість сучасних ICP є графічними. Але перші ICP використовувалися ще до того, як стали широко застосовуватися операційні системи з графічним інтерфейсом – вони були засновані на текстовому інтерфейсі з використанням функціональних і гарячих клавіш для виклику різних функцій (наприклад, Turbo Pascal, створений фірмою Borland).

Середовище розробки включає в себе:

- текстовий редактор;
- транслятор (компілятор та/або інтерпретатор);
- засоби автоматизації складання;
- відладчик.

IDE – це не просто текстовий редактор. У той час як текстові редактори коду, такі як Sublime або Atom, пропонують безліч зручних функцій, таких як підсвічування синтаксису, настроюваний інтерфейс і розширені засоби навігації, вони дозволяють тільки писати код. Для створення функціонуючих додатків як мінімум потрібен компілятор і відладчик [7].

IDE включає в себе ці компоненти, як і ряд інших. Деякі з них поставляються з додатковими інструментами для автоматизації, тестування та візуалізації процесу розробки. Термін «інтегроване середовище розробки» означає, що надається все необхідне для перетворення коду в функціонуючі програми.

Проаналізуємо найпоширеніше середовище для розробки ПЗ – Microsoft Visual Studio.

Microsoft Visual Studio – це інтегроване середовище розробки, ціна якої варіюється від $699 до $2900. Безліч версій цієї IDE здатні створювати всі типи програм, починаючи від веб-додатків і закінчуючи мобільними додатками, відеоіграми. Ця лінійка програмного забезпечення включає в себе безліч інструментів для тестування сумісності. Завдяки своїй гнучкості Visual Studio є відмінним інструментом для студентів і професіоналів. Один з найстаріших програмних продуктів для створення як консольних додатків, так і володіють графічним інтерфейсом. Додавання сторонніх плагінів дозволяє серйозно розширити функціональність середовища, в тому числі до кроссплатформенного стану.

Підтримувані мови: Ajax, ASP.NET, DHTML, JavaScript, Jscript, Visual Basic, Visual C#, Visual C++, Visual F#, XAML та інші. Інтерфейс Microsoft Visual Studio наведений на рис. 4.1 [8].

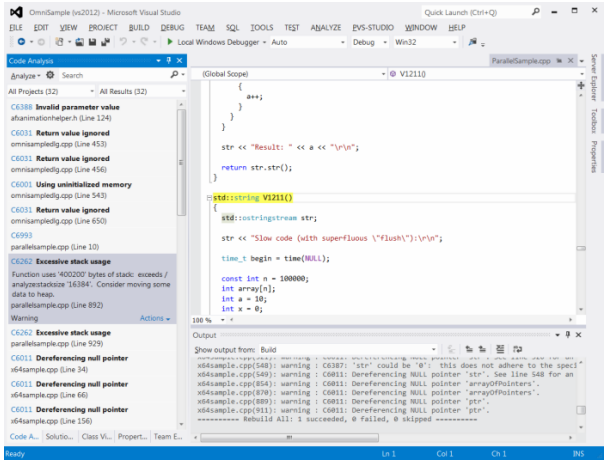

Рисунок 4.1 – Інтерфейс Microsoft Visual Studio

Особливості:

- величезна бібліотека розширень, яка постійно збільшується;
- IntellliSense;
- настроюється панель і закріплюється вікна;
- простий робочий процес і файлова ієрархія;
- статистика моніторингу продуктивності в режимі реального часу;

- інструменти автоматизації;
- легкий рефакторинг і вставка фрагментів коду;
- підтримка розділеного екрану;
- список помилок, який спрощує налагодження;
- перевірка затвердження при розгортанні додатків за допомогою CkickOnce, Windows Installer або Publish Wizard.

До недоліків можна віднести те, що оскільки Visual Studio є складною IDE, для відкриття та запуску додатків потрібні значні ресурси. Тому на деяких пристроях внесення простих змін може зайняти багато часу. Для простих завдань доцільно використовувати компактний редактор засіб розробки PHP. Новачкові буде просто неможливо самостійно розібратися з Visual Studio без проходження спеціальних курсів і читання літератури. Це продукт скоріше для досвідчених розробників, що звертають увагу на якість редактора і функції тестування.

Далі розглянемо середовище – NetBeans.

Безкоштовне середовище розробки з відкритим вихідним кодом. Підходить для редагування існуючих проектів або створення нових [8]. NetBeans пропонує простий drag-and-drop інтерфейс, який поставляється з великою кількістю зручних шаблонів проектів. Середовище в основному використовується для розробки Java додатків, але можна встановлювати пакети, що підтримують інші мови. Підтримують мови програмування C, C++, C++ 11, Fortan, HTML 5, Java, PHP та інші. Інтерфейс NetBeans наведено на рис. 4.2 [9].

Особливості:

- інтуїтивний drag-and-drop інтерфейс;
- динамічні та статичні бібліотеки;
- інтеграція декількох сесій GNU-відладчиа з підтримкою коду;
- можливість здійснювати віддалене розгортання;
- сумісність з платформами Windows, Linux, OS X і Solaris;
- підтримка Qt Toolkit;

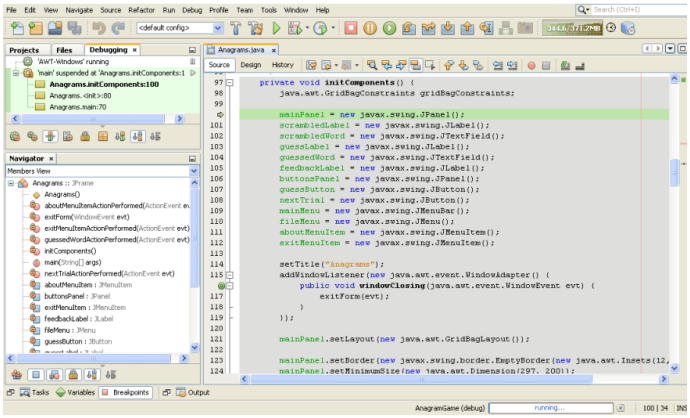

Рисунок 4.2 – Інтерфейс NetBeans

- підтримка Fortan і Assembler;

- підтримка цілого рядку компіляторів, включаючи Clang / LLVM, Cygwin, GNU, MinGW і Oracle Solaris Studio.

До недоліків можна віднести те, що це безкоштовне середовище розробки споживає багато пам’яті, тому може працювати повільно на деяких ПК.

Далі проаналізуємо PyCharm.

PyCharm розроблений командою JetBrains. Користувачам надається безкоштовна версія Community Edition, 30-денна безкоштовна ознайомча версія Professional Edition і річна підписки за $213-$690 на версію Professional Edition. Комплексна підтримка коду і аналіз роблять PyCharm кращою IDE для Python-програмістів. Підтримуються: AngularJS, CoffeeScript, CSS, Python, HTML, JavaScript, Node.js, TypeScript. Інтерфейс PyCharm наведено на рис. 4.3 [10].

Особливості:

- сумісність з операційними системами Windows, Linux і Mac OS;

- поставляється з Django IDE;

- легко інтегрується з Git, Mercurial і SVN;

- настроюваний інтерфейс з емуляцією VIM;

- відладчики JavaScript, Python і Django;

- підтримка Google App Engine.

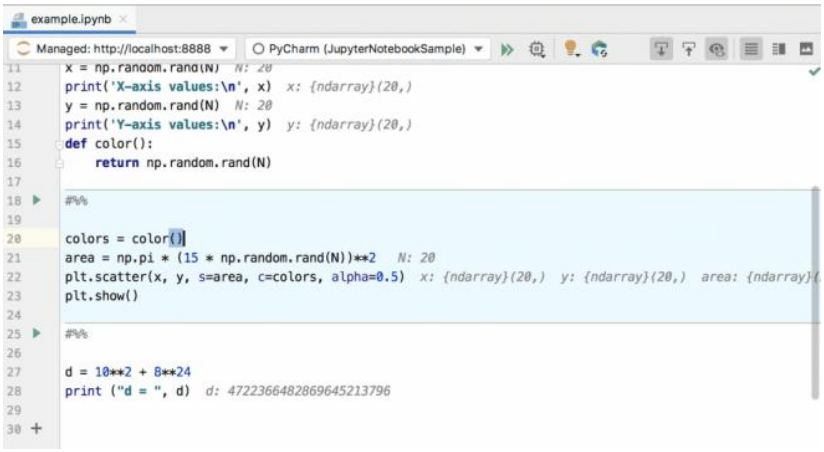

Рисунок 4.3 – Інтерфейс PyCharm

До недоліків можна віднести те, що користувачі скаржаться, що це середовище розробки Python містить деякі помилки, такі як періодично не працює функція автоматичного заповнення, що може доставити певні незручності.

Таким чином, можна зробити висновок, що для розробки інтерфейсу автоматизованої системи інтерфейсу користувача найбільш підходить середовище Visual Studio бо як мінімум має зручний та інтуїтивно зрозумілий інтерфейс та підтримка мови програмування C#, а також має простий робочий процес і файлову ієрархію.

4.2. Опис програмної реалізації

Для програмної реалізації було обрано середовище Visual Studio, в якому створено проект Windows Form (.NET Framework) (рис. 4.4) з платформою .NET Framework 4.7.2.

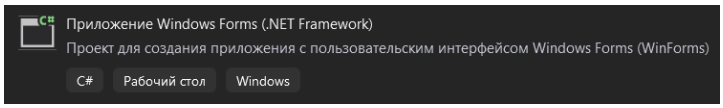

Рисунок 4.4 – Створення проекту

Після завантаження створеного проекту було додано 31 форму, з них одна початкова та дві після закінчення кожного з розділів (теоретичного/практичного), 15 – для теоретичних питань, 13 – для практичних завдань.

Для відображення кроків алгоритму, форми заповнені елементами, постійними є *Label* та *Button*. В залежності від питання/завдання кожен крок має різні елементи, щоб відповісти.

Для теоретичних питань було використано *RadioButton* (дозволяє студентові вибрати єдиний варіант), *CheckBox* (дозволяє студентові вибрати декілька варіантів) та *TableLayoutPanel* (панель, в якій вміст динамічно відображається у сітці, що складається з рядків та стовпців) з використанням *Label* і *CheckBox*. На рисунках 4.5-4.7 містять розміщення елементів на типових кроках з вибором однієї та декількох відповідей, а також у вигляді таблиці.

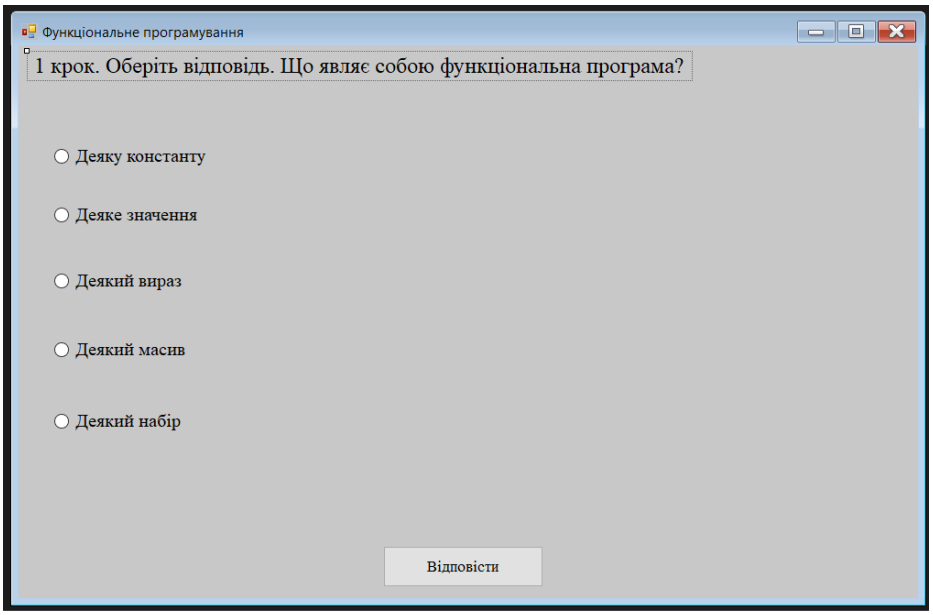

Рисунок 4.5 – Розміщення елементів в теоретичному питанні з вибором однієї відповіді

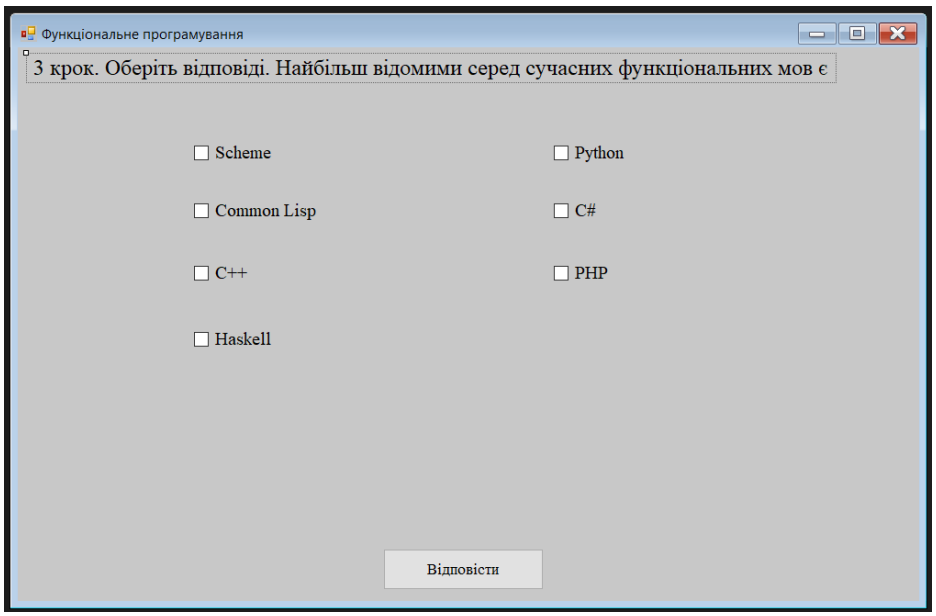

Рисунок 4.6 –Розміщення елементів в теоретичному питанні з вибором декількох відповіді

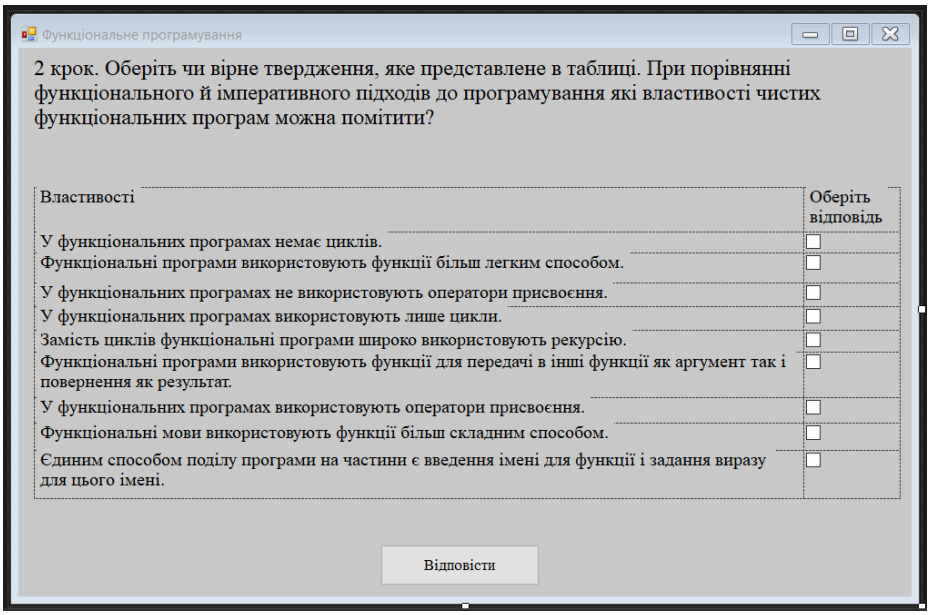

Рисунок 4.7 –Розміщення елементів в теоретичному питанні, де присутній вибір декількох відповідей з таблиці

Для практичних завдань було використано *TextBox* (надає поле для ведення відповіді) та *ComboBox* (надає поле для вибору відповіді із

випадаючого списку). На рисунку 4.8-4.10 показано розміщення елементів на типових кроках в практичній частині.

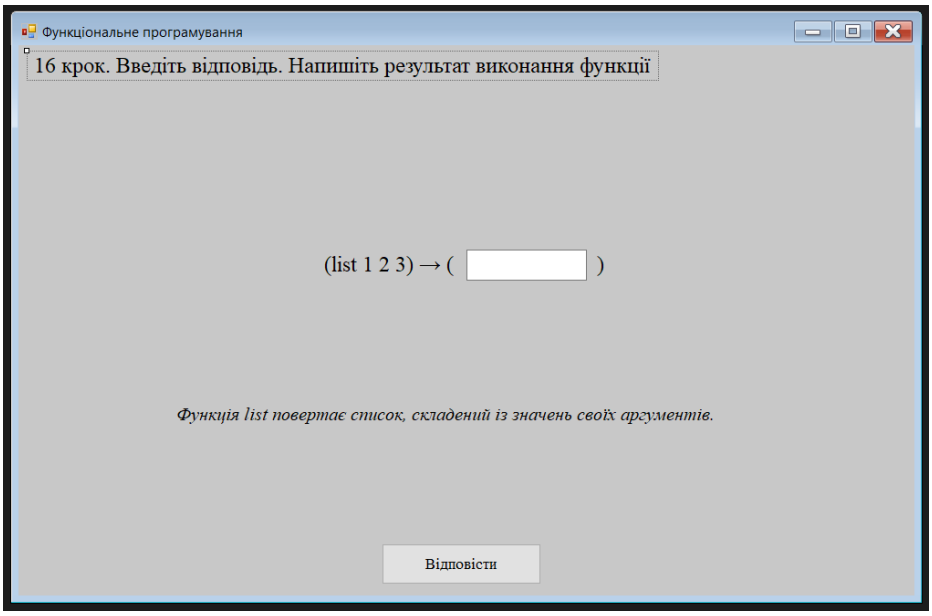

Рисунок 4.8 –Розміщення елементів в практичній частині з веденням відповіді

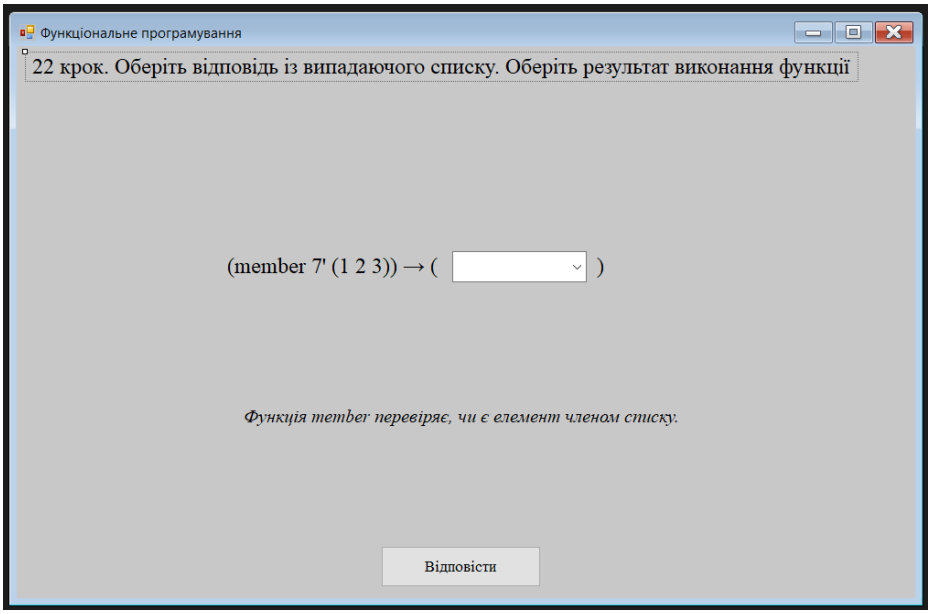

Рисунок 4.9 –Розміщення елементів в практичній частині з вибором відповіді із випадаючого списку

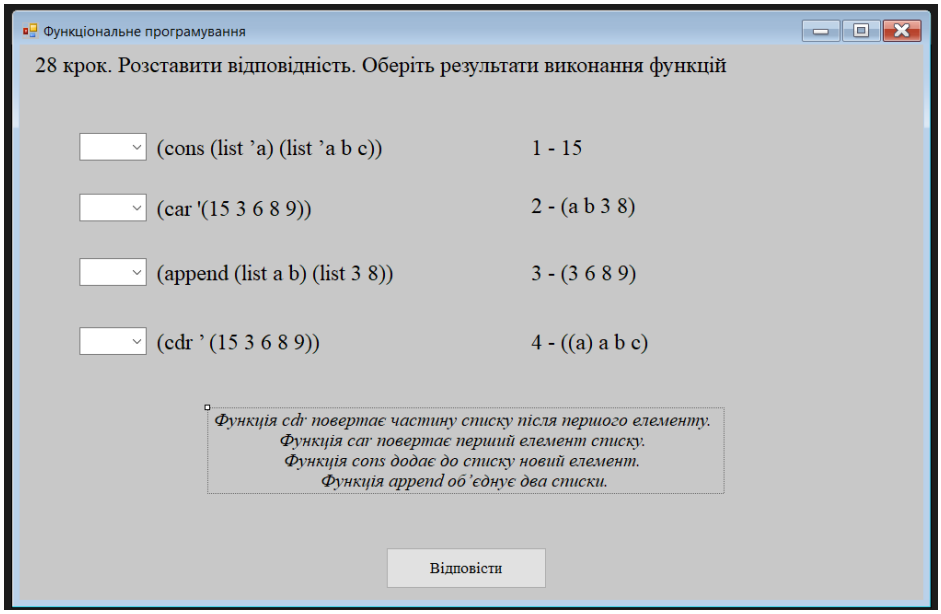

Рисунок 4.10 –Розміщення елементів в практичній частині з вибором відповіді із випадаючого списку

Так як при відкриті форми перед студентом не повинно бути обраної відповіді, то для кожного *CheckBox, RadioButton* подія *checked* приймає значення *false*. Для *TextBox* початкове значення є пробіл. Для елементу *ComboBox* властивості *items* першим варіантом додано пусте значення.

Для перевірки правильності обраної або веденої відповіді розроблені умови з допомогою умовної конструкції *if...else*. На рисунках 4.11-4.14 показано перевірки для питань з вибором однієї, декількох відповідей, а також з вибором із випадаючого списку та ведення до порожнього поля.

Так як студента необхідно інформувати про результат перевірки відповіді на правильність, то було використано вікно-повідомлення (діалогове вікно) *MessageBox* (модальне вікно, яке блокує інші дії в додатку, доки студент не зачинить його). На рисунку 4.15 показана реалізація вікна-повідомлення при вірній відповіді, 4.16 – вікно-повідомлення про хибну відповідь, 4.17 – вікно-повідомлення, коли відповідь ще не обрана студентом.

```
if (radioButton3.Checked)
{
    MessageBox.Show("Відповідь вірна!", "Увага!",
        MessageBoxButtons.OK,
        MessageBoxIcon.Information,
        MessageBoxDefaultButton.Button1);
    Err.ero++;
    Q_2 q_2 = new Q_2();
    q_2.Show();
    Hide();
}
else if (radioButton1.Checked || radioButton2.Checked ||
    radioButton4.Checked || radioButton5.Checked)
{
    MessageBox.Show("Відповідь не вірна!", "Увага!",
        MessageBoxButtons.OK,
        MessageBoxIcon.Information,
        MessageBoxDefaultButton.Button1);
    Q_2 q_2 = new Q_2();
    q_2.Show();
    Hide();
}
```

Рисунок 4.11 – Перевірка на правильність питання з однією правильною відповіддю

```
if (checkBox3.Checked || checkBox5.Checked || checkBox6.Checked || checkBox7.Checked ||
(checkBox1.Checked && checkBox2.Checked && checkBox3.Checked && checkBox4.Checked &&
checkBox5.Checked && checkBox6.Checked && checkBox7.Checked))
{
    MessageBox.Show("Відповідь не вірна!", "Увага!",
       MessageBoxButtons.OK,
       MessageBoxIcon.Information,
       MessageBoxDefaultButton.Button1);
  Q_4 q_4 = new Q_4();
  q_4.Show();
Hide();
}
else if (checkBox1.Checked && checkBox2.Checked && checkBox4.Checked)
{
    MessageBox.Show("Відповідь вірна!", "Увага!",
      MessageBoxButtons.OK,
      MessageBoxIcon.Information,
      MessageBoxDefaultButton.Button1);
    Err.ero++;
    Q_4 q_4 = new Q_4();
    q_4.Show();
    Hide();
}
```

Рисунок 4.12 – Перевірка на правильність питання з декількома правильними відповідями

```
if ((textBox1.Text == "1 2 3") || (textBox1.Text == "123"))
{
    MessageBox.Show("Відповідь вірна!", "Увага!",
        MessageBoxButtons.OK,
        MessageBoxIcon.Information,
        MessageBoxDefaultButton.Button1);
    P_2 p_2 = new P_2();
    p_2.Show();
    Hide();
}
else if ((textBox1.Text != "1 2 3") || (textBox1.Text != "123"))
{
    MessageBox.Show("Відповідь не вірна!", "Увага!",
        MessageBoxButtons.OK,
        MessageBoxIcon.Information,
        MessageBoxDefaultButton.Button1);
    label4.Visible = true;
}
```

Рисунок 4.13 – Перевірка на правильність завдання з веденням відповіді

```
if (comboBox1.SelectedIndex == 4 && comboBox2.SelectedIndex == 1 && comboBox3.SelectedIndex == 2 && comboBox4.SelectedIndex == 3)
{
    MessageBox.Show("Відповідь вірна!", "Увага!",
        MessageBoxButtons.OK,
        MessageBoxIcon.Information,
        MessageBoxDefaultButton.Button1);
    Form3 f_3 = new Form3();
    f_3.Show();
    Hide();
}
else if (comboBox1.SelectedIndex == -1 && comboBox2.SelectedIndex == -1 && comboBox3.SelectedIndex == -1 && comboBox4.SelectedIndex == -1)
{
    MessageBox.Show("Оберіть відповідь!", "Увага!",
        MessageBoxButtons.OK,
        MessageBoxIcon.Information,
        MessageBoxDefaultButton.Button1);
}
```

Рисунок 4.14 – Перевірка на правильність завдання з вибором відповіді із випадаючого списку

```
MessageBox.Show("Відповідь вірна!", "Увага!",
    MessageBoxButtons.OK,
    MessageBoxIcon.Information,
    MessageBoxDefaultButton.Button1);
```

Рисунок 4.15 – Вікно-повідомлення при вірній відповіді

```
MessageBox.Show("Відповідь не вірна!", "Увага!",
    MessageBoxButtons.OK,
    MessageBoxIcon.Information,
    MessageBoxDefaultButton.Button1);
```

Рисунок 4.16 – Вікно-повідомлення при хибній відповіді

```
MessageBox.Show("Оберіть відповідь!", "Увага!",
    MessageBoxButtons.OK,
    MessageBoxIcon.Information,
    MessageBoxDefaultButton.Button1);
```

Рисунок 4.17 – Вікно-повідомлення, коли студент ще не обрав відповідь

Перехід між формами реалізований з допомогою закриття активного вікна та відкриття наступного. На рисунку 4.18 показана реалізація переходу.

```
P_9 p_9 = new P_9();
p_9.Show();
Hide();
```

Рисунок 4.18 – Перехід до наступного кроку

Для реалізації підрахунку кількості вірних відповідей, які дав студент було створено публічний статичний клас *Err* із публічною статичною зміною *ero* (рис. 4.19).

```
public static class Err
{
    public static int ero = 0;
}
```

Рисунок 4.19 – Реалізація підрахунку вірних відповідей

4.3.Опис роботи навчального тренажера

Для того, щоб розпочати роботу з навчальним тренажером першим кроком його необхідно завантажити з дистанційного навчального курсу «Теорія програмування» та запустити.

Після того, як навчальний тренажер завантажився, перед студентом з’являється перша форма (перший крок алгоритму), яка містить інформацію про тему, розробника та кнопку «Розпочати» з допомогою якої можна перейти далі. На рисунок 4.20 демонструє дану форму.

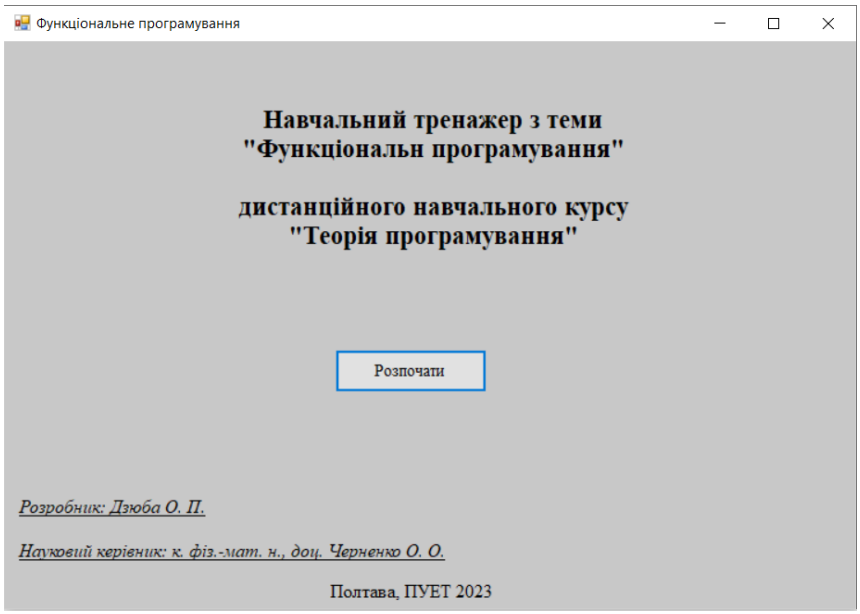

Рисунок 4.20 – Перша форма (перший крок алгоритму)

Далі навчальний тренажер переходить до першого питання в якому необхідно обрати одну правильну відповідь. На рисунку 4.21 показано дане питання.

Коли студент відповів, він отримує інформаційне повідомлення про вірність як показано на рисунку 4.22, а коли хибно, то повідомлення, яке має вигляд як на рисунку 4.23. Доки не буде на інформаційному повідомлені натиснута кнопка «Ок», то перехід до наступного кроку не відбудиться. Якщо

користувач (студент) вирішив натиснути кнопку «Відповісти» без обраної відповіді, то він також отримує вікно-повідомлення вигляд який показано на рисунку 4.24.

Після того, як натиснули кнопку «Ок» відбувається перехід до другого питання в якому необхідно обрати правильне твердження із таблиці. Рисунок 4.25 демонструє вигляд даного питання в навчальному тренажері.

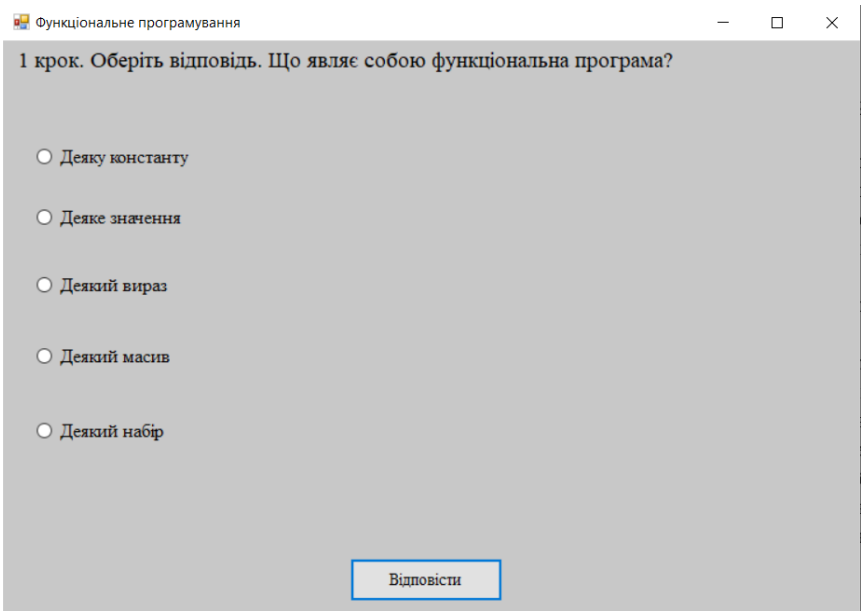

Рисунок 4.21 – Перший крок алгоритму (питання з однією вірною відповіддю)

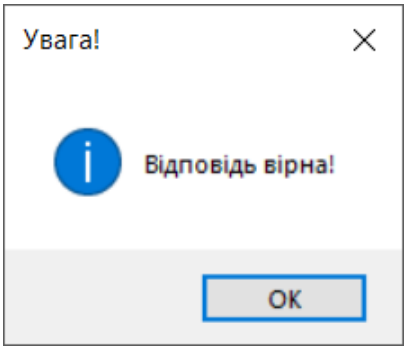

Рисунок 4.22 – Інформаційне вікно-повідомлення при вірній відповіді

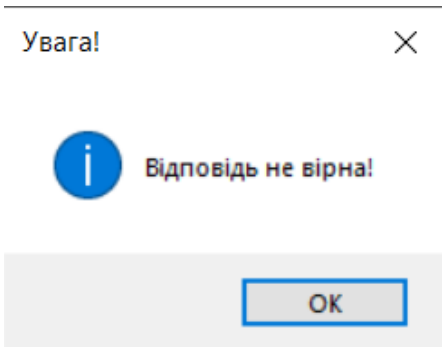

Рисунок 4.23 – Інформаційне вікно-повідомлення при хибній відповіді

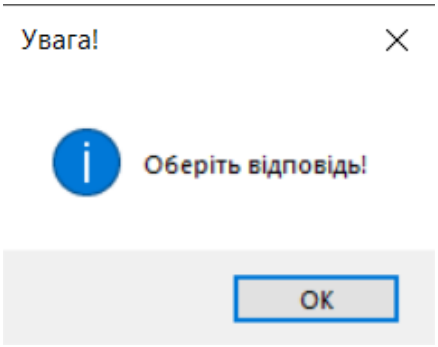

Рисунок 4.24 – Інформаційне вікно-повідомлення при не обраній відповіді

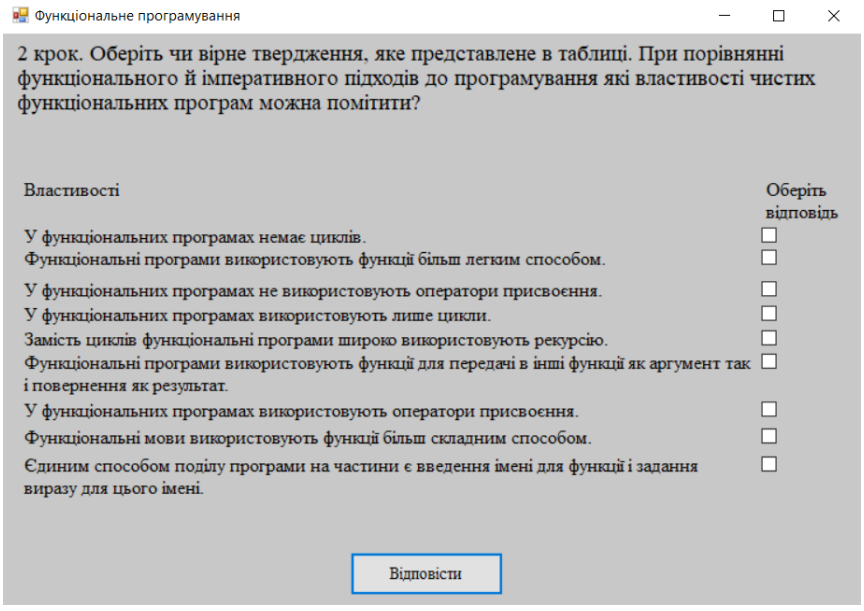

Рисунок 4.25 – Другий крок алгоритму (питання з вибором правильного твердження із таблиці)

Також в перевірці теоретичних знань було розроблено питання з вибором декількох відповідей із запропонованих. Особливістю цих питань є те, що студентові, необхідно обрати більше однієї відповіді. На рисунку 4.26 показано третє питання алгоритму, а саме з вибором декількох відповідей.

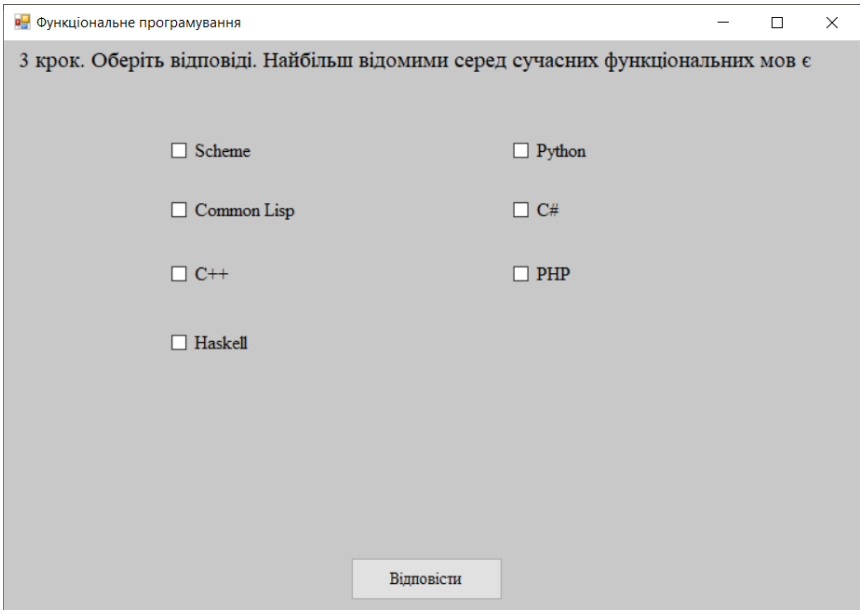

Рисунок 4.26 – Третій крок алгоритму (питання з вибором декількох відповідей)

Після того, як студент дав відповіді на всі 15 теоретичних питань, перед ним з’являється форма (рис. 4.27). Ця форма інформує про те, що студент закінчив питання теоретичного матеріалу з теми, також показує кількість вірних відповідей на питання та має дві кнопки «Завершити роботу», «Практичні завдання».

Основною частиною завдань в практичний частині є ведення відповіді до порожнього поля на формі. На рисунку 4.28 продемонстровано перше практичне завдання з веденням відповіді.

Перевірка на правильність веденої відповіді відбувається інакше, бо поки студент не дасть вірну відповідь, то не зможе перейти до наступного кроку навчального тренажеру. Також при невірній відповіді з’являється підказка на формі, яка дасть змогу зрозуміти, що виконує функція. На рисунку 4.29

показано невірна відповідь на перше практичне завдання з підказкою до функції.

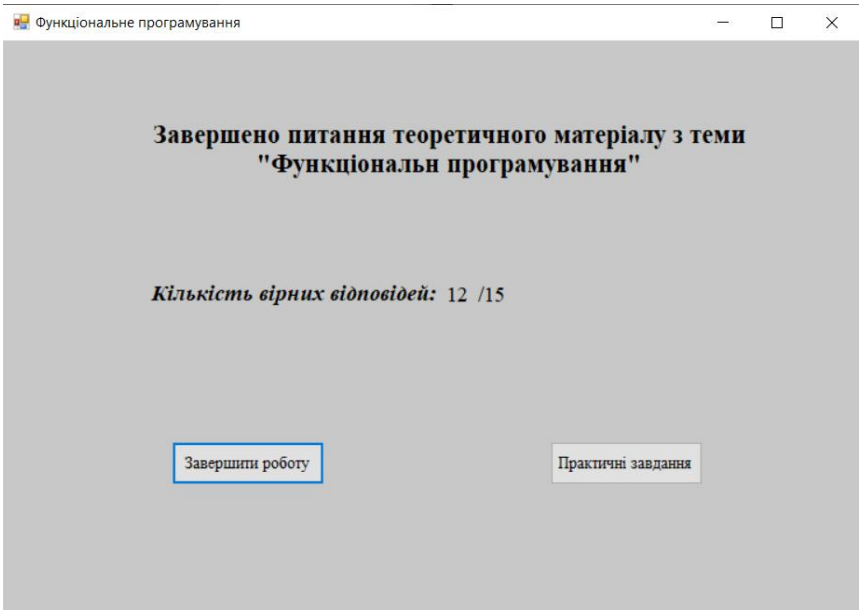

Рисунок 4.27 – Шістнадцятий крок алгоритму (форма переходу)

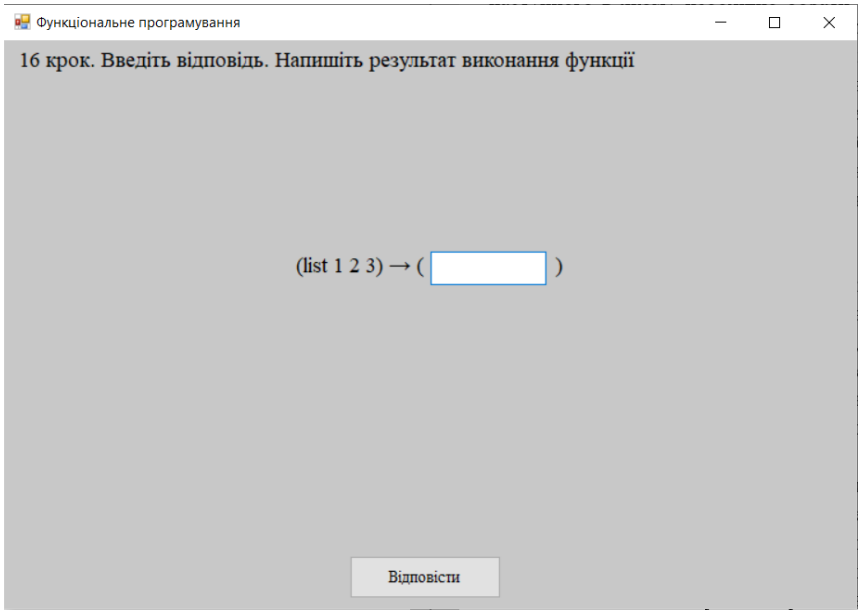

Рисунок 4.28 – Шістнадцятий крок алгоритму (завдання на ведення відповіді)

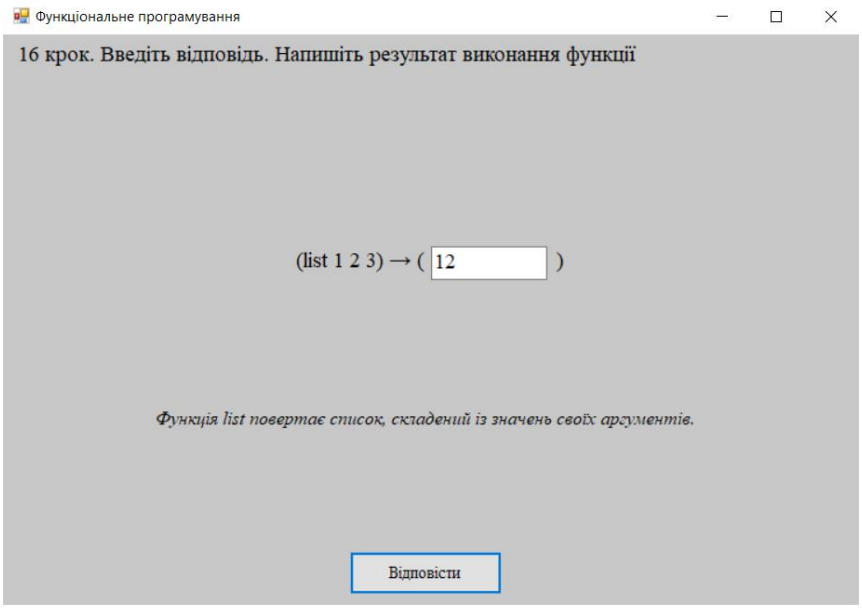

Рисунок 4.29 – Практичне завдання з невірною відповіддю та підказкою

Після 6 практичних кроків з’являється два завдання на вибір відповіді із випадаючого списку (рис. 4.30).

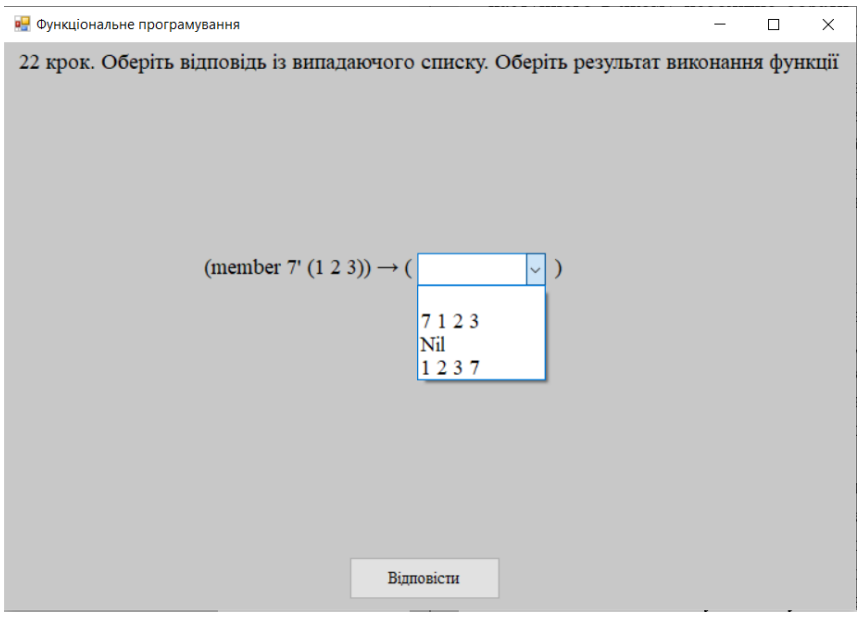

Рисунок 4.30 – Двадцять другий крок алгоритму (завдання з вибором відповіді із випадаючого списку)

Останнім кроком в практичній частині є правильно розташувати відповідність функції до її виконання. На рисунку 4.31 показано вигляд цього завдання.

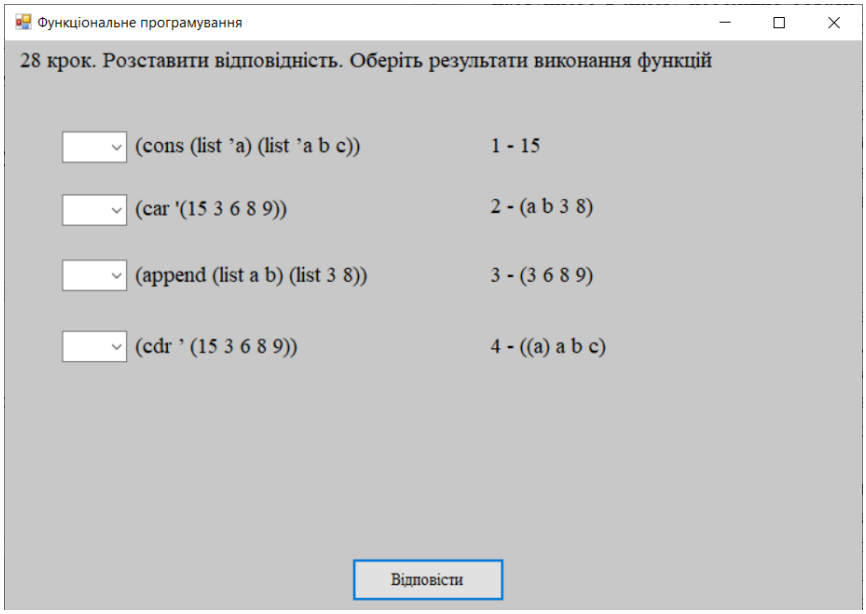

Рисунок 4.31 – Двадцять восьмий крок алгоритму (завдання із розставленням відповідності)

Після того, як студент відповів на останнє завдання практичної частини, перед ним з'являється форма (рис. 4.32), яка дає можливість завершити роботу з навчальним тренажером або повернутися до початку та розпочати все знову. Форма містить текстове повідомлення та два елементи для переходу відповідно.

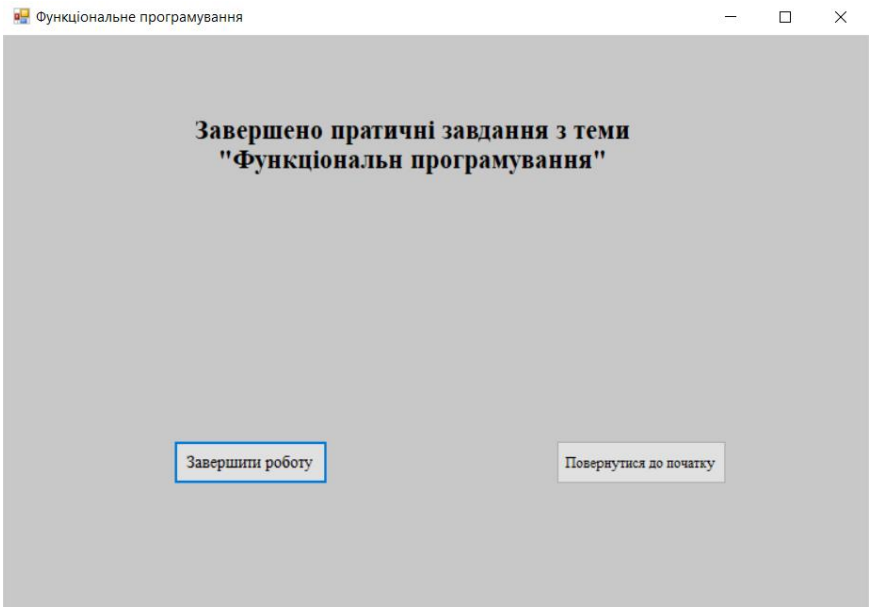

Рисунок 4.32 – Кінцева форма інформування

ВИСНОВКИ

В ході виконання кваліфікаційної роботи виконано головну задачу, а саме алгоритмізовано та програмно реалізовано навчальний тренажер з теми «Функціональне програмування» дистанційного навчального курсу «Теорія програмування».

Для виконання головної задачі було виконані всі етапи, а саме:

1. Описано об’єкт дослідження в постановці завдання.
2. Проведено інформаційний огляд, де було розглянуто види комп’ютерних тренажерів, а також навчальні тренажери, які розроблені в університеті їх переваги та недоліки.
3. Розроблено алгоритм навчального тренажера з теми «Функціональне програмування».
4. Розроблено блок-схему роботи описаного алгоритму.
5. Проаналізовано середовища розробки та мови програмування, які мають найбільшу сумісність з дистанційною платформою.
6. Після складання алгоритму, реалізовано навчальний тренажер на обраній мові програмування C# з візуальною реалізацією в Microsoft Visual Studio.
7. Протестований кожен крок роботи навчального тренажера на виявлення помилок.

Робота пройшла апробацію на XLVI Міжнародній науковій студентській конференції «Актуальні питання розвитку науки та забезпечення якості освіти у XXI столітті» (за підсумками науково-дослідних робіт студентів за 2022 рік), яка пройшла 25 квітня 2023 року, про що засвідчує отриманий сертифікат.

Програмна реалізація навчального тренажера передана до дистанційного відділу ПУЕТ та імплементовано у навчальний курс з «Теорії програмування», про що засвідчує акт впровадження.

СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Melnychuk I. M. (2008) Naukovo-metodychni zasady vprovadzhennya interaktyvnykh tekhnolohiy u vyshchiy shkoli [Scientific and method basis of implementation of interactive technologies in higher institution]. Bulletin of Education of National pedagogical university after M. P. Drahomanov. Vol. 11. P. 183–190.

2. Алексов С. В. Алгоритмізація та програмування тренажера «Дерава розбору» дистанційного навчального курсу «Теорія програмування» / С. В. Алексов, О. О. Черненко. – Полтава: Кафедра ММСІ ПУЕТ, 2020 – 85с. – Режим доступу: http://dspace.puet.edu.ua/bitstream/123456789/8998/1/Aleksov_Sergey_Viktorovich-KN21m-1%20%283%29.pdf

3. Цехмейстер В. О. Тренажер з теми «Метод обрізання основ» дистанційного навчального курсу «Теорія програмування» та розробка його програмного забезпечення / В. О. Цехмейстер, О. О. Черненко. – Полтава: Кафедра ММСІ ПУЕТ, 2020 – 35с. – Режим доступу: http://dspace.puet.edu.ua/bitstream/123456789/9028/1/Tsekhmeister_Vadym_Oleksandrovych-KN21i-1.pdf

4. Закіров Р. Р. Програмне забезпечення для тренажера з теми «Лямбда-вирахування і типізація» дистанційного навчального курсу «Теорія програмування» / Р. Р. Закіров, О. О. Черненко. – Полтава: Кафедра ММСІ ПУЕТ, 2021 – 69с. – Режим доступу: http://dspace.puet.edu.ua/bitstream/123456789/10353/1/Zakirov_Renat_Raifovich-KNIT-51.pdf

5. Волосевич М. В. Тренажер з теми «Марківські підстановки» дистанційного навчального курсу «Теорія алгоритмів» та розробка його програмного забезпечення / М. В. Волосевич, О. О. Черненко. – Полтава: Кафедра ММСІ, 2021 – 63с. – Режим доступу:

http://dspace.puet.edu.ua/bitstream/123456789/9032/1/Volosevich_Maxim_Vasilyevich-KN22m-1%20%281%29.pdf

6. Будяй В. О. Тренажер з теми «Математичні основи теорії алгоритмів» дистанційного навчального курсу «Теорія програмування» та розробка його програмного забезпечення / В. О. Будяй, О. О. Черненко. – Полтава: Кафедра ММСІ, 2021 – 39с. – Режим доступу: http://dspace.puet.edu.ua/bitstream/123456789/11348/1/Budyay_Vitaliy_Oleksiyovych-KN21i-1.pdf

7. Комп'ютерні тренажерні системи [Електронний ресурс] – https://doc4web.ru/informatika/kompyuterni-trenazherni-sistemi.html – Назва з екрану.

8. Рибалко М. А., Іванова Є. А. Сучасні засоби розробки бізнес-додатків/М. А. Рибалко // Інформаційне суспільство: сучасний стан та перспективи розвитку. – 2018. – С. 345-347.

9. Сидорова М. Н. Сучасні середовища розробки програмного забезпечення: огляд та аналіз / М. Н. Сидорова // Вісник освітнього косорціуму Середньоруський університет. Інформаційні технології. – 2017. – № 2. – С. 71-73.

10. Хеффельфінгер Д. Розробка програм Java EE в NetBeans 7. / Д. Хеффельфінгер. – Litres, 2017. – 332 с.

11. Медведєв В. В. Розробка та оптимізація програми для обчислення конструктивного коефіцієнта енергетичної ефективності EEDI / В. В. Медведєв // Технічна експлуатація водного транспорту: проблеми та шляхи розвитку. – 2019. - № 1-2.

12. Лекція 11. Функціональне програмування [Електронний ресурс] - http://www2.el.puet.edu.ua/st/mod/page/view.php?id=158364 – Назва з екрану.

13. Лекції. Програмування, комп'ютери і кібернетика. Етапи розв'язання задач з використанням ЕОМ [Електронний ресурс] – https://project.zu.edu.ua/ReadData.php?h_id=28 – Назва з екрану.

14. Ольховська О. В. Методичні рекомендації до виконання кваліфікаційної роботи для студентів за освітньою програмою «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» ступеня бакалавра / О. В. Ольховська, О. О. Черненко – Полтава: ПУЕТ, 2022. – 71с.