ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

ФАКУЛЬТЕТ ЕКОНОМІКИ І МЕНЕДЖМЕНТУ
ФОРМА НАВЧАННЯ ДЕННА
КАФЕДРА МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ
ТА СОЦІАЛЬНОЇ ІНФОРМАТИКИ

Допускається до захисту
Завідувач кафедри _____ О.О. Ємець
(підпис)
«_____»_____ 2021 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО бакалаврської роботи
на тему

Розробка програмного забезпечення тренажеру з теми «Побудова блок-схем алгоритмів лінійної структури» дистанційного навчального курсу «Програмування ІІ».

Спеціальність 122 "Комп'ютерні науки"

Виконавець роботи Осіпов Сергій Вітальович_____
«___»_____2021 р.
(підпис)

Науковий керівник к.ф.-м.н., доц., Ємець Олександра Олегівна
_____ «___»_____2021 р.
(підпис)

ПОЛТАВА 2021 р.

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ «ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

ЗАТВЕРДЖУЮ

Завідувач кафедри _____ О.О. Ємець
(підпис)

«_____» _____20__ р.

ЗАВДАННЯ ТА КАЛЕНДАРНИЙ ГРАФІК

ВИКОНАННЯ БАКАЛАВРСЬКОЇ РОБОТИ

Студента зі спеціальності 122 «Комп'ютерні науки»

Прізвище, ім'я, по батькові Осіпов Сергій Віталійович

1. Тема «Розробка програмного забезпечення тренажеру з теми «Побудова блок-схем алгоритмів лінійної структури» дистанційного навчального курсу «Програмування ІІ», затверджена наказом ректора № 121-Н від «1» вересня 2020 р.

Термін подання студентом бакалаврської роботи «24» травня 2021 р.

2.Вихідні дані до бакалаврської роботи: публікації за темою роботи; методичні рекомендації; стандарти.

3. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Вступ.
1. Постановка задачі.
2. Інформаційний огляд.
2.1. Огляд тренажерів професійно-орієнтованих дисциплін спеціальності «Комп'ютерні науки».
2.2. Позитивні риси розглянутих тренажерів.
2.3. Недоліки розглянутих тренажерів.
2.4. Необхідність та актуальність теми.
3. Теоретична частина.
3.1. Алгоритм тренажеру.
3.2. Блок-схема алгоритму.
4. Практична частина.
4.1. Опис програми.
4.2. Інструкція по роботі з програмою.
Висновки.

4. Перелік графічного матеріалу (з точним визначенням кількості блок-схем, іншого графічного матеріалу) Блок-схема алгоритму (4-6 листів).

5. Консультанти розділів бакалаврської роботи

Розділ	П.І.Б., посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

Вступ	Ємець Ол-ра О.	05.09.2020	05.09.2020
1. Постановка задачі	Ємець Ол-ра О.	05.09.2020	05.09.2020
2. Інформаційний огляд	Ємець Ол-ра О.	05.09.2020	05.09.2020
3. Теоретична частина	Ємець Ол-ра О.	05.09.2020	05.09.2020
4. Практична частина	Ємець Ол-ра О.	05.09.2020	05.09.2020

6. Календарний графік виконання бакалаврської роботи

Зміст роботи	Термін виконання	Фактичне виконання
1. Вступ	04.05.21	
2. Вивчення методичних рекомендацій і стандартів та звіт керівнику	1.10.20	
3. Постановка задачі	10.10.20	
4. Інформаційний огляд джерел бібліотек та інтернету	1.11.20	
5. Теоретична частина	13.01.21	
6. Практична частина	15.04.21	
7. Закінчення оформлення	04.05.21	
8. Доповідь студента на кафедрі	25.05.21	
9. Доробка (за необхідності), рецензування	07.06.21	

Дата видачі завдання «5» вересня 2020 р.

Студент _____ Осіпов С. В.

(підпис)

Науковий керівник _____ к. ф.-м. н., доц. Ємець Ол-ра О.

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Результати захисту бакалаврської роботи

Бакалаврська робота оцінена на _____

(балів, оцінка за національною шкалою, оцінка за ECTS)

Протокол засідання ЕК № ____ від «____» _____ 20___ р.

Секретар ЕК_____ _____

(підпис) (ініціали та прізвище)

РЕФЕРАТ

Записка: 45 с., 28 рис., 1 додаток (на 2 сторінках), 1 таблиця, 12 джерел.

Предмет розробки – програма-тренажер для навчання побудови блок-схем алгоритмів лінійної структури.

Мета роботи – «Розробка програмного забезпечення тренажеру з теми «Побудова блок-схем алгоритмів лінійної структури» дистанційного навчального курсу «Програмування ІІ»

Методи, які були використані для розв'язування задачі – TypeScript фреймворк Angular та фреймворк для розробки нативних десктоп додатків – Electron.

Під час створення тренажера використовувалося інтегроване середовище розробки WebStorm.

«Розробка програмного забезпечення тренажеру з теми «Побудова блок-схем алгоритмів лінійної структури»

Ключові слова: ТРЕНАЖЕР, БЛОК-СХЕМА, ЦИКЛ, ANGULAR.

ЗМІСТ:

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні позначення, символи, одиниці, скорочення, терміни	Пояснення умовних позначень, символів, одиниць, скорочень, термінів
Алгоритм	Це чіткий опис послідовних дій, які необхідно виконати для розв’язання задачі.
Блок-схема	Це наочне графічне зображення алгоритму, в якому окремі етапи алгоритму зображуються за допомогою різних геометричних

	фігур, а зв’язки між етапами зображуються за допомогою стрілок, що з’єднують ці фігури.
	Символ «термінатор». Показує початок, зупинку або кінець блок-схеми.
	Символ «дані». Відображає введення або виведення даних.
	Символ «процес». Відображає обробку даних будь-якого типу
	Символ «рішення». Відображає рішення або функцію типу «перемикач», яка має один вхід і ряд альтернативних виходів, з яких лише один може бути активізованим після обчислення або перевірки умов, зазначених в середині цього символу. Відповідні результати обчислень (або позначки виконання/невиконання умов) мають бути записані поруч з лініями, які відображають ці шляхи.
	Символ «лінія». Показує потік даних або керування. За необхідності або для підвищення зручності читання блок-схем можуть бути добавлені стрілки-вказівники

ВСТУП

На даний момент, інформаційні технології являють собою одну з найбільш цікавих та затребуваних галузей, що глибоко проникає в усі можливі сфери нашого життя та щоденно перетинається як з функціонуванням світових корпорацій, так і відіграє важливу роль в буденних справах звичайного користувача. До того ж, інформаційні технології - це відносно нова галузь, тому вона ще знаходиться на етапі безперервного стрімкого розвитку, який має бути доволі тривалим. Професії, пов’язані з інформаційними технологіями вважаються дуже затребуваними та високооплачуваними, чим обумовлена зростаюча популярність галузі та велика кількість людей, що кожного дня вирішує приєднатися і починає свій шлях у світі кодів та розробок. Наприклад, відповідно до статистики 2018 року, кількість вже існуючих членів IT-суспільства, поповнилася на 2,5 млн нових програмістів [1].

Особливістю сфери інформаційних технологій є велика різноманітність, тому у залежності від конкретного відгалуження, також варіюється поріг входження. Доволі часто у нових розробників виникають складнощі в розумінні роботи алгоритмів. Для вирішення цієї проблеми найбільш доступним способом, доцільним є використання програмного забезпечення, що формуватиме візуальне відображення роботи алгоритмів. Графічний спосіб представлення алгоритмів, тобто їх візуальне відображення, відбувається за допомогою використання спеціальних графічних засобів, що називаються блок-схемами. Зазвичай, блок-схеми використовують у роботі з послідовними функціональними підходами. За використання об’єктно-орієнтованого підходу, блок-схеми не здатні повною мірою відобразити функціонал алгоритму.

Більшість програмістів починають крокувати інформаційними технологіями саме через ознайомлення з основами ООП. Одним з доволі важких моментів для початківців - є розуміння сутності структури алгоритмів та взаємна взаємодія компонентів програм, чим зумовлена потреба у нових, більш досконалих підходах щодо подання навчальних матеріалів.

Об’єкт розробки у проекті – це тренажер.

Предмет розробки – тренажер з теми «Побудова блок-схем алгоритмів лінійної структури» для дистанційного навчального курсу.

Мета роботи – створити програму-тренажер з теми «Побудова блок-схем алгоритмів лінійної структури» для дистанційного курсу «Програмування ІІ».

Для реалізації мети слід виконати такі задачі:

1) ознайомитися з тренажерами схожої тематики та проаналізувати їх;

2) ознайомитися зі стандартом по блок-схемам та теоретичним матеріалом зі створення блок-схем алгоритмів та умовних операторів мови С++;

3) підібрати або розробити приклад(и) тренажеру, розробити алгоритм та блок-схема тренажеру(ів), створити програму реалізацію алгоритму тренажеру(ів).

Використані при розробці програми методи – мова програмування С++, середовище програмування Borland Builder, пакет інженерної графіки MS Visio. Готовність до впровадження – програмний продукт повність завершено.

Структура пояснювальної записки. Записка складається з 4 змістовних частин. Перша частина має назву «Постановка задачі» та містить вимоги до дипломного проекту. Друга частина – «Інформаційний огляд». В ній показано знайдені розробки аналогічної тематики. Ці розробки проаналізовані. Третя частина має назву «Теоретична частина». Тут подано умову завдання тренажеру, алгоритм тренажеру, блок-схему алгоритму тренажеру. Четверта частина має назву «Практична частина». Тут викладено опис створення програми та копії екранів кроків тренажеру.

Обсяг пояснювальної записки – 58 сторінок.

1. ПОСТАНОВКА ЗАДАЧІ

Алгоритмом називається точний, однозначний та зрозумілий опис послідовності дій, що необхідно виконати виконавцю для вирішення поставленого завдання. Слово «алгоритм» походить від імені математика Аль Хорезмі, який сформулював правила виконання арифметичних дій. Спочатку під алгоритмом розуміли тільки правила виконання чотирьох арифметичних дій над числами. В подальшому це поняття стали використовувати в загальному, для позначення послідовності наперед визначених дій, що призводять до вирішення всіх поставлених завдань.

Якщо розглядати алгоритми обчислювальних процесів, як окрему групу алгоритмів, то а такому випадку об'єктами, до яких застосовувався алгоритм,

є дані в цифровому форматі. Алгоритм рішення обчислювальної задачі представляє собою сукупність правил перетворення вихідних цифрових даних в деякий результ.

Основними властивостями алгоритму є:

— детермінованість (визначеність). Результати обчислювального процесу, при заданих тих самих вихідних даних завжди будуть однозначні. Завдяки цій властивості процес виконання алгоритму носить механічний характер;

— результативність. Дана властивість вказує на наявність таких вихідних даних, для яких виконаний за заданим алгоритмом обчислювальний процес повинен через скінченне, зліченне число кроків завершитись та видати шуканий результат;

— масовість. Ця властивість передбачає, що алгоритм повинен бути використаний для вирішення всіх завдань відповідного типу;

— дискретність. Це означає те, що весь алгоритмом обчислювального процесу повинен бути поділений на окремі етапи, можливість виконання яких виконавцем (комп'ютером) не викликає сумнівів.

При всьому різноманітті алгоритмів розв'язання задач в них можна виділити три основні види обчислювальних процесів:

— лінійний;

— розгалужений;

— циклічний.

Лінійним називається такий обчислювальний процес, при якому всі етапи рішення задачі виконуються в природному порядку проходження записи цих етапів. Розгалуженим називається такий обчислювальний процес, в якому вибір напрямку обробки інформації залежить від вихідних або проміжних даних (від результатів перевірки виконання будь-якого логічного умови).

Циклом називається багаторазово повторювана дсукупність обчислень. Обчислювальний процес, що містить один або кілька циклів, називається циклічним. За кількістю виконання цикли поділяються на цикли з певним

(заздалегідь заданим) числом повторень та цикли з невизначеним числом повторень. Кількість повторень останніх залежить від дотримання деякої умови, що задає необхідність виконання циклу. При цьому умова може перевірятися на початку циклу - тоді мова йде про цикл з передумовою, або в кінці - тоді це цикл з умовою поста.

Окрім загально визначених існує ще декілька типів алгоритмів:

Імовірнісний (стохастичний) алгоритм дає напрямок вирішення задачі декількома шляхами або способами, що приводять до ймовірного досягнення результату.

Евристичний алгоритм - це такий тип алгоритмів, при якому досягнення кінцевого результату програми дій однозначно не визначено, так само як не позначена вся послідовність дій, так само які і не виділені всі дії виконавця. До евристичних алгоритмах відносять, наприклад, інструкції та розпорядження. У цих алгоритмах використовуються універсальні логічні процедури та способи прийняття рішень, засновані на аналогії, асоціаціях або минулому досвіді виконавця по вирішенню подібних завдань.

Допоміжний (підлеглий) алгоритм (процедура) - алгоритм, раніше і цілком використовується при алгоритмізації конкретного завдання. У деяких випадках при наявності однакових послідовностей вказівок (команд) для різних даних з метою скорочення запису також виділяють допоміжний алгоритм. Для більшої зрозумілості та уникнення подвійних трактувань окремих кроків алгоритму опис алгоритм повинен бути формалізований за деякими правилами на основі конкретних образотворчих засобів.

Серед існуючих способів задання алгоритмів належать такі:

— словесний;

— формульно-словесний;

— графічний;

— мова операторних схем;

— алгоритмічна мова.

Найбільшого поширення завдяки своїй наочності отримав графічний спосіб запису алгоритмів на основі блок-схем. Блок-схемою називається графічне зображення логічної структури алгоритму, в якому кожен крок процесу обробки інформації представлений у вигляді геометричних символів (блоків), що мають певну конфігурацію в залежності від характеру виконуваних операцій. Перелік символів, їх найменування, які відображаються ними функції, форма та розміри визначаються відповідними стандартами.

Таким чином, алгоритм виникає при вирішенні проблеми (інформування, управління). Для цього будується математична модель проблеми і на її базі ставиться задача та будуються методи її розв’язування. Під теорією алгоритмів розумітимемо систему фактів про властивості відповідних математичних об’єктів, якими подають алгоритм, у тому числі про взаємозв’язки між ними (їх алгебру, структури тощо).

Щоб алгоритм виконав своє призначення, його необхідно будувати за певними правилами. Тому потрібно говорити все ж не про властивості алгоритму, а про правила побудови алгоритму, або про вимоги, що пред'являються до алгоритму. Перше правило - при побудові алгоритму насамперед необхідно задати множину об'єктів, з якими буде працювати алгоритм. Формалізоване (закодоване) представлення цих об'єктів носить назву даних. Алгоритм приступає до роботи з певним набором даних, які називаються вхідними, і в результаті своєї роботи видає дані, які називаються вихідними. Таким чином, алгоритм перетворює вхідні дані у вихідні.

Поки відсутні формалізовані вхідні дані, не можливо побудувати алгоритм. Друге правило - для роботи алгоритму потрібна пам'ять. У пам'яті розміщуються вхідні дані, з якими алгоритм починає працювати, проміжні дані та вихідні дані, які є результатом роботи алгоритму. Пам'ять є дискретною, тобто складається з окремих комірок. Поіменована комірка пам'яті має назву змінної. В теорії алгоритмів розміри пам'яті не обмежуються, тобто вважається, що ми можемо надати алгоритму будь-який необхідний для

роботи обсяг пам'яті. Третє правило - дискретність. Алгоритм будується з окремих послідовних кроків (дій, операцій, команд). Множина кроків, з яких складено алгоритм, повинна бути визначеною і зрозумілою виконавцю. Четверте правило - детермінованість. Після кожного кроку необхідно вказувати, який крок виконується наступним, або давати команду зупинки. П'яте правило - збіжність (результативність). Алгоритм повинен завершувати роботу після деякого числа зліченних кроків. При цьому необхідно вказати, що вважати результатом роботи алгоритму.

При проектуванні алгоритмів використовуються два основних підходи. Метод послідовної деталізації завдання («зверху-вниз») полягає в тому, що вихідне складне завдання розбивається на підзадачі. Кожна з підзадач розглядається та вирішується окремо. Якщо будь-які з підзадач складні, вони також розбиваються на підзадачі. Процес триває до тих пір, поки підзадачі не зведуть до елементарних. Рішення окремих підзадач потім збираються в єдиний алгоритм рішення вихідної задачі. Метод широко використовується, так як дозволяє вести розробку загального алгоритму одночасно кільком програмістам, вирішальним локальні підзадачі. Це необхідна умова швидкої розробки програмних продуктів.

Складальний метод («знизу-вгору») полягає у створенні великої кількості програмних модулів, що реалізують рішення типових задач. При вирішенні складного завдання програміст може використовувати розроблені модулі в якості допоміжних алгоритмів (процедур). У багатьох системах програмування вже існують подібні набори модулів, що істотно спрощує та прискорює проектування складного алгоритму. Для швидкого читання алгоритму слід його записувати у зрозумілій для відповідного кола виконавців формі, серед найбільш розповсюджених форм задання алгоритмів є такі як.

Словесний опис. Даний підхід представляє структуру алгоритму природною мовою. Наприклад, будь-який прилад побутової техніки (праска, електропила тощо) має інструкцію по експлуатації (словесний опис алгоритму, відповідно до якого даний прилад повинен використовуватися).

Ніяких правил складання словесного опису не існує. Запис алгоритму здійснюється в довільній формі на природній мові. Даний спосіб опису не має широкого розповсюдження, так як строго не формалізуємо (під «формальним» розуміється то, що опис абсолютно повне і враховує всі можливі ситуації, які можуть виникнути в ході вирішення); допускає неоднозначність тлумачення при описі деяких дій; даний опис може бути перенасичений описовим характером. Наприклад алгоритм вибору одягу відносно температури навколишнього середовища:"визначити температуру повітря; якщо температура нижче 0, то надіти шубу, інакше надіти куртку".

Псевдокод - опис структури алгоритму на природній, частково формалізованій мові, що дозволяє виділити основні етапи вирішення задачі, перед точним його записом на мові програмування. У псевдокоді використовуються деякі формальні конструкції та загальноприйнята математична символіка. Строгих синтаксичних правил для запису псевдокоду не існує. Це полегшує запис алгоритму при проектуванні та дозволяє описати алгоритм, використовуючи будь-який набір команд. Проте в псевдокоді зазвичай використовуються деякі конструкції, властиві формальним мовам, що полегшує перехід від псевдокоду до запису алгоритму на мові програмування. Єдиного або формального визначення псевдокоду не існує, тому можливі різні варіанти псевдокоду, що відрізняються набором використовуваних слів і конструкцій. Задання алгоритму за допомогою графів.

Використовуючи матриці інциденцій і суміжності, інші властивості графу обчислюють характеристики графу чи його частин — шляхів, дерев тощо. Ці характеристики мають свої інтерпретації та аналогії на характеристиках алгоритму (складність, швидкодія тощо). Наприклад, знаючи час виконання операцій, які відображені вузлами та гілками графу досліджуваного алгоритму, та використавши поняття шляху можна побудувати алгоритм обчислення часу роботи досліджуваного алгоритму; автоматичним підрахунком числа вузлів, гілок можна встановити складність алгоритму (надаючи певним операціям відповідну "вагу").

Блок-схема - опис структури алгоритму за допомогою геометричних фігур з лініями-зв'язками, які показують порядок виконання окремих інструкцій. Цей спосіб має ряд переваг. Завдяки наочності, він забезпечує «читабельність» алгоритму та явно відображає порядок виконання окремих команд. У блок-схемі кожній формальній конструкції відповідає певна геометрична фігура або пов'язана лініями сукупність фігур.

На практиці в якості виконавців алгоритмів використовуються спеціальні автомати - комп'ютери. Тому алгоритм, призначений для виконання на комп'ютері, повинен бути записаний на зрозумілій йому мові. І тут на перший план висувається необхідність точного запису команд, яка не залишає місця для довільного тлумачення їх виконавцем. Отже, мова для запису алгоритмів повинен бути формалізований. Така мова прийнято називати мовою програмування, а запис алгоритму на цій мові - програмою для комп'ютера.

Програма, створювана людиною - програмістом, являє собою текст, що складається із знаків, як правило букв, цифр і спеціальних знаків. Знаки в тексті програми часто об'єднані в послідовності - ключові слова, слова об'єднані в пропозиції мови програмування - оператори. Кожен оператор, як правило, записується в окремий рядок тексту програми. Таким чином текстове програмування являє собою ієрархічну послідовність знаків, слів, операторів, що записуються і читаються послідовно, як звичайний текст людської писемності.

При оформленні матеріалів дипломної роботи потрібно дотримуватись методичних рекомендацій щодо оформлення роботи студента зі спеціальності «Комп’ютерні науки».

Зміст роботи повинен відповідати завданню затвердженим керівником та завідувачем кафедри[2].

Викладені матеріалі повинні відповідати таким пунктам:

– чіткість, логічність та послідовність зробленого матеріалу;

– чітке формулювання думок, яке дозволить уникнути неясності матеріалу;

– рекомендації та пропозиції які потрібно проаналізувати.

Робота повинна містити:

– актуальність даної теми повинна відповідати вимогам сучасності;

– проаналізованість методів дослідження, та порівняльної характеристики;

– опис роботи розробленого програмного забеспечення;

– наявність блок-схеми програмного продукту;

– англомовний варіант тренажеру.

Мета дипломного проектування – розробити тренажер з теми «Побудова блок-схем алгоритмів лінійної структури» дистанційного навчального курсу «Програмування ІІ». В зв’язку з цим слід:

1) ознайомитися зі стандартом та теоретичним матеріалом по створенню блок-схем алгоритмів.

2) ознайомитися з тренажерами схожої тематики. Проаналізувати їх, уникати негативних моментів оглянутих програм; взяти (якщо це можливо) позитивні риси з переглянутих додатків.

3) ознайомитись з темою «Умовні оператор мови С++». Вивчити всі можливі умовні оператори та підібрати приклади програм (або алгоритмів описаних іншим способом, наприклад, словесно).

4) два-чотири підібраних прикладів реалізувати у дипломній роботі. Для цього створити блок-схеми для алгоритмів прикладів, розробити алгоритми тренажерів та їх блок-схеми, створити програми тренажерів.

У пояснювальній записці описати створені програми. Оскільки, тренажер призначений для дистанційного навчання за дистанційними курсами, то дотримуватися позначок та термінології, яка зазначена у відповідних темах відповідних дистанційних курсів [1, 2]

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Огляд розробок, аналогічних темі дипломної роботи

Популярність дистанційної освіти в останні роки різко зросла. Ця форма навчання є найбільш гнучкою та доступною для багатьох бажаючих отримати знання. Багато сказано на користь дистанційної освіти, і, мабуть, не менше про недоліки подібної форми навчання.

Серед суттєвих переваг дистанційної форми навчання можна відзначити наступні:

– можливість навчатися у будь-який час. Студент, який навчається дистанційно, може самостійно вирішувати, коли і скільки часу упродовж семестру йому приділяти на вивчення матеріалу. Він будує для себе індивідуальний графік навчання;

– можливість навчатися в будь-якому місці. Студенти можуть вчитися, не виходячи з дому чи офісу, перебуваючи у будь-якій точці планети. Щоб приступити до навчання, необхідний лише комп'ютер з доступом в Інтернет. Відсутність необхідності щодня відвідувати навчальний заклад - безсумнівний плюс для людей з обмеженими можливостями здоров'я, для проживаючих у важкодоступних місцевостях, батьків з маленькими дітьми;

– навчання без відриву від основної діяльності. Для навчання зовсім не обов'язково брати відпустку на основному місці роботи, виїжджати у відрядження. Також дистанційно можна навчатися на декількох курсах чи у декількох навчальних закладах одночасно;

– можливість навчатися у своєму темпі. Не обов’язково навчатися у тому ж темпі, що й інші студенти. Студент завжди може повернутися до вивчення більш складних питань, кілька разів подивитися відео-лекції, перечитати переписку з викладачем, а вже відомі йому теми може пропустити. Головне, успішно проходити проміжні та підсумкові атестації;

– доступність навчальних матеріалів. Доступ до всієї необхідної літературі відкривається студенту після реєстрації в системі дистанційного навчання, або він отримує навчальні матеріали електронною поштою. Зникає проблема нестачі чи відсутності підручників, навчальних посібників чи методичок;

– мобільність. Зв'язок з викладачами, репетиторами здійснюється різними способами: як on-line, так і off-line. Проконсультуватися з викладачем за допомогою електронної пошти іноді ефективніше та швидше, ніж призначити особисту зустріч при очному або заочному навчанні;

– навчання в спокійній обстановці. Проміжна атестація студентів дистанційних курсів проходить у формі on-line тестів. Тому в студентів менше причин для хвилювань. Виключається можливість суб'єктивної оцінки: на систему, яка перевіряє правильність відповідей на питання тесту, не вплине успішність студента з інших предметів, його соціальний статус та інші чинники;

– індивідуальний підхід. При традиційному навчанні викладачеві досить важко приділити необхідну кількість уваги всім студентам групи, підлаштуватися під темп роботи кожного. Використання дистанційних технологій підходить для організації індивідуального підходу. Крім того, що студент сам обирає собі темп навчання, він може оперативно отримати у викладачавідповіді на виникаючі питання;

– дистанційна освіта дешевша. Якщо порівнювати вартість навчання на заочній і дистанційній формах навчання, то дистанційна скоріш за все буде дешевшою. Студенту не доводиться оплачувати дорогу, проживання, а у випадку з зарубіжними вузами не потрібно витрачатися на візу і закордонний паспорт;

– зручність для викладача. Вчителі, репетитори, викладачі, що займаються педагогічною діяльністю дистанційно, можуть приділяти увагу більшій кількості студентів і працювати, навіть перебуваючи у відрядженні чи на конференції за кордоном.

Разом з тим, дистанційне навчання не позбавлене і ряду недоліків:

– необхідна сильна мотивація. Практично весь навчальний матеріал студент-дистанційник освоює самостійно. Це вимагає достатньої сили волі, відповідальності і самоконтролю. Швидше за все, ніхто його підганяти чи заохочувати до навчання не стане. Підтримувати потрібний темп навчання без контролю з боку вдається не всім;

– нестача практичних вмінь та навиків. Досить проблемно якісно організувати дистанційне навчання за напрямами підготовки та спеціальностями, на яких передбачена велика кількість практичних занять. Навіть найсучасніші комп’ютерні тренажери не замінять майбутнім менеджерам «живої» практики;

– дистанційна освіта не підходить для розвитку комунікабельності. При дистанційному навчанні особистий контакт студентів один з одним і з викладачами мінімальний, а то і цілком відсутній. Тому така форма навчання

не підходить для розвитку комунікабельності, впевненості, навичок роботи у команді;

– проблема ідентифікації студента.

Поки найефективніший спосіб простежити за тим, чи студент самостійно здавав іспити чи заліки, - це відеоспостереження, що не завжди можливо. Тому на підсумкову атестацію студентам доводиться особисто приїжджати до вузу або його філії [3].

Сьогодні широко використовується велика номенклатура систем дистанційного навчання та управління дистанційним навчанням як з відкритим кодом (умовно безкоштовних), так і платних, широковживаних та вузькоорієнтованих.

ATutor – модульна система дистанційним керуванням навчанням з відкритим кодом. Поширюється на основі GNU General Public License. Для 14 установки необхідно мати комп'ютер з веб-сервером Apache 1.3.x, PHP версії > 4.2.0 та MySQL версій > 3.23.x і > 4.0.12 (версії 4.1.x і 5.x офіційно не підтримуються). Система розроблена із врахуванням доступності та можливістю адаптації за бажанням користувача. Щодо операційної системи сервера, обмежень немає – система є кросплатформеною.

Claroline – платформа дистанційного навчання та електронної діяльності з відкритим кодом. Аналогічно з ATutor, поширюється на основі GNU General Public License. Сумісна з такими операційними системами, як Linux, Mac і Windows. Забезпечує інтуїтивно простий інтерфейс для адміністрування. В основу організації Claroline LMS покладено концепцію просторів, пов'язаних з курсом чи педагогічною діяльністю. Кожен з просторів забезпечений інструментарієм для створення, організації та управління навчальними матеріалами; можливостями для забезпечення взаємодії між користувачами тощо.

Live@EDU – система дистанційного навчання, виконана з використанням технології Active Server Pages на платформі Microsoft. Для установки та коректної роботи системи серверна частина повинна бути

забезпеченою ОС Microsoft Windows NT Server 4.0, базою даних Microsoft SQL Server 7.0 та Microsoft Internet Information Server 4.0. Клієнтська частина повинна мати установлену ОС, Lviv Polytechnic National University Institutional Repository яка забезпечує доступ до мережі Інтернет та браузер, що обслуговує протокол HTTP версії 3.0, а також програмне забезпечення для перегляду і створення лекційних матеріалів. eFront є новим поколінням систем електронного навчання, яка об'єднує в собі функції системи управління навчанням та системи створення та управління навчальних матеріалів. Використовується для організації навчального процесу у навчальних закладах, а також для підвищення кваліфікації, атестації та відбору працівників у різномасштабних організаціях.

Система побудована на трьох типах користувачів – Адміністратор, Викладач та Студент. Moodle (Modular Object-Oriented Dynamic Learning Environment) – пакет модульного програмного забезпечення з відкритим кодом (ліцензія GNU GPL), який призначений для створення курсів дистанційного навчання та web-сайтів. Ця програма управління дистанційним навчанням орієнтована на взаємодію між викладачем та студентом, також використовується для підтримки очних курсів.

Moodle може бути встановленим на будь-який комп'ютер, який підтримує PHP та роботу із СУБД MySQL, PostgreSQL, Microsoft SQL Server; програмне забезпечення є кросплатформеним. SharePointLMS – система дистанційного навчання, розроблена на потужній багатофункціональній платформі MS Office SharePoint Server 2007. Є комплексним рішенням, яке об'єднує всіх користувачів (викладачі, студенти, адміністратори тощо) у єдиний інформаційно-навчальний простір та забезпечує інструментарій для спільної роботи. На відміну від Moodle, Claroline та ATutor, система є платною. Використовується не лише навчальними закладами та центрами навчання, а й підприємствами, організаціями, державними структурами [4].

Розглянемо роботи, які навчають студентів побудові блок-схем. Серед таких робот можна виділити тренажер «Побудова блок-схем алгоритмів

лінійної структури» (на прикладі мови Pascal), створений студенткою спеціальності «Інформатика» ПУЕТ Мордасовою І. у 2019 р. Тренажер (рис. 2.1-2.6) містить один приклад – задано алгоритм кодом (рис. 2.2), створити за ним блок-схему. На рис. 2.3 показано перше питання. Коли користувач вказує правильну відповідь, то йде перехід далі, коли невірну, то з'являється вікно як на рис. 2.4. Помилку слід усунути.

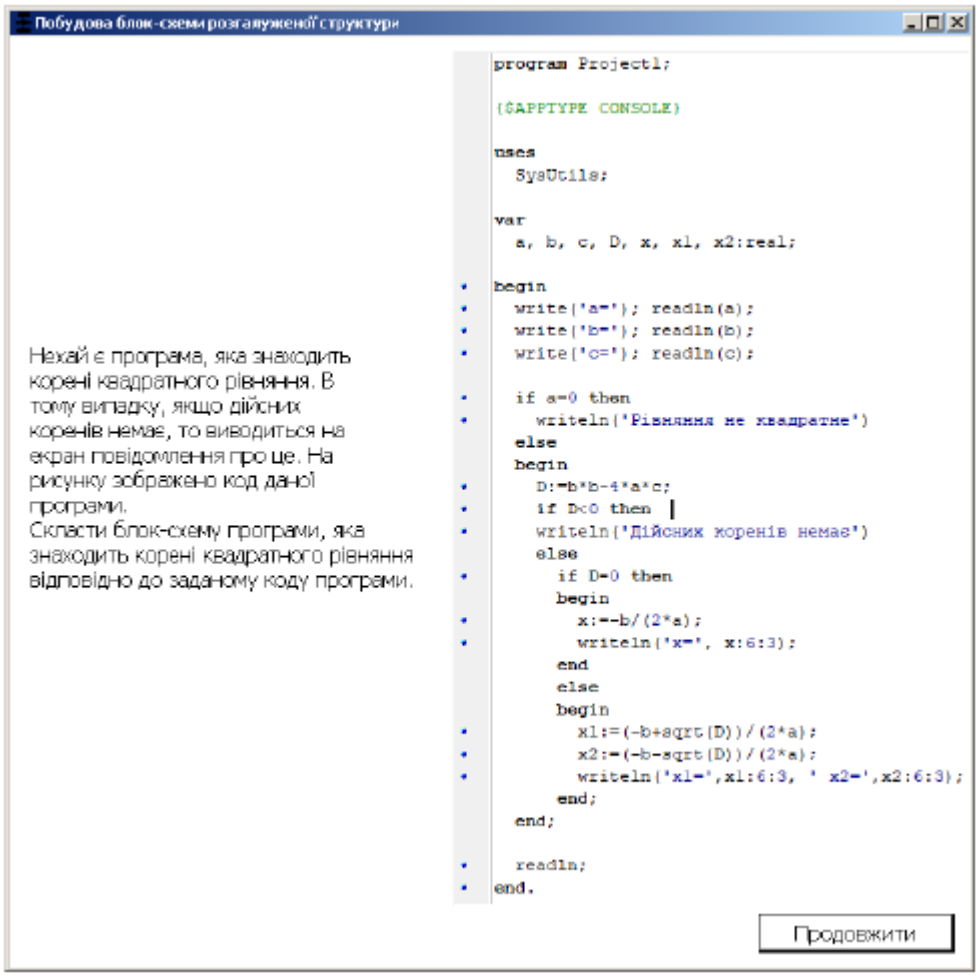


Рисунок 2.2 – Умова прикладу

Після кожного кроку з'являється фрагмент блок-схеми, наприклад, як на рис. 2.5. В кінці тренінгу показується повністю готова блок-схеми, побудована за умовою прикладу (рис. 2.6). Можна також виокремити тренажер «Побудова блок-схем алгоритмів лінійної структури на прикладі циклу for» (на прикладі мови Pascal), створений студентом спеціальності «Інформатика» ПУЕТ Гмизою Б. у 2019 р.

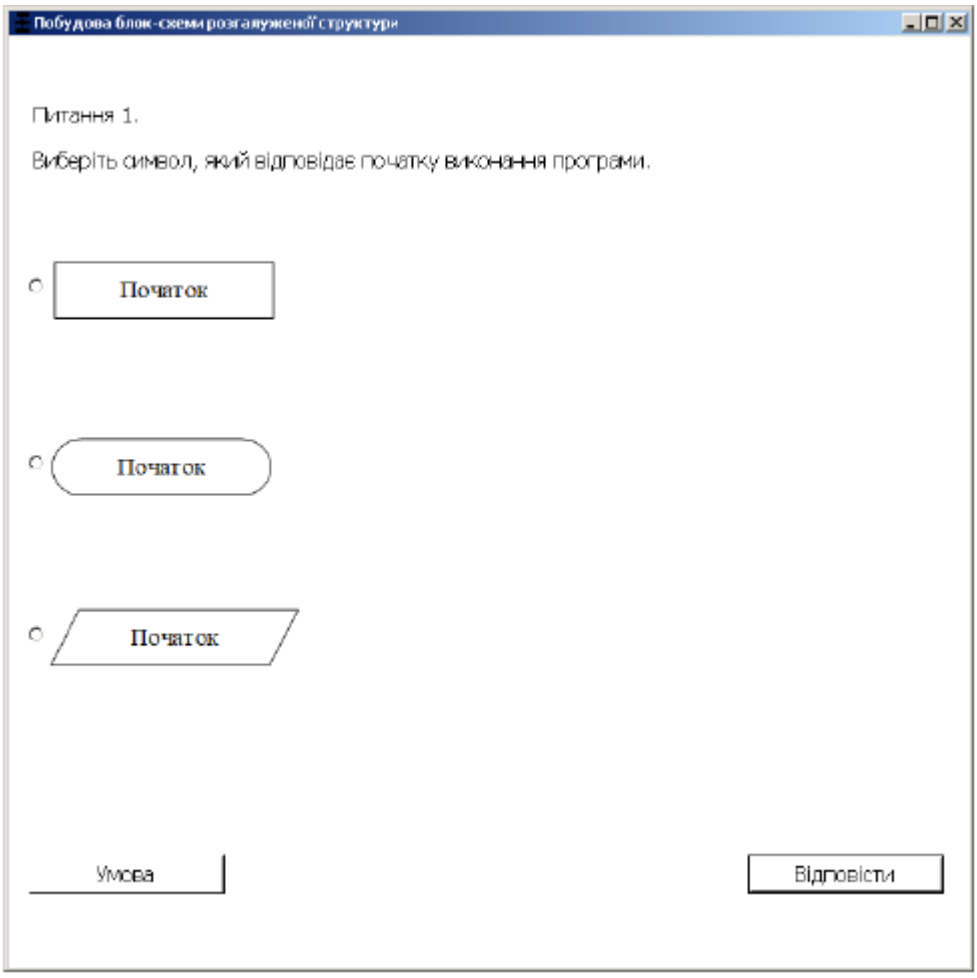


Рисунок 2.3 – Питання 1

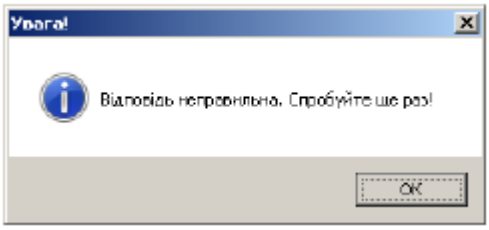


Рисунок 2.4 – Обробка невірної відповіді

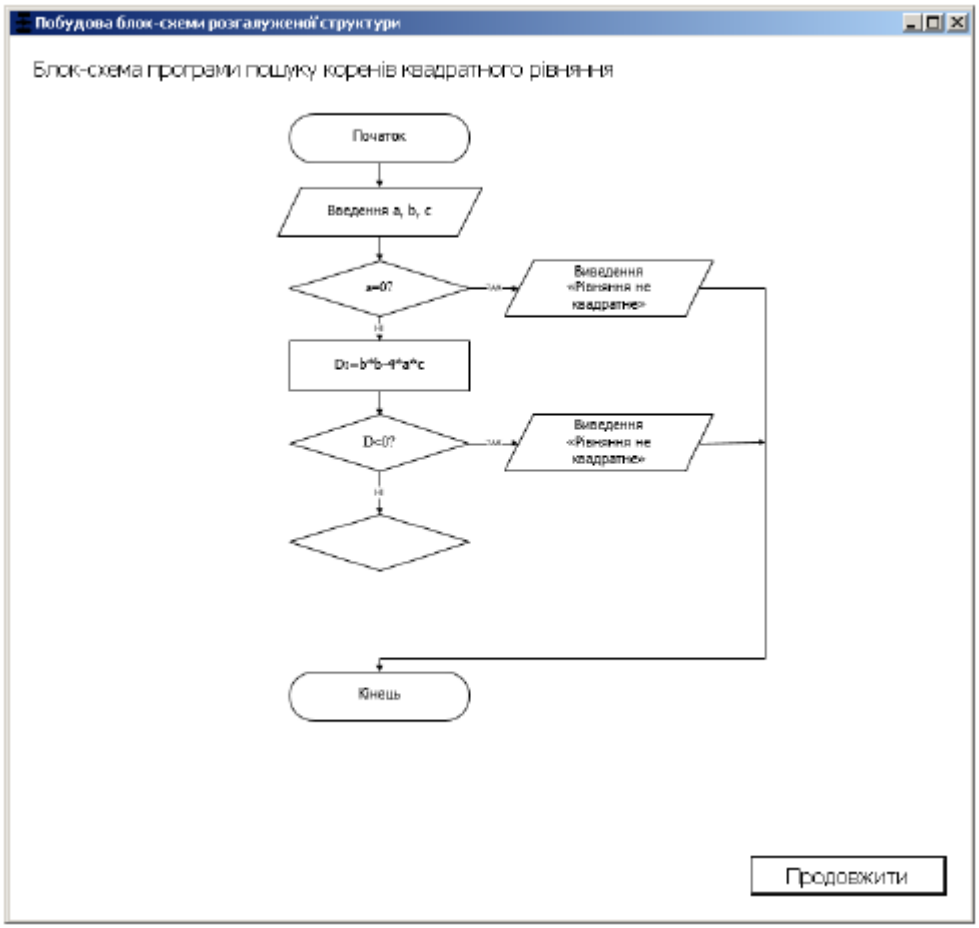


Рисунок 2.5 – Фрагмент побудованої після чергового кроку блок-схеми

В основі роботи цього симулятору (рис. 2.7-2.11) така ж ідея як і в першому. Є один приклад, в якому алгоритм задається кодом мови програмування Pascal (рис. 2.8). Слід побудувати блок-схему. На рис. 2.9 показано одне з питань тренінгу. Після натиснення кнопки «Перевірити», кольором вказується вірні та невірні відповіді (рис. 2.10). Червоним – хибні, зеленим – вірні. Кнопка «Перевірити» змінює назву на «Наступне питання». Натиснувши кнопку вдруге, відбувається перехід до наступного кроку.

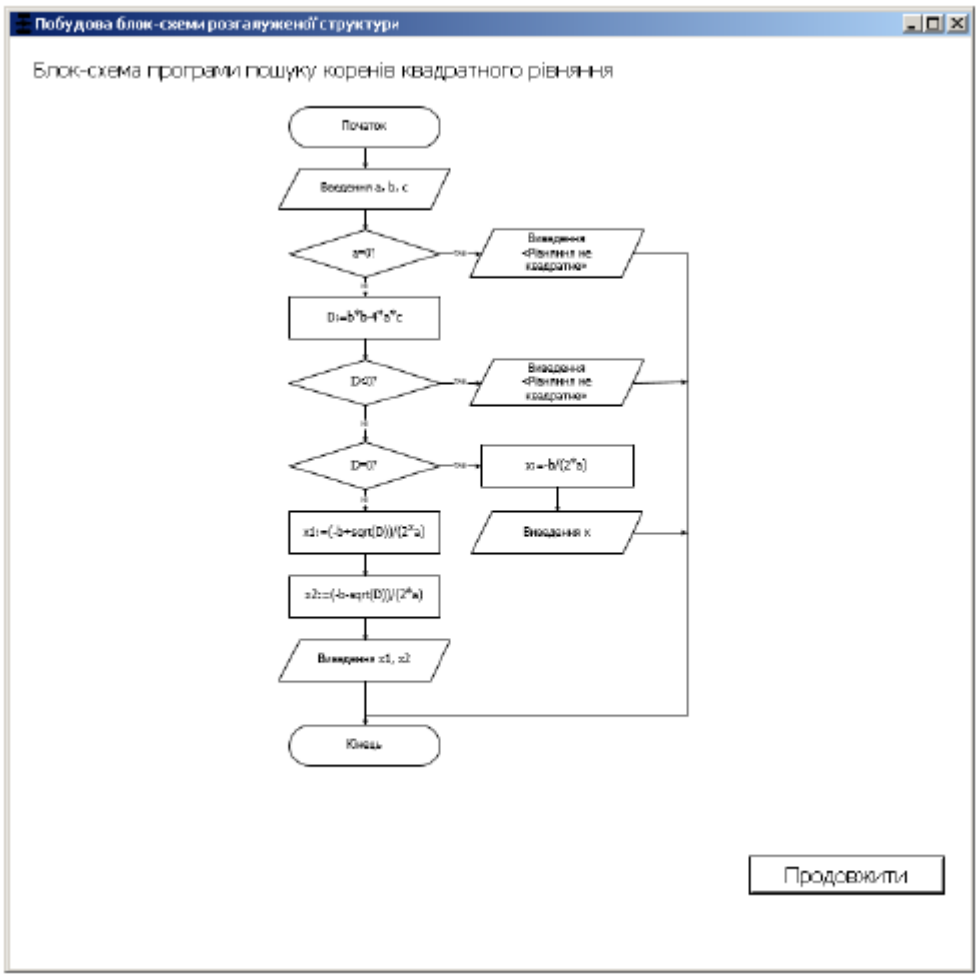


Рисунок 2.6 – Результат

Побудований фрагмент блок-схеми тут не з’являється, лише у кінці тренінгу показується повна блок-схеми, створена за умовою прикладу (рис. 2.11). Третя робота «БлокСхемник» (рис. 2.12-2.19) була знайдена в мережі інтернет і є візуалізатором, який за кодом будує блок-схему. Програма може намалювати блок-схему за кодом трьох мов: Pascal, C, C++. На рис. 2.12-2.13 показано робота додатку для двох прикладів, на базі яких в цій випусковій роботі буде будуватися тренажер. Але побудовані БлокСхемником рисунки не відповідають стандарту.

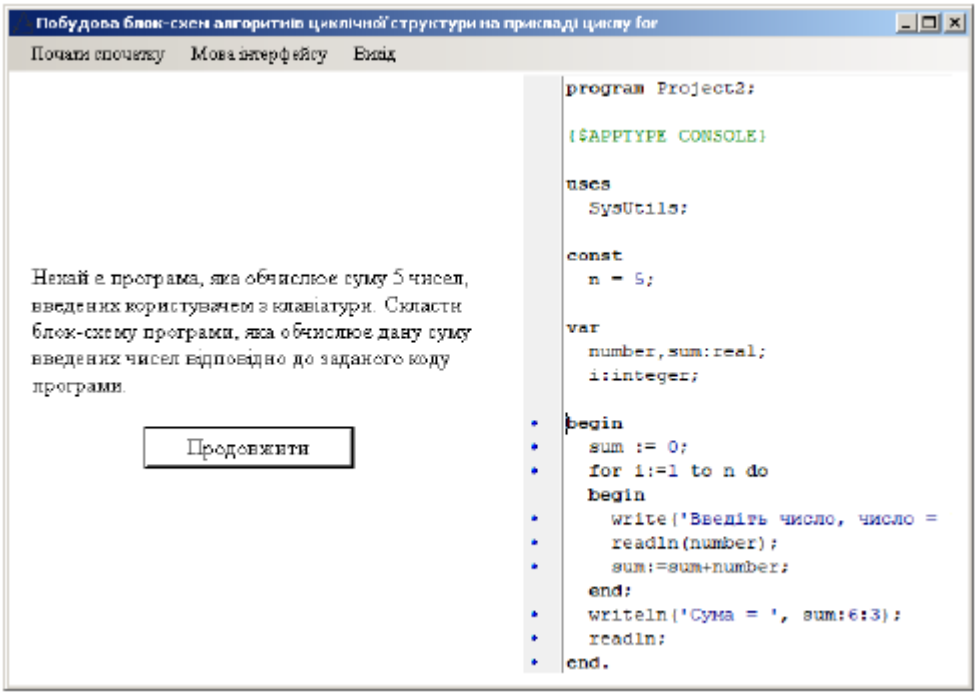


Рисунок 2.8 – Умова прикладу

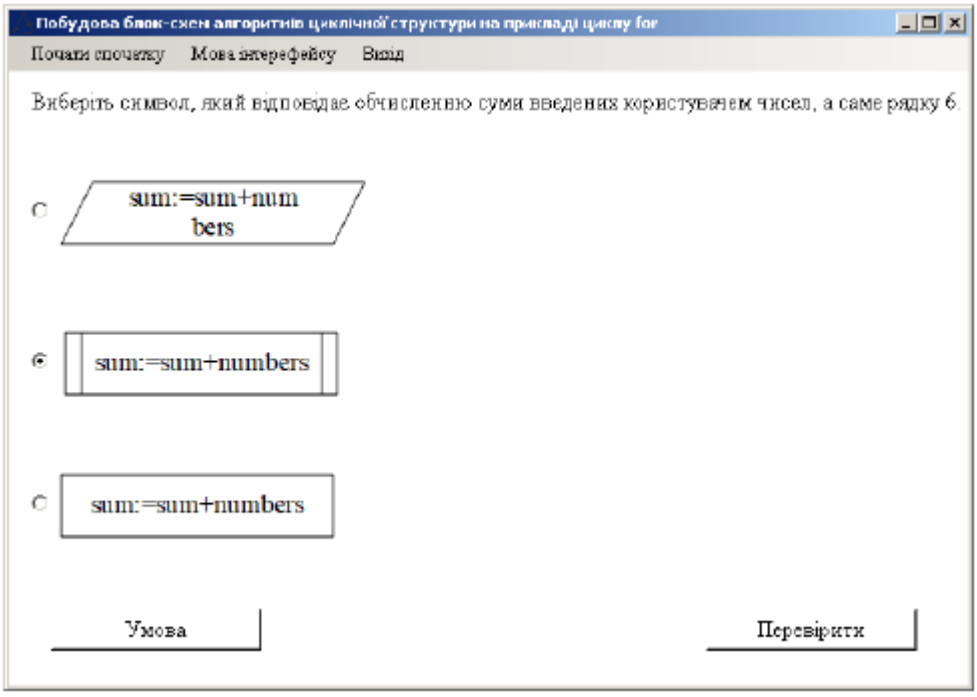


Рисунок 2.9 – Питання тренінгу

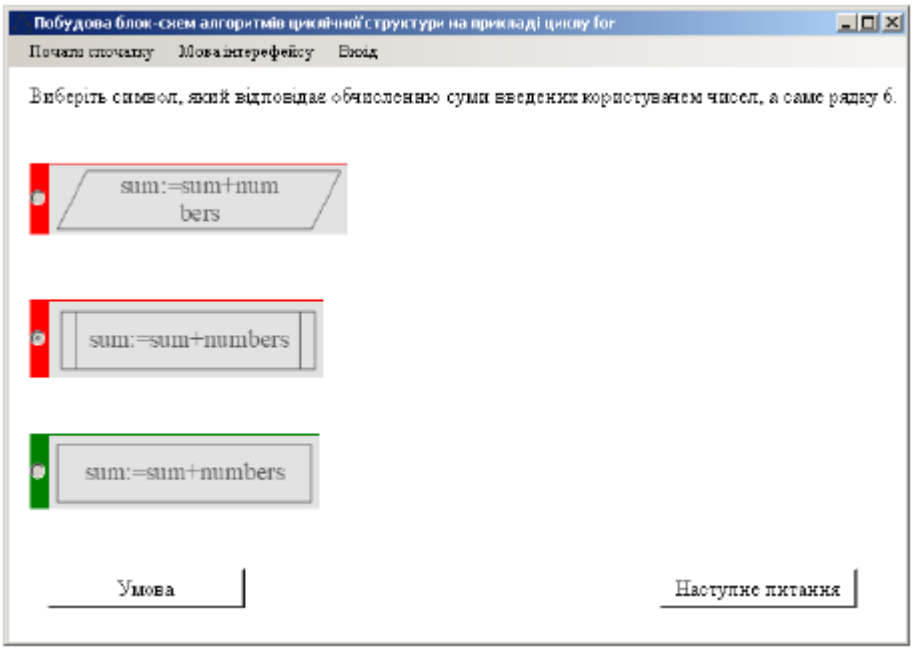


Рисунок 2.10 – Перевірка наданої відповіді

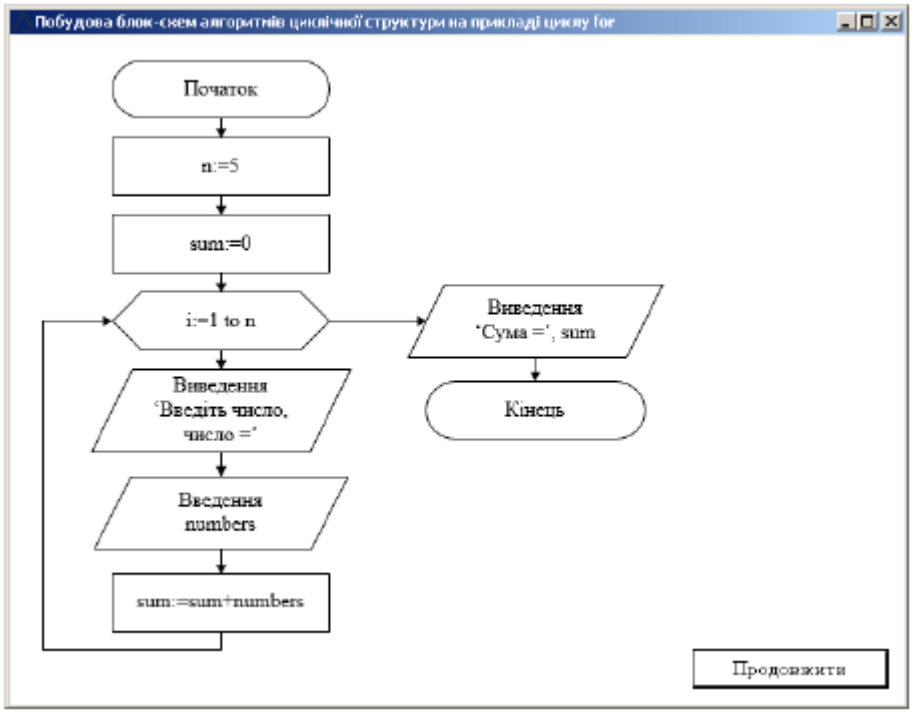


Рисунок 2.11 – Результат

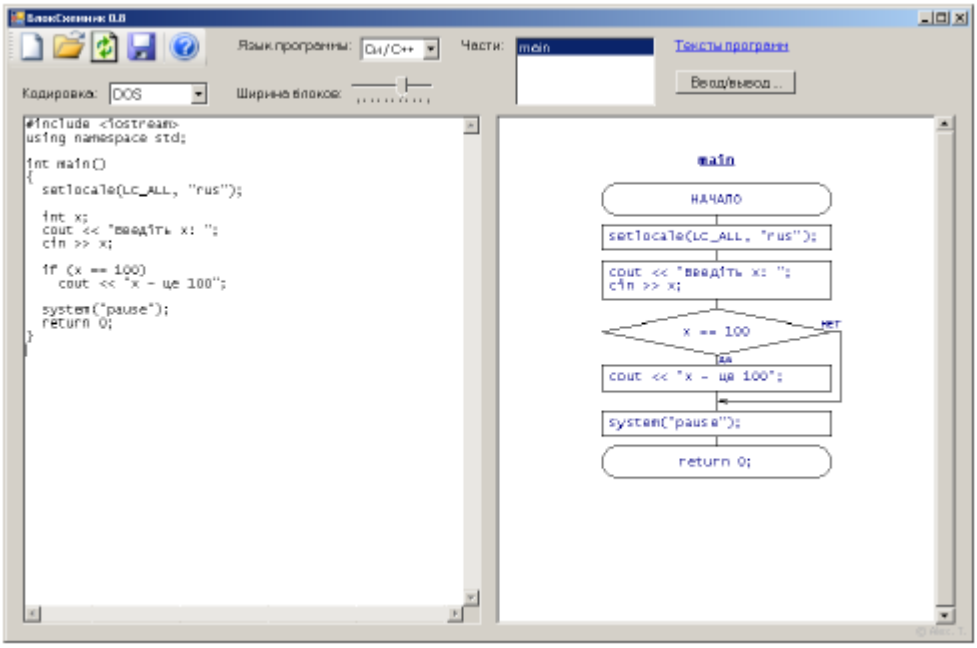


Рисунок 2.12 – Приклад зі структурою if

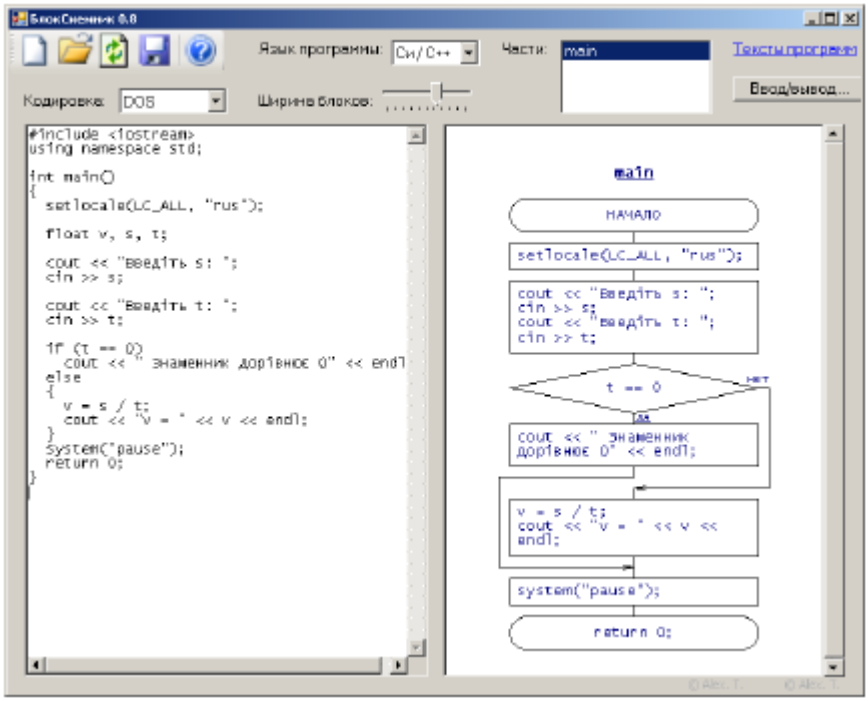


Рисунок 2.13 – Приклад зі структурою if/else

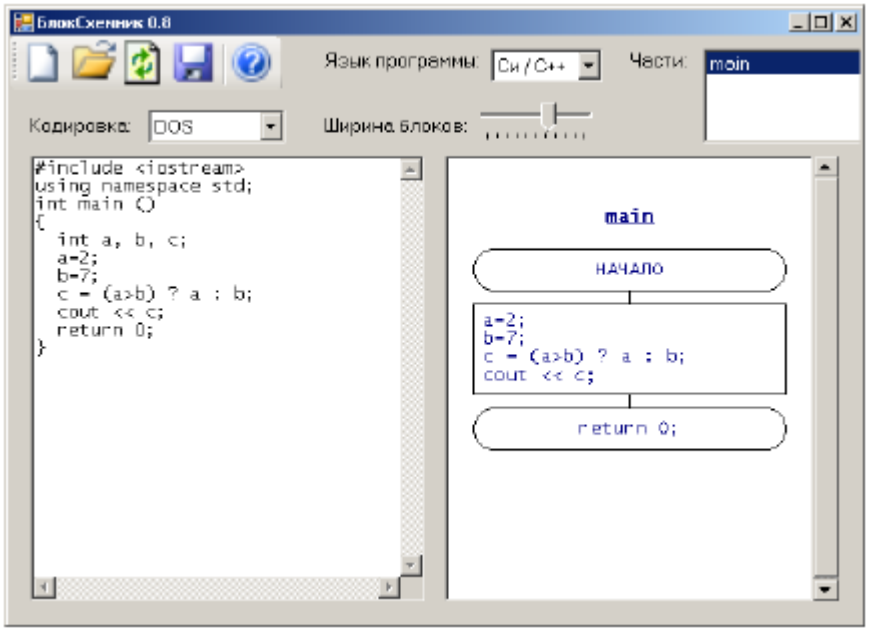


Рисунок 2.14 – Приклад з тринарним умовний оператором

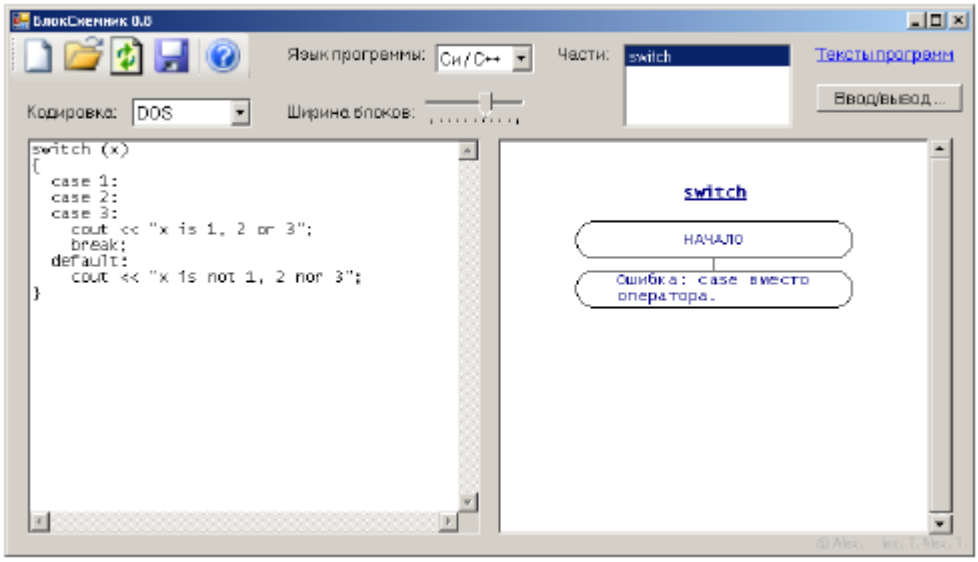


Рисунок 2.15 – Приклад зі структурою switch

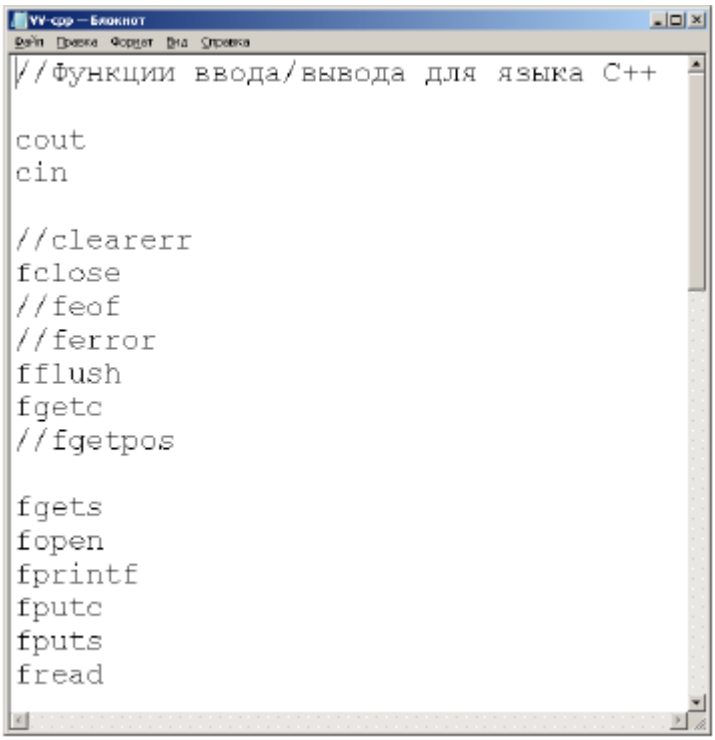


Рисунок 2.16 – Команди мови C++

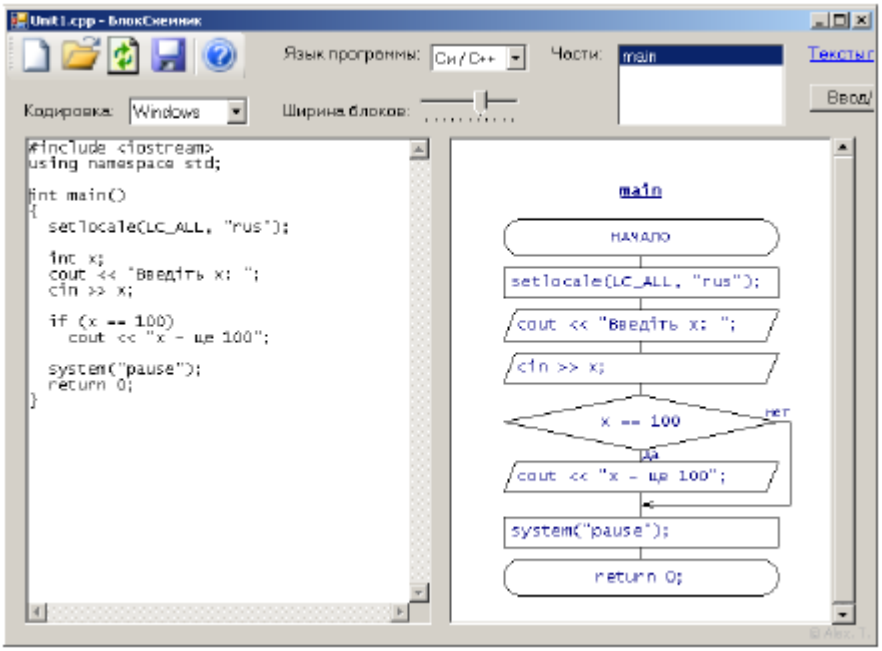


Рисунок 2.17 – Приклад зі структурою if після зміни налаштувань

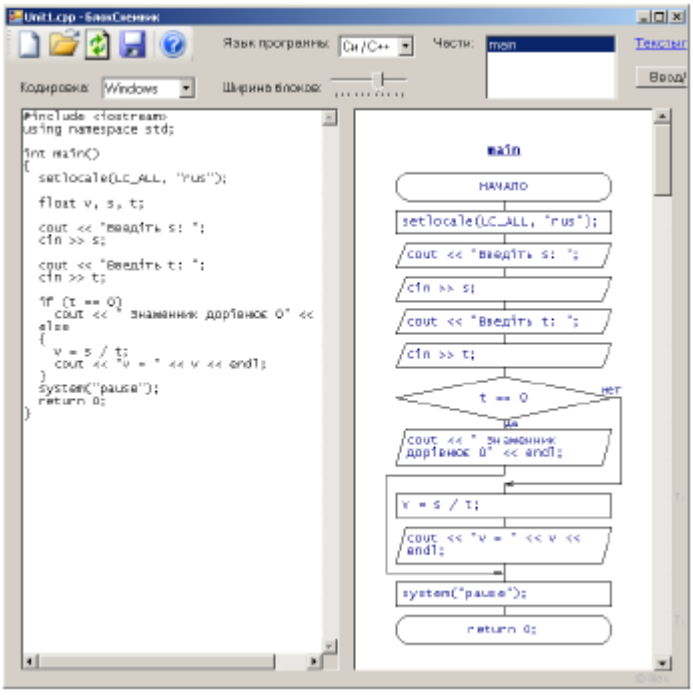


Рисунок 2.18 – Приклад зі структурою if/else після зміни налаштувань

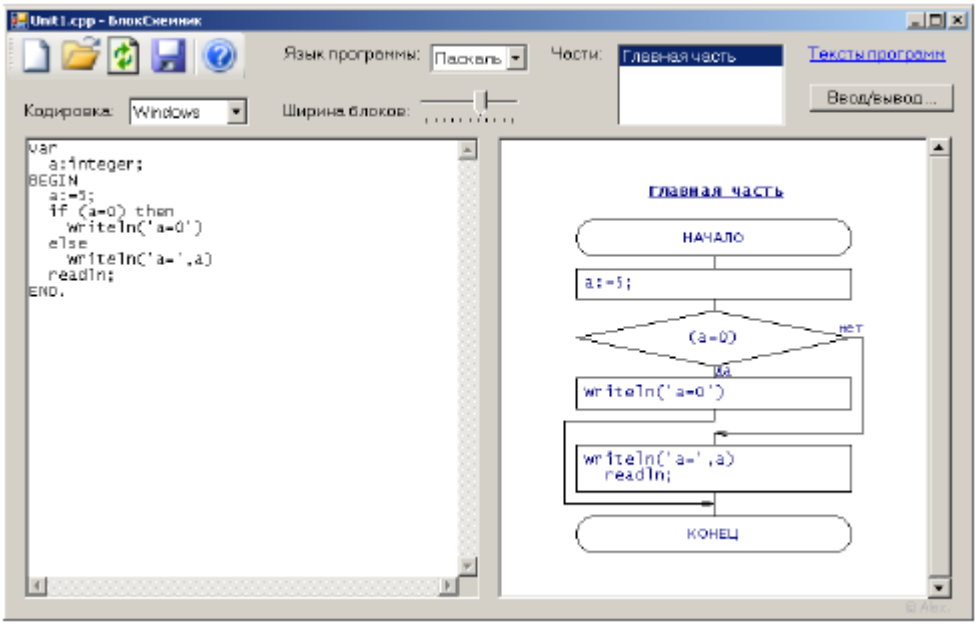


Рисунок 2.18 – Приклад на мові Pascal

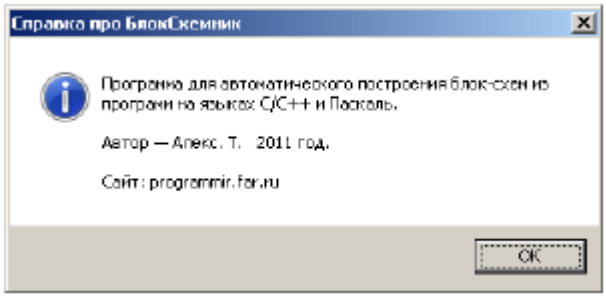


Рисунок 2.19 – Довідка про розробника

Разом з програмою йдуть два текстові файли, в яких перераховані команди вводу та виводу на мові C++ (перший файл) та Pascal (другий файл). Після зміни першого файлу (додано команди cout, cin, рис. 2.16) програма для прикладів з випускової роботи будує більш точні блок-схеми (рис. 2.17-2.18). Інші можливості БлокСхемника: зміна ширину символів блок-схеми; оновлення рисунку; зберігання рисунку (блок-схеми) у окремому файлі; розпізнавання функцій та побудова окремих блок-схем для різних функцій; зміна кодуванння.

2.2. Позитивні аспекти переглянутих робіт

Тренажер Мордасової І.:

1) Після кожного кроку з’являється фрагмент побудованої блок-схеми.

2) Є інформації про розробника.

Тренажер Гмизи Б.

1) Правильна відповідь виділяється зеленим кольором, не вірна – червоним.

2) Програма самостійно виправляє помилки та йде на наступний крок.

Блок-схемник

1) Можливість зчитування коду з файл cpp тощо.

2) Розпізнавання трьох мов: C, C++, Pascal.

3) Розпізнавання функцій у коді та малювання окремних блок-схем для кожної функції.

4) Можливість збереження як картинки блок-схеми.

5) Можливість зміни ширини символів та оновлення картинки.

2.3. Недоліки переглянутих робіт

Тренажер Мордасової І.:

1) Немає пояснення помилки, тільки фіксується, що помилка є.

2) Немає пояснення, як з трьох (або більше) наданих відповідей вірна. Слід вгадувати.

3) Немає фіксації, що користувач не надав відповіді.

4) Немає нумерація рядків ні в завданні, ні протягом тренінгу. А отже, або неможливо відповідати на питання, або слід вгадувати з контексту питання та відповідей.

5) Питання 10 (вибір декількох відповідей з 4 наданих) – багато варіантів для вгадування.

6) Питання 10 (містить помилку) – немає дужок у знаменнику. 1 і 2 відповідь ідентичні, 3 і 4 теж однакові.

7) Помилка на останньому кроці – немає стрілки на лінії, що йде справа наліво, як цього вимагає стандарт.

Тренажер Гмизи Б.

1) Немає заявленого англійськомовного та російськомовного інтерфейсу.

2) Немає нумерація рядків ні в завданні, ні протягом тренінгу.

3) Немає пояснення помилки.

4) Немає фіксації, що користувач не надав жодної відповіді.

5) Після кожного кроку не з’являється фрагмент побудованої блок-схеми.

6) У ітоговій блок-схемі помилка – немає стрілки на лініях, що йдуть справа наліво та знизу догори, як цього вимагає стандарт.

7) Немає нумерації кроків.

8) Немає інформації про розробника.

Блок-схемник

1) Не розпізнає фрагмент коду (рис. 2.15).

2) Не розпізнає тринарний умовний оператор (рис. 2.14).

3) Є помилки у використанні символів (для команд cout та cin повинні відображатися паралелограми, рис. 2.22-2.13).

4) Є порушення вимог стандарту, зокрема, різна висота символів.

5) Є помилки для коду на Pascal. Не розпізнаються команди readln, writeln (рис. 2.18).

2.4. Необхідність та актуальність теми

Переглянуті додатки, містять суттєві помилки, тому не можуть бути використані у навчальному процесі. Крім того, перші дві роботи побудовані на прикладах з кодом іншої мови програмування. Отже, є необхідність в розробці тренажеру з побудови блок-схем алгоритмів лінійної структури на прикладах програм мовою С++. Ця задача є доцільною та актуальною.

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Алгоритм програми

Умова прикладу (код програми) видима впродовж тренінгу. Якщо користувач надав вірну відповідь, то виводиться підтвердження «Правильно!», з’являється побудована частина блок-схеми, відбувається перехід до наступного кроку алгоритму. Якщо користувач надав хибну відповідь, то виводиться «Помилка!» і пояснення помилки. Користувач знову відповідає на питання. Приклад 1. Створити блок-схему програми:

```
1.  #include <iostream>
2.  using namespace std;

3.  int main()
4.  {
5.    setlocale(LC_ALL, "rus");

6.    int x;
7.    cout << "Введіть x ";
8.    cin >> x;

9.    if (x == 100)
10.     cout << "x - це 100";

11.   system("pause");
12.   return 0;
13. }
```

Рисунок 3.1 – Умова прикладу №1

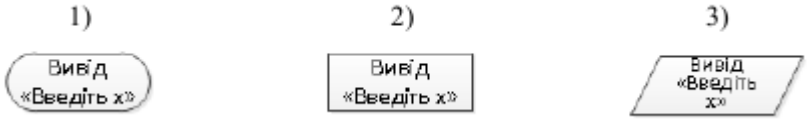
1. Блок-схема починається символом

Правильна відповідь – 1.

Пояснення помилки: «Початок блок-схеми відображається символом «термінатор», який показано у відповіді 1.».

Рисунок 3.2 – Блок-схема, що створюється

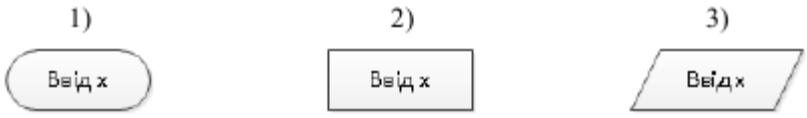
2. Виводу фрази (рядок 7) відповідає символ:



Правильна відповідь – 3.

Пояснення помилки: «Вивід інформації відображається символом «дані», який зображується у вигляді паралелограма (відповідь 3).».

Рисунок 3.3 – Блок-схема, що створюється 3. Введенню x (рядок 8) відповідає символ:

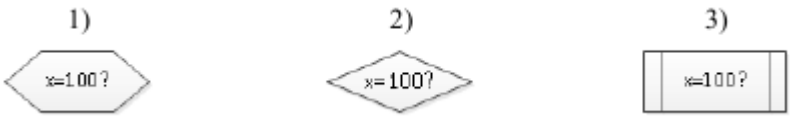


Правильна відповідь – 3.

Пояснення помилки: «Введення значень змінних також відображається символом «дані», який зображується у вигляді паралелограма (відповідь 3).».

Рисунок 3.4 – Блок-схема, що створюється

4. Перевірці умови (рядок 9) відповідає символ:

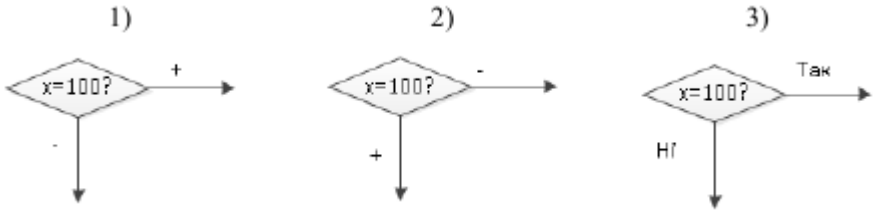


Правильна відповідь – 2.

Пояснення помилки: «Перевірка умови зображується символом «рішення», який зображується у вигляді ромба (відповідь 2).».

Рисунок 3.5 – Блок-схема, що створюється

5. Оберіть можливі варіанти подальшої побудови блок-схеми (рядок 9):



Правильна відповідь – 1, 2, 3.

Пояснення помилки: «Стандарт не регламентує написи та підписи символів. Тому можна позначати +/-, Так/Ні або іншим чином. Стандарт не регламентує, куди вести лінію – вниз чи вправо – у випадку виконання чи не виконання умови. Тому всі три відповіді є вірними.».

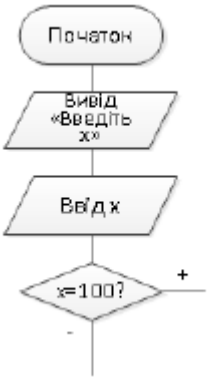
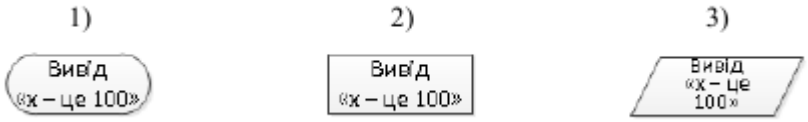


Рисунок 3.6 – Блок-схема, що створюється

6. Виводу фрази (рядок 10) відповідає символ:



Правильна відповідь – 3.

Пояснення помилки: «Вивід інформації відображається символом «дані», який зображується у вигляді паралелограма (відповідь 3).».

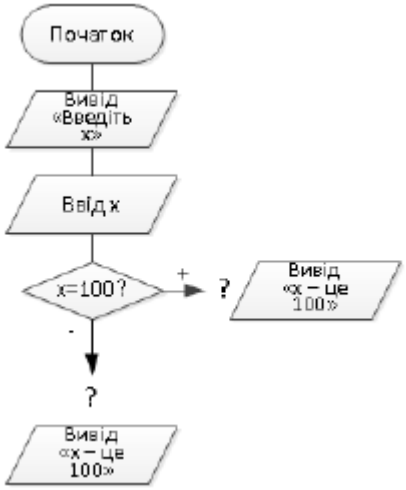
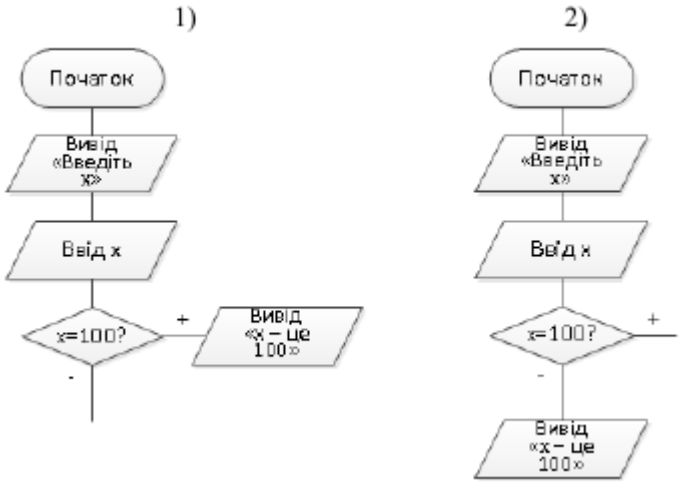


Рисунок 3.7 – Блок-схема, що створюється

7. Виконанню умови (рядки 9-10) відповідає блок-схема:



Правильна відповідь – 1.

Пояснення помилки: «Згідно коду (рядки 9-10) вивід відбувається в тому випадку, коли умова виконується. Отже вірна відповідь – це 1.».

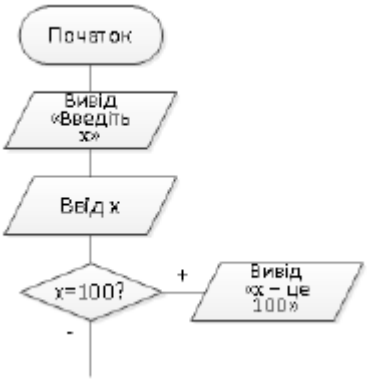


Рисунок 3.8 – Блок-схема, що створюється

8. Блок-схема закінчується символом

Правильна відповідь – 1.

Пояснення помилки: «Кінець блок-схеми (як і початок) відображається символом «термінатор», який показано у відповіді 1.».

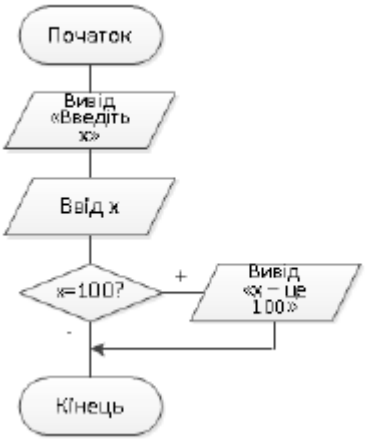


Рисунок 3.9 – Побудована блок-схема

Алгоритм другого прикладу представлено у додатку А.

3.2. Блок-схема алгоритму

Було створено блок-схему алгоритму тренажеру для першого прикладу. Блок-схема показана у додатку Б (рис. Б.1-Б.4).

4. ПРАКТИЧНА ЧАСТИНА

4.1. Опис роботи програмного продукту

Робота тренажеру відображена рисунками 4.1-4.29, В.1-В.14.

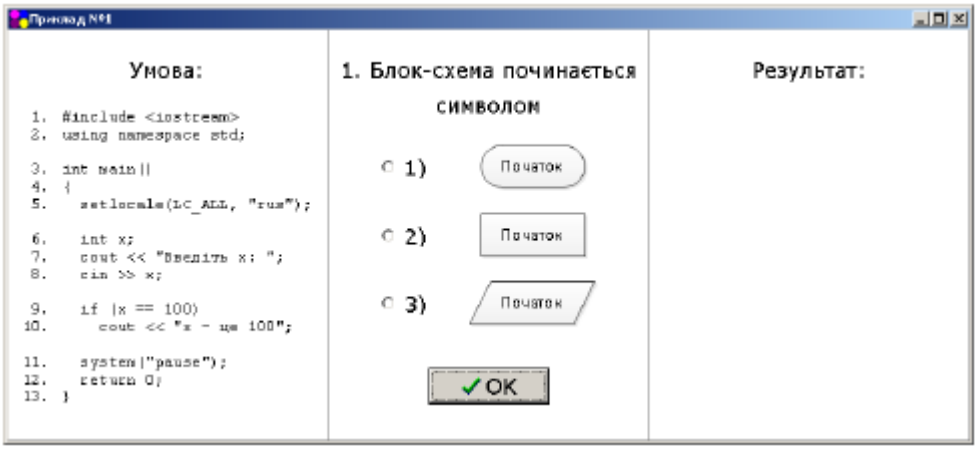


Рисунок 4.2 – Питання №1

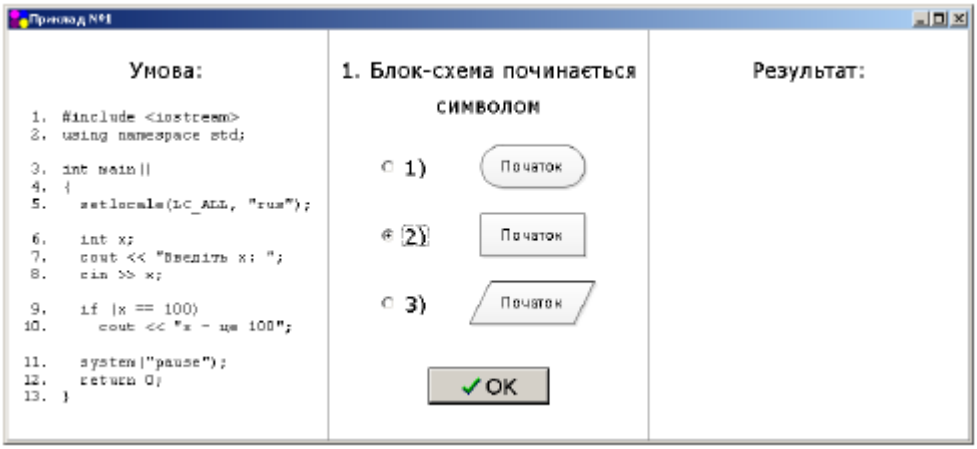


Рисунок 4.3 – Питання №1 (з помилкою)

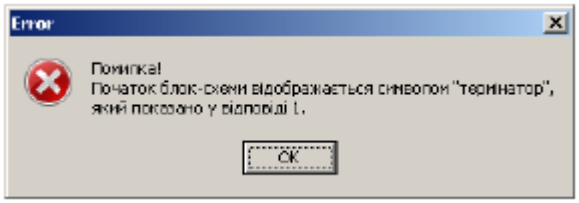


Рисунок 4.4 – Пояснення похибки

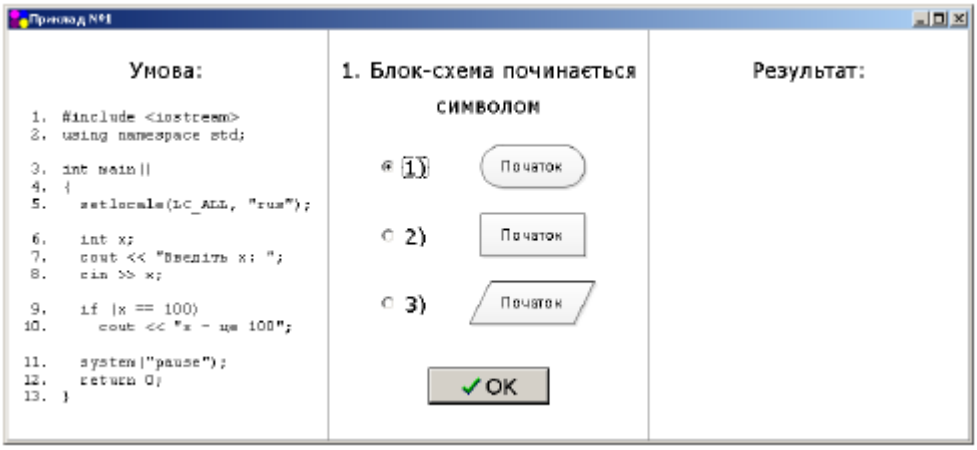


Рисунок 4.5 – Питання №1 (з вірною відповіддю)

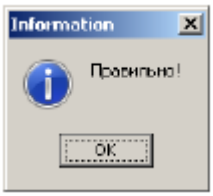


Рисунок 4.6 – У випадку вірності відповіді

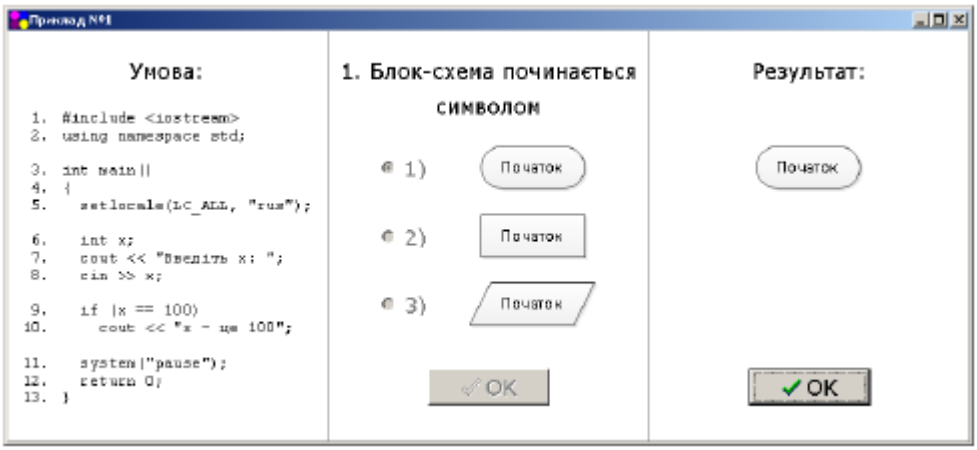


Рисунок 4.7 – Питання №1 (з побудованим фрагментом блок-схеми)

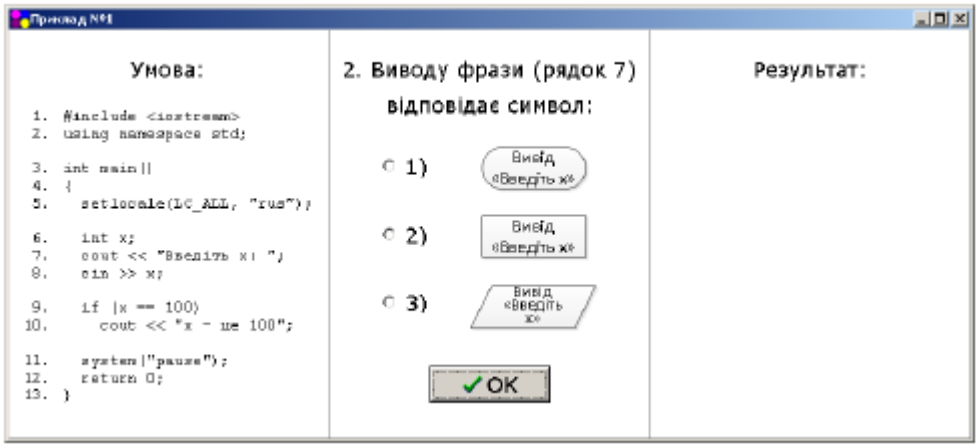


Рисунок 4.8 – Питання №2

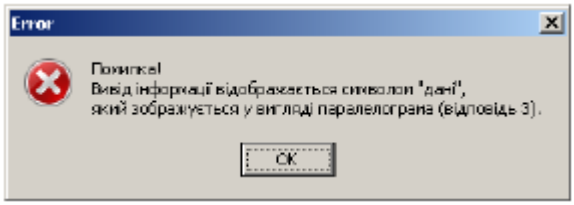


Рисунок 4.9 – Пояснення похибки

Рисунок 4.10 – Питання №2 (з побудованим фрагментом блок-схеми)

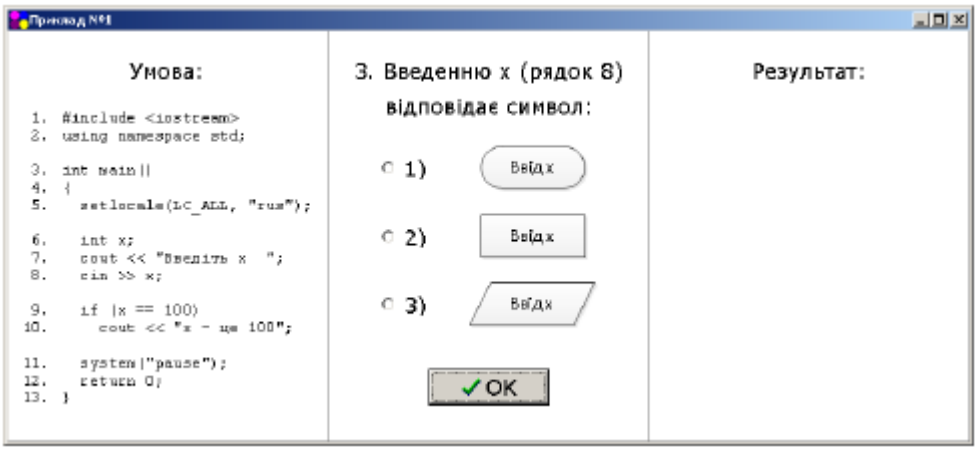


Рисунок 4.11 – Питання №3

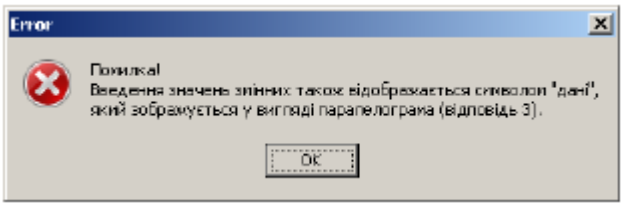


Рисунок 4.12 – Пояснення похибки

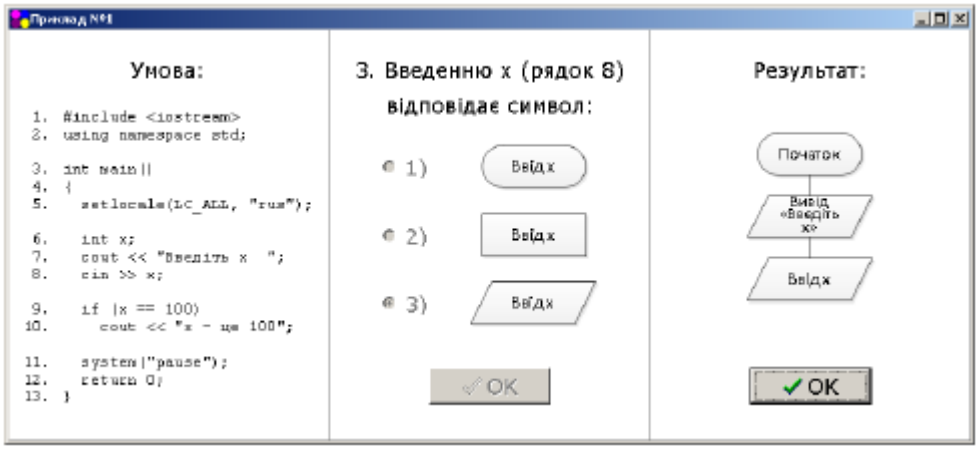


Рисунок 4.13 – Питання №3 (з побудованим фрагментом блок-схеми)

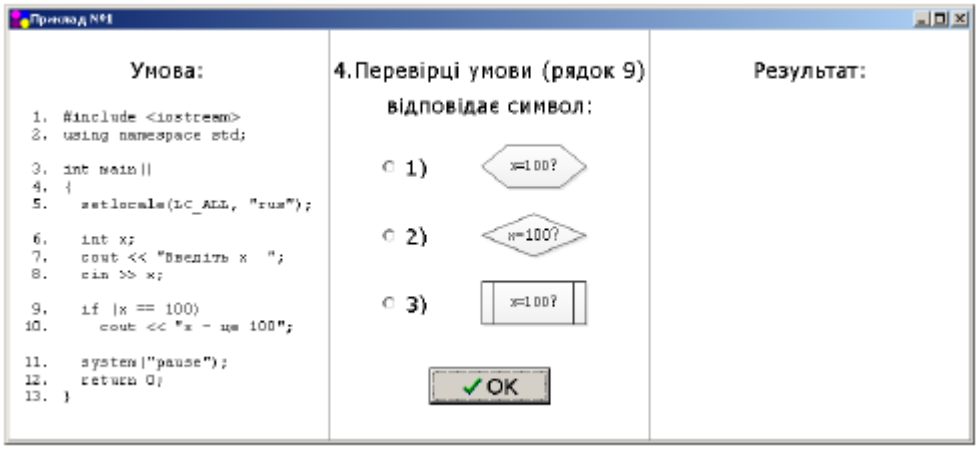


Рисунок 4.14 – Питання №4

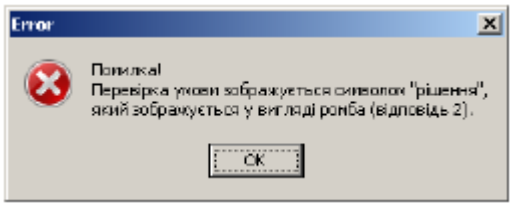


Рисунок 4.15 – Пояснення похибки

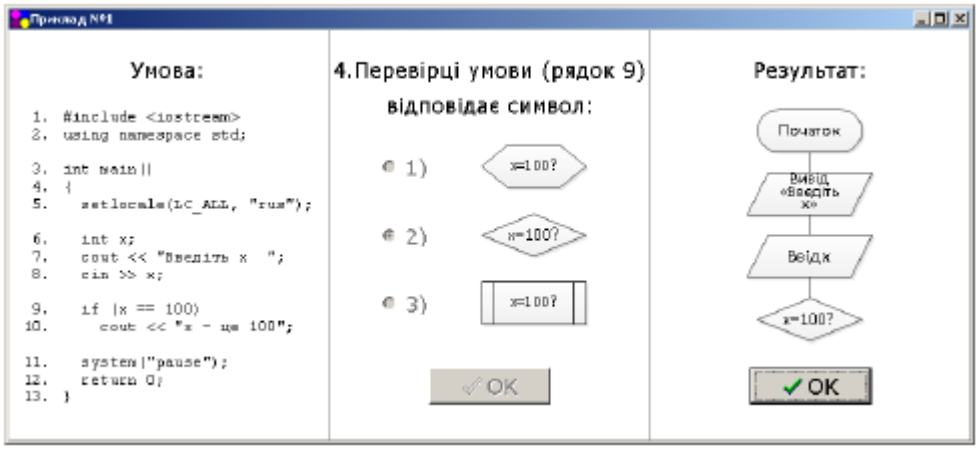


Рисунок 4.16 – Питання №4 (з побудованим фрагментом блок-схеми)

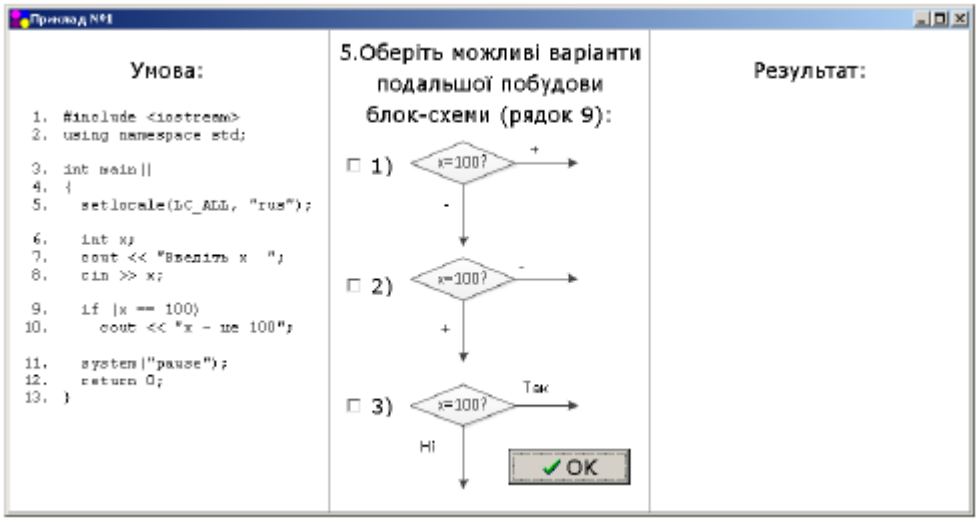


Рисунок 4.17 – Питання №5

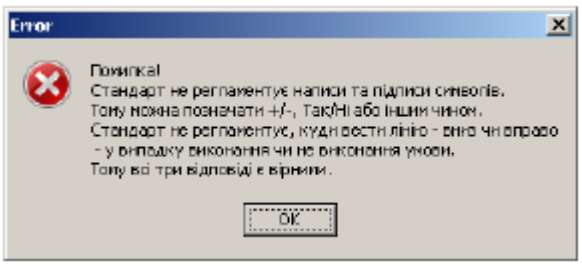


Рисунок 4.18 – Пояснення похибки

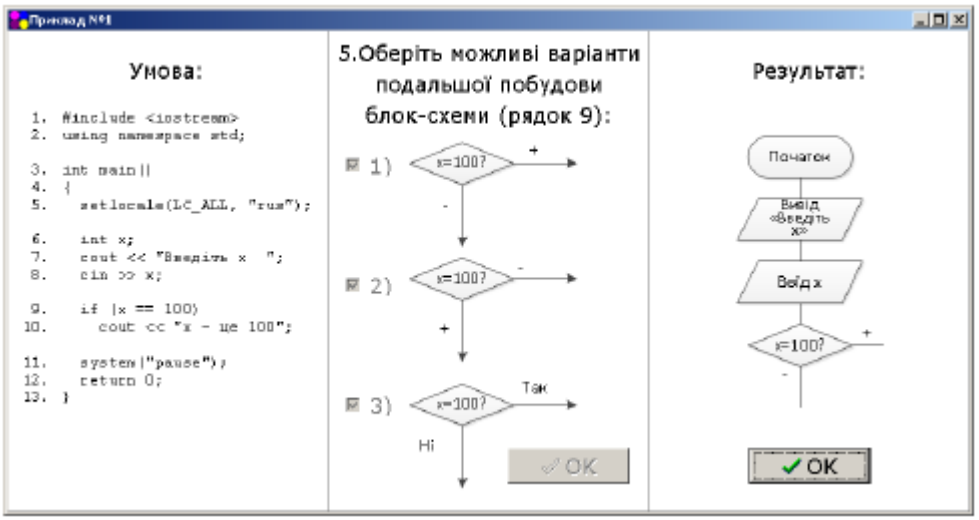


Рисунок 4.19 – Питання №5 (з побудованим фрагментом блок-схеми)

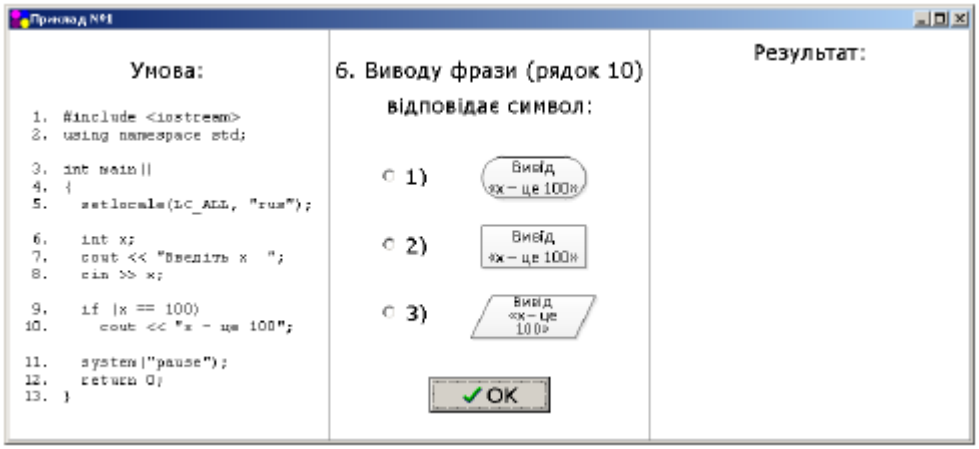


Рисунок 4.20 – Питання №6

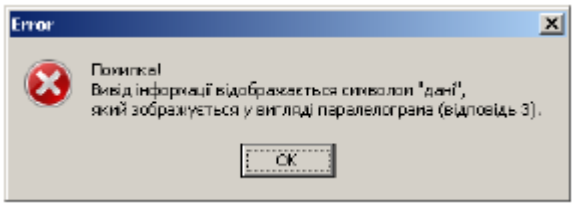


Рисунок 4.21 – Пояснення похибки

Рисунок 4.22 – Питання №6 (з побудованим фрагментом блок-схеми)

Рисунок 4.23 – Питання №7

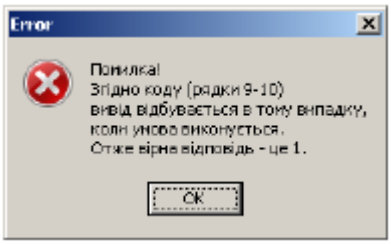


Рисунок 4.24 – Пояснення похибки

Рисунок 4.25 – Питання №7 (з побудованим фрагментом блок-схеми)

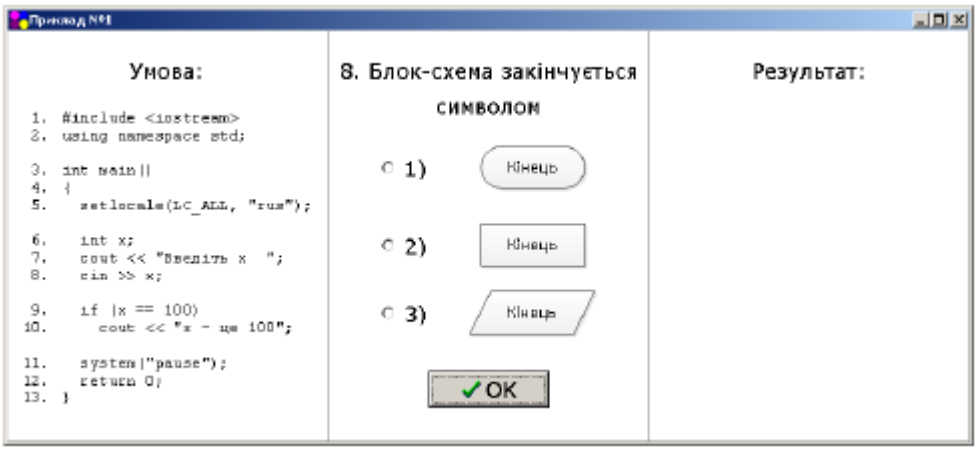


Рисунок 4.26 – Питання №8

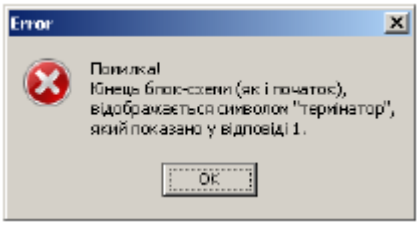


Рисунок 4.27 – Пояснення похибки

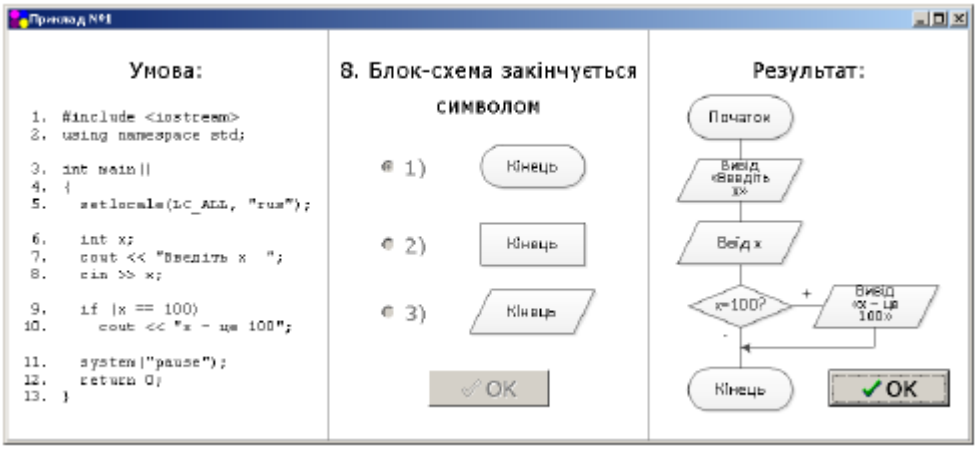


Рисунок 4.28 – Питання №8 (з побудованою блок-схемою)

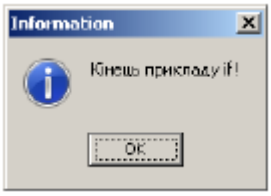


Рисунок 4.29 – Останнє повідомлення

4.2. Опис процесу створення програми

Для створення програми була обрана мова C++ та середовище програмування Borland Builder. Розглянемо роботу програми для першого питання. Форма для цього кроку (як і для всіх наступних) поділена на три частини.

В першій частині (лівій) представлена умова завдання (код програми). В другій частині (центральній) подані питання та варіанти відповідей. В третій частині (правій) показано фрагмент блок-схеми, що створюється. Рисунок стає видимим після надання правильної відповіді, і показує новий символ або лінії, що додались до блок-схеми на цьому кроці. Користувач спочатку працює з центральної частиною, де надає свою відповідь і натискає кнопку «OK». Якщо відповідь помилкова, то з’являється пояснення помилки. Користувач повинен виправити відповідь на вірну та знову натиснути кнопку «OK». Якщо

відповідь вірна, то з’являється повідомлення про це, центральна частина форми блокується від змін, стає активною права частина.

У правій частині з’являється рисунок та кнопку. Після перегляду та аналізу рисунку користувач натискає кнопку «ОК». Відбувається перехід до наступного кроку. Для першої кнопки «ОК» був створений код:

```
// питання 1
// кнопка «ОК» (у центральній частині вікна)
void __fastcall TForm2::BitBtn1Click(TObject *Sender)
{
 if (RadioButton1->Checked==true)
 {
  MessageDlg("Правильно!", mtInformation, TMsgDlgButtons() << mbOK, 0);
  Image2->Visible=true;
  BitBtn2->Visible=true;
  BitBtn2->SetFocus();

  BitBtn1->Enabled=false;
  RadioButton1->Enabled=false;
  RadioButton2->Enabled=false;
  RadioButton3->Enabled=false;
```

```
 }
 else
 {
  MessageDlg("Помилка!\nПочаток блок-схеми відображається символом
\"термінатор\",\який показано у відповіді 1.", mtError, TMsgDlgButtons()
<< mbOK, 0);
 }
}
```

Якщо відповідь вірна (тут це перша надана альтернатива) RadioButton1-> Checked==true, то з’являється повідомлення. Повідомлення створено за допомогою стандартної процедури MessageDlg: MessageDlg("Правильно!", mtInformation, TMsgDlgButtons() << mbOK, 0). Далі стає видимою картинка у правій частині: Image2->Visible=true. Стає видимою картинка у правій частині:

BitBtn2->Visible=true. Ця кнопка приймає фокус (стає активною в першу чергу): BitBtn2-> SetFocus().

Усі компоненти центральної частини блокуються (стають неактивними для будь-яких дій користувача): BitBtn1->Enabled=false; RadioButton1->Enabled=false; RadioButton2->Enabled=false; RadioButton3->Enabled=false. Якщо відповідь користувача невірна (обрано другу, третю альтернативи або не обраної жодної з трьох), то з'являється повідомлення про помилки та її пояснення: MessageDlg("Помилка!\nПочаток блок-схеми відображається символом \"термінатор\",\nякий показано у відповіді 1.", mtError, TMsgDlgButtons() << mbOK, 0). Тут символ \n означає перехід на наступний рядок, а символ \" – вивід лапок у повідомленні. Для другої кнопки «ОК» був створений код:

```
// кнопка «ОК» (у правій частині вікна)
void __fastcall TForm2::BitBtn2Click(TObject *Sender)
{
   Form3->Show();
   Form2->Hide();
}
```

Поточне вікно програми приховується: Form2->Hide(), а наступне (з наступним питанням) з'являється на екрані: Form3->Show().

ВИСНОВКИ

В дипломному проекті відбулось знайомство зі створенням блок-схем алгоритмів; з умовними операторами мови С++.

Серед умовних операторів в С++ можна виділити чотири структури:

- if;
- if/else;
- switch;
- (condition : true ? false).

Було підібрано декілька прикладів програм з використанням вказаних структур. Для дипломного проекту було обрано два з них: зі структурами if та if/else. Для них створено блок-схему та алгоритм тренажеру.

Алгоритм складається з восьми кроків для прикладу 1 та дванадцяти кроків для прикладу 2. Алгоритм тренажеру закодовано мовою С++. Програма перевірена на помилки та описки. Продукт повністю готовий до використання. Процес створення та роботи програмного додатку задокументовано та викладено у пояснювальній записці до дипломного проекту.

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ:

1. Абрамов С. А. Задачи по программированию // С. А. Абрамов, Г. Г. Гнездилова, Е. Н. Капустина, М. И. Селюн// М.: Наука, 1988. – 256с.
2. Александров А.Г. Оптимальные и адаптивные системы.— М.: Высш. школа, 1989.— 263 с.
3. Алферова. З.В. Теория алгоритмов.- М.: Статистика, 1973.- 164 с.
4. Баженова И.Ю. Язык программирования Delphi// И.Ю. Баженова// АО "Диалог-МИФИ", 1997. – 366с.
5. Бартлетт Н. Программирование на Delphi Путеводитель // Н. Бартлетт А. Лесли, С. Симкин / Издательство НИПФ "ДиаСофт Лтд.",1996. – 116с.
6. Березин Б.И., Начальный курс Delphi // Б.И.Березин, С.Б.Березин // М.: ДИАЛОГ-МИФИ, 1996. – 331с.
7. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання: ДСТУ 7.1-2006. – [Чинний від 2007-07-01]. – К. : Держспоживстандарт України, 2007. – 47 с.
8. Бондарев В.М. Основы программирования // В.М. Бондарев, В.И. Рублинецкий, Е.Г. Качко // Харьков: Фолио, Ростов н/Д: Феникс, 1997. – 446с.
9. Брукшир Дж. Гленн Введение в компютерные науки.— М.: Издательский дом "Вильямс", 2001.— 668 с.
10. Вебер Дж. Технология С++в подлиннике // Дж. Вебер //QUE Corporation, 1996. – 256с.
11. Вирт Н. Алгоритмы + структуры данных = программы.— М.: Мир, 1985.-406 с.
12. Вирт Н. Алгоритмы и структуры данных. // Н. Вирт // М.: Мир, 1989. - 345с.
13. Волш А. И. Основы программирования на С++для World Wide Web // А. И. Волш // Издательство "Диалектика",1996. – 458с.

14. Гладков В. П. Задачи по информатике на вступительном экзамене в вуз и их решения: Учебное пособие // В. П. Гладков // Пермь: Перм. техн. ун-т, 1994. – 516с.
15. Грин Д., Кнут Д. Математические методы анализа алгоритмов.- М.: Мир, 1987.- 120 с.
16. Грогоно П. Программирование на языке Delphi // П. Грогоно // М.: Мир, 1982. – 216с.
17. Громов Г.Р. Очерки информационной технологии.— М.: ИнфАрт, 1992. – 336 с.
18. Дагене В.А. 100 задач по программированию // В.А.Дагене, Г.К. Григас, К.Ф. Аугутис // М.: Просвещение, 1993. – 106с.
19. Джамса К. Библиотека программиста Java // К.Джамса // ООО "Попурри", 1996. – 656с.
20. Драґан Я.П., Сікора Л.С., Яворський Б.І. Основи сучасної теорії стохастичних сигналів: енергетична концепція, математичний апарат, фізичне тлумачення. - Львів: ЕБТЕС, 1999.- 132 с.
21. Ємець О. О. Методичні рекомендації до виконання бакалаврської роботи для студентів за освітньою програмою «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки та інформаційні технології» галузь знань - 12 «Інформаційні технології» / О.О.(Олег) Ємець. – Полтава : РВВ ПУЕТ, 2017. – 71 с.
22. Заварыкин В.М. Основы информатики и вычислительной техники // В.М. Заварыкин, В.Г. Житомирский, М.П. Лапчик // М.: Просвещение, 1989. – 556с.
23. Капітонова Ю.В., Кривий С.Л., Летичевський О.А. та. ін. Основи дискретної математики.— Київ: Наук. думка, 2002.— 560 с.
24. Касаткин В. Н. Информация. Алгоритмы. ЭВМ // В. Н. Касаткин // М.: Просвещение, 1991. – 219с.
25. Кен А. Язык программирования Delphi // А. Кен, Дж. Гослинг // Издательство "Питер-Пресс", 1997. – 378с.

26. Керниган Б. Язык программирования Delphi // Б. Керниган, Д. Ритчи // Пер. с англ. — М.: Финансы и статистика, 1992. – 391с.
27. Кириллов А.А. Элементы теории представлений. - М.: Наука,1972.336с.
28. Кузьмин И.В., Березюк И.Т., Фурманов К.К., Шаронов В.Б. Синтез вычислительных алгоритмов управления и контроля. – Л.: Техніка, 1975. – 248 с.
29. Лавров И.А., Максимова Л.Л. Задачи по теории множеств, математической логике и теории алгоритмов. – М.: Наука, 1984. – 224с.
30. Ляхович В.Ф. Руководство к решению задач по основам информатики и вычислительной техники // В.Ф. Ляхович// М.: Высшая школа, 1994. – 127с.
31. Марков А. С. «Базы данных. Введение в теорию и методологию // А. С. Марков, К.Ю. Лисовский // Финансы и статистика»-2006,-Р. 24-35.
32. Марков А. С. «Базы данных. Введение в теорию и методологію // А. С. Марков, К.Ю. Лисовский // Финансы и статистика»-2006, - Р. 24-35.
33. Мейнджер Дж. Delphi Основы программирования // Дж. Мейнджер // Издательская группа BHV, Киев,1997. – 346с.
34. Методичні вказівки до написання техніко-економічного розділу дипломних проектів освітньо-кваліфікаційного рівня «ба калавр» напряму підготовки 6.050102 комп’ютерна інженерія/ І.Р. Паздрій // Тернопіль: ТНЕУ, 2014. – 37 с.
35. Методичні рекомендації до виконання дипломного проекту з освітньо-кваліфікаційного рівня ―Бакалавр‖ напряму підготовки 6.050102 «Комп’ютерна інженерія» фахового спрямування «Комп’ютерні системи та мережі» / О.М. Березький, Л.О.Дубчак, Р.Б. Трембач, Г.М. Мельник, Ю.М. Батько, С.В. Івасьєв // Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2014.–65 с.

36. Миков А. И. Информатика. Введение в компьютерные науки // А. И. Миков // Пермь: Изд-во ПГУ, 1998. – 442с.
37. Могилев А. В. Информатика: Учеб. пособие для студ. пед. вузов // А. В. Могилев, Е. К. Хеннера.// М.: Изд. центр «Академия», 1999. – 629с.
38. Нотон П. JAVA:Справ.руководство// П.Нотон, А.Тихонова.-М.:БИНОМ:Восточ.Кн.Компания,1996:Восточ.Кн.Компания. - 447с.
39. Нотон П. Полный справочник по Java // П. Нотон, Г. Шилдт// McGraw-Hill,1997, Издательство "Диалектика",1997. – 556с.
40. Ренеган Э.Дж. 1001 адрес WEB для программистов :Новейший путеводитель программиста по ресурсам World Wide Web:Пер.// Э.Дж. Ренеган /с англ..-Минск:Попурри,1997.-512с.
41. Родли Дж. Создание Java-апплетов // Дж. Родли // The Coriolis Group,Inc.,1996, Издательство НИПФ "ДиаСофт Лтд.",1996. – 466с.
42. Семакина И. Г. Информатика. Задачник-практикум: В 2 т. // И. Г. Семакина, Е.К.Хеннера // М.: Лаборатория Базовых Знаний, 1999. – 476с.
43. Сергиенко И.В. Математические модели и методы решения задач дискретной оптимизации.— Киев: Наук. думка, 1988.— 472 с.
44. Сигорский В.П. Математический аппарат инженера.— К.: Техника, 1975.— 768 с.
45. Сокольский М.В. Все об Intranet и Internet // М.В. Сокольский // М.:Элиот,1998. - 254с.ил.
46. Тассел Д. Стиль, разработка, эффективность, отладка и испытание программ // Д. Тассел // М.: Мир, 1981. – 56с.
47. Томас М. Секреты программирования для Internet на Java // М.Томас, П. Пратик, А. Хадсон, Д. Болл// Ventana Press, Ventana Communications Group, U.S.A., 1997. – 396с.
48. Томович Р., Вукобратович М. Общая теория чувствительности.— М.: Сов. радио, 1972.—240 с.

49. Тюрин Ю.Н. Анализ данных на компьютере. // Ю.Н. Тюрин, А.А. Макаров, В.Э.Фигурнова // М.: ИНФРА-М, Финансы и статистика, 1995. - 384с.
50. Флэнэген Д. Java in a Nutshell // Д. Флэнэген // O'Reilly & Associates, Inc., 1997, Издательская группа BHV, Киев, 1998. – 473с. 33.Чен М.С. Программирование на С++:1001 совет: Наиболее полное руководство по Java и Visual J++: Пер.с англ.// М.С.Чен, С.В. Грифис, Э.Ф. Изи./ - Минск:Попурри,1997.- 640с.
51. Эферган М. С++: справочник // М. Эферган // QUE Corporation, 1997, Издательство "Питер Ком", 1998. – 256с.
52. Яворський Б. І. Математичні основи радіоелектроніки/ В 3-х частинах. – Тернопіль: ТПІ, 1996. – 336 с.

ДОДАТКИ

Програмний код

Main.java

```
package sample;
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
public class Main extends Application {
@Override
public void start(Stage primaryStage) throws Exception{
Parent root =
FXMLLoader.load(getClass().getResource("ChooseLang.fxml"));
primaryStage.setTitle("Головне меню/Main Menu");
primaryStage.setScene(new Scene(root, 500, 300));
primaryStage.show();
}
public static void main(String[] args) {
launch(args);
}
}
```

GolovneMenu.fxml

```
package sample;
import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
```

```
import javafx.scene.Scene; i
mport javafx.scene.control.Button;
import javafx.stage.Stage;
public class Controller {
@FXML
private ResourceBundle resources;
@FXML private URL location; @FXML
private Button Button;
@FXML void initialize() {
assert Button != null : "fx:id=\"Button\" was not injected: check your FXML file 'GoloveMenu.fxml'.";60
Button.setOnAction(event-> {
Button.getScene().getWindow().hide();
FXMLLoader
loader = new FXMLLoader(); loader.setLocation(getClass().getResource("1_vopros.fxml"));
try {
loader.load();
} catch (IOException
e) { e.printStackTrace();
}
Parent root = loader.getRoot();
Stage Practic = new Stage();
Practic.setTitle("Головне меню");
Practic.setScene(new Scene(root));
Practic.show();
});
}
}
```

Controller.java

```
package sample;
import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.geometry.Orientation;
import javafx.geometry.Pos;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.RadioButton;
import javafx.scene.control.ToggleGroup;
import javafx.scene.layout.FlowPane;
import javafx.scene.layout.StackPane;
import javafx.stage.Modality;
import javafx.stage.Stage;
import javafx.fxml.FXMLLoader;
public class Controller1 {
@FXML
 private ResourceBundle resources;
@FXML
 private URL location;
@FXML
private Button Button;
@FXML
private RadioButton RadioButton1;
@FXML
```

```
        private RadioButton RadioButton2;
        @FXML
        private RadioButton RadioButton3;
        @FXML
        void initialize() {
        assert Button != null : "fx:id=\"Button\" was not injected: check your FXML file '1_vopros.fxml'.";
        assert RadioButton1 != null : "fx:id=\"RadioButton1\" was not injected: check your FXML file '1_vopros.fxml'.";
        assert RadioButton2 != null : "fx:id=\"RadioButton2\" was not injected: check your FXML file '1_vopros.fxml'.";
        assert RadioButton3 != null : "fx:id=\"RadioButton3\" was not injected: check your FXML file '1_vopros.fxml'."; ToggleGroup group = new
        ToggleGroup();
        RadioButton1.setToggleGroup(group);
        RadioButton2.setToggleGroup(group);
        RadioButton3.setToggleGroup(group);
        Button.setOnAction(event ->{
        if (RadioButton1.isSelected()){
        {
        RadioButton1.getScene().getWindow().hide();
        FXMLLoader loader = new FXMLLoader();
loader.setLocation(getClass().getResource("2.fxml"));
        try {
        loader.load();
        } catch (IOException e) {
        e.printStackTrace(); }
        Parent root = loader.getRoot();
        Stage Practic = new Stage();
        Practic.setTitle;
```

```
Practic.setScene(new Scene(root));
Practic.show();
}
}
else if (RadioButton2.isSelected()){
Label lb = new Label("Не вірно!\n ");
Button bt = new Button ("Спобувати ще!");
bt.setOnAction(new EventHandler() {
@Override
public void handle(ActionEvent event) {
bt.getScene().getWindow().hide();
}});
FlowPane root = new FlowPane(Orientation.VERTICAL, lb, bt);
root.setAlignment(Pos.CENTER);
Scene secondScene = new Scene(root, 500, 500 );
// New window (Stage)
Stage newWindow = new Stage();
newWindow.setTitle("Second Stage"); newWindow.setScene(secondScene);
//
Specifies the modality for new window.
newWindow.initModality(Modality.APPLICATION_MODAL);
newWindow.show(); }
else if (RadioButton3.isSelected()){
Label lb = new Label("Не вірно\n "); Button bt = new Button ("Спобувати
ще!");
bt.setOnAction(new EventHandler() {
@Override public void handle(ActionEvent event) {
bt.getScene().getWindow().hide();
 }});
FlowPane root = new FlowPane(Orientation.VERTICAL, lb, bt);
```

```
root.setAlignment(Pos.CENTER);
Scene secondScene = new Scene(root, 500, 500);
// New window (Stage)
Stage newWindow = new Stage();
newWindow.setTitle("Second Stage");
newWindow.setScene(secondScene);
// Specifies the modality for new window.
newWindow.initModality(Modality.APPLICATION_MODAL);
newWindow.show();
}
else {
Label lb = new Label("Оберіть відповідь!\n \n ");
Button bt = new Button ("Повренутись назад!");
bt.setOnAction(new EventHandler() {
@Override
public void handle(ActionEvent event)
{ bt.getScene().getWindow().hide();
}});
FlowPane root = new
FlowPane(Orientation.VERTICAL, lb, bt);
root.setAlignment(Pos.CENTER);
Scene secondScene = new Scene(root, 500, 500);
// New window (Stage)
Stage newWindow = new Stage();
newWindow.setTitle("Second Stage");
newWindow.setScene(secondScene);
// Specifies the modality for new window.
newWindow.initModality(Modality.APPLICATION_MODAL);
newWindow.show();
}
```

```
});
}
```