ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ

Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____ Олена ОЛЬХОВСЬКА
(підпис)

«___»_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТРЕНАЖЕРУ З ТЕМИ «ОСНОВНІ МЕТОДОЛОГІЇ ПРОГРАМУВАННЯ» ДИСТАНЦІЙНОГО НАВЧАЛЬНОГО КУРСУ «ТЕОРІЯ ПРОГРАМУВАННЯ»

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

Виконавець роботи Банасюкевич Лілія Сергіївна
_____«____»_____ 202_ р.
(підпис)

Науковий керівник к. ф.-м. н., доцент, Черненко Оксана Олексіївна
_____«____»_____ 202_ р.
(підпис)

Рецензент

ПОЛТАВА 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних позначень, скорочень, символів
Visual Basic	Засіб розробки програмного забезпечення, створений і підтримуваний корпорацією Microsoft, який складається з мови програмування і середовища розробки.
Visual Studio	Інтегроване середовище розробки програмного забезпечення.
Adobe Photoshop CS6	Графічний редактор розроблений і поширюваний фірмою Adobe Systems.
Windows Forms	Інтерфейс програмування додатків (API), відповідальний за графічний інтерфейс користувача і є частиною Microsoft .NET Framework.
Click	Для того щоб всі кнопки працювали та натискались в тренажері, для кожної кнопки створені події «Click».
TextBox	Текстове поле.
if..esle..then	Конструкція, яка дозволяє виконувати деякі дії, не тільки якщо вихідна умова вірна, але і якщо вона неправильна.

ВСТУП

На сьогодні про розвиток вищої освіти в Україні точаться у двох напрямах: роботи університетів в умовах війни і повоєнної відбудови. Повномасштабна війна стала величезним викликом для освітньої галузі, з'явились нові фактори впливу, до яких освіту ніхто не готував. Саме використання нових сучасних комунікативних технологій може вплинути на якість освіти у воєнний та післявоєнний період. Українці в Європі вже зауважили, наскільки сучасною виявилась наша банківська система. «ДіЯ» також змусила багатьох забути прочерги в соціальних установах, про які тепер доводиться згадувати українцям у Європі. Тож диджиталізація освітніх процесів може стати шансом для мільйонів українських студентів продовжити отримувати якісні знання в українських закладах вищої освіти, а значить продовжувати бути ментально пов'язаними зі своєю батьківщиною. Це дозволить зменшити відтік талановитих молодих людей за кордон. Протягом вже останніх 20 років, відбувається процес переходу від традиційного навчання до навчання на базі комп'ютерних технологій. [2]

Мета бакалаврської роботи – програмна реалізація програмного забезпечення тренажеру з теми «Основні методології програмування» дистанційного курсу «Теорія програмування».

Об'єкт розробки – алгоритм, блок-схема, програмна реалізація програмного забезпечення тренажеру.

Предмет розробки – тренажер з теми «Основні методології програмування» дистанційного навчального курсу «Теорія програмування».

Бакалаврська робота складається з чотирьох розділів. У першому розділі описано постановку задачі для реалізації навчального тренажеру. У другому розділі описано про технології та методології програмування. У

третьому розділі розглянуто теоретичний матеріал з теми тренажеру, детально описаний алгоритм роботи та блок-схема роботи тренажеру. У четвертому розділі описано процес програмної реалізації тренажеру та інструкція користувача.

Обсяг пояснювальної записки: 59 стор., в т.ч. основна частина - 55 стор., джерела - 20 назв.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ

Головним завданням бакалаврської роботи – є розробка програмного забезпечення тренажеру з теми «Основні методології програмування» дистанційного курсу «Теорія програмування».

Завдання бакалаврської роботи:

- Розглянути матеріал про «Основні методології програмування»;
- Створити алгоритм тренажеру;
- Створити блок-схему тренажеру;
- Розписати процес створення тренажеру;
- Розглянути результат програмної реалізації елементів тренажеру.

Під час розробки програмного забезпечення тренажеру, потрібно реалізувати такі вікна:

- При програмній реалізації елементів тренажеру, треба врахувати наступні форми описані нижче:
- Перша титульна форма тренажеру, на цій формі буде відображена інформація про тему тренажеру, кнопка для початку тренінгу, виведена інформація про студента та наукового керівника курсового проекту з фаху;
- Наступні форми тренажеру - це питання тренажеру, питання у вигляді тестів, на які потрібно студентові дати правильну або неправильну відповідь;
- Останнє вікно тренажеру користувачеві відкривається в кінці тренінгу, на останній формі написано привітання про успішне проходження: «Вітаємо! Ви успішно пройшли тренажер з теми «Основні методології програмування» дистанційного навчального курсу «Теорія програмування»

РОЗДІЛ 2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Технології та методології програмування

Методологія програмування є фундаментом, на якому будуються конкретні *технології програмування* – сукупність виробничих процесів, що приводить до створення необхідного програмного забезпечення, а також опис цієї сукупності процесів. В технології програмування акцент робиться саме процесах розробки програм (технологічних процесах) у порядку їх проходження. Слід зазначити, що для однієї методології може існувати декілька технологій програмування.

Як і будь-яка інша технологія, технологія програмування являє собою набір технологічних інструкцій, що включають:

- вказівку послідовності виконання технологічних операцій;
- перерахування умов, при яких виконується та чи інша операція;
- описи самих операцій, де для кожної операції визначені вихідні дані, результати, а також інструкції, нормативи, стандарти, критерії та методи оцінки і т. п.

Структурний підхід до програмування являє собою сукупність технологічних прийомів, що охоплюють виконання всіх етапів розробки програмного забезпечення. Так процес розробки імперативних програм включає наступні етапи:

1) постановка задачі;
2) її формалізація;
3) вибір методу вирішення задачі;
4) розробка алгоритму програми;
5) розробка тексту програми;
6) тестування і налагодження програми.

Постановка задачі. Полягає у формулюванні опису умови задачі,

визначенні її вхідних і вихідних даних, а також форми видачі результатів. Для цього треба дати відповіді на питання:

- Що дано?
- Які дані допустимі ?
- Які результати і у якому вигляді повинні бути отримані ?
- За яких умов можливе одержання необхідних результатів, а за яких – ні ?
- Які результати будуть вважатися правильними ?

Так як задача формулюється в термінах предметної області, її необхідно перевести на мову понять, більш близьких до програмування. Задача повинна бути сформульована чітко, щоб її тлумачення було однозначним.

Постановка задачі завершується створенням технічного завдання (ТЗ), а потім зовнішньої специфікації програми, що включає в себе:

- опис вихідних даних і результатів (типи, формати, точність, спосіб передачі, обмеження);
- опис задачі, що реалізується програмою;
- спосіб звернення до програми;
- опис можливих аварійних ситуацій і помилок користувача.

Таким чином, програма розглядається як чорний ящик, для якого визначені функція і вхідні і вихідні дані.

Формалізація задачі. На цьому етапі розгорнутий опис задачі формалізується - заміняється її *математичною моделлю* - математичними залежностями (наприклад, рівнянням, системою рівнянь тощо), які зв’язують вхідні дані з вихідними результатами, або *інформаційною моделлю*, в якій встановлюється залежність між даними та способами їх отримання (наприклад, бази даних - БД).

При побудові моделі виділяються характеристики задачі, істотні з точки зору розгляду, тобто виконується її абстрагування. Ці

характеристики повинні представлятися в моделі з необхідною повнотою і точністю.

Вибір методу вирішення задачі. Виконується аналіз усіх існуючих методів вирішення поставленої задачі, зокрема, для математичної моделі – аналіз та аргументований вибір відповідного математичного / числового методу. За наявності декількох методів найкращий вибирається виходячи з критеріїв складності, ефективності, точності і т.д. залежно від конкретних задач, що стоять перед програмістом.

На цьому етапі також визначається середовище, в якому буде виконуватися програма: вимоги до апаратури, яка використовується операційна система та інше програмне забезпечення.

Велике значення при розробці програмного забезпечення має правильний вибір мови програмування. При цьому слід враховувати наступні чинники:

- функціональне призначення системи;
- склад технічних засобів системи;
- необхідну швидкість роботи програми;
- обмеження на об'єм використовуваної пам'яті;
- можливість розбиття програми на модулі;
- необхідність перенесення на інших типів ЕОМ;
- наявність ефективних трансляторів;
- необхідність використання вже розроблених пакетів програм;
- використовувані операційні системи;
- необхідність роботи із зовнішніми пристроями;
- типи даних, з якими працює програма та ін.

Головний критерій вибору мови програмування — скорочення терміну розробки та зниження трудомісткості на етапі супроводу. Найкраще використовувати одну мову для написання всіх модулів системи. Вибір мови значною мірою залежить від класу розв’язуваних задач та наявності

достатніх бібліотек стандартних програм для обробки інформації відповідного типу та організації. Оскільки зараз усі мови програмування мають практично однакові можливості обробки інформації, то при виборі мови беруться до уваги професійні можливості розробників та наявного ПЗ.

Розробка алгоритму програми. Неформально *алгоритмом* називається опис у зрозумілій для виконавця формі послідовності операцій над вхідними даними, виконання яких забезпечує досягнення поставленої мети. Цей опис повинен бути абсолютно повним і враховувати всі можливі ситуації, які можуть зустрітися по ходу вирішення задачі.

Алгоритм може представляти собою деяку послідовність обчислень, а може - послідовність дій не математичного характеру. Але в кожному випадку повинні бути визначені початкові умови і те, що потрібно отримати.

Алгоритм розв'язання задачі розбивається на декілька більш простих частин (підзадач). Їх виділяють таким чином, щоб програмування підзадач було незалежним. Складається блок-схема програми, де виділяють головну і підлеглі підзадачі і зв'язки між ними. Тут же встановлюють, які початкові дані отримує кожна підзадача і які результати вона видає.

Розробка тексту програми. Для опису алгоритмів на зрозумілій ПК мові, використовують мови програмування.

Етап розробки тексту програми саме і полягає у кодуванні алгоритму на обраній мові програмування. На цьому етапі основну увагу приділяють питанням синтаксичної правильності та забезпеченню документованості програмного тексту, що дозволяє будь-якому програмісту легко читати і розуміти програмний текст у процесі розробки та супроводу.

Тестування і налагодження програми. Частину синтаксичних помилок, допущених при написанні програми, виявляє транслятор. Правильність же виконання обчислень і передачі керування (логічна

перевірка) виявляються програмістом в процесі *налагодження* програми шляхом порівняння очікуваних і отриманих результатів. ***Налагодження*** програми – це процес виявлення і усунення семантичних помилок при її написанні.

Даний етап передбачає багаторазове виконання програми з різними варіантами даних, які вона може обробляти. Дані спеціально добираються таким чином, щоб можна було виявити якнайбільше помилок, якщо такі існують. Така цілеспрямована перевірка працездатності програми називається *тестуванням*. Тобто, ***тестування*** - перевірка програми при різних варіантах вхідних даних.

Налагодження і тестування програми - дуже трудомісткий і кропіткий процес. Він займає в загальній роботі майже половину часу.

З розвитком ООП, а також засобів графічного інтерфейсу на ПК, виникла така технологія програмування, як *візуальне програмування*. Дана технологія дозволяє програмісту легко і швидко будувати наочний графічний інтерфейс для своїх програм на основі стандартного набору шаблонів об’єктів, що можуть графічно відображатися на екрані.

Не слід також плутати технологію програмування з методологією програмування. Хоча в обох випадках вивчаються методи, але в технології програмування методи розглядаються "зверху" - з погляду організації технологічних процесів, а в методології програмування методи розглядаються "знизу" - з погляду основ їхньої побудови. [3]

РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Методології програмування

Кожна парадигма програмування є підходом до вирішення задач, на базі якого виникає певна *методологія програмування*. З того часу, як програмування стало достатньо масовою професійною діяльністю, розробки загальноприйнятої методології програмування, що підвищує продуктивність роботи програмістів, і, що саме головне, якість програмних продуктів, стала достатньо актуальною задачею.

Методологія програмування - це об'єднана єдиним філософським підходом сукупність методів, що застосовуються в процесі створення програм.

З кожною методологією можна пов'язати деякі характерні для неї атрибути, а саме:

- парадигму програмування, що визначає основне джерело ефективності методології;
- узгоджену, зв'язну множину методів, через які реалізується дана методологія;
- концепції (поняття, ідеї), що підтримують методи і дозволяють більш точно їх визначити.

Метод – це сукупність теоретичних принципів і практичних прийомів, які потрібно застосувати для виконання певної задачі.

Так основними методами, через які реалізується імперативна парадигма програмування, є:

- *метод зміни станів*, що полягає в послідовному зміні станів пам’яті (змінних) і підтримується концепцією алгоритму;
- *метод управління потоком виконання*, який полягає в покроковому

контролі управління і підтримується концепцією потоку виконання.

Самий ранній стиль розробки імперативних програм характеризувався *неструктурованістю* - код програми представлявся єдиним безперервним блоком; переходи до потрібних секцій програми виконувалися за допомогою операторів переходу (завичай, goto). Подібні проекти отримали назву BS-програм – абревіатура від "bowl of spaghetti" – блюдо спагеті, бо саме так виглядала готова програма при спробі зобразити всі переходи між її операторами.

Хорошими програмістами на той час вважалися ті, хто створював достатньо хитромудрі програми, які займали мінімальний об'єм пам'яті та швидко виконувалися. Це було цілком природно, враховуючи тодішні можливості обчислювальної техніки. Однак, такі програми було важко (якщо взагалі можливо) зрозуміти іншим фахівцям. Інколи навіть автори таких програм з плином часу з трудом розуміли власне творіння. Тому при розробці таких програм виникали проблеми забезпечення якості: утворення «спагетті-коду», складність його перевірки і тестування.

Із збільшенням складності задач розмір програм постійно збільшувався. Типовою стала ситуація, коли в різних місцях програми виникала необхідність у виконанні однієї і тієї ж послідовності дій. Такі дії могли бути досить складними і представлятися великими фрагментами програми. Для забезпечення компактності і підвищення наочності програм було запропоновано фрагменти програми, що повторюються, оформляти у вигляді спеціальних синтаксичних конструкцій – *підпрограм* (процедур або функцій), які за необхідності могли бути викликані для виконання з будь-якого місця основної програми. Методологія програмування, заснована на концепції виклику підпрограми, що є логічно завершеним фрагментом програми, отримала назву *процедурного програмува́ння*.

Основними ідеями, на яких грунтується процедурне програмування, є:

- використання механізмів передачі параметрів і повертання результатів;
- розподіл областей дії змінних програми (локальні, глобальні),
- використання різних моделей пам'яті для зберігання змінних;
- блокова організація програм.

Основними методами, через які реалізується процедурне програмування, є:

- *метод процедурної декомпозиції*, який полягає у оформленні кожного логічно завершеного фрагменти програми у вигляді підпрограми, і грунтується на концепції виклику підпрограми;
- *метод блочності*, що розглядає програму як деяку ієрархічну структуру, яка складається з головної програми і множини вкладених блоків (підпрограм), і грунтується на концепції блокової організації програм;
- *метод локалізаціїї даних*, який полягає у видимості (доступності) даних, що обробляються у підпрограмі, лише в межах підпрограми, і грунтується на концепції розподілу областей дії змінних програми;
- *метод специфікації інтерфейсів*, який полягає у формалізованому описі входів, виходів і підпрограм;
- *механізм раннього зв'язування*, який полягає у статичному зв'язуванні підпрограм (на етапі компіляції).

Моделлю обробки даних, характерною для процедурного програмування, є послідовне виконання підпрограм. Коротко це можна представити наступною нотацією:

програма = послідовність процедур, кожна з яких є послідовністю елементарних дій і викликів процедур.

Використання процедурного підходу при розробці програм дозволяє зменшити об'єм програмного коду, виключивши його дублювання, зробити

програми прозорішими, прискорити процес їх написання і тестування. Використання підпрограм дозволило розбивати великі задачі на підзадачі і таким чином спростило написання великих програм. Завдяки даному підходу програми набули більш впорядкованого характеру.

Наступним кроком розвитку програмування стало підвищення структурованості програм – *структурне програмування*. Дана методологія - це сукупність методів проектування та написання програм за жорсткими правилами, дотримання яких підвищує продуктивність праці програмістів, поліпшує читабельність і полегшує процес тестування програм.

Витоки цієї методології сягають 70-х років XX століття, коли швидкими темпами почала зростати складність програм, а отже, постала потреба у методах подолання такої складності. Вона ґрунтується на теоремі Бома-Якопіні[1] (1966 р.), які довели, що будь-який виконуваний алгоритм може бути перетворений до структурованого вигляду, тобто такого виду, коли хід його виконання визначається тільки за допомогою трьох структур управління: послідовностей, розгалужень і циклів.

Методологія структурного програмування реалізується через методи:

- структурного кодування (структурування програм);
- низхідного проектування (спадного проектування, проектування «зверху-вниз»);
- блочного програмування;
- паралельного документування.

Метод структурного кодування передбачає, що будь-яка програма представляє собою структуру, побудовану із трьох типів базових конструкцій:

- *послідовне виконання* – одноразове виконання операцій в тому порядку, в якому вони записані в тексті програми;

[1] Коррадо Бом і Джузеппе Якопіні - італійські математики

- *розгалуження* – одноразове виконання однієї із двох або більшої кількості операцій, в залежності від виконання деякої заданої умови;
- *цикл* – багаторазове виконання однієї і тієї ж операції до тих пір, доки виконується деяка задана умова (умова продовження циклу).

В програмі базові конструкції можуть бути вкладені одна в одну довільним чином, але ніяких інших засобів керування послідовністю виконання операцій не передбачається. Зокрема, неприпустимим є використання оператора безумовного переходу (GOTO).

Метою структурування є перетворення неструктурованої програми на еквівалентну їй структуровану, тобто таку, що складається з обмеженого набору керуючих алгоритмічних структур. Методи структурування ґрунтуються на поняттях функціонального вузла, а також на поняттях простої, елементарної і складеної програми.

Елементарна програма — це проста програма, що не містить блоків, які складаються більше ніж з одного вузла. Слід зазначити, що існує обмежена кількість елементарних програм, з яких основними є три програми, які зображені на рис. 4.

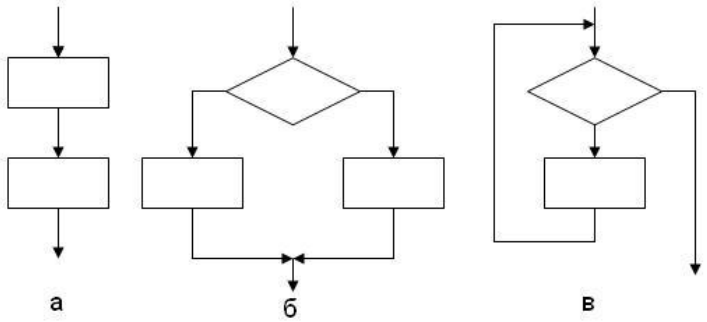


Рис. 4. Основні елементарні програми: послідовність (а), вибір (б), цикл із передумовою (в)

Якщо функціональний вузол елементарної програми замінити елементарною програмою, утвориться складена програма. Якщо вузол складеної програми замінити елементарною програмою, знов утвориться складена програма. В такий спосіб із будь-якої множини елементарних

програм утворюється певний клас складених програм. Складена програма, побудована з певної множини елементарних програм, називається структурованою.

Позначимо літерою S клас складених програм, базисна множина якого містить такі елементарні програми, як послідовність (рис. 6, а), вибір (рис. 6, б) і цикл із передумовою (рис. 6, в). Теорема про структурування стверджує, що будь-яку просту програму шляхом покрокового перетворення можна замінити функціонально еквівалентною структурованою програмою, що належить означеному вище класу S. Згадане покрокове перетворення здійснюється за переліченими нижче правилами:

1) *правило простоти* - створення програми слід починати із простої програми;
2) *правило пакетування* - кожну дію можна замінити двома послідовними діями (вихід одного функціонального вузла з'єднується із входом наступного);
3) *правило вкладання* - кожну дію можна замінити будь-якою структурою керування (будь-який блок може бути замінений на структуру керування вибору або повторення);
4) правила 2 і 3 можна застосовувати у будь-якій послідовності необмежену кількість разів.

Принцип застосування правил пакетування та вкладання до простої програми продемонстровано на рис. 5.

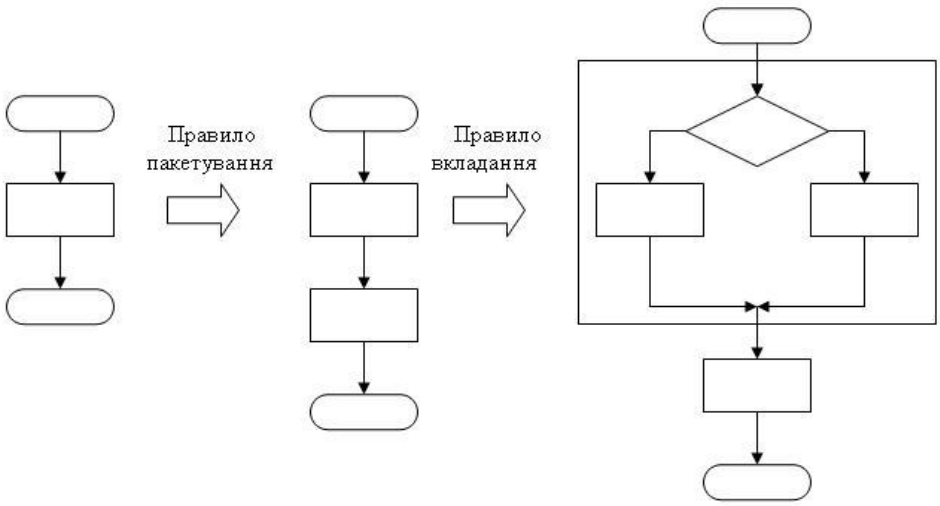


Рис. 5. Застосування правил пакетування та вкладання до простої програми

Для перетворення неструктурованих програм у структуровані використовуються такі методи:

- дублювання кодів,
- введення змінної стану,
- метод булевих ознак.

Метод дублювання кодів застосовується для перетворення неструктурованих програм, що не містять циклів. Для отримання структурованої програми дублюють ті блоки (модулі), у які можна увійти з декількох точок.

Розглянемо неструктуровану програму, блок-схема якої зображена на рис. 6. Блоки позначені ідентифікаторами модулів.

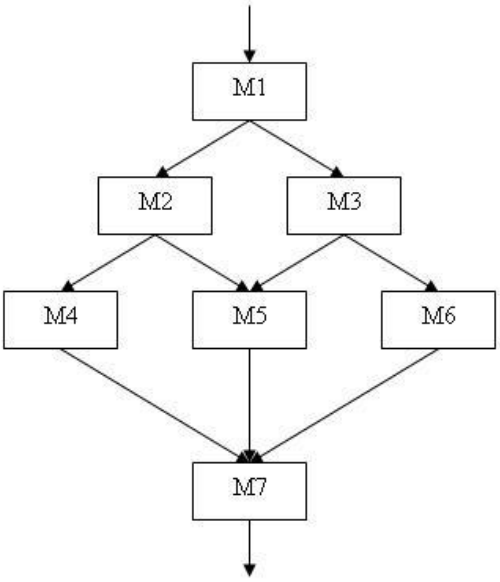


Рис. 6. Блок-схема неструктурованої програми

Із даної блок-схеми видно, що модуль М5 не може розпочати своєї роботи доти, доки не стануть відомими результати роботи модулів М2 і М3.

Перетворимо зображену на рис. 8 неструктуровану програму, застосовуючи правила пакетування та вкладання. Зобразимо початкову програму як просту, що складається з елементарних програм типу вибору. Користуючись правилом вкладання, розширимо блоки МХ і МУ, замінивши їх елементарними програмами типу вибору. У результаті заміни модуль М5 продублюється (рис. 7).

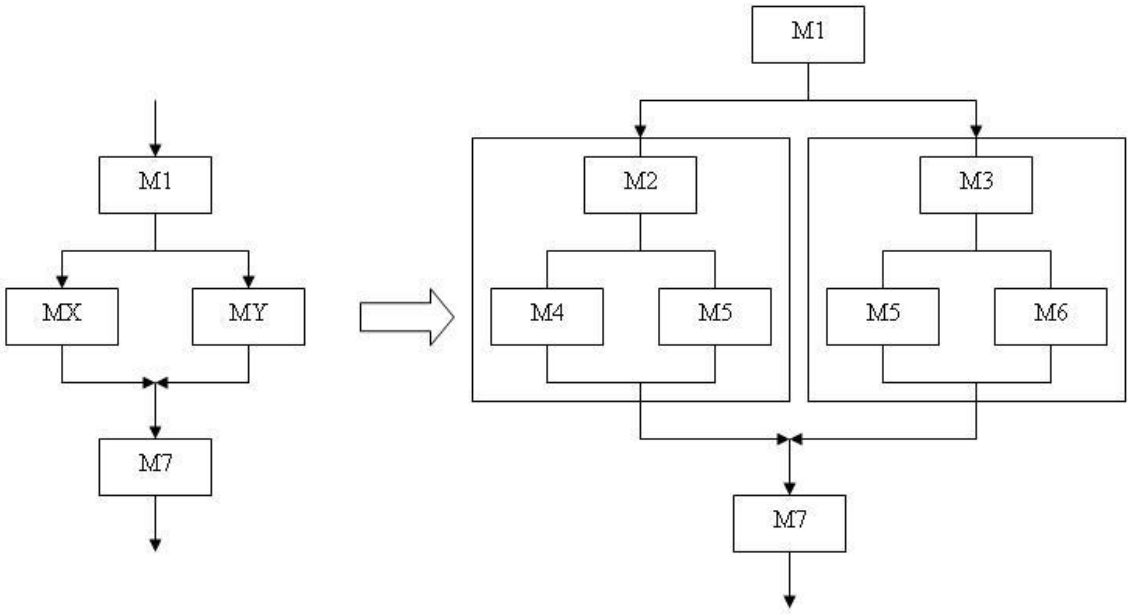


Рис. 7. Процес перетворення неструктурованої програми у структуровану

Розглянемо блок-схему неструктурованої програми, що містить цикли (рис. 8). Кожному блоку приписано довільний номер.

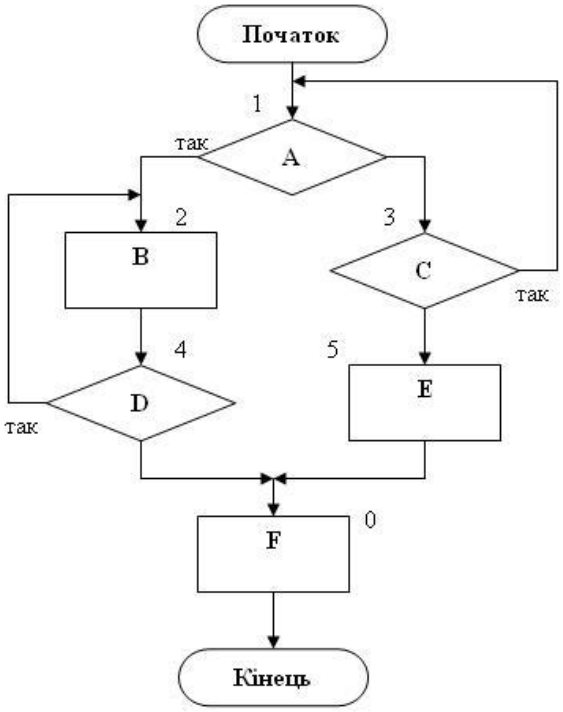


Рис. 8. Блок-схема неструктурованої програми, що містить цикли

У програму вводиться нова змінна і. Цій змінній присвоюється номер

блока, на який потрібно передати керування. Зокрема, якщо істинна умова A, то цій змінній присвоюється номер блока B, тобто 2, якщо істинна умова C, змінній і присвоюється 1 — номер блока A.

Логічні блоки неструктурованої блок-схеми замінюються елементарними програмами типу вибору, що послідовно перевіряють значення змінної стану. Гілки «так» програм типу вибору мають містити оператори присвоєння нових значень змінній стану і оператори перевірки певних умов початкової програми.

Блок-схему неструктурованої програми зображено на рис. 8, а блок-схему відповідної структурованої програми - на рис. 9.

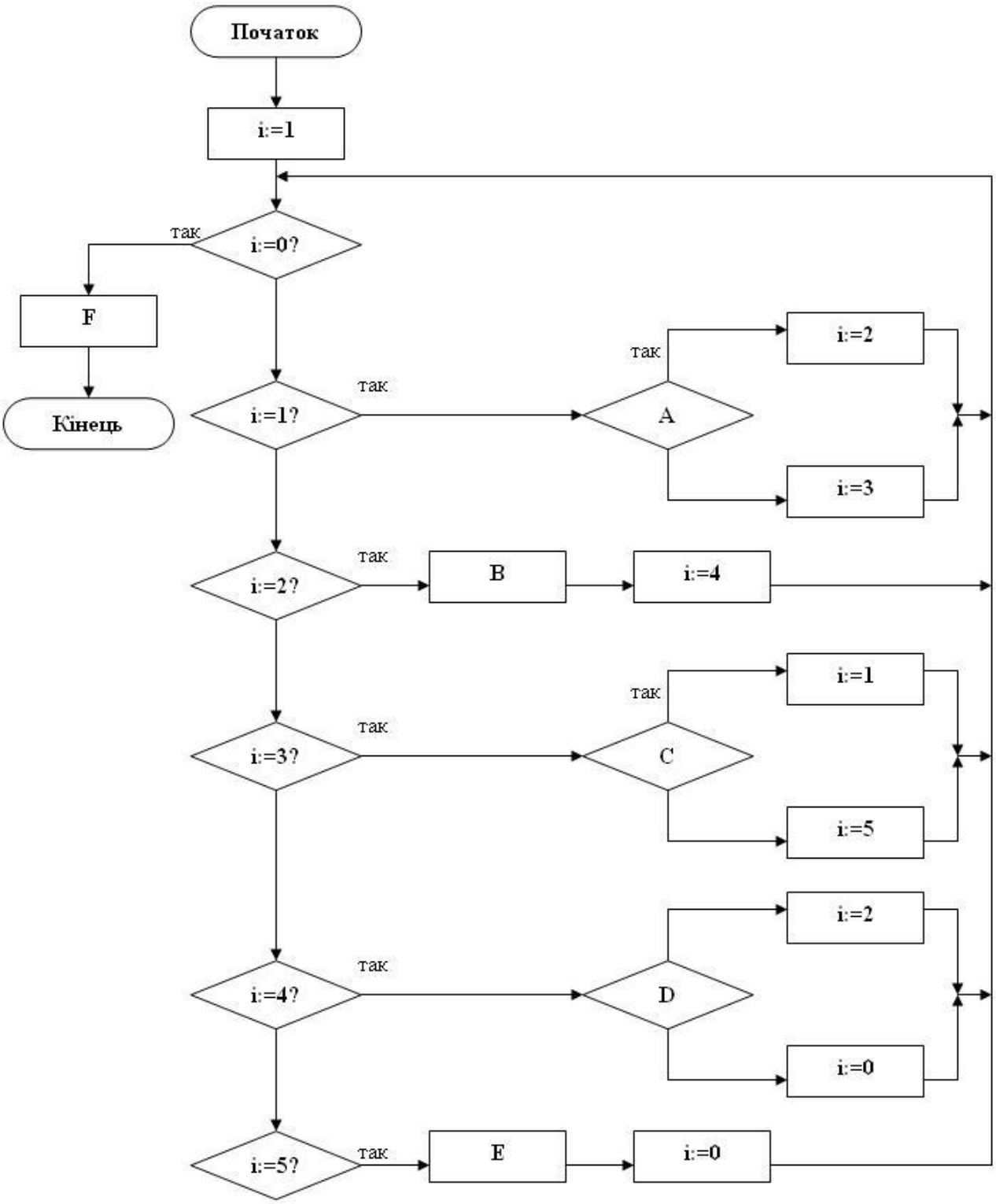


Рис. 9. Блок-схема структурованої циклічної програми

Метод булевої ознаки використовується для перетворення неструктурованих програм, що містять цикли. В програму вводиться додаткова змінна, яка набуває значення true чи false. Доки змінна ознаки зберігає це значення, виконання циклу триває. Значення змінної ознаки

всередині циклу модифікується лише за певних умов.

Блочне програмування — це організація програми у вигляді сукупності відносно незалежних складових частин - блоків, якими фактично є підпрограми. Як показує практика, великі задачі простіше вирішуються, якщо розглядати їх як сукупність менших задач.

Як правило, програмний блок розв’язує порівняно нескладну задачу, логічно незалежну від інших задач. Його властивості:

- відповідає лише одній задачі;
- має один вхід і один вихід;
- має порівняно невеликий розмір;
- доступний за своїм ідентифікатором;
- може викликати інший блок (не в усіх мовах програмування);
- підтримує незалежність функціонування (заміна підпрограми на аналогічну не впливає на всю програму).

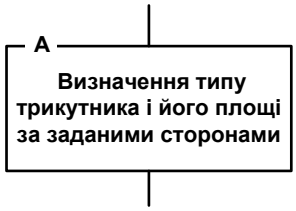
Тобто, кожна програма представляється лінійною послідовністю блоків. Кожен блок складається з одного або декількох інших блоків, кожен із яких має також рівно один вхід і рівно один вихід і сам може розглядатися як блок. Сенс блоків, що мають один вхід і один вихід, полягає в тому, що можна вийняти їх, змінити їх начинку і вставити назад без зміни інших сполучень в програмі.

Блоки повинні бути незалежними в межах інтерфейсу програми і структури даних. Практика показала, що чим вищий ступінь незалежності підпрограм, тим простіше розібратись в окремих програмних блоках і в програмі в цілому; тим менша ймовірність так званого *хвильового ефекту* - появи нових помилок при виправленні старих або при внесенні змін у програму. Тому не варто без крайньої необхідності використовувати в підпрограмах глобальні змінні. Всі зв'язки між підпрограмами мають підтримуватися через списки параметрів.

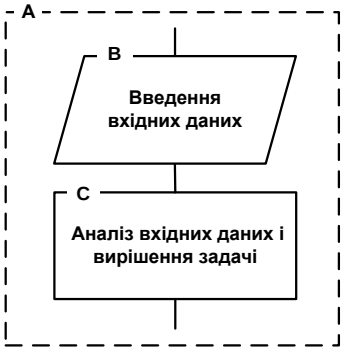
В основу *методу низхідного проектування* покладено структурну декомпозицію[2] задачі із застосуванням принципу покрокової (поетапної) деталізації. Спадне проектування передбачає, що розробка програм ведеться методом «зверху вниз», від загального до деталей. Спочатку задача розглядається як єдиний блок, що виражає загальне призначення програми. Далі ця задача поділяється на декілька дрібніших підзадач в тому порядку, в якому вони повинні виконуватися. Потім кожна з підзадач розбивається на свої підзадачі, що належать другому рівню деталізації і т.д. Процес покрокової деталізації підзадач здійснюється до тих пір, поки підзадачі чергового рівня не стануть досить простими для незалежного розв'язання (наприклад, кожній із них буде відповідати окрема команда мови програмування).

Приклад низхідного проектування алгоритму визначення типу трикутника і його площі за заданими сторонами.

Етап 1. Задача представляється у виді єдиного блоку (A), що містить її постановку:

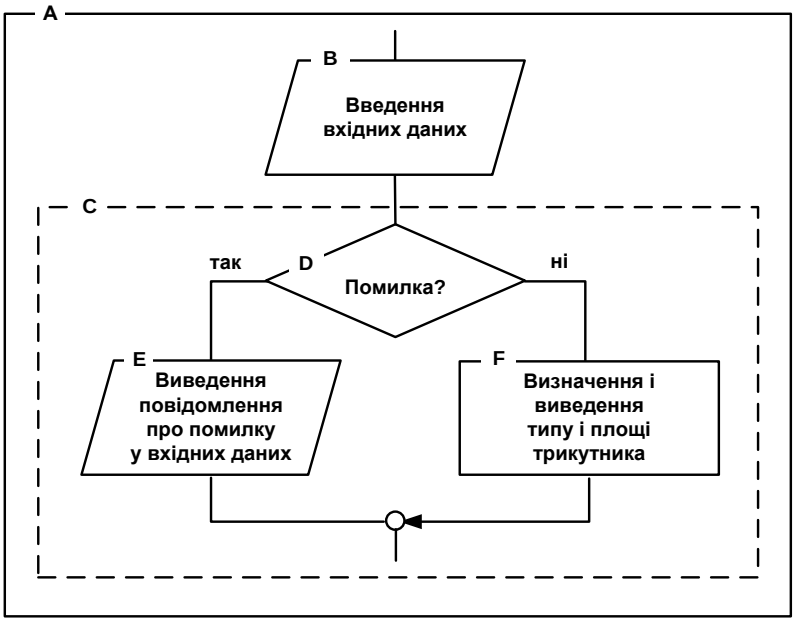


Етап 2. Декомпозиція задачі на підзадачі: введення вхідних даних (B) і власне її вирішення (C):

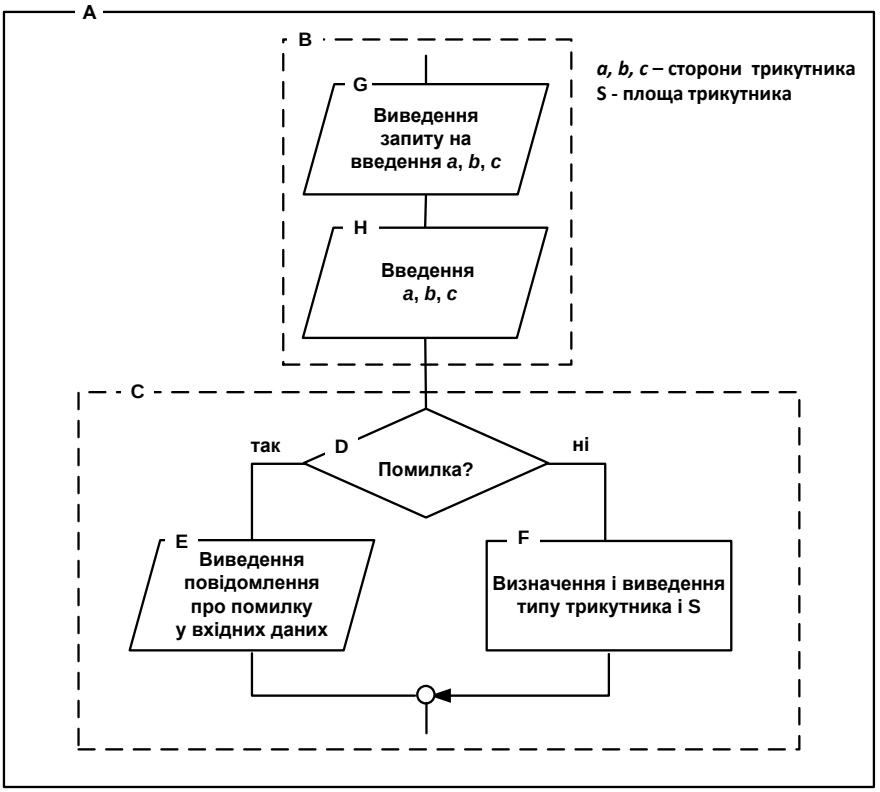


[2] *Декомпозиція* - розбиття на частини.

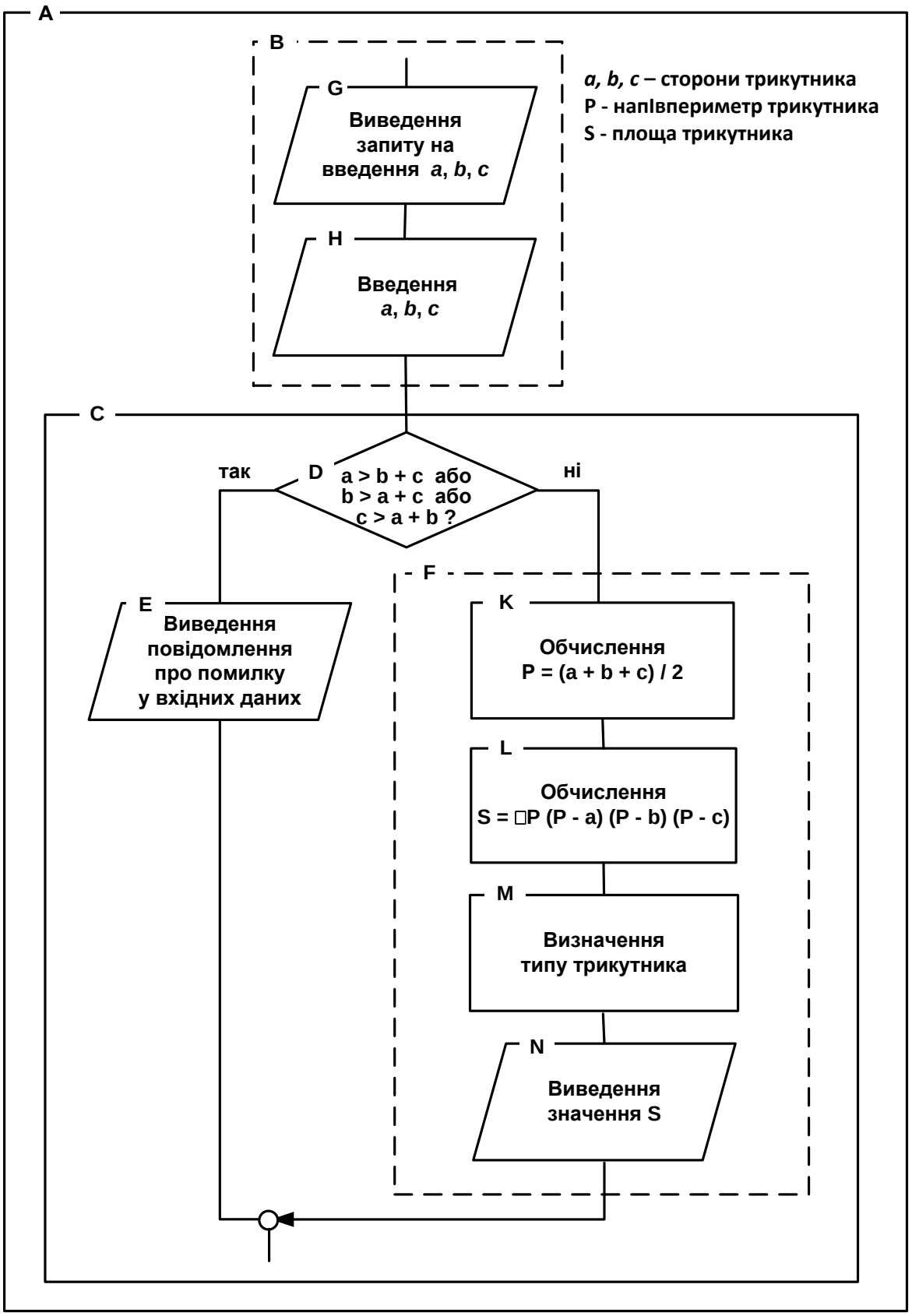
Етап 3. Подальша деталізація структури програми в частині аналізу введених даних (блок C):



Етап 4. Деталізація структури програми в частині введення даних (блок B).

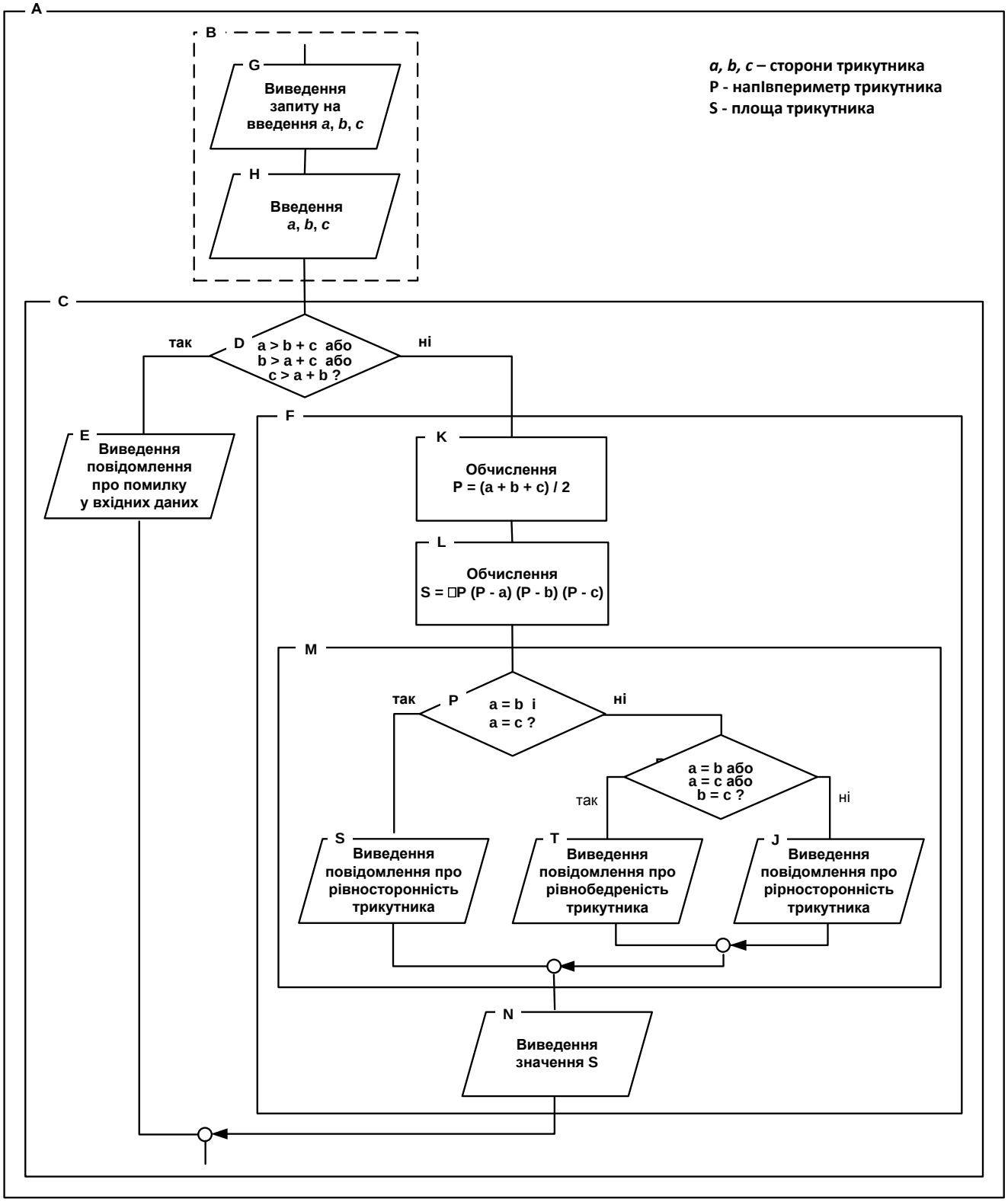


Етап 5. Деталізація структури програми в частині задання умов аналізу (блок D) та визначення площі трикутника (блок F).



Етап 6. Деталізація структури програми в частині визначення типу

трикутника (блок M):



Процедурне структурне програмува́ння передбачає використання принципу *процедурної декомпозиції* на всіх рівнях проектування

програмної системи. У відповідності із даним принципом кожен логічно цілісний програмний блок, отриманий на певному рівні деталізації, оформлюється як *підпрограма*. Наприклад, підпрограма обчислення визначника матриці, підпрограма знаходження суми елементів ряду тощо.

Розробка процедурної структурованої програми ведеться покроково, методом "зверху вниз" з використанням механізму так званих «заглушок» - підпрограм, які нічого не роблять. При цьому спочатку пишеться основна програма, у якій замість кожного логічного зв'язного фрагмента тексту вставляється виклик підпрограми, яка буде виконувати цей фрагмент. Замість справжніх, працюючих підпрограм, в програму вставляються "заглушки". Отримана програма перевіряється та налагоджується. Після того, як програміст переконається, що підпрограми викликаються в правильній послідовності, тобто загальна структура програми вірна, підпрограми-"заглушки" послідовно замінюються на реально працюючі, причому розробка кожної підпрограми ведеться тим же методом, що і основної програми. Розробка закінчується тоді, коли не залишиться жодної "заглушки", яка не була б видалена.

Таким чином, «заглушки» дозволяють перевірити логіку програми верхнього рівня до реалізації наступного. Тобто на кожному кроці розробки програми існує працездатний «каркас», який поступово обростає деталями. Така послідовність гарантує, що на кожному етапі розробки програміст одночасно має справу з доступною для огляду і зрозумілою йому множиною фрагментів і може бути впевнений, що загальна структура всіх більш високих рівнів програми вірна. При супроводженні та внесенні змін у програму з'ясовується, в які саме процедури потрібно внести зміни, і вони вносяться, не зачіпаючи безпосередньо не пов'язані з ними частини програми. Це дозволяє гарантувати, що при внесенні змін і виправленні помилок не вийде з ладу якась частина програми, що знаходиться в даний момент поза зоною уваги програміста.

Таким чином, при структурному програмуванні програма ієрархічно структурується і розробляється шляхом послідовного уточнення на кожному рівні ієрархії. В основу цього процесу, окрім принципу ієрархічності, покладено принципи абстрагування.

Абстрагування — це спрощений опис системи, в якому зосереджують увагу на певних властивостях і деталях, а на інші не зважають. Вдалою є та абстракція, що підкреслює суттєві деталі і відкидає несуттєві. Під час низхідного проектування програми на верхніх рівнях абстракції деталі реалізації приховуються, а на нижніх рівнях вони описуються конкретною мовою програмування.

Фактично такий підхід збільшив структурованість програм – велика програма стала сукупністю процедур-підпрограм. Одна підпрограма, головна, розпочинала роботу всієї програми.

Одним із прийомів формалізованого підходу до низхідного процедурного проектування є метод ієрархічних діаграм, що позначається абревіатурою HIPO (від англ. *Hierarchical Input Processing Output* — діаграма входу, обробки, виходу). Згідно з цим методом структура всієї програми подається у вигляді дерева, в якому підпрограми зображуються вузлами, а їхні виклики — ребрами.

Методи низхідного проектування та блочного програмування регламентують процес побудови архітектури програм на макрорівні. Методи структурування, навпаки, працюють на мікрорівні, регламентуючи процес створення програмного коду.

Сутність методу *паралельного документування* полягає у тому, що при розробці програм документація повинна створюватися одночасно із програмуванням, зокрема, у вигляді коментарів до програми. Документування програмного тексту дозволяє будь-якому програмісту легко читати і розуміти його у процесі розробки та подальшого супроводу.

Впровадження принципів структурного програмування зробило тексти

програм, навіть досить великих, більш читабельними. Серйозно полегшилось розуміння програм, з'явилася можливість розробки програм в нормальному промисловому режимі, коли програму може без особливих труднощів зрозуміти не тільки її автор, а й інші програмісти. Все це полегшило і прискорило процес розробки програмного забезпечення.

Розвитком ідей процедурного програмування, пов'язаним із зростанням об'єму і складності програмного забезпечення, стало так зване *модульне програмування*. Основна ідея модульного програмува́ння полягає у організації програм як сукупності незалежних блоків, які прийнято іменувати *модулями*.

Визначальним принципом модульного програмування є незалежність модулів. Це означає, що програма має бути поділена на модулі таким чином, щоб зв'язки між модулями були слабкими, а зв'язки всередині модулів — сильними. Модулі вважаються незалежними, якщо вони задовольняють таким вимогам:

- кожен модуль можна замінити іншим функціонально еквівалентним модулем, що має той самий інтерфейс;
- взаємозв'язки між модулями встановлені за ієрархічним принципом;
- усі структури даних інкапсульовані в модулі, тобто доступ до даних може здійснюватися лише через процедури та функції модуля;
- модуль має одну точку входу та одну точку виходу;
- модуль повертає керування тому програмному блоку, що його викликав.

При розбитті програмного забезпечення на модулі для кожного модуля визначається функціональність, що реалізовується ним, а також зв'язки з іншими модулями. При цьому функціональність модуля (**що** він повинен виконувати) відокремлюється від його реалізації (**як** це виконується). Зв'язки між модулями здійснюються через спеціальний інтерфейс, тоді як доступ до реалізації модуля (тіл підпрограм і деяких «внутрішніх»

змінних) заборонений. Тобто, знаючи, що робить модуль і як його викликати, можна без обмежень використовувати його, не маючи при цьому жодного уявлення, як саме він реалізований. Такий підхід розмежовує доступ до даних програми, зменшуючи кількість помилок, що виникають при роботі з ними.

Ще одна ідея модульного програмування полягає у групуванні підпрограм і даних, якими вони управляють, в окремо компільовані файли, які також називають модулями. Основна програма може підключати такі модулі і використовувати підпрограми, що містяться в них. Тим самим забезпечується можливість автономної реалізації і зберігання коду (в бібліотеках модулів), а також збирання програм з модулів.

Таким чином, основними ідеями, на яких грунтується модульне програмування, є:

- автономна реалізація модулів (в бібліотеках модулів),
- автономна компіляція,
- збирання програм з модулів.

Основними методами, через які реалізується модульне програмування, є:

- *метод незалежності модулів*, який полягає у розробці модулів, відносно ізольованих один від одного;
- *механізм пізнього зв'язування*, який полягає у динамічному виклику модулів (на етапі виконання).

Розбиття на модулі зменшує час перекомпіляції і полегшує процес налагодження програм, приховуючи несуттєві деталі за інтерфейсом модуля і дозволяючи налагоджувати програму по частинах. Інтерфейсом модуля є заголовки всіх підпрограм і описи доступних ззовні типів, змінних і констант. Використання модульного програмування істотно спрощує розробку програмного забезпечення, зокрема, шляхом розподілу процесу його розробки між групами розробників. Крім того, модулі надалі

без змін можна використовувати в інших розробках, що підвищує продуктивність роботи програмістів. У разі колективної розробки модульної програми робота розподіляється так: більш досвідчені програмісти відповідають за інтерфейс модулів, а менш досвідчені — за їх реалізацію. [3]

3.2. Алгоритм роботи тренажеру з теми «Основні методології програмування» дистанційного навчального курсу «Теорія програмування»

Під час запуску розробленого програмного забезпечення, на титульній формі студент зможе побачити інформацію інформацію про:

- розробника тренажеру;
- наукового керівника;
- тематика елементів тренажеру.

Якщо натиснути на кнопку, яка запускає тренінг, користувачу висвітиться перший крок алгоритму тренажеру, в якому висвітиться форма з першим питанням з лекційного матеріалу тематики тренажеру.

Перший крок. Користувачу відобразиться питання з теми, на нього можна дати відповідь натиснувши лише на один варіант відповіді: «Що таке методологія?»

1. Це принципи, сукупність ідей, методів і засобів, які визначають підхід до розробки програмного забезпечення; (вірна відповідь)
2. Сервер, на якому створені додатки, які використовують вашу БД, веб-сервіс;
3. Послуга з розміщення файлів користувача, за якої дані зберігаються на багатьох серверах, що розподілені в мережі.

Якщо користувач вибрав правильний варіант відповіді, то він може перейти на наступний крок. Якщо користувач вибрав неправильний варіант відповіді, то виводиться помилка: «Ваша відповідь невірна! Спробуйте ще раз!»

Другий крок. Користувачу відобразиться питання з теми, на нього можна дати відповідь натиснувши лише на один варіант відповіді: «Що таке розроблення програмного забезпечення (ПЗ)?»

1. Це послуга, що дозволяє створювати контент та зберігати його в Інтернеті;
2. Це складний процес, у якому беруть участь багато фахівців і який часто орієнтований на віддалену роботу.; (вірна відповідь)
3. Це послуга, що дозволяє приватним особам, розміщувати в Інтернеті рекламу.

Якщо користувач вибрав правильний варіант відповіді, то він може перейти на наступний крок. Якщо користувач вибрав неправильний варіант відповіді, то виводиться помилка: «Ваша відповідь невірна! Спробуйте ще раз!»

Третій крок. Користувачу відобразиться питання з теми, на нього можна дати відповідь натиснувши лише на один варіант відповіді: «Що таке системний аналіз розробки програмного забезпечення?»

1. Розміщення файлів користувача, за якої дані зберігаються на багатьох серверах, що розподілені в мережі;
2. Мова створення веб-сторінок;
3. Це дослідження вимог до ПЗ, його можливостей і представлення в загальному вигляді з подальшою деталізацією (вірна відповідь)

Якщо користувач вибрав правильний варіант відповіді, то він може перейти на наступний крок. Якщо користувач вибрав неправильний

варіант відповіді, то виводиться помилка: «Ваша відповідь невірна! Спробуйте ще раз!»

Четвертий крок. Користувачу відобразиться питання з теми, на нього можна дати відповідь натиснувши лише на один варіант відповіді: «Життєвий цикл розробки програмного забезпечення?»

1. Збір та аналіз вимог, дизайн, реалізація, тестування, підтримка; (вірна відповідь)
2. Збір та аналіз вимог, реалізація, дизайн, тестування;
3. Планування і контроль, аналіз та дизайн, імплементація, тестування, підтримка.

Якщо користувач вибрав правильний варіант відповіді, то він може перейти на наступний крок. Якщо користувач вибрав неправильний варіант відповіді, то виводиться помилка: «Ваша відповідь невірна! Спробуйте ще раз!»

П'ятий крок. Користувачу відобразиться питання з теми, на нього можна дати відповідь натиснувши лише на один варіант відповіді: «Яка методологія базується на кількох циклах повторення всіх етапів?»

1. Каскадна;
2. Ітераційна; (вірна відповідь)
3. Кластерна.

Якщо користувач вибрав правильний варіант відповіді, то він може перейти на наступний крок. Якщо користувач вибрав неправильний варіант відповіді, то виводиться помилка: «Ваша відповідь невірна! Спробуйте ще раз!»

Шостий крок. Користувачу відобразиться питання з теми, на нього можна дати відповідь натиснувши лише на один варіант відповіді: «Якою

методологією користуються у випадку, коли основні вимоги остаточно визначені, зрозумілі та зафіксовані?»

1. Кластерною;
2. Ітераційною;
3. Каскадною (вірна відповідь)

Якщо користувач вибрав правильний варіант відповіді, то він може перейти на наступний крок. Якщо користувач вибрав неправильний варіант відповіді, то виводиться помилка: «Ваша відповідь невірна! Спробуйте ще раз!»

Сьомий крок. Користувачу відобразиться питання з теми, на нього можна дати відповідь натиснувши лише на один варіант відповіді: «Тестування програмного забезпечення - це: »

1. Перевірка та порівняння поведінки програми з вимогами та очікуваним результатом; (вірна відповідь)
2. Перевірка програми на наявність критичних помилок в коді;
3. Пошук і виправлення всіх помилок в програмі.

Якщо користувач вибрав правильний варіант відповіді, то він може перейти на наступний крок. Якщо користувач вибрав неправильний варіант відповіді, то виводиться помилка: «Ваша відповідь невірна! Спробуйте ще раз!»

Восьмий крок. Користувачу відобразиться питання з теми, на нього можна дати відповідь натиснувши лише на один варіант відповіді: «Оберіть вірне твердження стосовно якості: »

1. Якісний дизайн, який подобається більшості користувачів продукту;

2. Сукупність властивостей продукції, які обумовлюють її придатність, задовольнити певні потреби відповідно до призначення; (вірна відповідь)
3. Програма є якісною, якщо її протестували.

Якщо користувач вибрав правильний варіант відповіді, то він може перейти на наступний крок. Якщо користувач вибрав неправильний варіант відповіді, то виводиться помилка: «Ваша відповідь невірна! Спробуйте ще раз!»

Дев'ятий крок. Користувачу відобразиться питання з теми, на нього можна дати відповідь натиснувши лише на один варіант відповіді: «Чому потрібно тестувати програмне забезпечення? »

1. Помилка в програмному забезпеченні може призвести до втрат грошей та часу;
2. Для того, щоб переконатись, що програма виконує ті функції, які вона має виконувати і вирішує ті задачі, які повинна вирішувати;
3. Все перелічене вище. (вірна відповідь)

Якщо користувач вибрав правильний варіант відповіді, то він може перейти на наступний крок. Якщо користувач вибрав неправильний варіант відповіді, то виводиться помилка: «Ваша відповідь невірна! Спробуйте ще раз!»

Десятий крок. Користувачу відобразиться питання з теми, на нього можна дати відповідь натиснувши лише на один варіант відповіді: «Яка методологія створення ПЗ описує послідовний процес виконання всіх етапів проекту?»

1. Каскадна; (вірна відповідь)
2. Інкрементальна;
3. Ітераційна.

Якщо користувач вибрав правильний варіант відповіді, то він може перейти на наступний крок. Якщо користувач вибрав неправильний варіант відповіді, то виводиться помилка: «Ваша відповідь невірна! Спробуйте ще раз!»

Одинадцятий крок. Користувачу відкриється питання з лекції, можна вписати з клавіатури відповідь на питання та натиснути кнопку «наступне питання»: «Яка із методологій створення ПЗ описана… основні вимоги остаточно визначені, зрозумілі й зафіксовані?»

Якщо користувач вписав правильний варіант відповіді, він зможе відкрити наступний крок. Якщо користувач вписав неправильний варіант відповіді, то йому виведеться помилка: «Неправильно! Підказка: К_ск_дна*»

Дванадцятий крок. Користувачу відкриється питання з лекції, можна вписати з клавіатури відповідь на питання та натиснути кнопку «наступне питання»: «Яка програма являє собою деякий вираз, виконання програми означає обчислення цього виразу?»

Якщо користувач вписав правильний варіант відповіді, він зможе відкрити наступний крок. Якщо користувач вписав неправильний варіант відповіді, то йому виведеться помилка: «Неправильно! Підказка: Ф_нкці_нальн_»

Тринадцятий крок. Користувачу відкриється питання з лекції, можна вибрати відповідь натиснувши лише на один варіант відповіді: «Зведення теоретичних основ і правил програмування, що не зв'язані з конкретною мовою, називається?»

Якщо користувач вписав правильний варіант відповіді, він зможе відкрити наступний крок. Якщо користувач вписав неправильний варіант

відповіді, то йому виведеться помилка: «Неправильно! Підказка: П_рад_гм_ю пр_грамув_ння»

Чотирнадцятий крок. Користувачу відкриється питання з лекції, можна вибрати відповідь натиснувши лише на один варіант відповіді: «Які системи контролю версій (СКВ) можуть надавати доступ до кількох віддалених репозиторіїв, що уможливлює одночасну співпрацю розробників?»

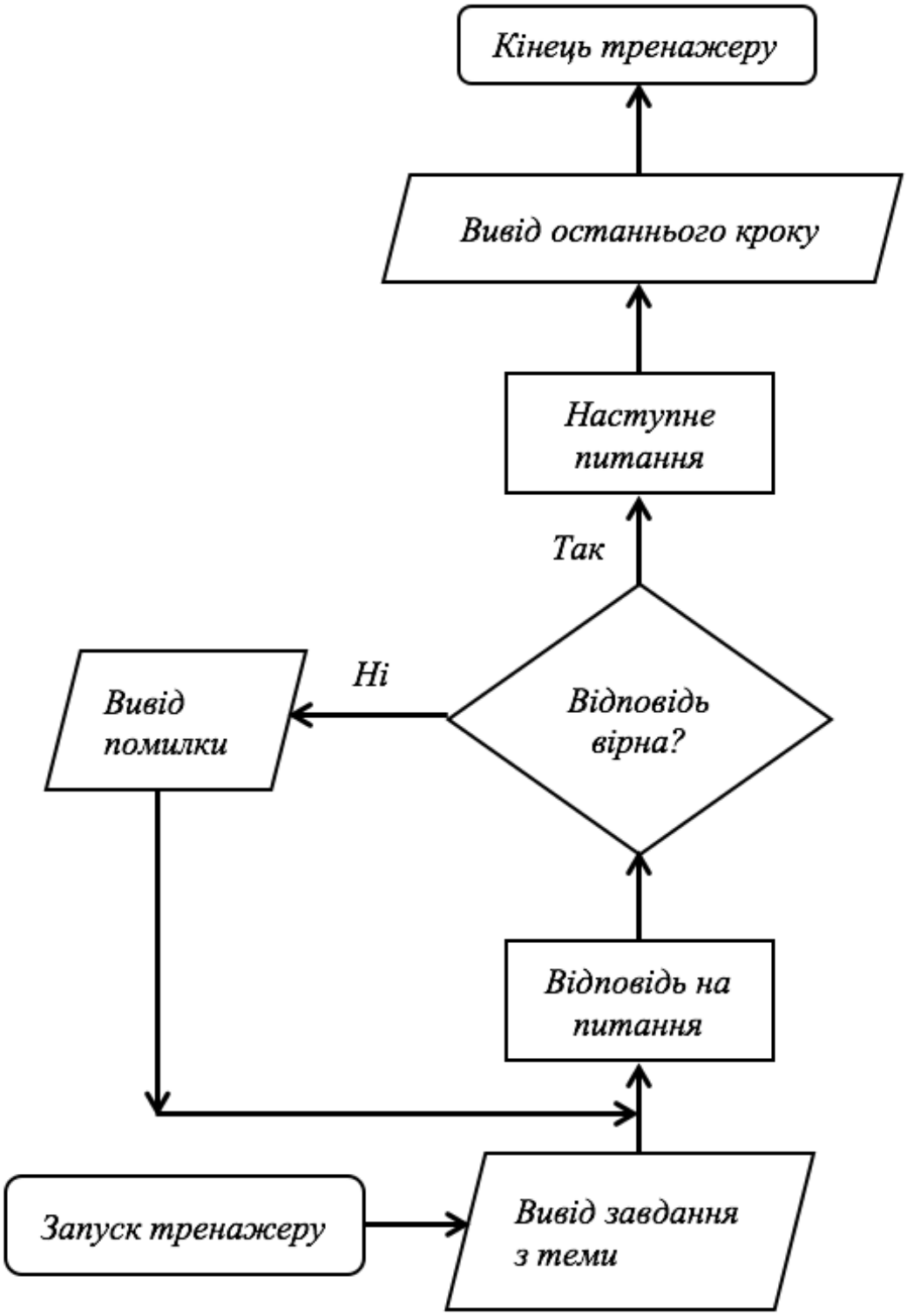
Якщо користувач вписав правильний варіант відповіді, він зможе відкрити наступний крок. Якщо користувач вписав неправильний варіант відповіді, то йому виведеться помилка: «Неправильно! Підказка: Розподілені»

П'ятнадцятий крок. Користувачу відкриється питання з лекції, можна вибрати відповідь натиснувши лише на один варіант відповіді: «Програми не містять циклів, а використовують рекурсію. Якій методології програмування належить дана властивість?»

Якщо користувач вписав правильний варіант відповіді, він зможе відкрити наступний крок. Якщо користувач вписав неправильний варіант відповіді, то йому виведеться помилка: «Неправильно! Підказка: Ф_нкц_ональн_й»

Після п'ятнадцятого кроку відкривається підсумкова сторінка тренажеру, на якій написано: «Вітаємо! Ви успішно пройшли тренажер з теми «Основні методології програмування» дистанційного навчального курсу «Теорія програмування».

3.3. Блок-схема тренажеру



РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА

4.1 Опис програмної реалізації

Програму тренажер було реалізовано на мові програмування Visual Basic, в середовищі розробки Visual Studio. Потім створюємо новий проект Windows Forms (рис. 4.1)

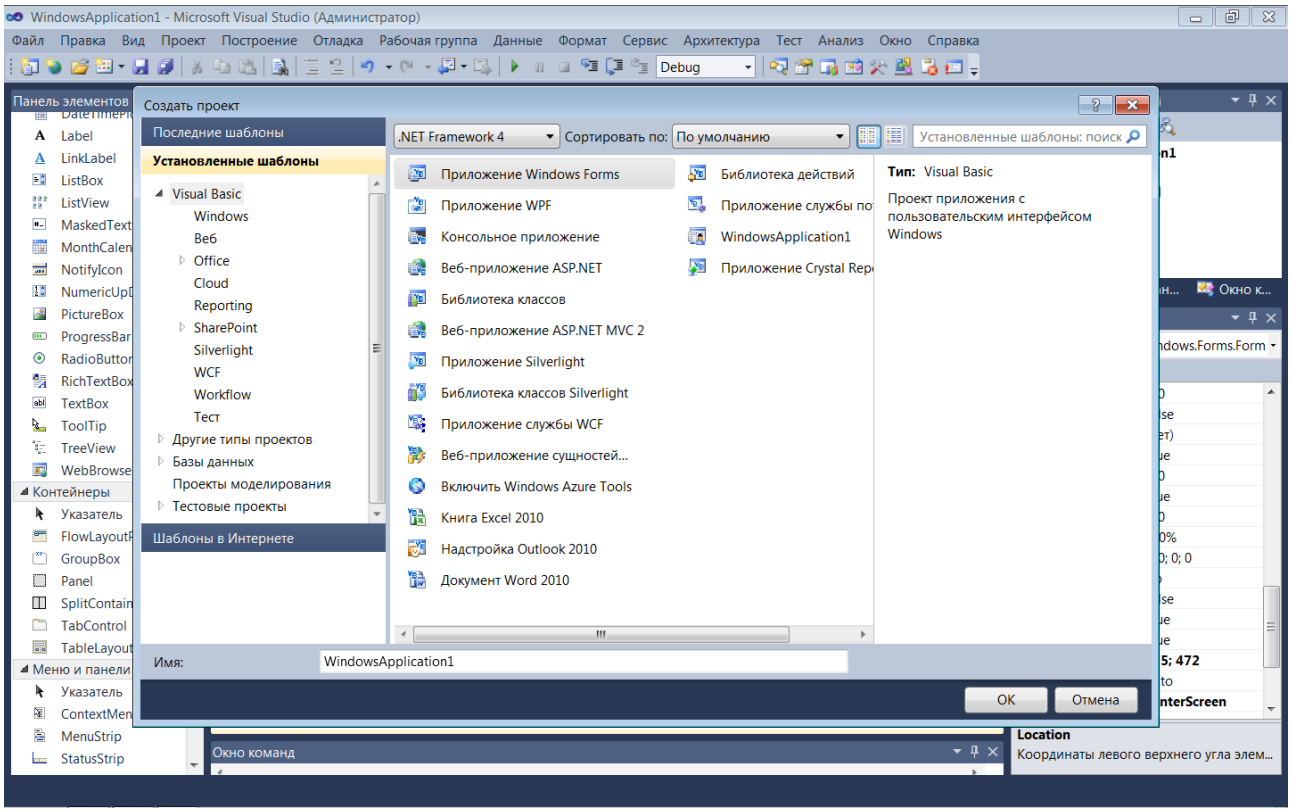


Рисунок 4.1 – процес створення нового проекту Windows Forms.

Після створення нового проекту в середовищі розробки, далі розробляється титульна стартова форма тренажеру (рис. 4.2)

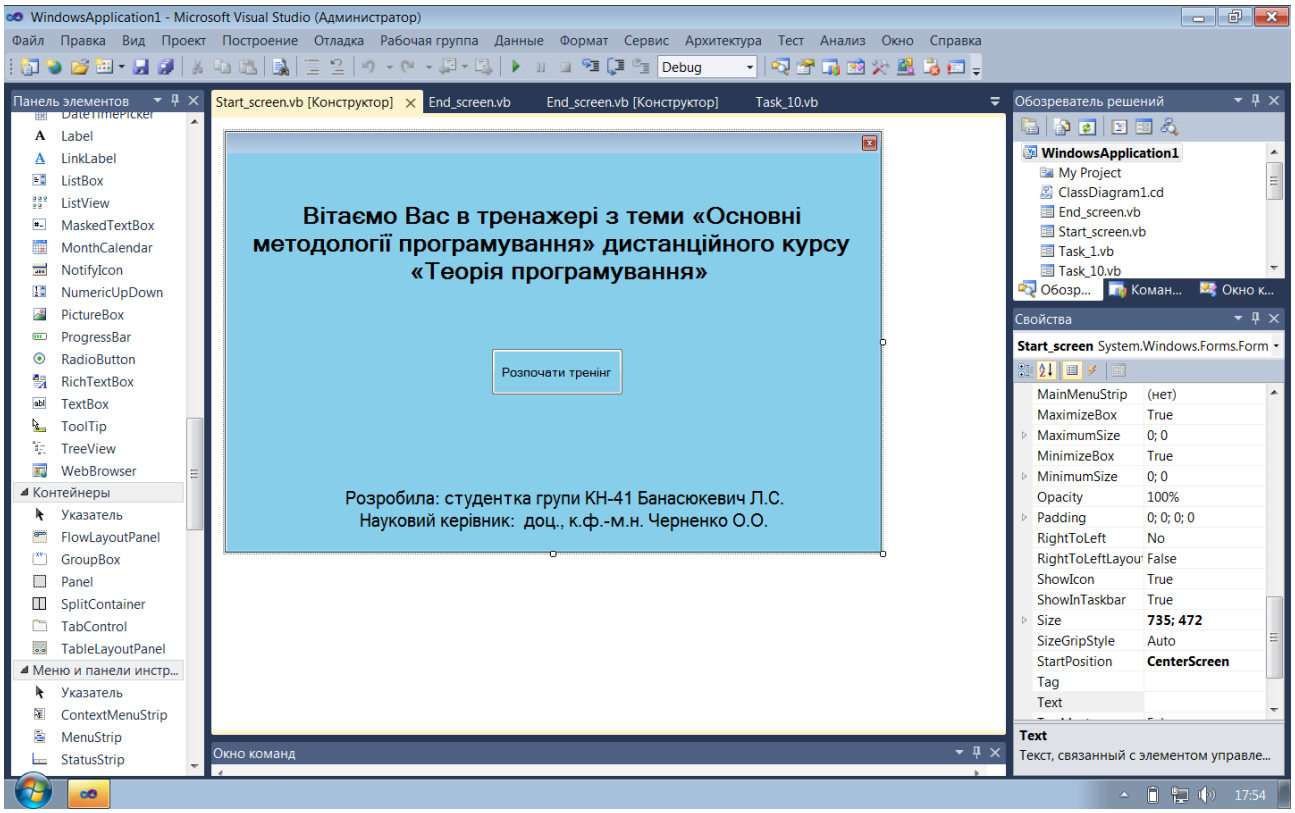


Рисунок 4.2 – титульна стартова форма тренажеру.

Потім, після титульної форми, потрібно реалізувати перше питання з тематики тренажеру, потрібно запрограмувати наступну форму, це буде перший крок алгоритмізації (рис. 4.3)

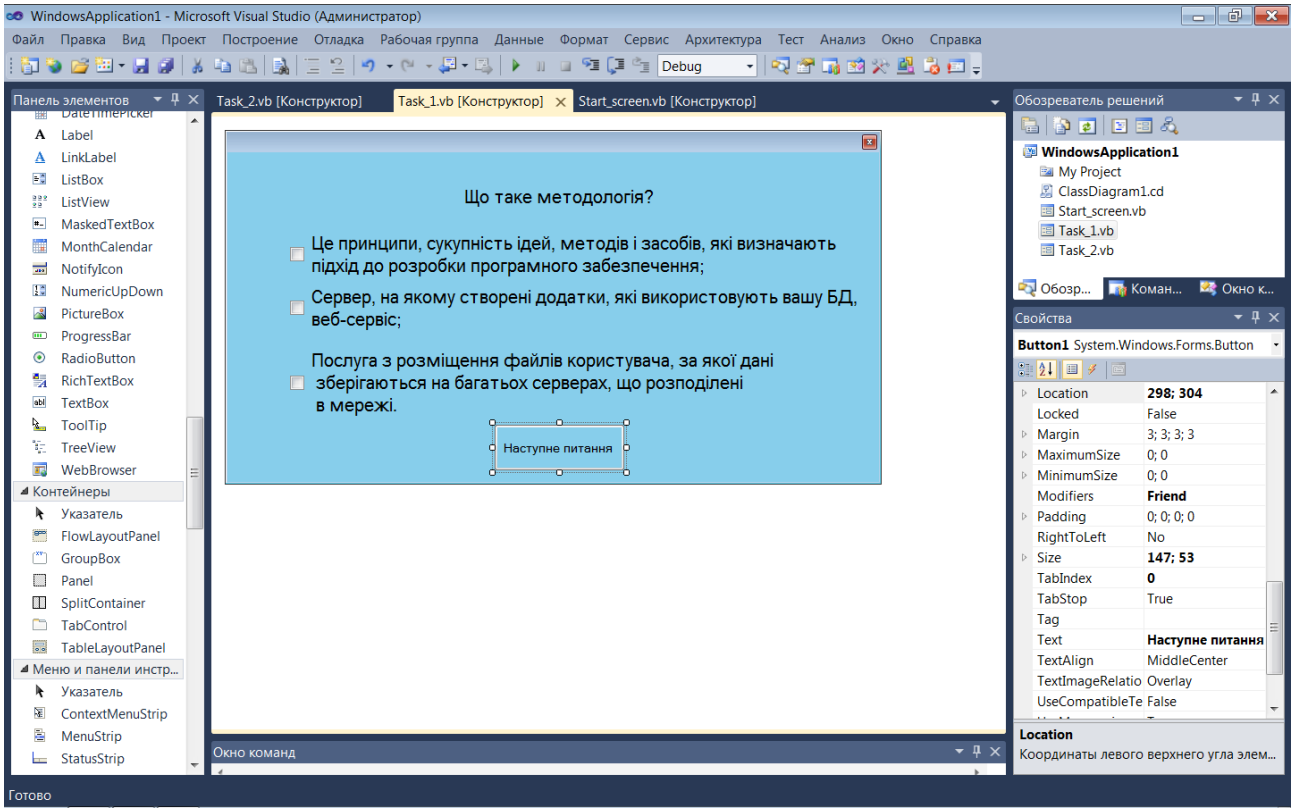


Рисунок 4.3 – перший крок алгоритмізації тренажеру.

Потім, після першого завдання, потрібно реалізувати друге питання з тематики тренажеру (рис. 4.4)

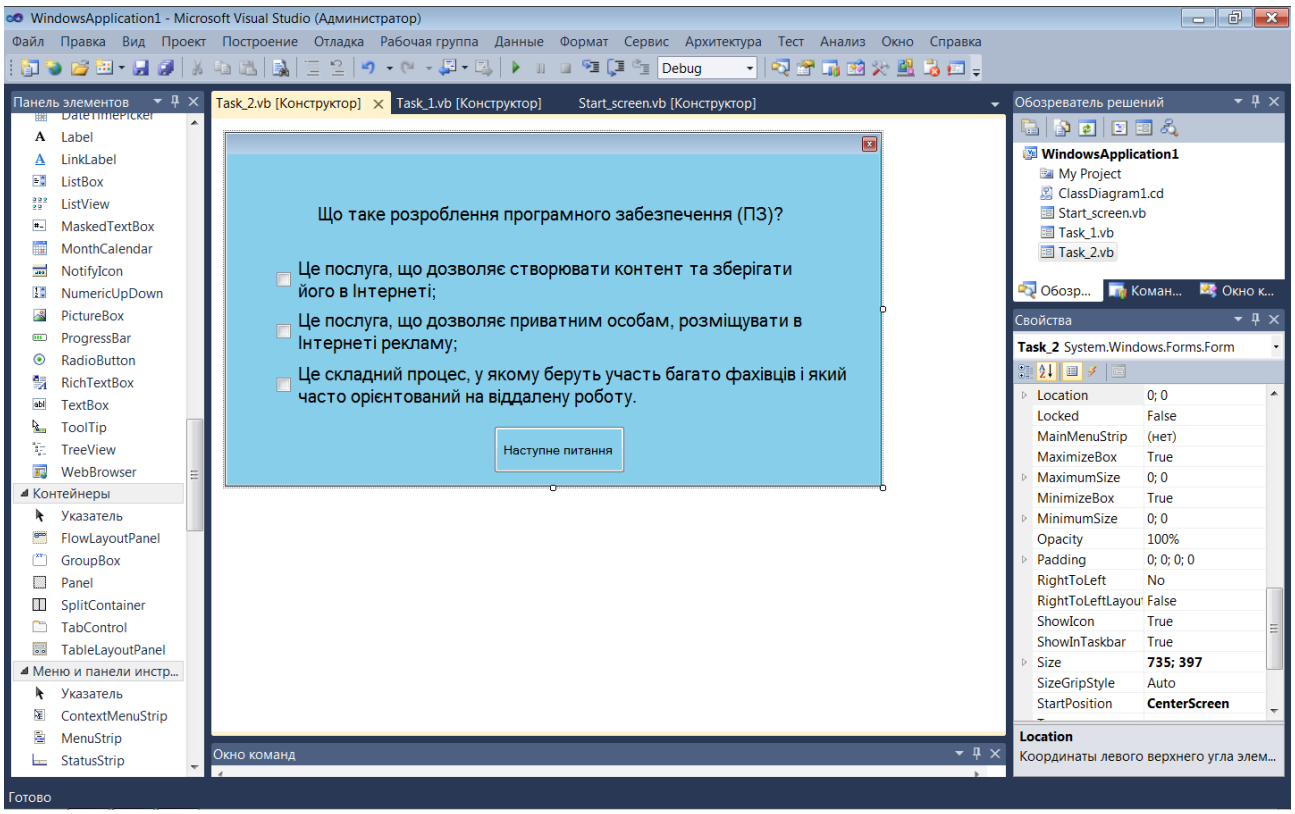


Рисунок 4.4 – другий крок алгоритмізації тренажеру.

За всю логіку роботи тренажеру, програмну реалізацію, можливість вибирати або вписувати відповіді відповідає наступний блок програмного коду. (рис. 4.5)

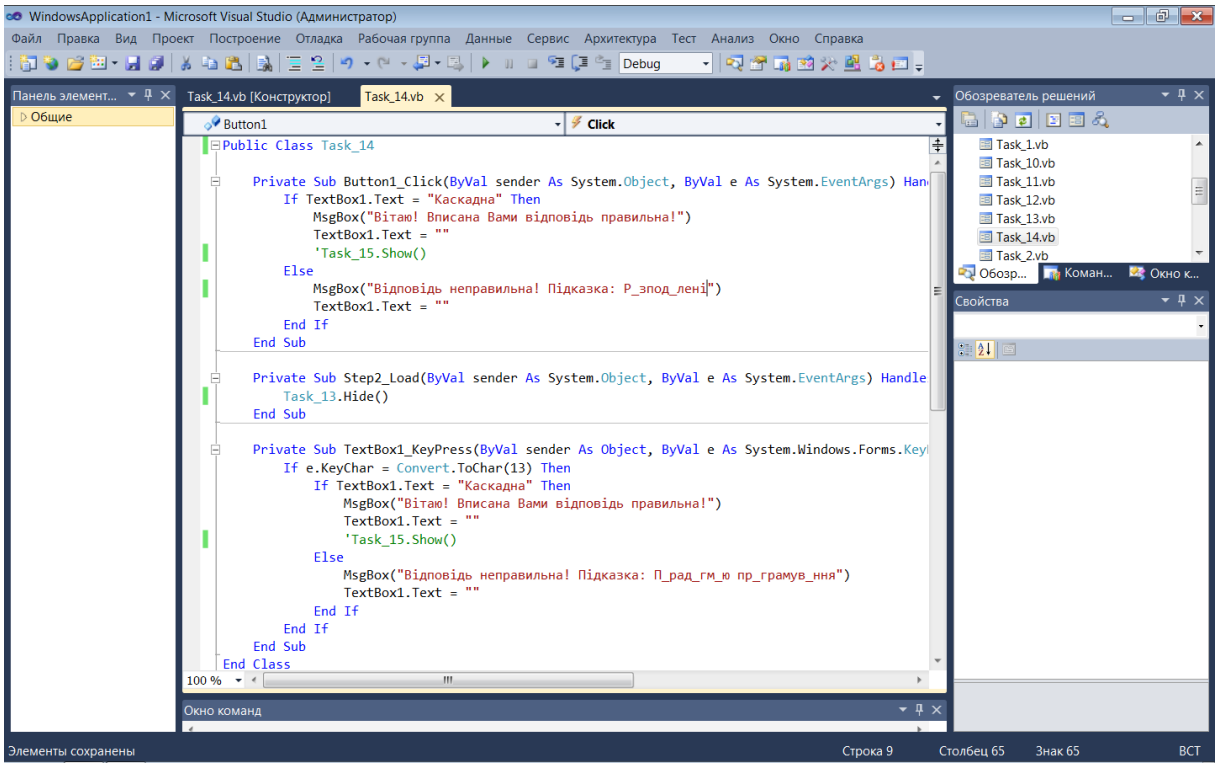


Рисунок 4.5 – блок програмного коду.

4.2. Інструкція по використанню навчального тренажеру

Після запуску навчального тренажеру, студенту відображається стартова форма тренажеру з кнопкою «Розпочати тренінг» (рис. 4.6)

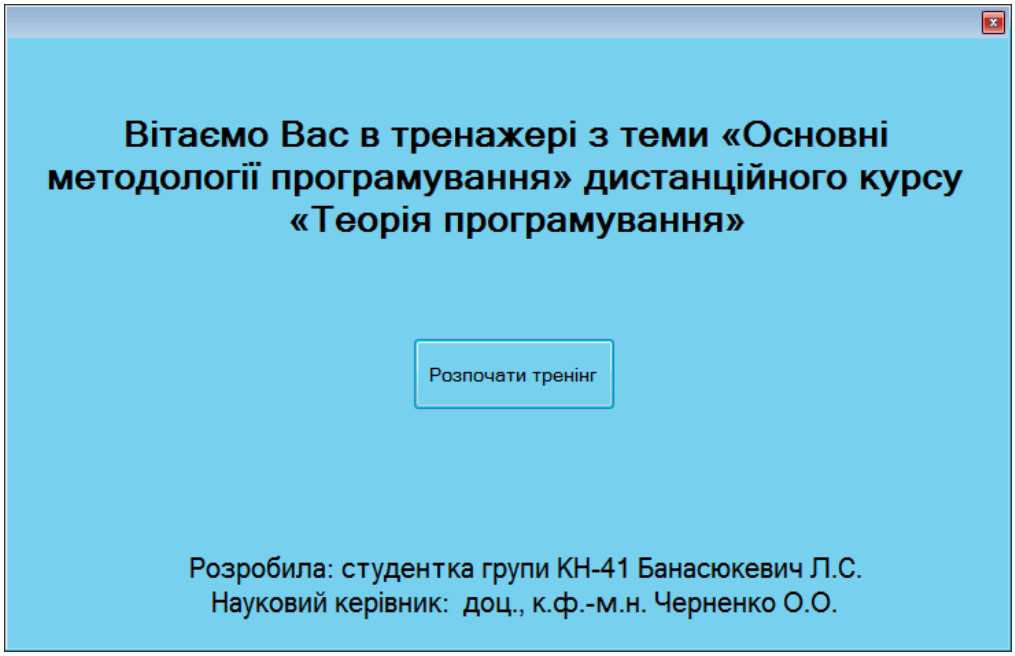


Рисунок 4.6 – стартова форма.

Після натискання кнопки «Розпочати тренінг», студенту відкривається перше питання, він може відповісти натиснувши на якесь питання (рис. 4.7)

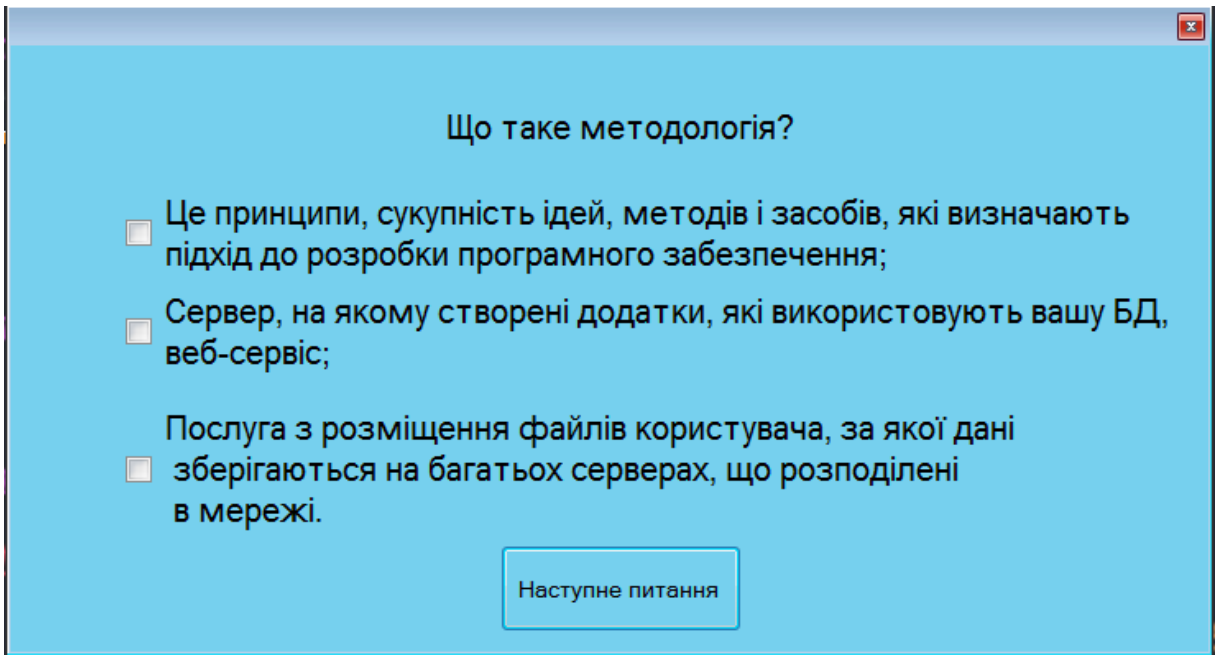


Рисунок 4.7 – перше питання.

Після вибору відповіді студент натискає кнопку «Наступне питання», якщо вибрана відповідь вірна, висвітиться додатково вікно (рис. 4.8)

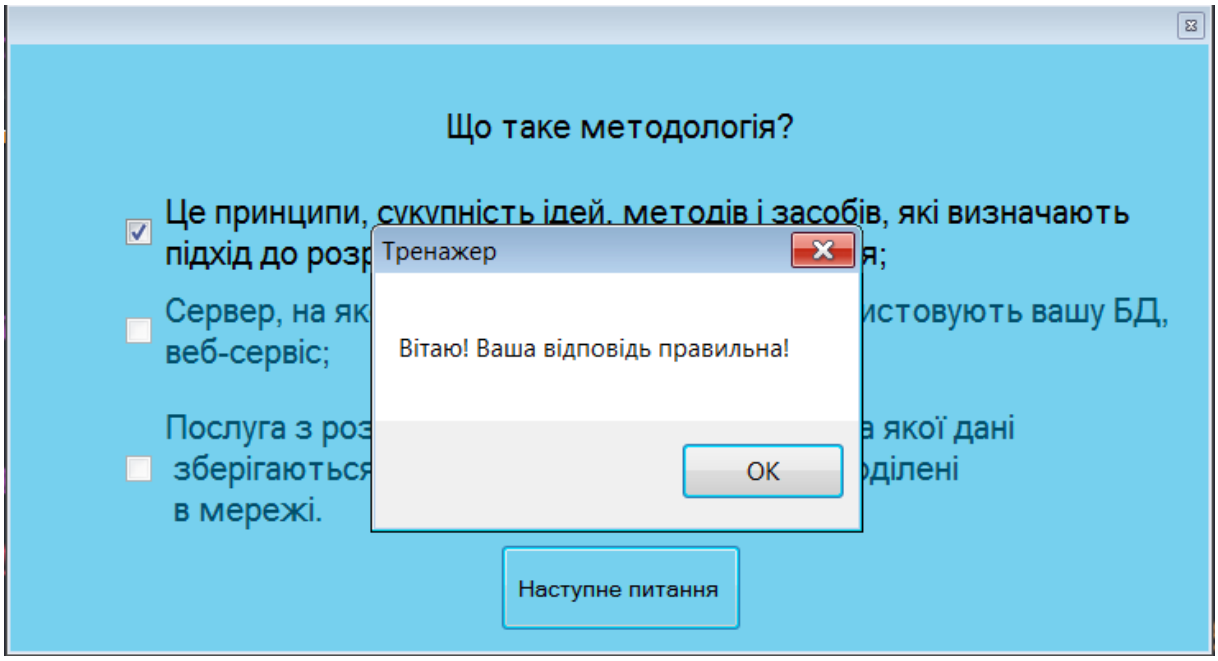


Рисунок 4.8 – вікно вірної відповіді.

Якщо студент відповів невірно, висвітиться помилка (рис. 4.9)

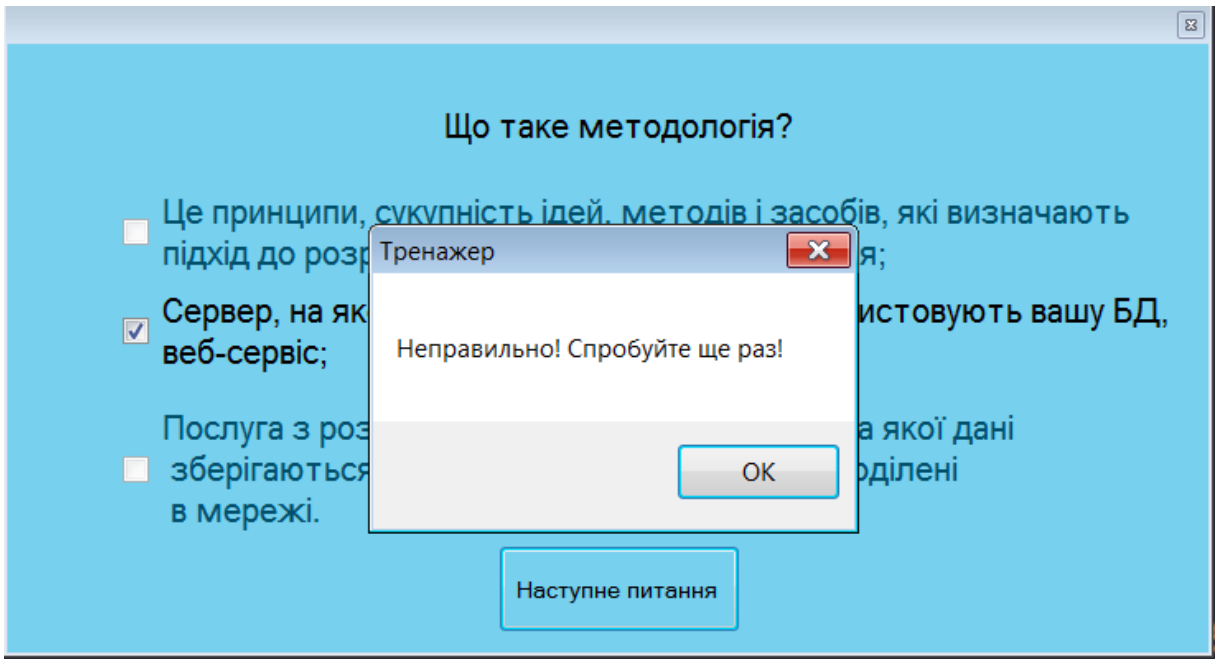


Рисунок 4.9 – вікно невірної відповіді.

Після правильної відповіді, користувач натискає на наступне питання з теми (рис. 5.0)

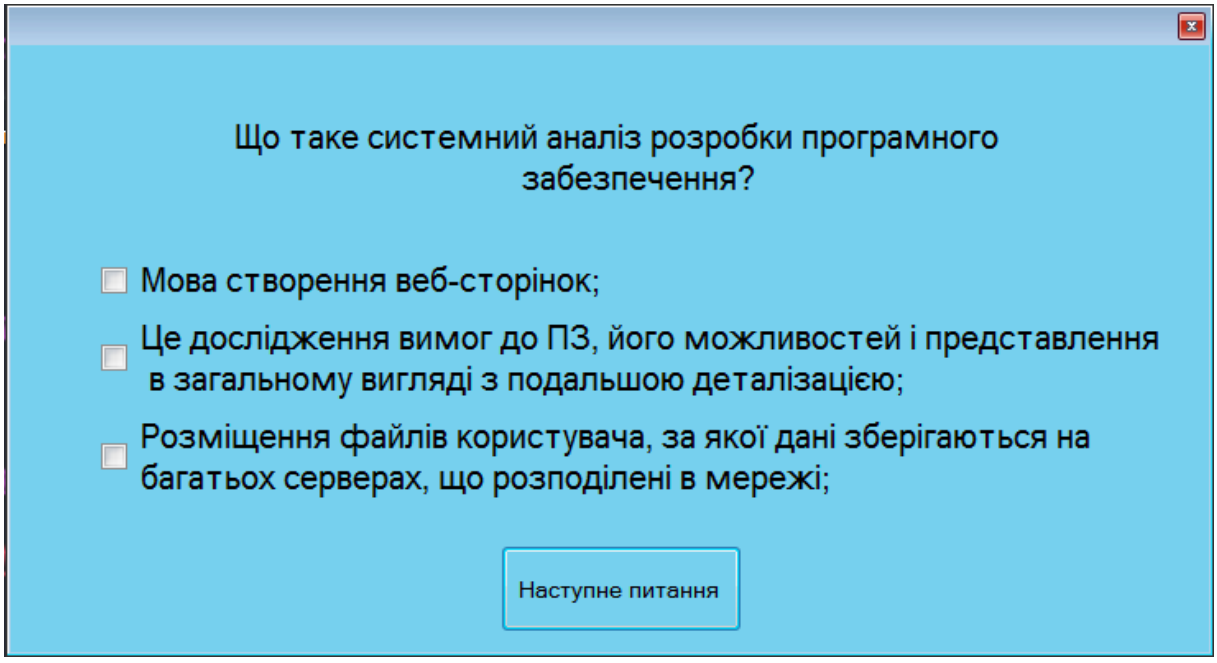


Рисунок 5.0 – вікно питання номер 2.

Потім після правильної відповіді користувачу відкриється питання номер 3 (рис. 5.1)

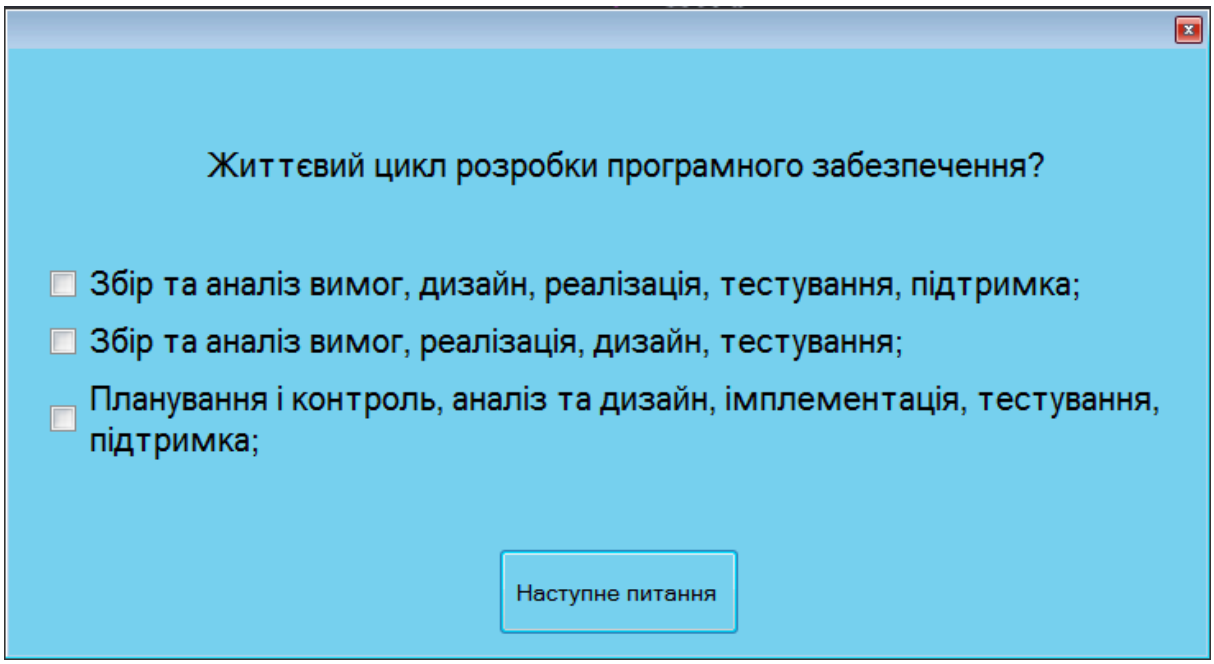


Рисунок 5.1 – вікно питання номер 3.

Після 3 питання користувачу по порядку відкривається 4 питання з теми (рис. 5.2)

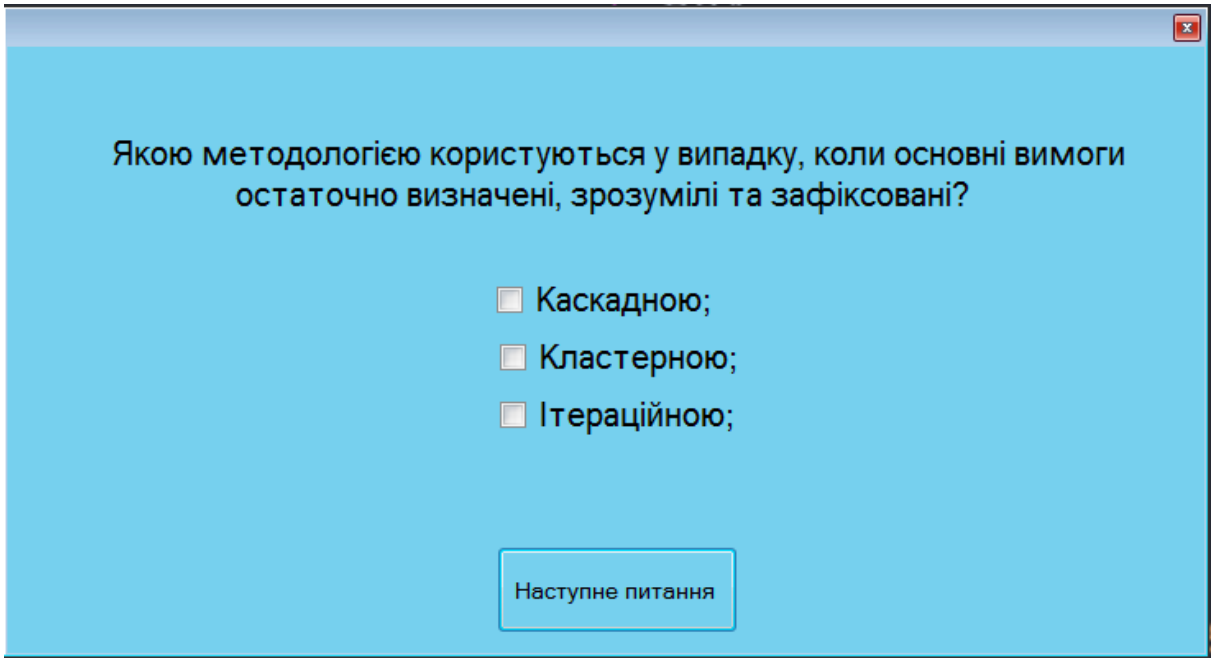


Рисунок 5.2 – вікно питання номер 4.

Якщо користувач вибрав правильний варіант відповіді, йому відкриється питання номер 5 (рис. 5.3)

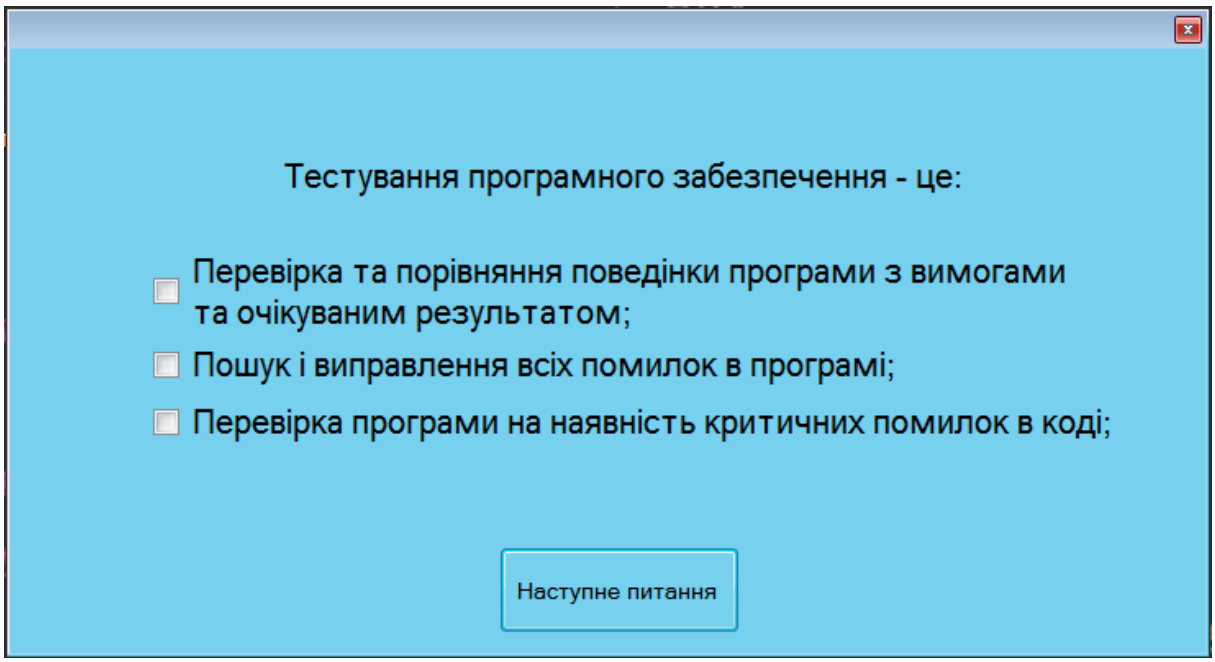


Рисунок 5.3 – вікно питання номер 5.

Після 5 питання, по порядку відкривається питання номер 6 (рис. 5.4)

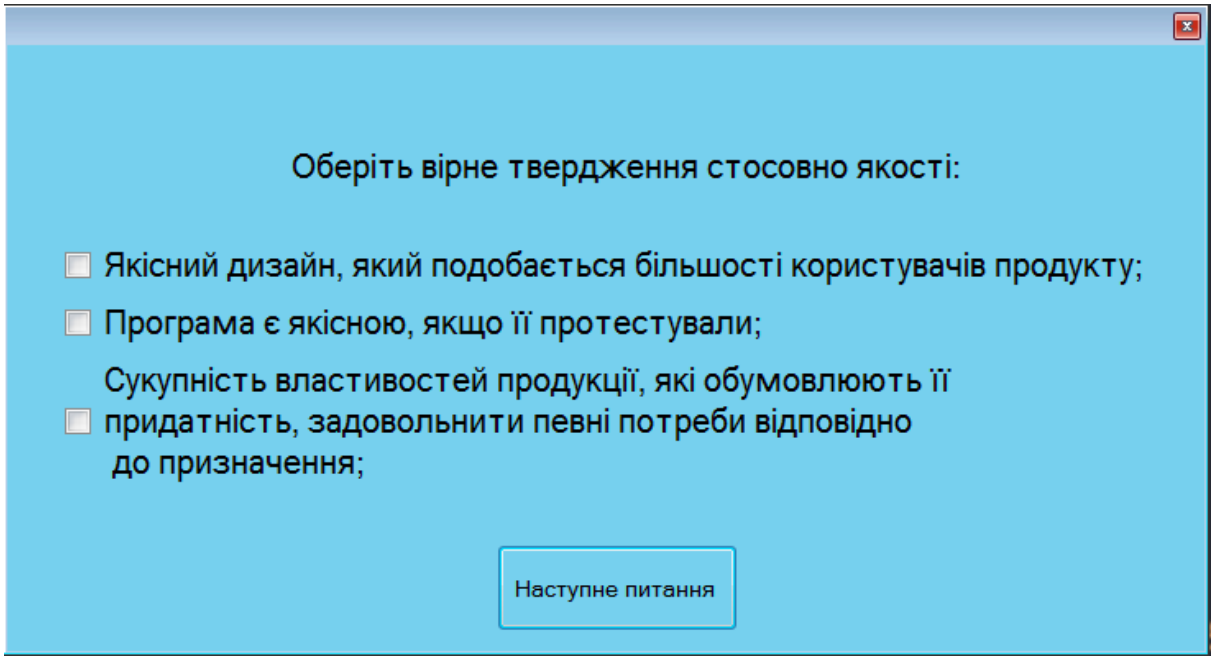


Рисунок 5.4 – вікно питання номер 6.

Потім користувачеві відкривається питання номер 7 (рис. 5.5)

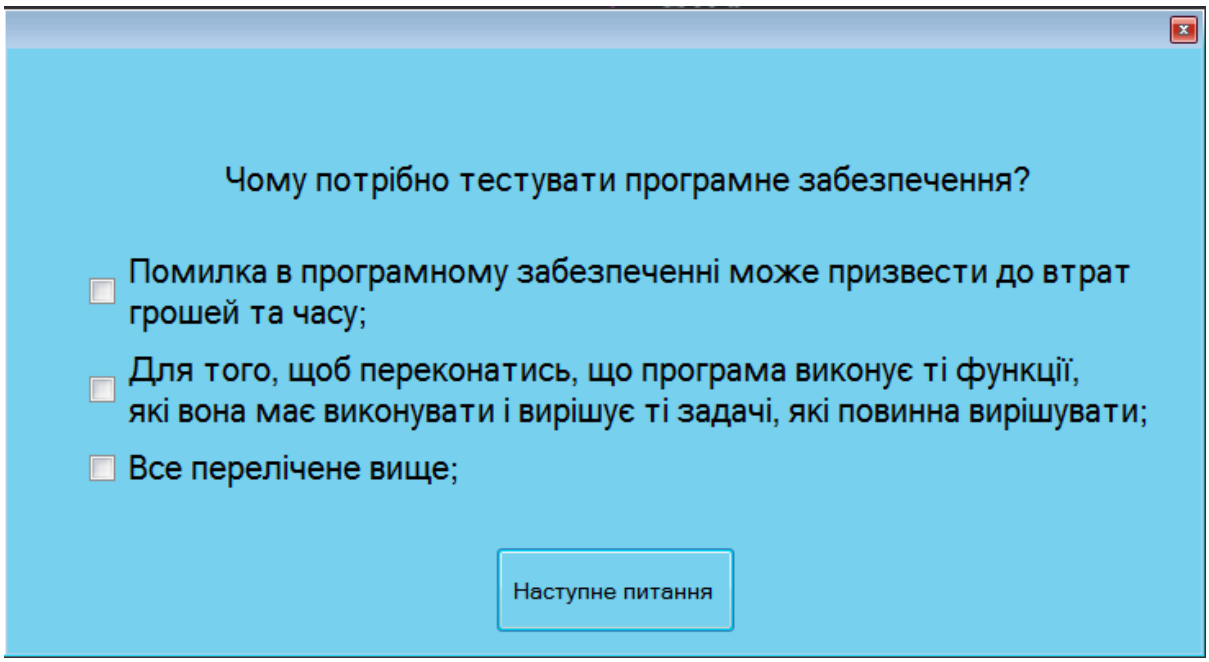


Рисунок 5.5 – вікно питання номер 7.

Потім користувачеві відкривається питання номер 8 (рис. 5.6)

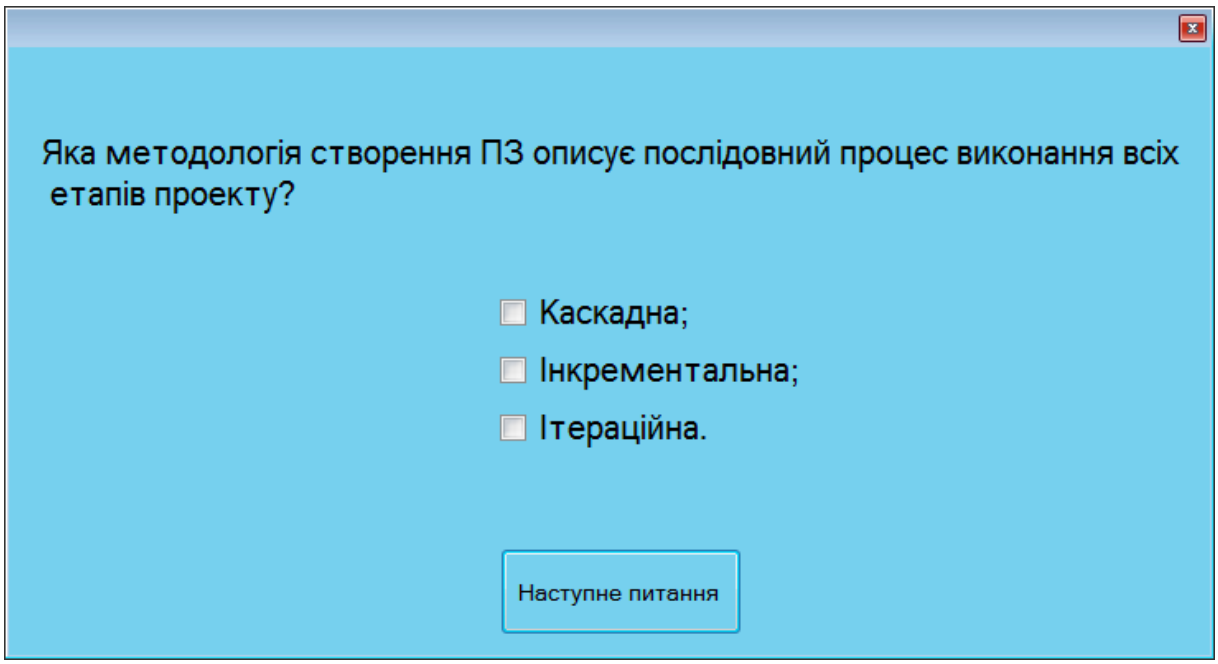


Рисунок 5.6 – вікно питання номер 8.

Аналогічно користувач відповідає на інші питання тренажеру, потім після десятого питання, користувач натискає кнопку «ОК», відкриється блок питань, відповідь на них можна вписати з клавіатури. (рис. 5.7)

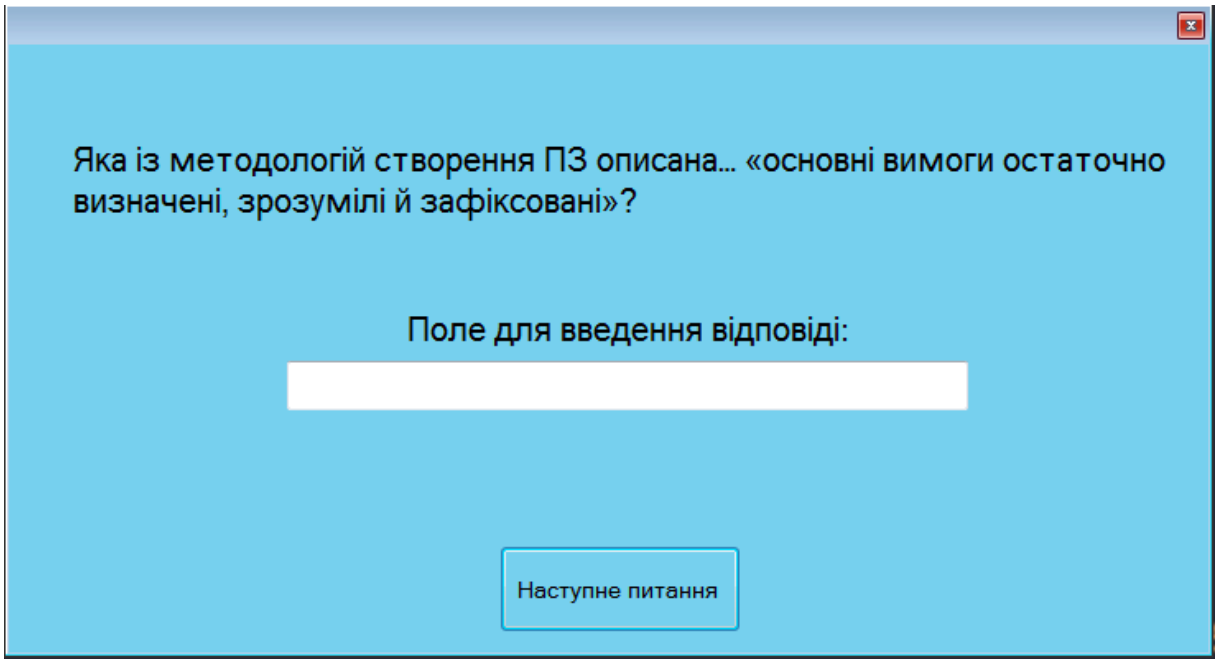


Рисунок 5.7 – вікно питання номер 11.

Після введення відповіді студент натискає кнопку «Наступне питання» або натискає кнопку «Enter», якщо вписана відповідь вірна, висвітиться вікно (рис. 5.8)

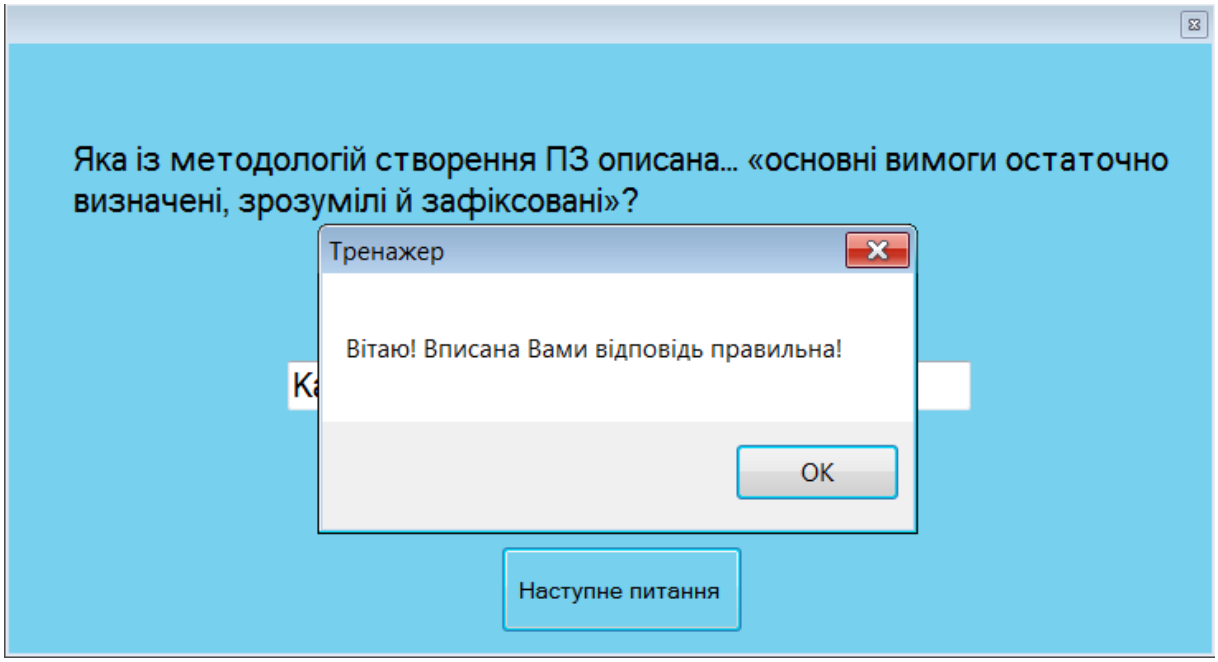


Рисунок 5.8 – вікно правильної відповіді.

Якщо студент вписав невірну відповідь, висвітиться помилка і підказка (рис. 5.9)

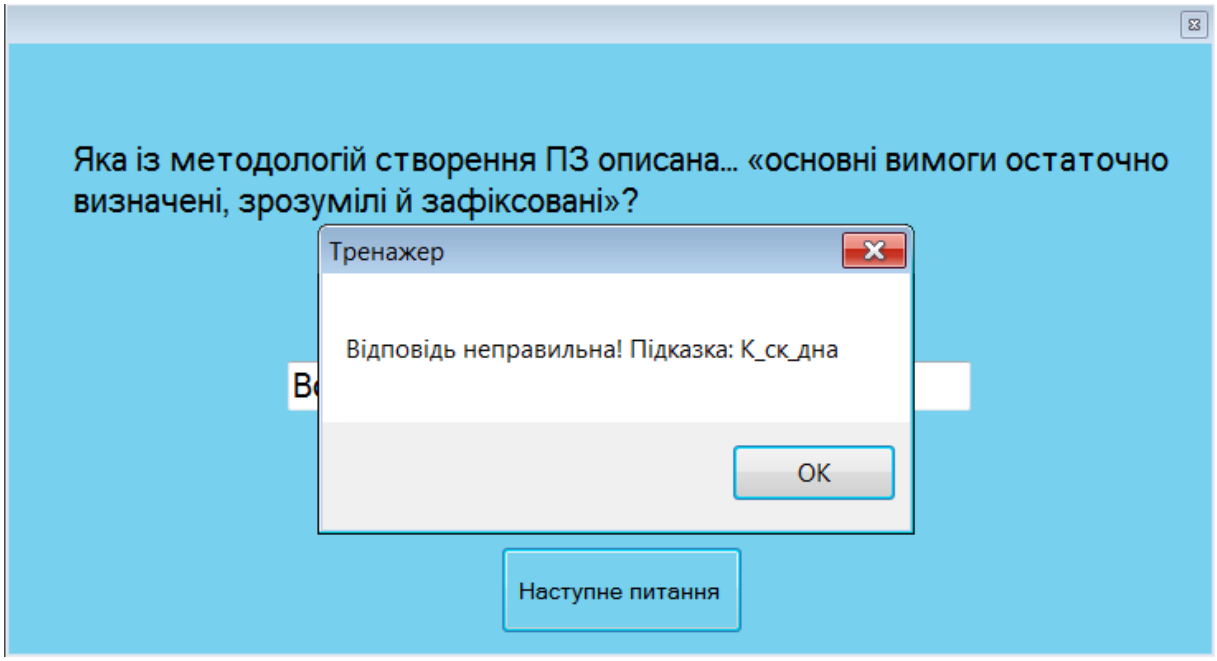


Рисунок 5.9 – вікно неправильної відповіді.

Потім користувач аналогічно вписує відповідь на інші питання, в кінці тренінгу відкриється підсумкове вікно. (рис. 6.0)

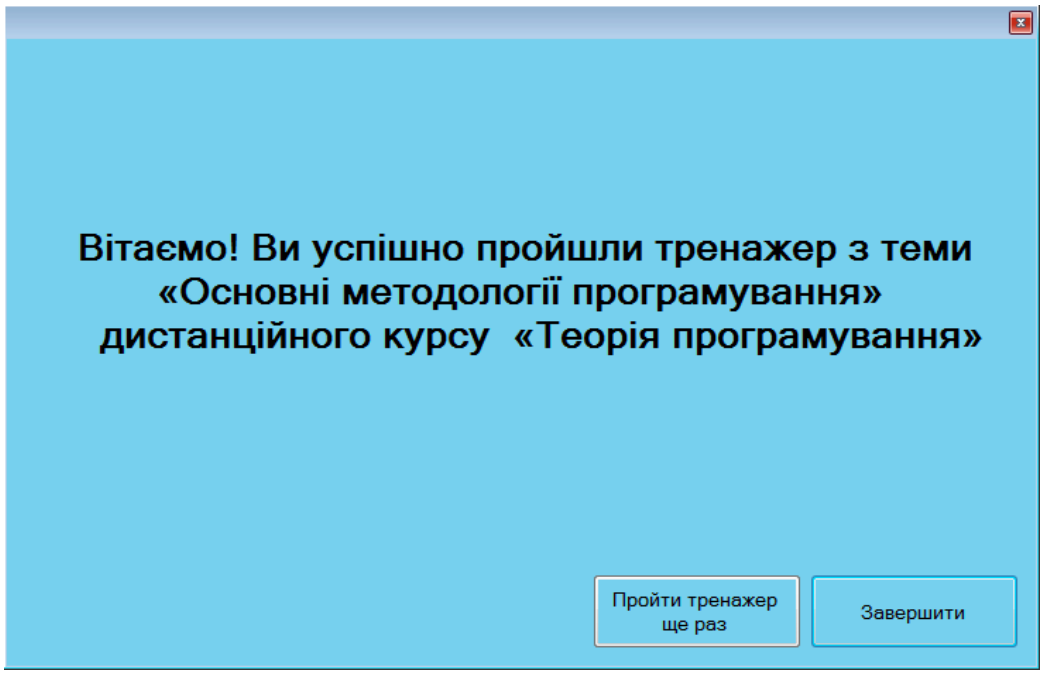


Рисунок 6.0 – підсумкове вікно тренажеру.

4.3. Обґрунтування вибору програмних засобів

Microsoft Visual Basic — засіб розроблення програмного забезпечення, створений і підтримуваний корпорацією Microsoft, який складається з мови програмування і середовища розроблення. Мова Visual Basic успадкувала дух, стиль і частково синтаксис свого предка - мови Бейсік, в якої є чимало діалектів. У той же час Visual Basic поєднує в собі процедури та елементи об'єктно-орієнтованих та компонентно-орієнтованих мов програмування. Середовище розробки VB містить інструменти для візуального конструювання користувальницького інтерфейсу.

Visual Basic одночасно і улюблений, і зневажений багатьма програмістами. Visual Basic вважається ідеальним засобом швидкої розробки прототипів програми, розробки додатків баз даних і взагалі для компонентного способу створення програм, що працюють під управлінням операційних систем родини Microsoft Windows.

Перше визнання серйозними розробниками Visual Basic отримав після виходу версії 3 - VB3. Остаточне визнання як повноцінного засобу програмування для Windows - при виході версії 5 - VB5. Версія VB6, що входить до складу Microsoft Visual Studio 6.0, стала по-справжньому зрілим і функціонально багатим продуктом. Після цього розробники з Microsoft суттєво змінили напрямок розвитку даної технології.

Visual Basic.NET не дозволяє програмувати по-старому, бо по суті є зовсім іншою мовою, такою самою, як і будь-яка інша мова програмування для платформи .NET. Індивідуальність мови і її переваги (простота, скромність створення програм, легкість використання готових компонент) при використанні в середовищі .NET не мають такого значення, як раніше - усе зосереджено на можливостях самої системи .NET, на її бібліотеці класів. Тому сьогодні (лютий 2008) треба говорити про класичний Visual Basic, його діалекти Visual Basic for Applications (VBA) і Visual Basic Scripting Edition (VBScript) і про мову для платформи .NET - Visual

Basic.NET. Ця мова дуже сильно прив'язана до свого середовища розроблення й до операційної системи Windows, оскільки вона є виключно інструментом написання Windows-додатків. Прив'язаність до середовища полягає в тому, що існує велика кількість засобів, призначених для допомоги й зручності програмування: вбудований відладчик, перегляд змінних і структур даних на льоту, вікно налагодження, спливаюча підказка при наборі тексту програми (Intellisense). Всі ці переваги роблять марним і навіть неможливим використання Visual Basic поза середовищем для розроблення, наприклад, у звичайному текстовому редакторі. (VBA) це засіб програмування, практично нічим не відрізняється від класичного Visual Basic, що призначене для написання макросів та інших прикладних програм для конкретних програм. Найбільшу популярність здобув завдяки своєму використання в пакеті Microsoft Office. Широке розповсюдження Visual Basic for Applications в поєднанні з споконвічно недостатньою увагою до питань безпеки призвело до значного поширення макровірусов. (VBScript) Це скриптова мова, що є дещо урізаною версією звичайного Visual Basic. Використовується в основному для автоматизації адміністрування систем Windows і для створення сторінок ASP та сценаріїв для Internet Explorer.

ВИСНОВКИ

Отже, впровадження дистанційних технологій під час війни у навчальний процес спрямоване на глибше розуміння навчального матеріалу; формування таких компетенцій як: комунікативні (безпосереднє спілкування за допомогою засобів мережі), інформаційні (пошук інформації з різних джерел та можливість її критичного осмислення), самоосвіти (вміння навчатись самостійно). Як показує практика, якщо студент не навчиться самостійно приймати рішення, визначати зміст своєї навчальної діяльності та знаходити засоби її реалізації, він не зможе якісно оволодіти тією чи іншою дисципліною. Окрім того, дистанційне навчання виконує й виховну функцію – сприяє формуванню провідних якостей особистості: активність, самостійність, самовдосконалення, що є надзвичайно важливими якостями на сьогодні.

Результатом виконання бакалаврської роботи є розроблений навчальний тренажер з теми «Основні методології програмування» дистанційного курсу «Теорія програмування».

Завдання бакалаврської роботи виконано. Створено зручний навчальний тренажер для підготовки та навчання студентів з теми «Основні методології програмування» дистанційного курсу «Теорія програмування».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Черненко О. О. Курсовий проєкт із фаху: методичні рекомендації щодо оформлення пояснювальних записок до курсового проєкту для студентів спеціальності 122 Комп'ютерні науки освітня програма «Комп'ютерні науки» ступеня бакалавра, магістра / О. О. Черненко. – Полтава : ПУЕТ, 2022. – 58 с. – 1 електрон. опт. диск (CVD-ROM).
2. Оптимізація освітнього процесу шляхом впровадження дистанційного навчання під час воєнного стану [Електронний ресурс]. – Режим доступу: https://er.knutd.edu.ua/bitstream/123456789/20210/1/PIONBUG_2022_P051-052.pdf
3. Методології програмування [Електронний ресурс]. – Режим доступу: https://studfile.net/preview/5994723/page:2/
4. Моделі життєвого циклу, принципи і методології розробки програмного забезпечення [Електронний ресурс]. – Режим доступу: https://evergreens.com.ua/ua/articles/software-development-metodologies.html
5. Microsoft Visual Studio [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio
6. Microsoft Visual Basic [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Visual_Basic
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. Microsoft Visual Basic Step by Step [Електронний ресурс]. – Режим доступу: https://books.google.com.ua/books?hl=uk&lr=&id=Ap9CAwAAQBAJ&oi=fnd&pg=PT18&dq=visual+basic&ots=n5NfMNuX8q&sig=8OFEk2tHiXjaKynsUtD6SgQKabY&redir_esc=y#v=onepage&q=visual%20basic&f=false
9. Середовище розробки Microsoft Visual Studio [Електронний ресурс]. – Режим доступу: https://informatics.in.ua/programming_csharp/part_01.php

10. Програмування на Visual Basic [Електронний ресурс]. – Режим доступу: https://kursoviks.com.ua/bd_kompyuterni/article_post/65-lektsiya-osnovi-programuvannya-movoyu-visual-basic-6-0
11. Гнучка методологія розробки програмного забезпечення [Електронний ресурс]. – Режим доступу: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjCpvDixNn9AhUGpIsKHRqVDyQQFnoECAsQAQ&url=https%3A%2F%2Fwezom.com.ua%2Fua%2Fblog%2Fmetodologija-razrabotki-programmnogo-obespechenija&usg=AOvVaw2NDOTLSArFhpkFQIvpuw1S
12. Software development methodology [Електронний ресурс]. – Режим доступу: https://www.wiki-data.uk-ua.nina.az/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D1%96%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F
13. Методологія розробки та тестування програмного забезпечення [Електронний ресурс]. – Режим доступу: https://uk.myservername.com/software-development
14. Гнучка розробка програмного забезпечення [Електронний ресурс]. – Режим доступу: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwi3kOjyx9n9AhWSp4sKHZOPCpM4ChAWegQIFhAB&url=https%3A%2F%2Fprofilbaru.com%2Fuk%2F%25D0%2593%25D0%25BD%25D1%2583%25D1%2587%25D0%25BA%25D0%25B0_%25D1%2580%25D0%25BE%25D0%25B7%25D1%2580%25D0%25BE%25D0%25B1%25D0%25BA%25D0%25B0_%25D0%25BF%25D1%2580%25D0%25BE%25D0%25B3%25D1%2580%25D0%25B0%25D0%25BC%25D0%25BD%25D0%25BE%25D0%25B3%25D0%25BE_%25D0%25B7%25D0%25B0%25D0%25B1%25D0%25B5%25D0%25B7%25D0%25BF%25D0%25B5%25D1%258

7%25D0%25B5%25D0%25BD%25D0%25BD%25D1%258F&usg=AOvVaw3EUUM4W1WMs61hk1wwsLaf

15. Популярні гнучкі методології розробки ПЗ [Електронний ресурс]. – Режим доступу: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjBwqufyNn9AhVTgosKHcA7CI04FBAWegQIDRAB&url=http%3A%2F%2Fsj.dstu.dp.ua%2Farticle%2Fview%2F145898&usg=AOvVaw1ai0TdZ5qnZg8H5h6KGYqj
16. AGILE методологія розробки програмного забезпечення. [Електронний ресурс]. – Режим доступу: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjBwqufyNn9AhVTgosKHcA7CI04FBAWegQIKBAB&url=https%3A%2F%2Facts.kpi.ua%2Fsyllabuses%2Fuk%2FDiscipline%3Fid%3D435&usg=AOvVaw1UFqNBwZU49uWPsOq6cLKE
17. Життєвий цикл розробки програмного забезпечення [Електронний ресурс]. – Режим доступу: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjBwqufyNn9AhVTgosKHcA7CI04FBAWegQIFxAB&url=https%3A%2F%2Ficstudio.online%2Fpost%2Fetapi-zhittyevogo-ciklu-rozrobki-pz&usg=AOvVaw1RV_EeUNoMXtAG9ANu4YvW
18. Розробка програмного забезпечення [Електронний ресурс]. – Режим доступу: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwivm9SBydn9AhUC-yoKHQBeBM04HhAWegQIChAB&url=http%3A%2F%2Fcsc.knu.ua%2Fuk%2Ffiler%2Fcanonical%2F1641813837%2F1615%2F&usg=AOvVaw2j-mGTcWzgWvH40arKJ83J
19. 9 основних методологій розробки програмного забезпечення [Електронний ресурс]. – Режим доступу: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad

=rja&uact=8&ved=2ahUKEwj195qAytn9AhWtmIsKHYBKBNIQFnoECAoQAQ&url=https%3A%2F%2Fagile.yakubovsky.com%2Fua%2F2015%2F10%2F9-osnovnykh-metodolohiy-rozrobky-prohramnoho-zabezpechennya-agile%2F&usg=AOvVaw0bXatvniOFEwWKTfkxBm4t

20. Дистанційна освіта [Електронний ресурс]. – Режим доступу: http://osvita.ch.ua/distanciyna_osvita/

ДОДАТОК А. КОД ПРОГРАМИ