ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ БІЗНЕСУ
ТА СУЧАСНИХ ТЕХНОЛОГІЙ

ФОРМА НАВЧАННЯ ДЕННА

КАФЕДРА МАТЕМАТИЧНОГО МОДЕЛЮВАННЯТА СОЦІАЛЬНОЇ ІНФОРМАТИКИ

Допускається до захисту

Завідувач кафедри _____О.О. Ємець
(підпис)

«_____»_____ 2021 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО БАКАЛАВРСЬКОЇ РОБОТИ

на тему

ТРЕНАЖЕР З ТЕМИ «ПОБУДОВА БЛОК-СХЕМ АЛГОРИТМІВ ЦИКЛІЧНОЇ СТРУКТУРИ» ДИСТАНЦІЙНОГО НАВЧАЛЬНОГО КУРСУ «ІНФОРМАТИКА» ТА РОЗРОБКА ЙОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

зі спеціальності 122 «Комп'ютерні науки»

Виконавець роботи Бибка Богдан Миколайович _____ «___»_____2021 р.
(підпис)

Науковий керівник к.ф.-м.н., доц., Ємець Олександра Олегівна

_____ «___»_____ 2021 р.
(підпис)

ПОЛТАВА 2021 р.

РЕФЕРАТ

Записка: 45 с., 48 рис., 4 додатка (на 49 сторінках), 12 джерел.

Предмет розробки – тренажер з побудови блок-схем алгоритмів циклічної структури для дистанційного курсу «Інформатика» ПУЕТ.

Мета роботи – створення програми-тренажера з теми «Побудова блок-схем алгоритмів циклічної структури».

Методи розробки – для побудови та перевірки алгоритму прикладу – мова Object Pascal, середовище Delphi. Для створення програми-тренажеру – мову C++, середовище Borland Builder. Для рисування блок-схем та їх фрагментів – пакет інженерної графіки Microsoft Visio.

Розроблено алгоритми для двох прикладів тренажерів. Намальовано блок-схему алгоритму тренажеру.

Написано програму-тренажер для побудови блок-схем алгоритмів циклічної структури. Створено документацію по програмі.

Ключові слова: ТРЕНАЖЕР, БЛОК-СХЕМА, АЛГОРИТМ, ЦИКЛ WHILE … DO, ЦИКЛ REPEAT ... UNTIL.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні позначення, символи, одиниці, скорочення, терміни	**Пояснення умовних позначень, символів, одиниць, скорочень, термінів**
Цикл	Фрагмент програми, який виконується декілька разів поспіль.
Цикл `while … do`	*Цикл з передумовою*. Цикл, в якому умова перевіряється на початку циклу. Якщо умова істинна, то цикл виконується. Якщо умова хибна, то цикл не виконується.
Цикл `repeat … until`	*Цикл з післяумовою*. Цикл, в якому умова перевіряється у кінці циклу. Якщо умова хибна, то цикл виконується. Якщо умова істинна, то цикл не виконується.
Цикл `for`	*Цикл з лічильником*. Цикл, який виконується заданому наперед кількість разів.

ВСТУП

Актуальність теми. Епідемія коронавірусу призвела до того, що всі школи та вищі навчальні заклади перейшли на дистанційне навчання.

Такий вид навчання передбачає більше самостійного опановування дисциплін, а інколи і меншу кількість часу, яку викладач може виділити на студентів.

Опанувати дисципліну допоможуть навчальні матеріали, представлені в дистанційному курсі. Тренажери (симулятори) – це вид програмного забезпечення, який допомагає користувачу опановувати задачі, що носять переважно розрахунково-обчислювальний характер. Разом з тим і для інших видів задач можна придумати вдалі симулятори.

Отже, актуальність даної теми беззаперечна.

Об’єкт розробки – це програма-тренажер.

Предмет розробки – це тренажер з побудови блок-схем алгоритмів циклічної структури для дистанційного курсу «Інформатика» ПУЕТ.

Мета та задачі дипломного проектування. Мета полягає у дослідженні тем з побудови блок-схеми алгоритмів, «Циклічні оператори мови Pascal» та створенні програми-тренажера з теми «Побудова блок-схем алгоритмів циклічної структури» для заданого дистанційного курсу.

Методи розробки. Для побудови та перевірки алгоритмів прикладів було використано мову `Object Pascal` та середовище `Delphi`. Для створення програми-тренажеру було взято мову `C++` та середовище `Borland Builder`. Для рисування блок-схем та їх фрагментів був використаний пакет інженерної графіки `Microsoft Visio`.

Структура пояснювальної записки – це чотири частини. У постановці задачі сформульовані вимоги до дипломного проекту. В інформаційного огляді описуються тренажери, які навчаються користувачів побудові блок-схем. У

теоретичній частині викладено алгоритм тренажеру та зображена блок-схема алгоритму. У практичній частині подано опис програми та її скріншоти.

Обсяг пояснювальної записки. Обсяг – 45 сторінок основного тексту.

1. ПОСТАНОВКА ЗАДАЧІ

Під час дипломного проектування необхідно ознайомитися з правилами побудови блок-схем. Для цього вивчити стандарт «Схемы алгоритмов, данных и систем. Условные обозначения и правила выполнения: ГОСТ 19.701-90.» [1] та опрацювати матеріал дистанційного курсу «Інформатика. Частина 1» [3] з теми «Поняття алгоритму. Блок-схеми алгоритмів».

Після цього слід вивчити, як в мові програмування Object Pascal працюють циклічні оператори. При цьому опрацювати тему «Оператори повторення» дистанційного курсу [3].

Далі слід підібрати (взяти із підручників, дистанційного курсу або інших джерел; або розробити самостійно) декілька різнопланових прикладів програм з використанням циклів.

Для підібраних програм скласти блок-схеми алгоритмів.

Потім розробити алгоритм тренажеру та закодувати алгоритм обраною мовою програмування.

Тренажер повинен бути розрахований на студентів спеціальності «Комп’ютерні науки» або інших програмістів-початківців.

У тренажері повинно бути зрозуміло, яка відповідь є вірною, у чому полягає помилка. В кінці програма обов’язково повинна видавати кінцевий малюнок – побудовану за кодом блок-схему.

Для програми слід створити інструкцію по експлуатації та описати типові кроки створення програми-тренажеру.

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Огляд тренажерів подібної тематики

Розглянемо програми, що навчають створювати блок-схеми [4-10].

1). Тренажер з теми «Побудова блок-схем алгоритмів лінійної структури», створений ІТ-студентом ПУЕТ Шакуро Вадимом [10].

Призначений для курсу «Інформатика».

У тренажері три приклади: за алгоритмом, заданим кодом мовою Object Pascal, слід навчитися складати блок-схему (рис. 2.1).

Спочатку подана умова (рис. 2.1). Потім на кожному кроці слід обрати вірну відповідь (рис. 2.2-2.3). Якщо відповідь вірна, вона буде виділена зеленим кольором (рис. 2.2), інакше червоним (рис. 2.3.). Пояснення помилки немає. Похибки слід виправляти самостійно.

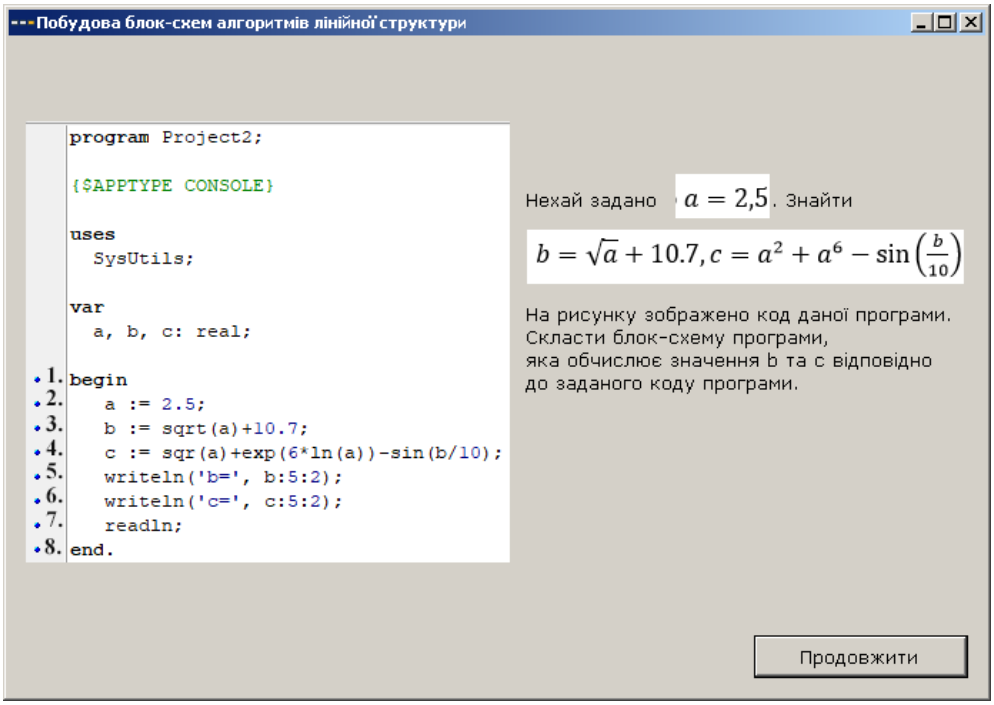

Рисунок 2.1 – Перший приклад (умова)

На кожному кроці користувач може звернутись до умови задачі.

В кінці користувач бачить підсумковий малюнок (рис. 2.4).

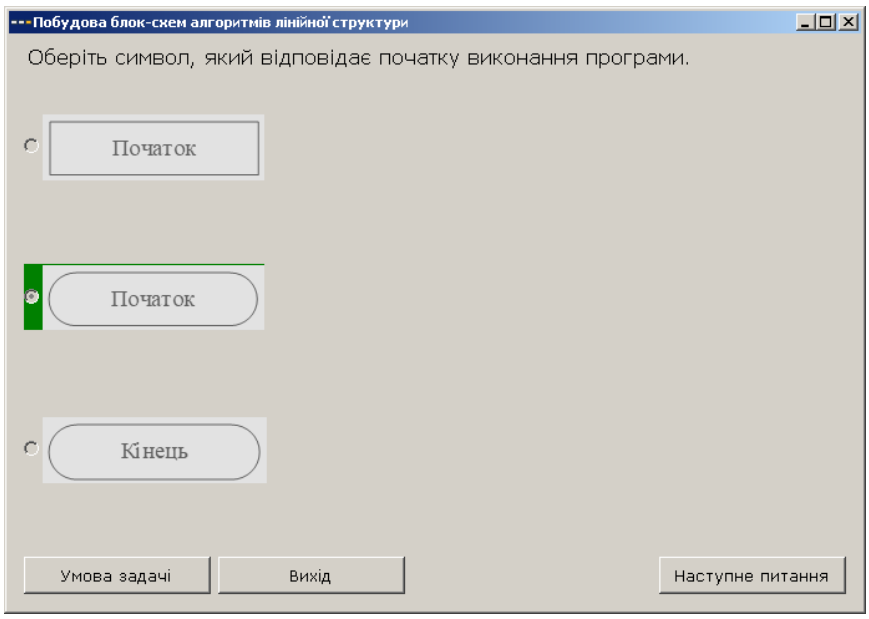

Рисунок 2.2 – Вірна відповідь крок 1)

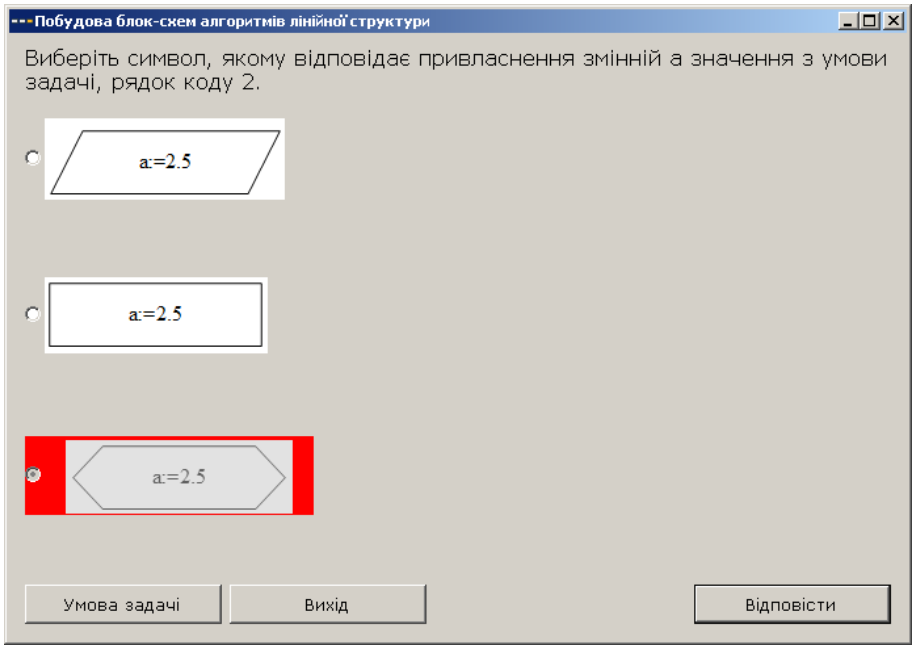

Рисунок 2.3 – Помилкова відповідь (крок 2)

2) Тренажер з теми «Побудова блок-схем алгоритмів розгалуженої структури», створений студенткою ПУЕТ Сузанською Аделіною [8]. Призначений для курсу «Програмування ІІ».

У тренажері одне завдання: за алгоритмом, заданим кодом мовою С++, слід навчитися складати блок-схему. Усього вісім кроків.

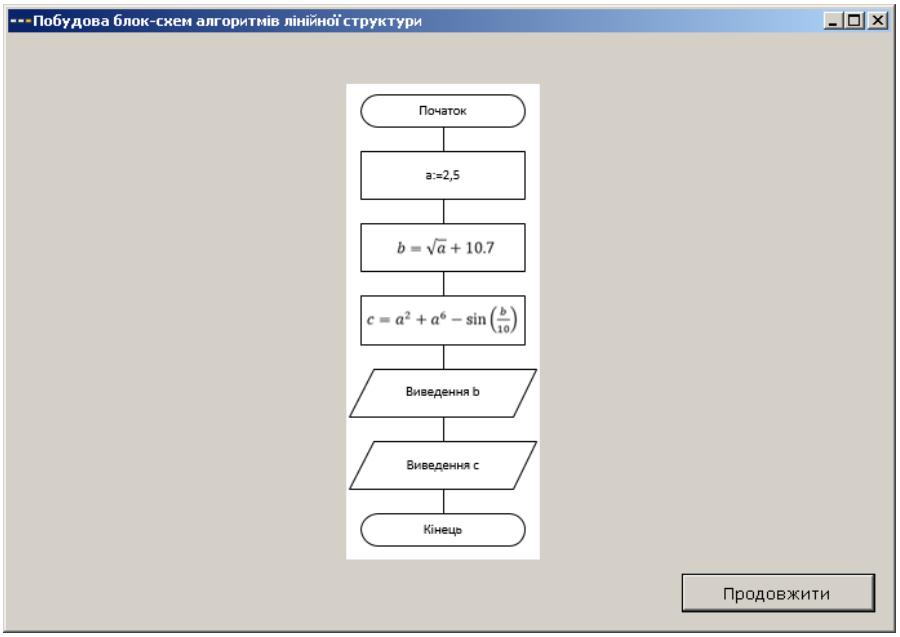

Рисунок 2.4 – Підсумок

Вікно програми на кожному кроці поділено на три частини (рис. 2.5, 2.8). У лівій – умова – код програми; посередині питання; у правій – фрагмент побудованої блок-схеми.

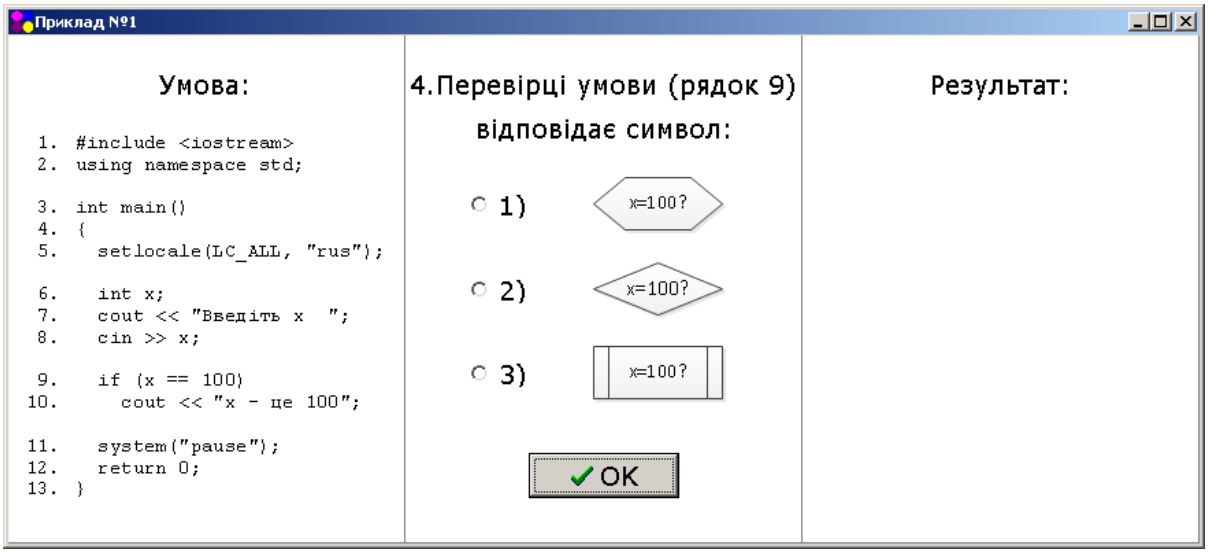

Рисунок 2.5 – Четвертий крок

На кожному кроці користувач відповідає на питання (обирає символ або фрагмент блок-схеми). Якщо відповідь помилкова, то з’являється пояснення помилки, яку користувач повинен виправити (рис. 2.6).

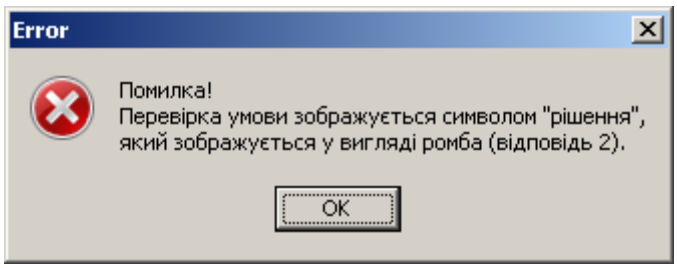

Рисунок 2.6 – Обробка помилки

Якщо відповідь вірна, на екрані виникає підтвердження (рис. 2.7), а після у правій частині вікна з’являється фрагмент блок-схеми з новим елементом, який додається як результат проходження кроку (рис. 2.8).

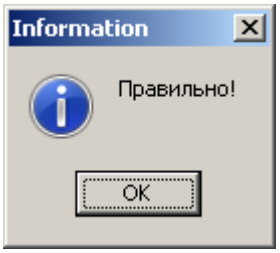

Рисунок 2.7 – У випадку вірності відповіді

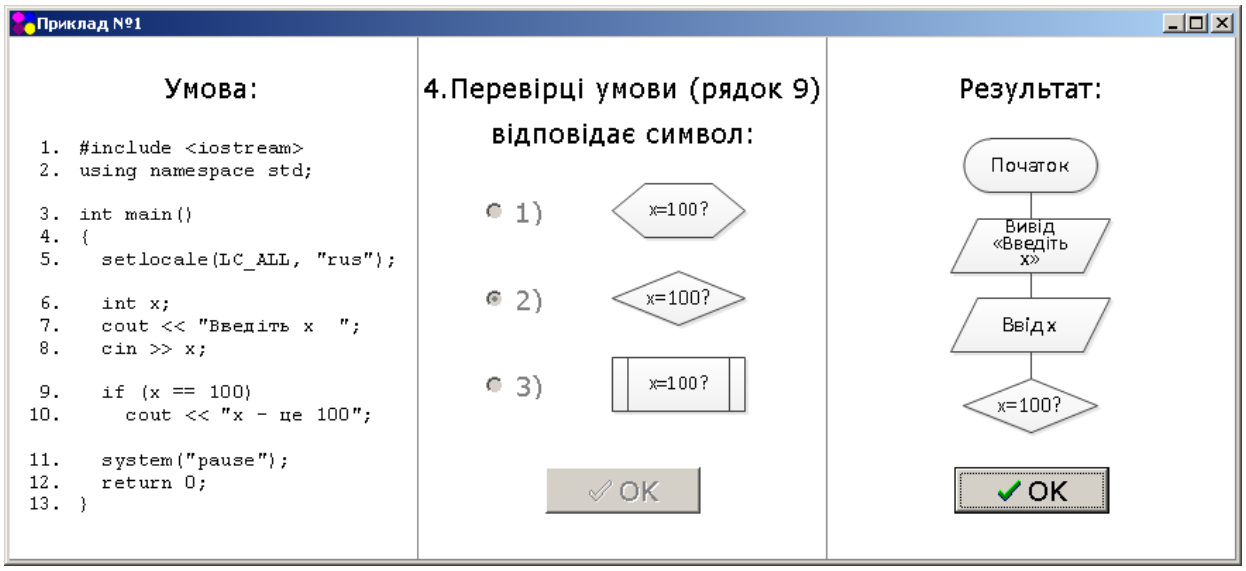

Рисунок 2.8 – Четвертий крок (з побудованим фрагментом блок-схеми)

3) Тренажер з теми «Побудова блок-схем алгоритмів циклічної структури на прикладі циклу «while», створений ІТ-студентом Недбайло Ярославом [7].

Призначений для курсу «Інформатика».

Фактично тема не розкрита. Програмна реалізація тренажеру погана і слабка.

2.2. Позитивні аспекти переглянутих програм

Програма № 1 Три різних приклади, що є достатньо для засвоєння теми. Використання кольорів пожвавлює програму. Є підсумковий малюнок. Умова прикладу весь часу доступна.

Програма №2 ґрунтовна пророблена. Умова весь час перед очима користувача. Різнопланові питання. У заданому питанні є посилання на номер рядка програми. Є пояснення помилки. Від користувача вимагається осмислення дій і самостійне виправлення помилок. Є підтвердження вірності відповіді. На кожному етапі побудови блок-схеми зразу видно, які символи додаються.

Програма № 3 позитивних рис не містить.

2.3. Негативні аспекти переглянутих програм

Програма № 1 містить помилки у коді програми. Відсутні пояснення помилки та яка вірна відповідь.

Програма № 2 недоліків немає. Але бажано було б доповнити тренажер іншими прикладами для закріплення засвоєння матеріалу.

Програма № 3 непридатна для використання через погану реалізацію.

2.4. Необхідність та актуальність теми

Аналізуючи існуючі розробки, бачимо, що для циклічних алгоритмів, які кодуються мовою `Object Pascal`, поки що не створено гідних тренажерів. Або таких немає у вільному доступі. Отже, тема даної випускової роботи актуальна, існує потреба в розробці даного тренажеру.

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Алгоритм тренажеру

Тренажер складається з двох прикладів.

Під час проходження тренінгу умови прикладів повинна бути доступною для перегляду.

Якщо відповідь вірна, то відбувається перехід до фрагменту блок-схеми.

У випадку помилкової відповіді на екрані з’являється пояснення помилки. Користувач повинен її виправити.

У випадку відсутності відповіді на екрані з’являється попереджуюче повідомлення з вказівкою обрати (надати) відповідь.

3.1.1. Приклад 1

На екрані з’являється умова завдання.

Умова: Побудувати блок-схему алгоритму, заданого алгоритмічною мовою Object Pascal:

```
const
  N = 5;
var
  a, Sum, i: integer;
BEGIN
  Sum := 0;
  i := 1;
  while (i <= N) do
  begin
    write(‘Введіть число: ’);
    readln(a);
    Sum := Sum + a;
    i := i + 1;
```

```
  end;
  writeln('Сума =', Sum);
  readln;
END.
```

Крок 1. Початок алгоритму відображається символом

а) 

б) 

в) 

Вірна відповідь – б.

Якщо відповідь хибна, то з'являється повідомлення: «Початок, кінець або зупинка алгоритму відображаються символом «термінатор», який має форму овалу зі сплюснутими сторонами.».

На екрані з'являється рисунок 3.1:

Рисунок 3.1 – Фрагмент блок-схеми

Крок 2. Рядок

```
N = 5;
```

відображається символом:

а) 

б)

в)

Вірна відповідь – а.

Якщо відповідь хибна, то з’являється повідомлення: «Усі маніпуляції з даними відображаються у символі «процес», який має форму прямокутника.».

На екрані з’являється рисунок 3.2:

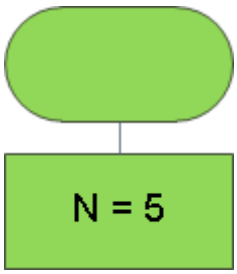

Рисунок 3.2 – Фрагмент блок-схеми

Крок 3. Рядок

```
Sum := 0;
```

відображається символом:

а)

б)

в)

Вірна відповідь – б.

Якщо відповідь невірна, то з’являється повідомлення: «Усі маніпуляції з даними відображаються у символі «процес», який має форму прямокутника без смуг.».

На екрані з’являється рисунок 3.3:

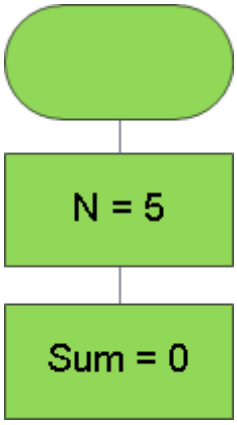

Рисунок 3.3 – Фрагмент блок-схеми

Крок 4. Рядок

```
i := 1;
```

відображається символом:

а)

б)

в)

Вірна відповідь – в.

Якщо відповідь хибна, то з’являється повідомлення: «Усі маніпуляції з даними відображаються у символі «процес», який має форму прямокутника без смуг.».

На екрані з’являється рисунок 3.4:

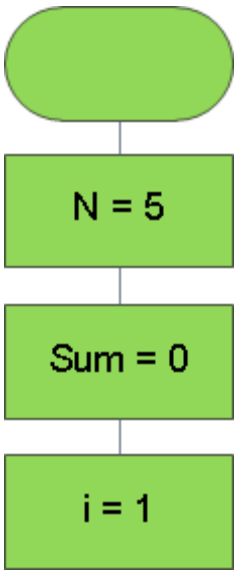

Рисунок 3.4 – Фрагмент блок-схеми

Крок 5. Рядок

```
while (i <= N) do
```

відображається символом:

а) 

б) 

в) 

Вірна відповідь – б.

Якщо відповідь хибна, то з’являється повідомлення: «Перевірка умови циклу відображається символом «рішення», який має форму ромба.».

На екрані з’являється рисунок 3.5:

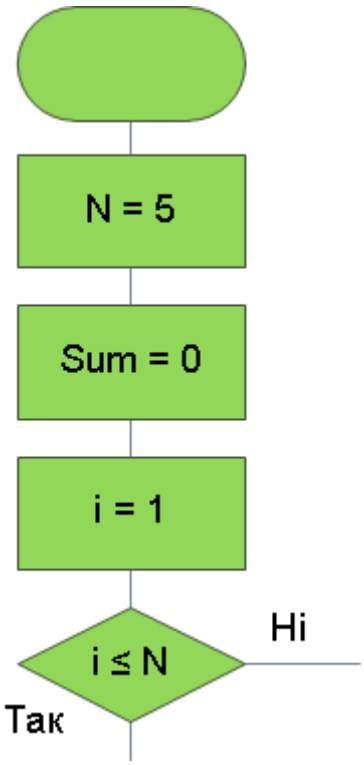

Рисунок 3.5 – Фрагмент блок-схеми

Крок 6. Рядок

```
write('Введіть число: ');
```

відображається символом:

а)

б)

в)

Вірна відповідь – а.

Якщо відповідь помилкова, то з'являється повідомлення: «Для виводу інформації використовується символ «дані», який має форму паралелограма.».

На екрані з'являється рисунок 3.6:

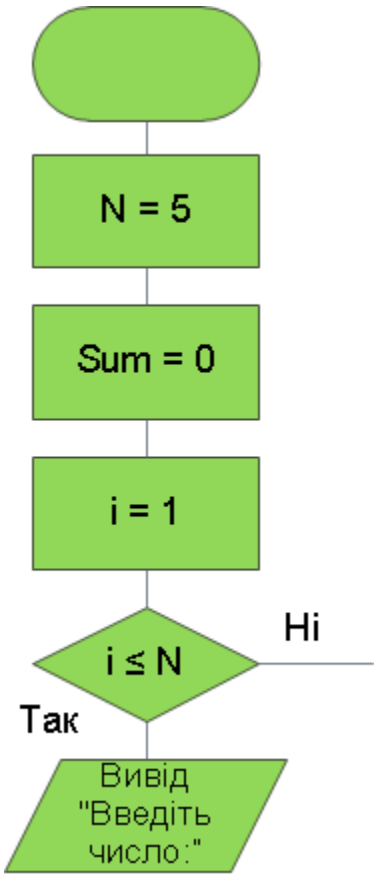

Рисунок 3.6 – Фрагмент блок-схеми

Крок 7. Рядок

```
readln(a);
```

відображається символом:

а)

б)

в)

Вірна відповідь – а.

Якщо відповідь помилкова, то з’являється повідомлення: «Для вводу інформації (як і для виводу) використовується символом «дані», який має форму паралелограма.».

На екрані з’являється рисунок 3.7:

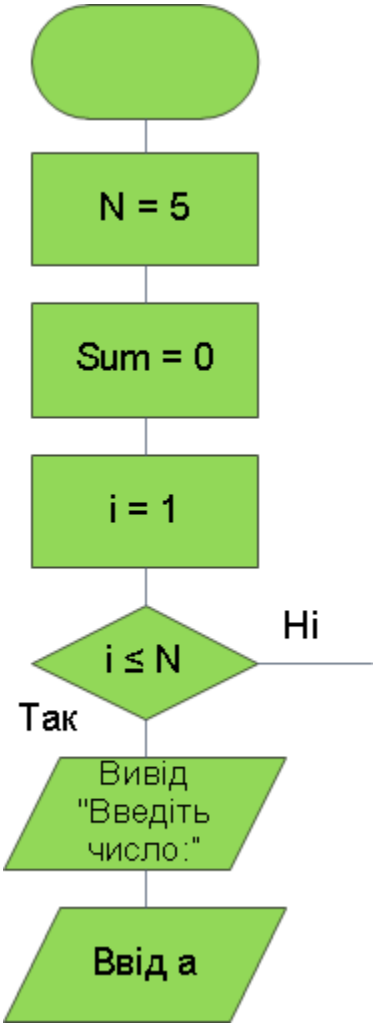

Рисунок 3.7 – Фрагмент блок-схеми

Крок 8. Рядок

```
Sum := Sum + a;
```

відображається символом:

а) 

б)

в)

Вірна відповідь – в.

Якщо відповідь помилкова, то з’являється повідомлення: «Усі арифметичні обчислення відображаються у символі «процес», який має форму прямокутника без смуг.».

На екрані з’являється рисунок 3.8.

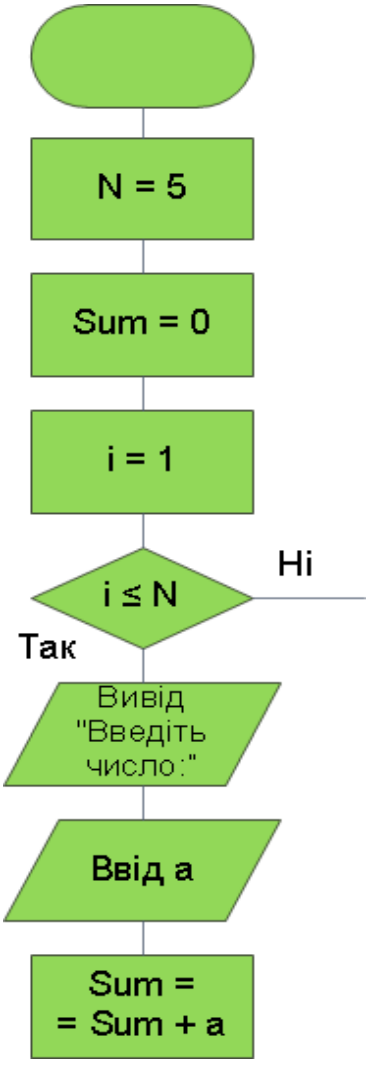

Рисунок 3.8 – Фрагмент блок-схеми

Крок 9. Рядок

```
i := i + 1;
```

відображається символом:

а) 

б) 

в) 

Вірна відповідь – б.

Якщо відповідь помилкова, то з'являється повідомлення: «Усі арифметичні обчислення відображаються у символі «процес», який має форму прямокутника без смуг.».

На екрані з'являється рисунок 3.9.

Крок 10. Рядок

```
writeln('Сума =', Sum);
```

відображається символом:

а) 

б) 

в) 

Вірна відповідь – в.

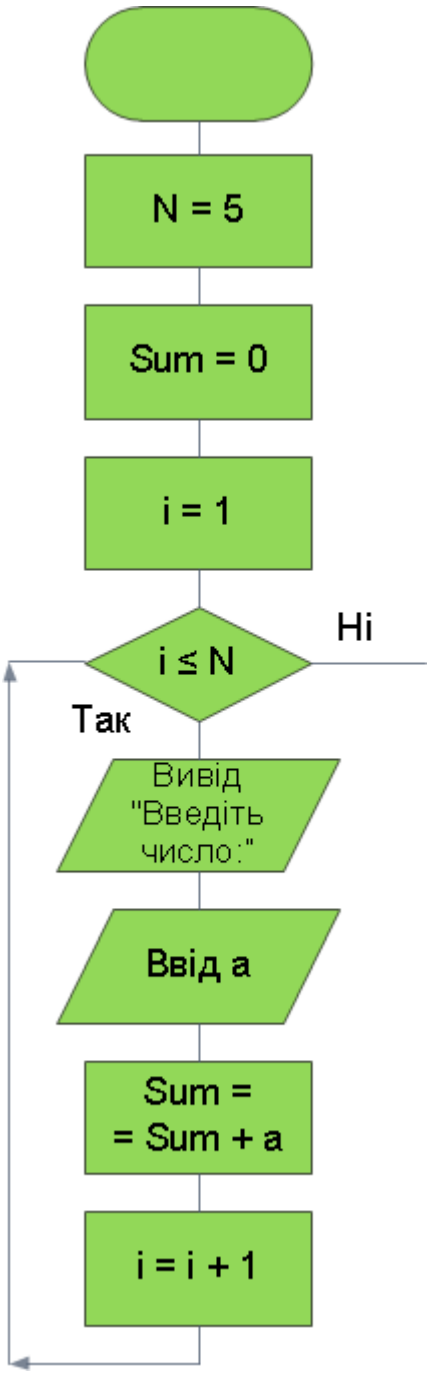

Рисунок 3.9 – Фрагмент блок-схеми

Якщо відповідь невірна, то з’являється повідомлення: «Для виводу інформації використовується символ «дані», який має форму паралелограма.».

На екрані з’являється рисунок 3.10:

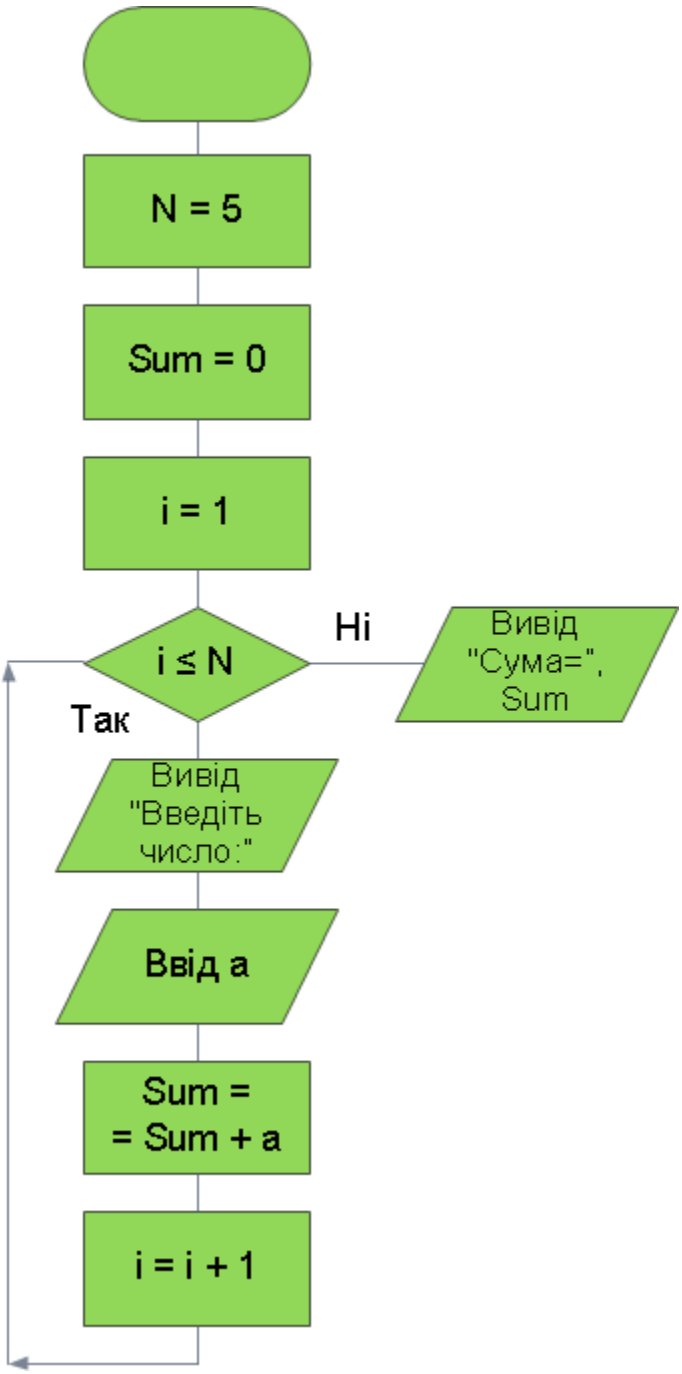

Рисунок 3.10 – Фрагмент блок-схеми

Крок 11. Кінець алгоритму відображається символом

а)

б)

в) 

Вірна відповідь – б.

Якщо відповідь хибна, то з’являється повідомлення: «Початок, кінець або зупинка алгоритму відображаються символом «термінатор», який має форму овалу зі сплюснутими сторонами.».

На екрані з’являється результуючий рисунок 3.11.

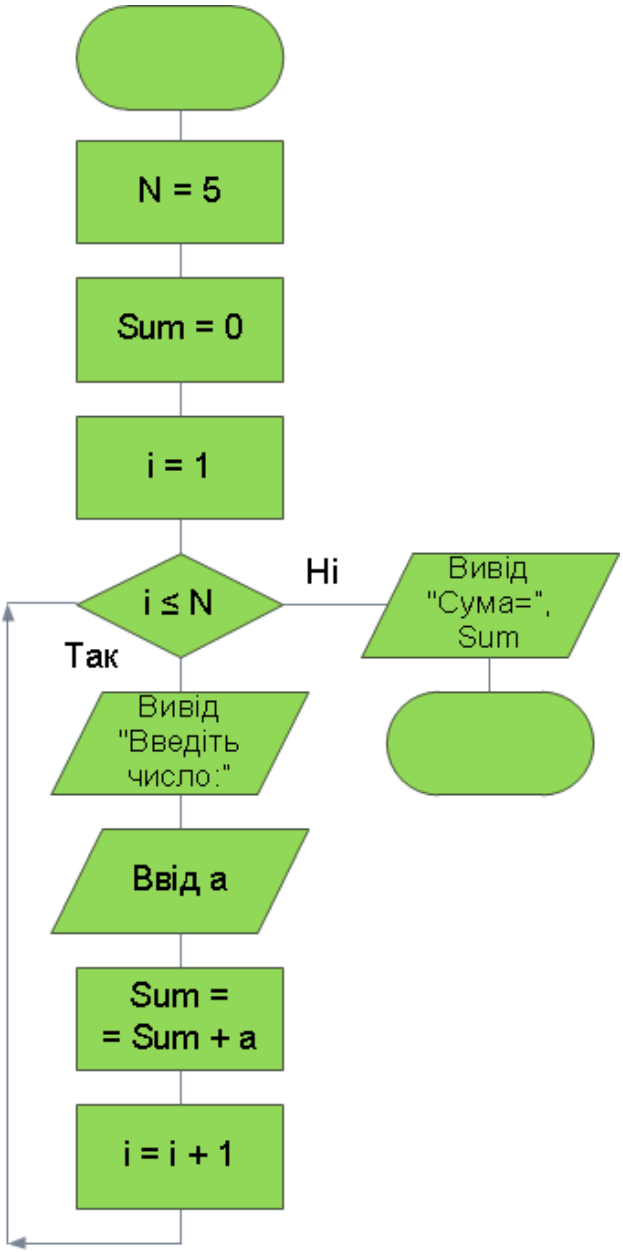

Рисунок 3.11 – Кінцева блок-схема

3.1.2. Приклад 2

Алгоритм другого прикладу представлено у додатку А.

3.2. Блок-схема алгоритму

На рис. 3.12 показана блок-схема алгоритму тренажеру у загальному випадку. На рис. Б.1-Б.3 показано кроки 1-3 прикладу 1.

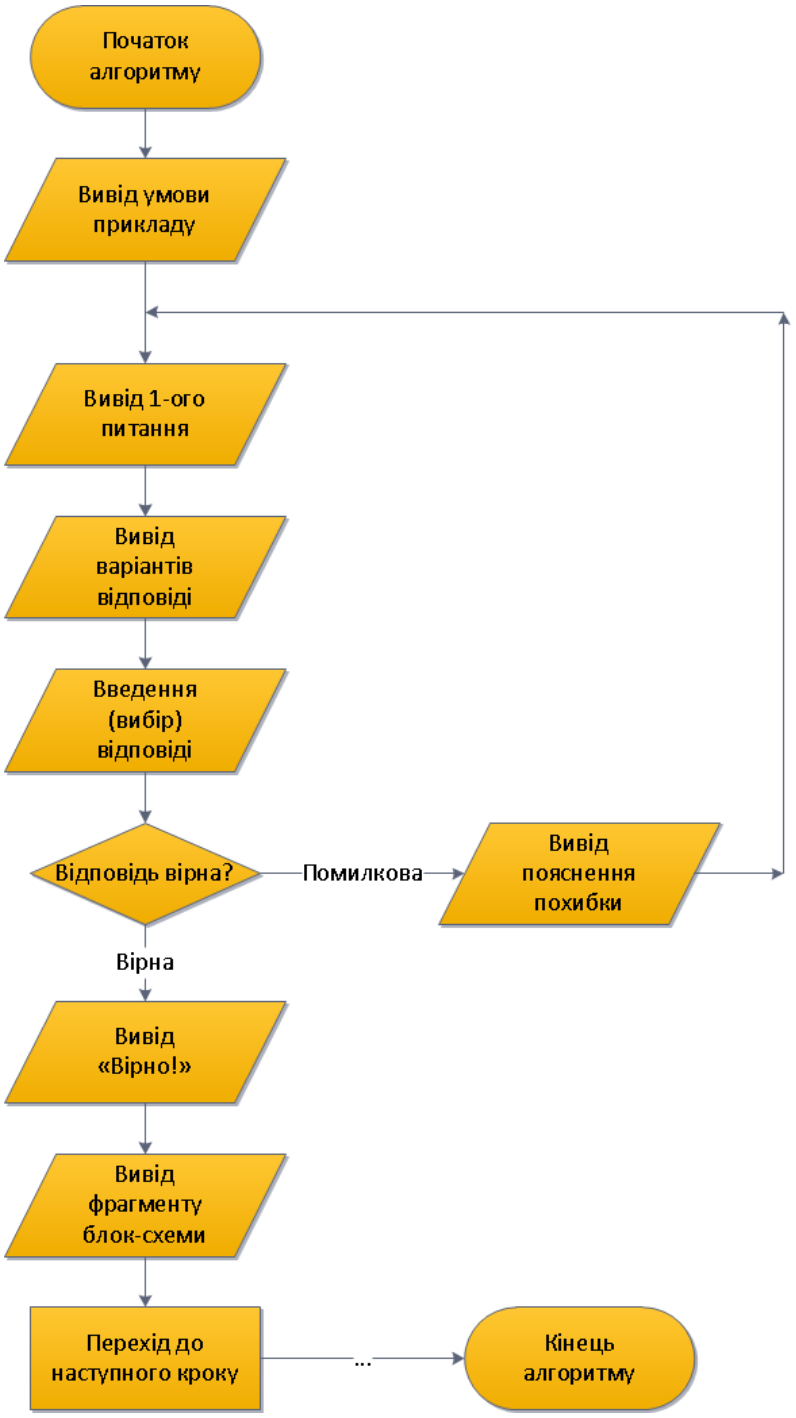

Рисунок 3.12 – Блок-схема алгоритму тренажера у загальному виді

4. ПРАКТИЧНА ЧАСТИНА

4.1. Опис програмної реалізації тренажеру

Для програмної реалізації розробленого алгоритму було використано середовище програмування `Borland Builder` разом з мовою `C++.`

Опишемо, як створювалась програма, використавши перший крок тренажеру прикладу 1 (рис. 4.1).

Рисунок 4.1 – Перший крок тренажеру прикладу 1

Ідея кроку полягає в тому, щоб користувач перетягнув потрібну картинку на місце, підсвічене жовтими кольором.

Для реалізації цієї концепції було використано принцип `Drag and Drop` (перетягуй та тягни).

Реалізовуючи цей принцип, слід визначити, що є приймачем, що є джерелом. Джерела – це картинки-альтернативи відповіді (тобто компоненти

Image1, Image2, Image3); приймач – це картинка з жовтим прямокутником (компонент Image4).

Для джерел властивість DragMode (режим перетягування) була змінена на dmAutomatic.

Для приймача було створено два оброблювача подій для подій DragOver (перетягування) та DragDrop (кінець перетягування).

Код події DragDrop:

```
void __fastcall TForm3::Image4DragDrop (TObject *Sender,
TObject *Source, int X, int Y)
{
  ((TImage*)Sender)  ->  Picture  =  ((TImage*)Source)->
Picture;
}
```

Цей код означає, що слід взяти картинку (за картинку відповідає властивість Picture), зображену в компонентові-джерелі (Source), та скопіювати її у компонент-приймач (Sender).

Для того, щоб розуміти, яку відповідь обрав користувач, було оголошено глобальну зміну answer.

```
int answer = 0;
```

Початкове значення змінною – це 0. Цей стан змінної буде сигналізувати, що відповідь не обрана.

Код події DragOver:

```
void __fastcall TForm3::Image4DragOver(TObject *Sender,
TObject *Source, int X, int Y, TDragState State, bool
&Accept)
{
```

```
  if (Source == Image1)
    answer=1;

  if (Source == Image2)
    answer=2;

  if (Source == Image3)
    answer=3;

  if (Source == (TImage*)Source)
    Accept = true;
}
```

В коді аналізується, яку відповідь надав користувач: якщо джерелом виступив компонент `Image1`, то зміна `answer` стала рівною 1 і так далі.

Останні два рядка коду означають, що в якості приймача можуть слугувати лише компоненти, що відносять до виду `Image`. Якщо це так, то компонент `Image` стає приймачем (`Accept = true`).

Перевірка відповіді користувача здійснюється при натисканні кнопки «Далі». Код події:

```
void __fastcall TForm3::BitBtn1Click(TObject *Sender)
{
  if (answer==0)
  {  MessageDlg("Увага!\nОберіть відповідь, перетягнувши символ блок-схеми\nна місце жовтого прямокутника!", mtWarning, TMsgDlgButtons() << mbOK, 0);
     return;
  }
  if (answer==2)
  {
     MessageDlg("Вірно!", mtInformation, TMsgDlgButtons() << mbOK, 0);
```

```
    Form4->Show();
    Form3->Hide();
  }
  else
  {
    MessageDlg("Не вірно!\nПочаток, кінець або зупинка
алгоритму\nвідображаються символом \"термінатор\",\nякий
має форму овалу зі сплюснутими сторонами.", mtError,
TMsgDlgButtons() << mbOK, 0);
  }
}
```

Якщо відповідь не надано (користувач не перетягнув жодну картинку на місце жовтого прямокутника, тобто змінна answer залишилась рівною нулю), то виводимо повідомлення про це. Користувач повинен обрати якусь відповідь.

Якщо відповідь вірна (тут це друга відповідь, тобто answer=2), то виводимо повідомлення про вірність, здійснюємо перехід до наступного вікна (метод Show) та приховуємо поточне вікно (метод Hide).

Якщо відповідь хибна, то виводимо повідомлення з поясненням похибки. Користувач знову повинен вказати відповідь, тобто перетягнути потрібну картинку.

Код програми прикладу 1 розміщено у додатку В.

4.2. Інструкція по роботі з тренажером

Як працює тренажер, видно з рисунків 4.2-4.28, В.1-В.12 (приклад 1), рис. В.13-В.44 (приклад 2).

У першому вікні (рис. 4.2) пропонується обрати один з двох прикладів. Далі виводиться умова: код програми з циклічним оператором (рис. 4.3). Слід створити блок-схему.

На кожному кроці тренажеру можна повернутися до умови (див., наприклад, рис. 4.4), натиснувши кнопку «Умова».

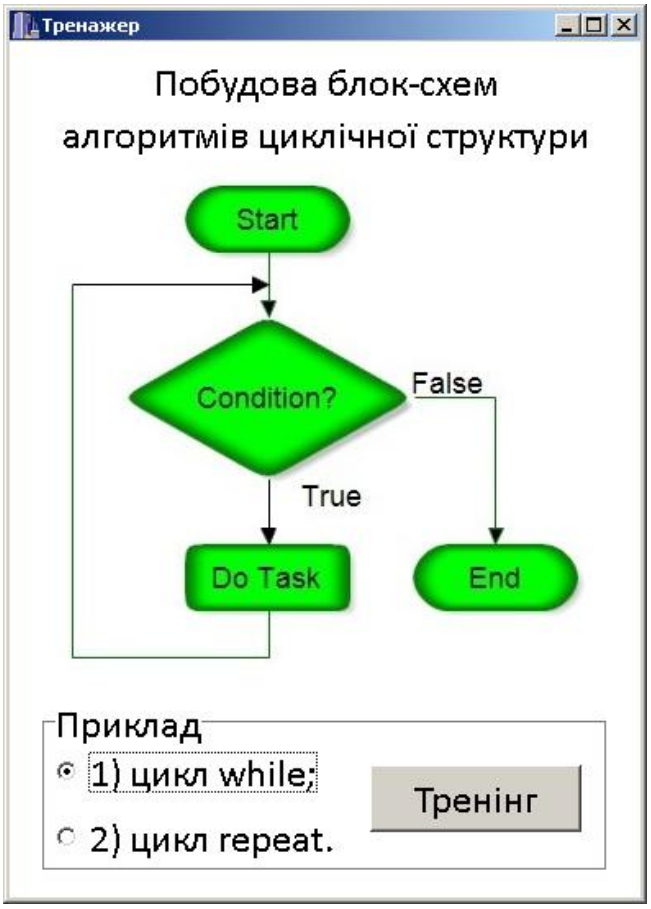

Рисунок 4.2 – Початкове вікно програми

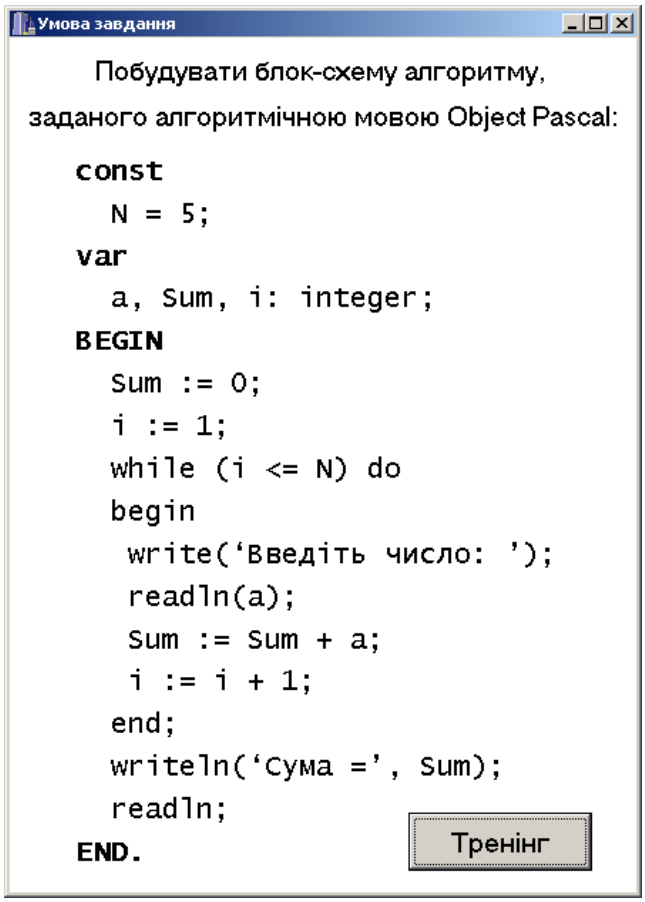

Рисунок 4.3 – Умова завдання

Рисунок 4.4 – Перше питання

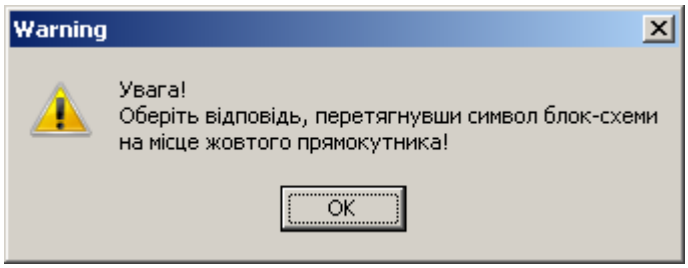

Рисунок 4.5 – Попередження про відсутність відповіді

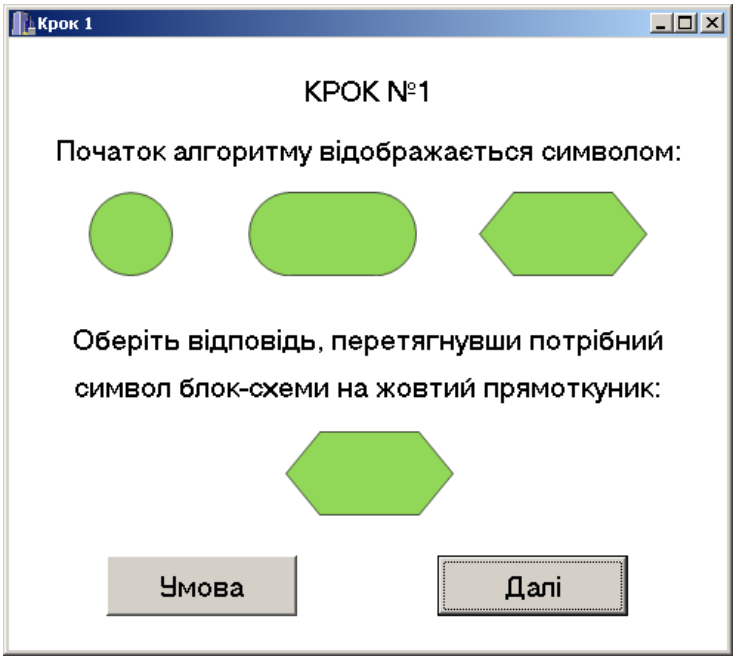

Рисунок 4.6 – Перше питання з хибною відповіддю

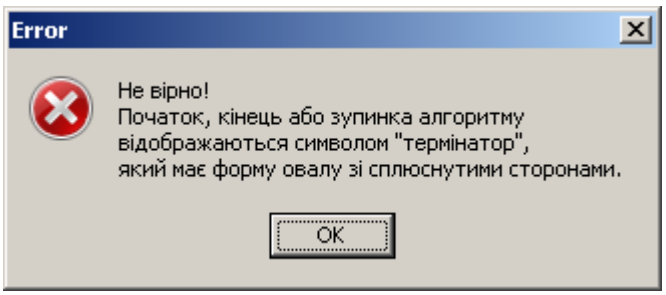

Рисунок 4.7 – Пояснення похибки (при першому питанні)

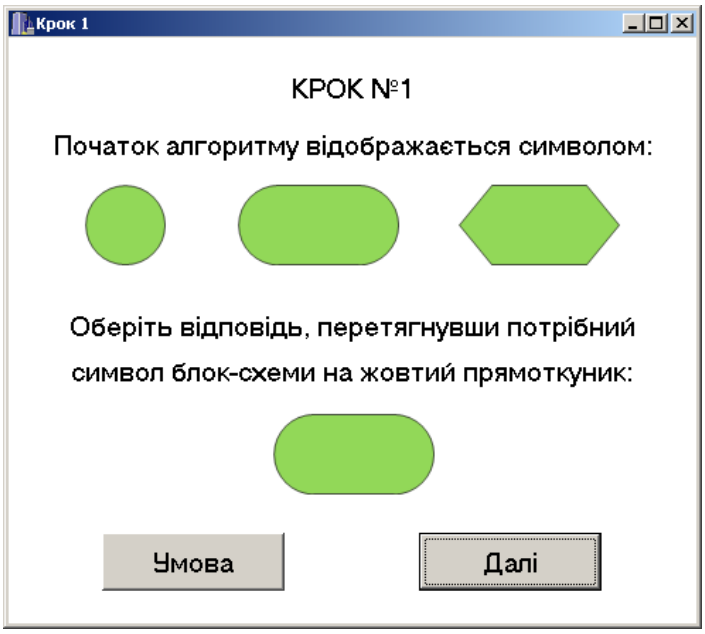

Рисунок 4.8 – Перше питання з вірною відповіддю

Рисунок 4.9 – Повідомлення про правильність відповіді

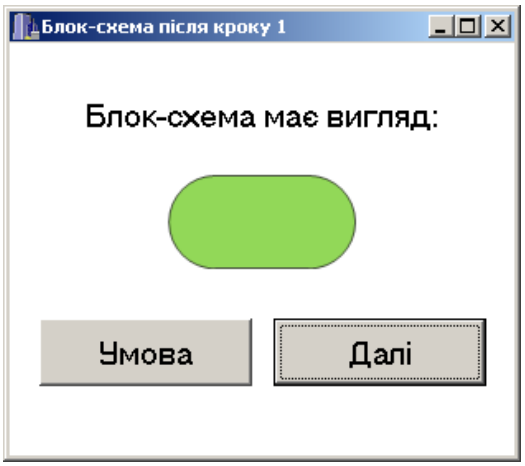

Рисунок 4.10 – Блок-схема після першого питання

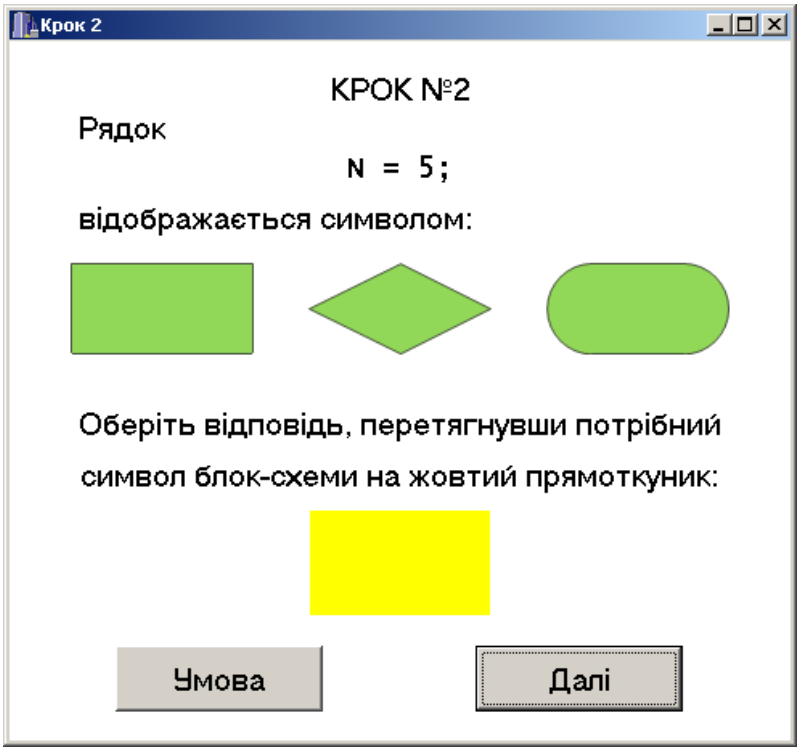

Рисунок 4.11 – Друге питання

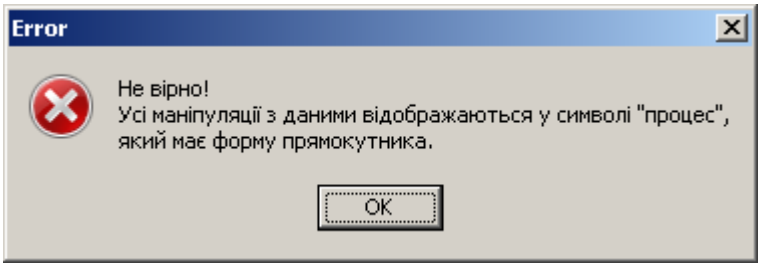

Рисунок 4.12 – Пояснення похибки (при другому питанні)

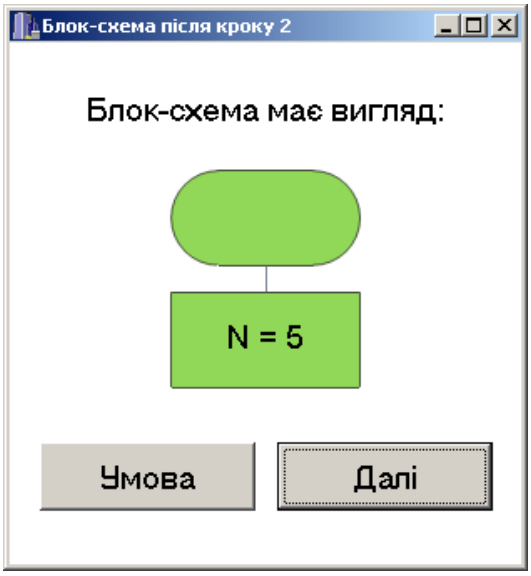

Рисунок 4.13 – Блок-схема після другого питання

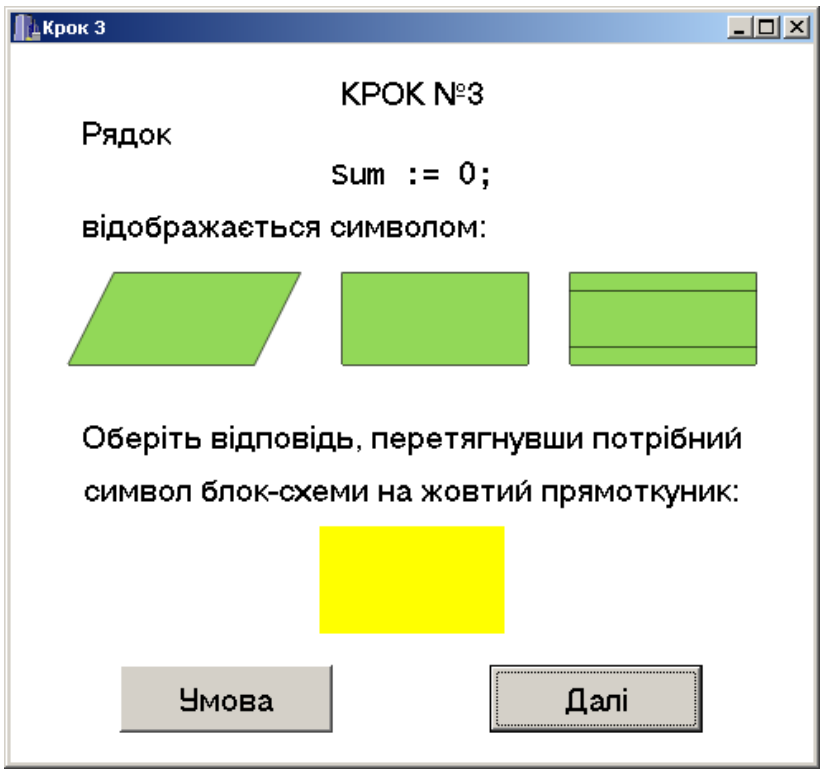

Рисунок 4.14 – Третє питання

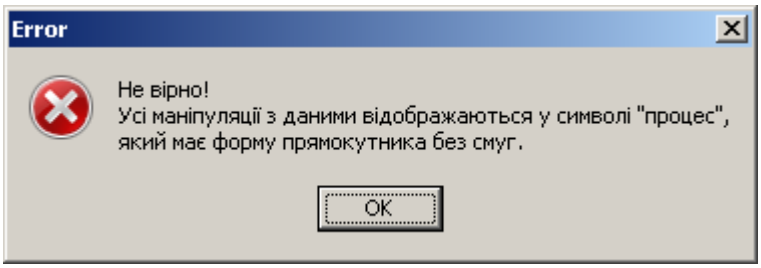

Рисунок 4.15 – Пояснення похибки (при третьому питанні)

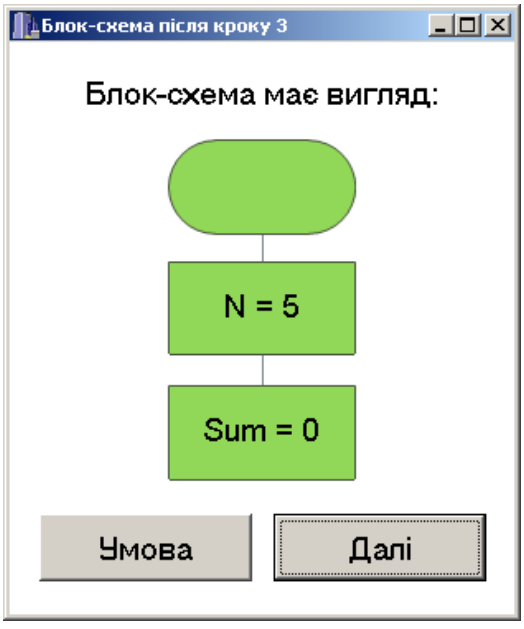

Рисунок 4.16 – Блок-схема після третього питання

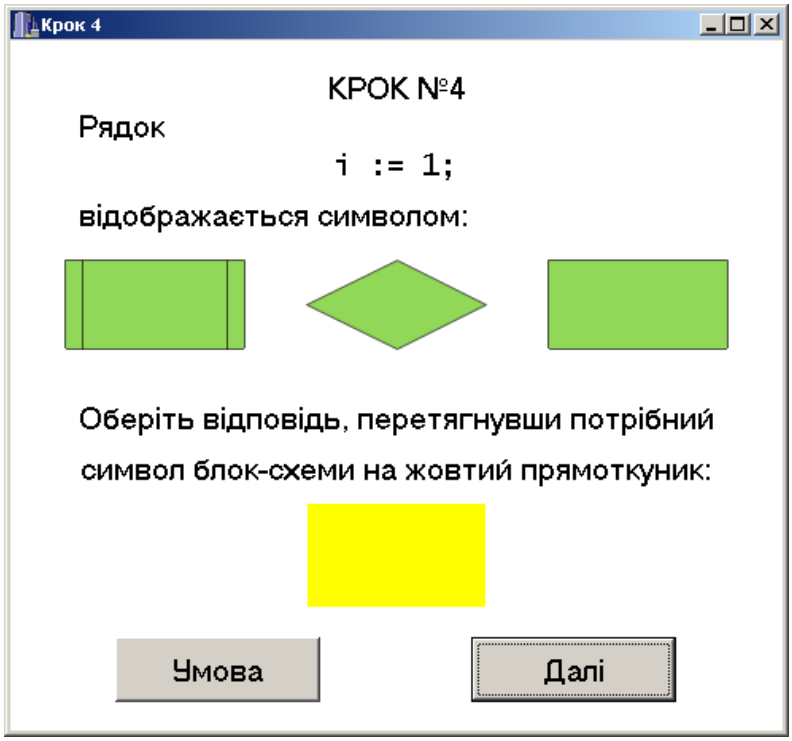

Рисунок 4.17 – Четверте питання

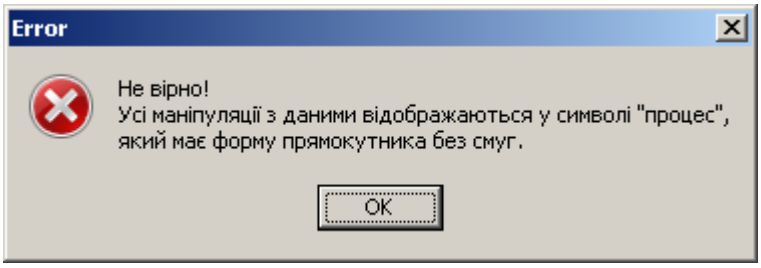

Рисунок 4.18 – Пояснення похибки (при четвертому питанні)

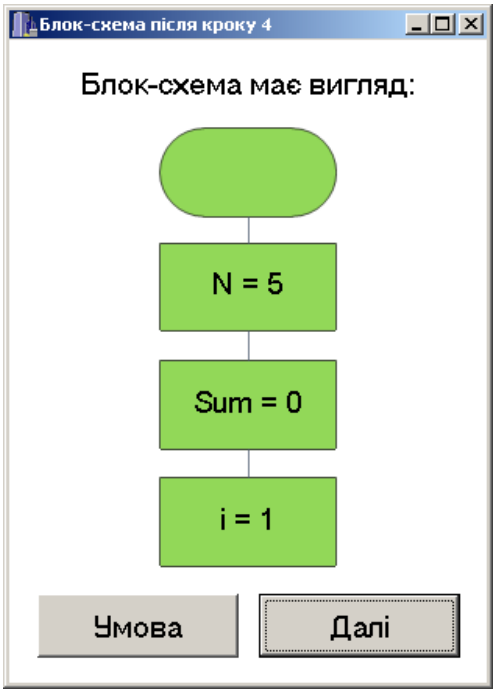

Рисунок 4.19 – Блок-схема після четвертого питання

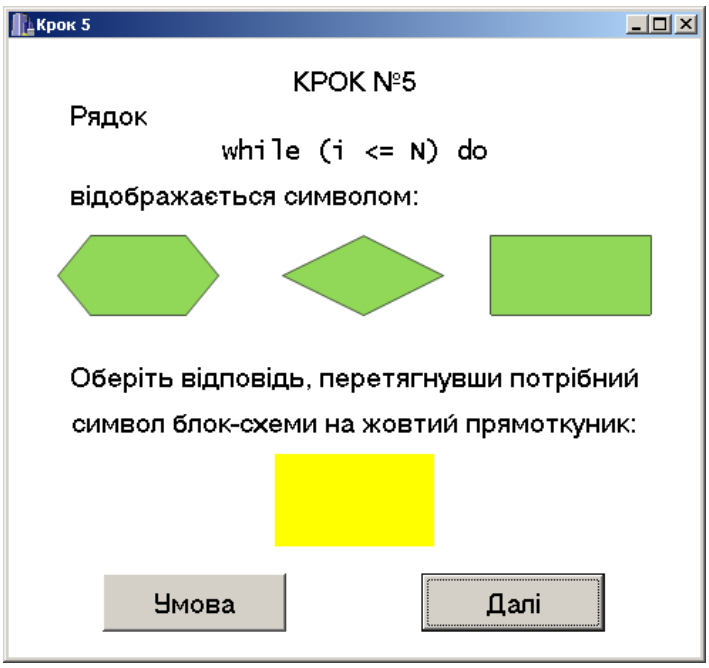

Рисунок 4.20 – П’яте питання

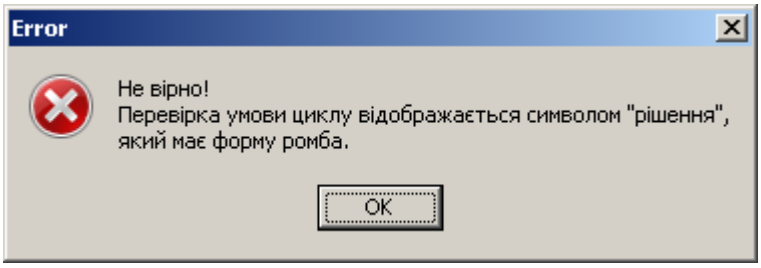

Рисунок 4.21 – Пояснення похибки (при п’ятому питанні)

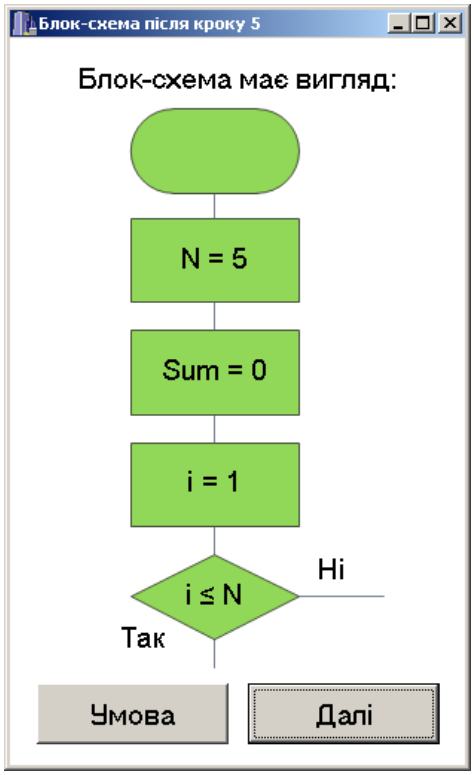

Рисунок 4.22 – Блок-схема після п’ятого питання

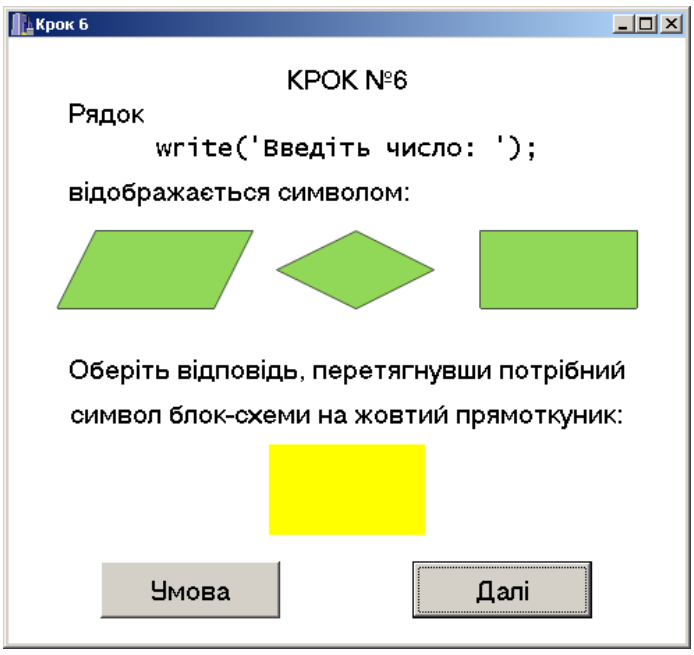

Рисунок 4.23 – Шосте питання

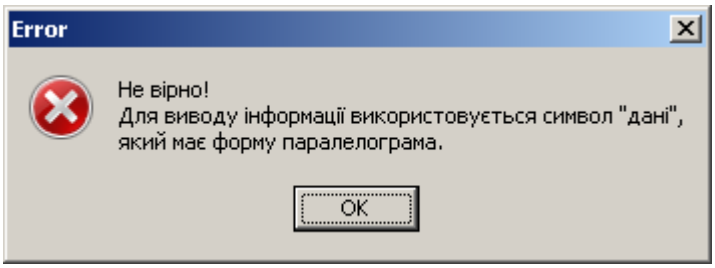

Рисунок 4.24 – Пояснення похибки (при шостому питанні)

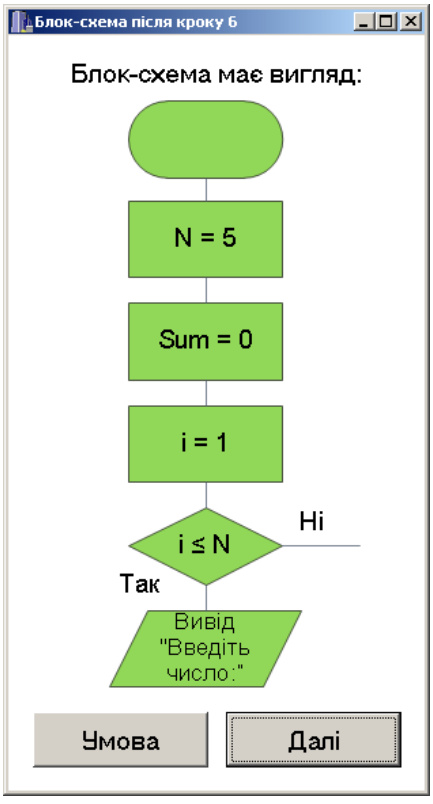

Рисунок 4.25 – Блок-схема після шостого питання

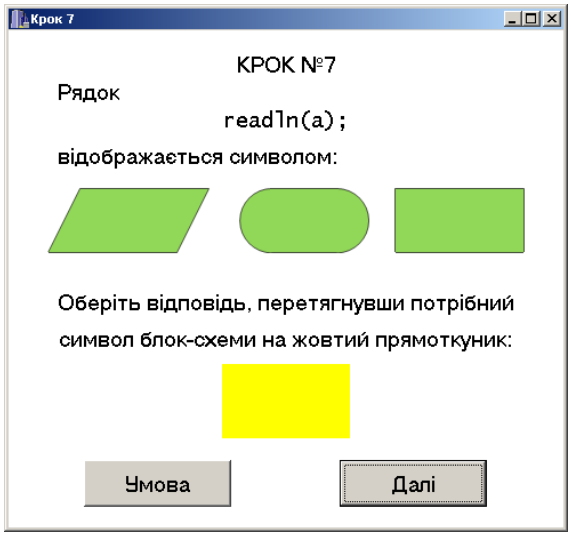

Рисунок 4.26 – Сьоме питання

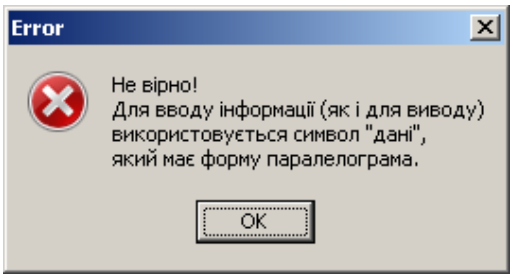

Рисунок 4.27 – Пояснення похибки (при сьомому питанні)

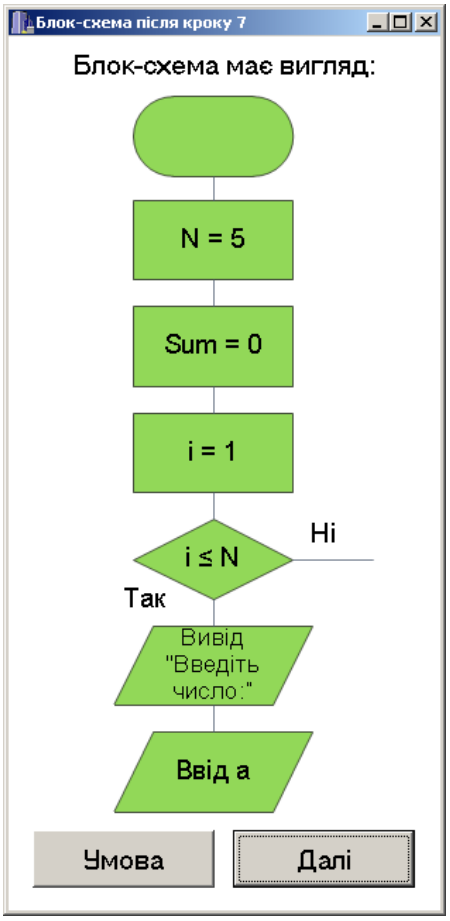

Рисунок 4.28 – Блок-схема після сьомого питання

На кожному кроці слід обрати вірну відповідь-рисунок і перетягнути обраний символ на прямокутник жовтого кольору, натиснути кнопку «Далі».

Якщо відповіді не обрано, а кнопка «Далі» натиснута, то з’являється попередження про некоректність дій користувача (рис. 4.5).

Якщо відповідь вірна (рис. 4.8), то з’являється відповідне повідомлення (рис. 4.9).

При помилці з’являється повідомлення з поясненням (див., наприклад, рис. 4.7). Користувач повинен виправити помилку і знову натиснути кнопку «Далі».

Після надання вірної відповіді з’являється блок-схема з цим символом (рис. 4.10). Тут також є можливість переглянути умови, натиснувши однойменну кнопку.

Після остатнього кроку – це вже буде повна блок-схема прикладу (рис. В.12).

ВИСНОВКИ

У випусковому проекті було досліджено тему побудови блок-схем алгоритмів циклічної структури на прикладі алгоритмів, записаних алгоритмічною мови Object Pascal.

Як відомо, в мові Pascal існують три циклічні структури:

- ✓ цикл while … do;
- ✓ цикл repeat … until;
- ✓ цикл for.

У проекті було взято приклад програми з використанням циклу while … do, в якій знаходиться сума п’яти чисел. Цю ж задачу було програмно реалізовано для другого прикладу, але з використанням циклу repeat … until.

Приклади було перевірено в середовищі Delphi. Впевнилися, що програми працюють вірно.

Після цього було побудовано блок-схеми алгоритмів обох прикладів.

Далі було розроблено алгоритми тренажеру для двох прикладів, намальовано блок-схему алгоритму тренажеру для першого прикладу.

Потім було здійснено програмну реалізацію тренажеру з використанням мови C++ та середовища Borland Builder.

Результати випускової роботи були опубліковані у збірнику матеріалів науково-практичного семінару «Комп’ютерні науки і прикладна математика (КНіПМ-2021)» [12].

Тренажер передано для впровадження у дистанційний курс «Інформатика. Частина 1» Полтавського університету економіки і торгівлі, що засвідчує акт впровадження.

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Схемы алгоритмов, данных и систем. Условные обозначения и правила выполнения: ГОСТ 19.701-90. – М.: Из-во стандартов, 1990. – 25 с.

2. Microsoft Visio 2013. – [Електронний ресурс]. – Режим доступу: http://office.microsoft.com/ru-ru/visio/FX103791368.aspx. – Назва з екрану.

3. Ємець О. О. Дистанційний курс Полтавського університету економіки та торгівлі «Інформатика. Частина 1» для студентів спеціальності «Комп'ютерні науки» / О. О. Ємець. – [Електронний ресурс].

4. Ємець О. О. Про розробку тренажерів для дистанційних курсів кафедрою ММСІ ПУЕТ / О. О. Ємець // Інформатика та системні науки (ІСН-2015): матеріали VI Всеукр. наук.-практ. конф. за міжн. участю (м. Полтава, 19-21 березня 2015 р.) / за ред. Ємця О. О. – Полтава: ПУЕТ, 2015. – С. 152-161. – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/2488.

5. Ємець О. О. Про розробку тренажерів для дистанційних курсів навчальних дисциплін спеціальності Комп`ютерні науки в ПУЕТ / О.О. Ємець, Є.М. Ємець, Ол-ра О. Ємець // Економіка сьогодні: проблеми моделювання та управління: матеріали X Міжн. наук.-практ. інтернет-конф. (м. Полтава, 19-20 листопада 2020 р.). – Полтава: ПУЕТ, 2021. – С. 365-370.

6. Мордасова І. В. Тренажер з теми «Побудова блок-схем алгоритмів розгалуженої структури» дистанційного навчального курсу «Інформатика» та розробка його програмного забезпечення / І. В. Мордасова, Ол-ра О. Ємець // Комп'ютерні науки і прикладна математика (КНіПМ-2019): матеріали наук.-практ. семінару. Випуск 3. / За ред. Ємця О. О. – Полтава: Кафедра ММСІ ПУЕТ, 2019. – С. 35-37. – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/7037.

7. Недбайло Я. І. Пояснювальна записка до дипломної роботи на тему «Тренажер з теми «Побудова блок-схем алгоритмів циклічної структури на прикладі циклу «do…while» дистанційного навчального курсу «Інформатика» та

розробка його програмного забезпечення» / Я. І. Недбайло. – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/8906.

8. Сузанська А. О. Тренажер «Побудова блок-схем алгоритмів розгалуженої структури» / А. О. Сузанська, Є. М. Ємець, О. О. Ємець // Комп'ютерні науки і прикладна математика (КНіПМ-2020): матеріали наук.-практ. семінару. Випуск 5. / За ред. Ємця О. О. – Полтава: Кафедра ММСІ ПУЕТ, 2019. – С. 56-61. – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/8906.

9. Хрупа О. І. Розробка програмного забезпечення з теми «Турнірне сортування» дистанційного навчального курсу «Алгоритми та структури даних»/ О. І. Хрупа, Ол-ра О. Ємець // Комп'ютерні науки і прикладна математика (КНіПМ-2019): матеріали наук.-практ. семінару. Випуск 3. / За ред. Ємця О. О. – Полтава: Кафедра ММСІ ПУЕТ, 2019. – С. 43-45. – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/7039.

10. Шакуро В. Є. Розробка програмного забезпечення з теми «Побудова блок-схема алгоритмів лінійної структури» дистанційного курсу «Інформатика»/ В. Є. Шакуро, О. О. Ємець // Комп'ютерні науки і прикладна математика (КНіПМ-2019): матеріали наук.-практ. семінару. Випуск 3. / За ред. Ємця О. О. – Полтава: Кафедра ММСІ ПУЕТ, 2019. – С. 40-42. – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/7038.

11. Ємець О. О. Методичні рекомендації до виконання бакалаврської роботи для студентів спеціальності 122 «Комп`ютерні науки та інформаційні технології» освітня програма «Комп`ютерні науки» галузь знань – 12 «Інформаційні технології» / О. О. Ємець. – Полтава: ПУЕТ, 2017. – 71 с.

12. Бибка Б. М. Тренажер «Побудова блок-схем алгоритмів циклічної структури на прикладі циклу while» / Б. М. Бибка, О. О. Ємець // Комп'ютерні науки і прикладна математика (КНіПМ-2021): матеріали наук.-практ. семінару. Випуск 6. / За ред. Ємця О. О. – Полтава: Кафедра ММСІ ПУЕТ, 2021. – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/10039.