ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ БІЗНЕСУ
ТА СУЧАСНИХ ТЕХНОЛОГІЙ

ФОРМА НАВЧАННЯ ЗАОЧНА

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 РОЗГАЛУЖЕНОЇ СТРУКТУРИ» ДИСТАНЦІЙНОГО НАВЧАЛЬНОГО КУРСУ «ПРОГРАМУВАННЯ ІІ» ТА РОЗРОБКА ЙОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

зі спеціальності 122 «Комп'ютерні науки та інформаційні технології»

Виконавець роботи Сузанська Аделіна Олександрівна _____ «___»_____2021 р.
(підпис)

Науковий керівник к.ф.-м.н., проф., Ємець Єлизавета Михайлівна
_____ «___»_____ 2021 р.
(підпис)

ПОЛТАВА 2021 р.

РЕФЕРАТ

Записка: 45 с., 57 рис., 4 додатки (на 27 сторінках), 6 джерел.

Предмет розробки – тренажер з теми «Побудова блок-схем алгоритмів розгалуженої структури» для дистанційного навчального курсу Полтавського університету економіки і торгівлі «Програмування ІІ».

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

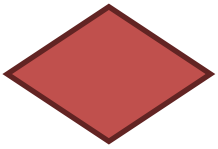
Методи розробки – мова програмування С++, середовище програмування Borland Builder, пакет інженерної графіки MS Visio.

Для двох прикладів створено алгоритми тренажерів, блок-схема та програмна реалізація.

Ключові слова: АЛГОРИТМ, БЛОК-СХЕМА, ТРЕНАЖЕР.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні позначення, символи, одиниці, скорочення, терміни	**Пояснення умовних позначень, символів, одиниць, скорочень, термінів**
Алгоритм	Це чіткий опис послідовних дій, які необхідно виконати для розв’язання задачі.
Блок-схема	Це наочне графічне зображення алгоритму, в якому окремі етапи алгоритму зображуються за допомогою різних геометричних фігур, а зв’язки між етапами зображуються за допомогою стрілок, що з’єднують ці фігури.
	Символ «термінатор». Показує початок, зупинку або кінець блок-схеми.
	Символ «дані». Відображає введення або виведення даних.
	Символ «процес». Відображає обробку даних будь-якого типу.
	Символ «рішення». Відображає рішення або функцію типу «перемикач», яка має один вхід і ряд альтернативних виходів, з яких лише один може бути активізованим після обчислення або перевірки умов, зазначених в середині цього символу. Відповідні результати обчислень (або позначки виконання/невиконання умов) мають бути записані поруч з лініями, які відображають ці шляхи.
	Символ «лінія». Показує потік даних або керування. За необхідності або для підвищення зручності читання блок-схем можуть бути добавлені стрілки-вказівники.

ВСТУП

Актуальність теми підтверджується тим, що слід розробити абсолютно новий засіб навчання, який корисний при дистанційній формі навчання.

Об’єкт розробки у проекті – це тренажер.

Предмет розробки – тренажер з теми «Побудова блок-схем алгоритмів розгалуженої структури» для дистанційного навчального курсу Полтавського університету економіки і торгівлі «Програмування ІІ».

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

Для реалізації мети слід виконати такі задачі: 1) ознайомитися з тренажерами схожої тематики та проаналізувати їх; 2) ознайомитися зі стандартом по блок-схемам та теоретичним матеріалом зі створення блок-схем алгоритмів та умовних операторів мови С++; 3) підібрати або розробити приклад(и) тренажеру, розробити алгоритм та блок-схема тренажеру(ів), створити програму реалізацію алгоритму тренажеру(ів).

Використані при розробці програми методи – мова програмування С++, середовище програмування Borland Builder, пакет інженерної графіки MS Visio.

Готовність до впровадження – програмний продукт повність завершено.

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

Обсяг пояснювальної записки – 45 сторінок.

1. ПОСТАНОВКА ЗАДАЧІ

Мета дипломного проектування – розробити тренажер з теми «Побудова блок-схем алгоритмів розгалуженої структури» дистанційного навчального курсу «Програмування ІІ».

В зв’язку з цим слід:

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

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

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

4) два-чотири підібраних прикладів реалізувати у бакалаврській роботі. Для цього створити блок-схеми для алгоритмів прикладів, розробити алгоритми тренажерів та їх блок-схеми, створити програми тренажерів. У пояснювальній записці описати створені програми.

Оскільки, тренажер призначений для дистанційного навчання за дистанційними курсами Полтавського університету економіки і торгівлі, то дотримуватися позначок та термінології, яка зазначена у відповідних темах відповідних дистанційних курсів [1, 2].

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Огляд розробок, аналогічних темі випускової роботи

Розглянемо роботи, які навчають студентів побудові блок-схем.

Серед таких робот можна виділити **тренажер «Побудова блок-схем алгоритмів розгалуженої структури»** (на прикладі мови Pascal), створений студенткою спеціальності «Інформатика» ПУЕТ Мордасовою І. у 2019 р.

Тренажер (рис. 2.1-2.6) містить один приклад – задано алгоритм кодом (рис. 2.2), створити за ним блок-схему.

На рис. 2.3 показано перше питання. Коли користувач вказує правильну відповідь, то йде перехід далі, коли невірну, то з’являється вікно як на рис. 2.4. Помилку слід усунути.

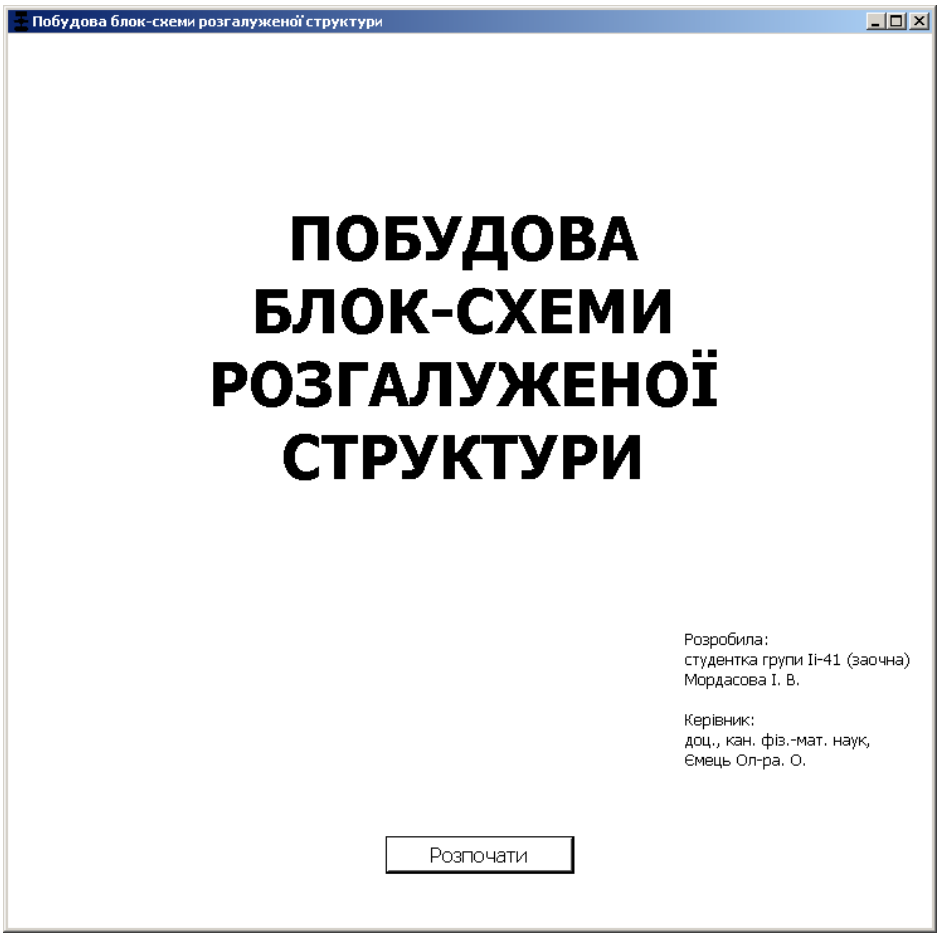


Рисунок 2.1 – Тренінг з побудови блок-схем розгалужених алгоритмів

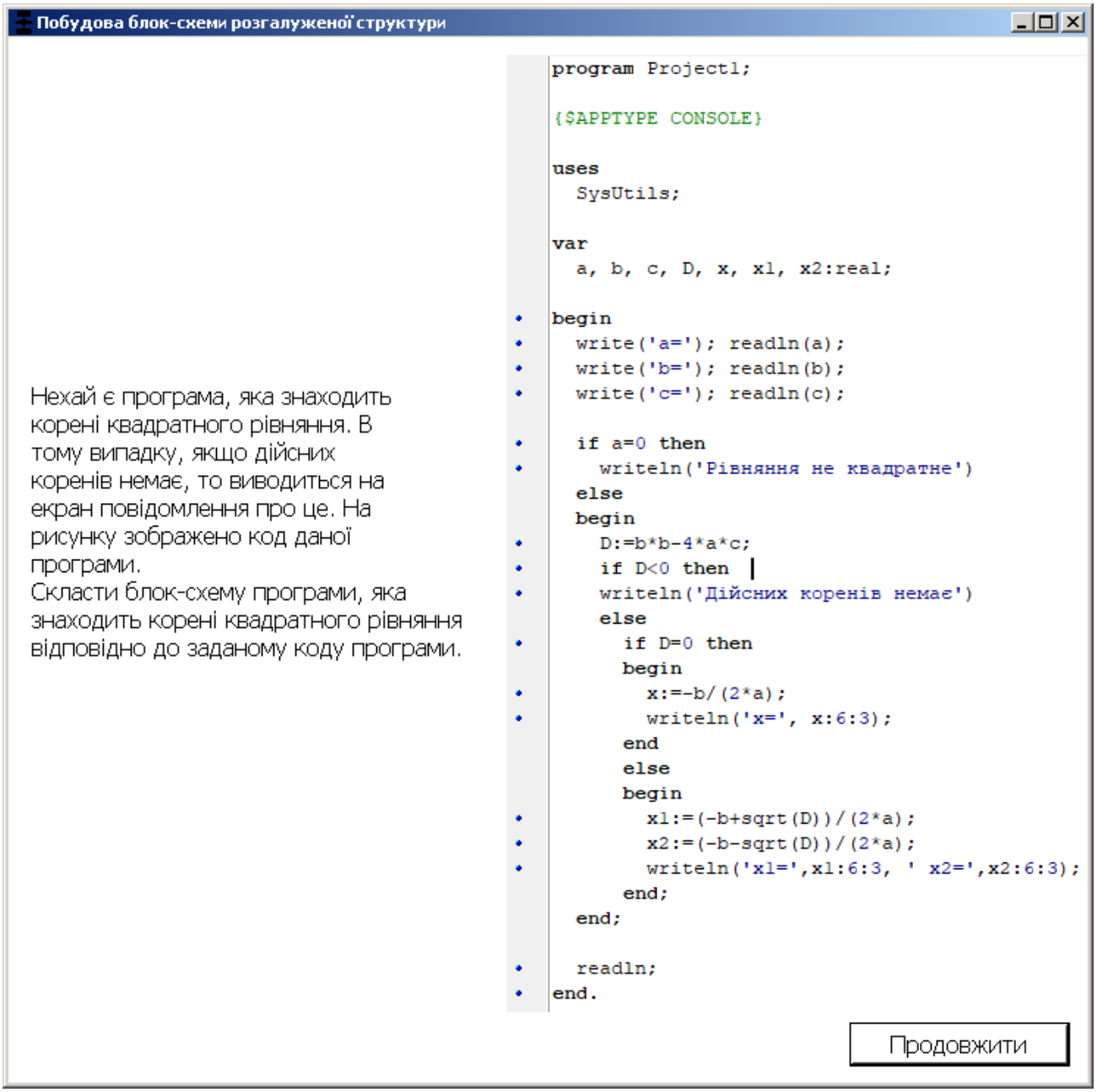


Рисунок 2.2 – Умова прикладу

Після кожного кроку з’являється фрагмент блок-схеми, наприклад, як на рис. 2.5.

В кінці тренінгу показується повністю готова блок-схеми, побудована за умовою прикладу (рис. 2.6).

Можна також виокремити **тренажер «Побудова блок-схем алгоритмів циклічної структури на пркиладі циклу for»** (на прикладі мови Pascal), створений студентом спеціальності «Інформатика» ПУЕТ Гмизою Б. у 2019 р.

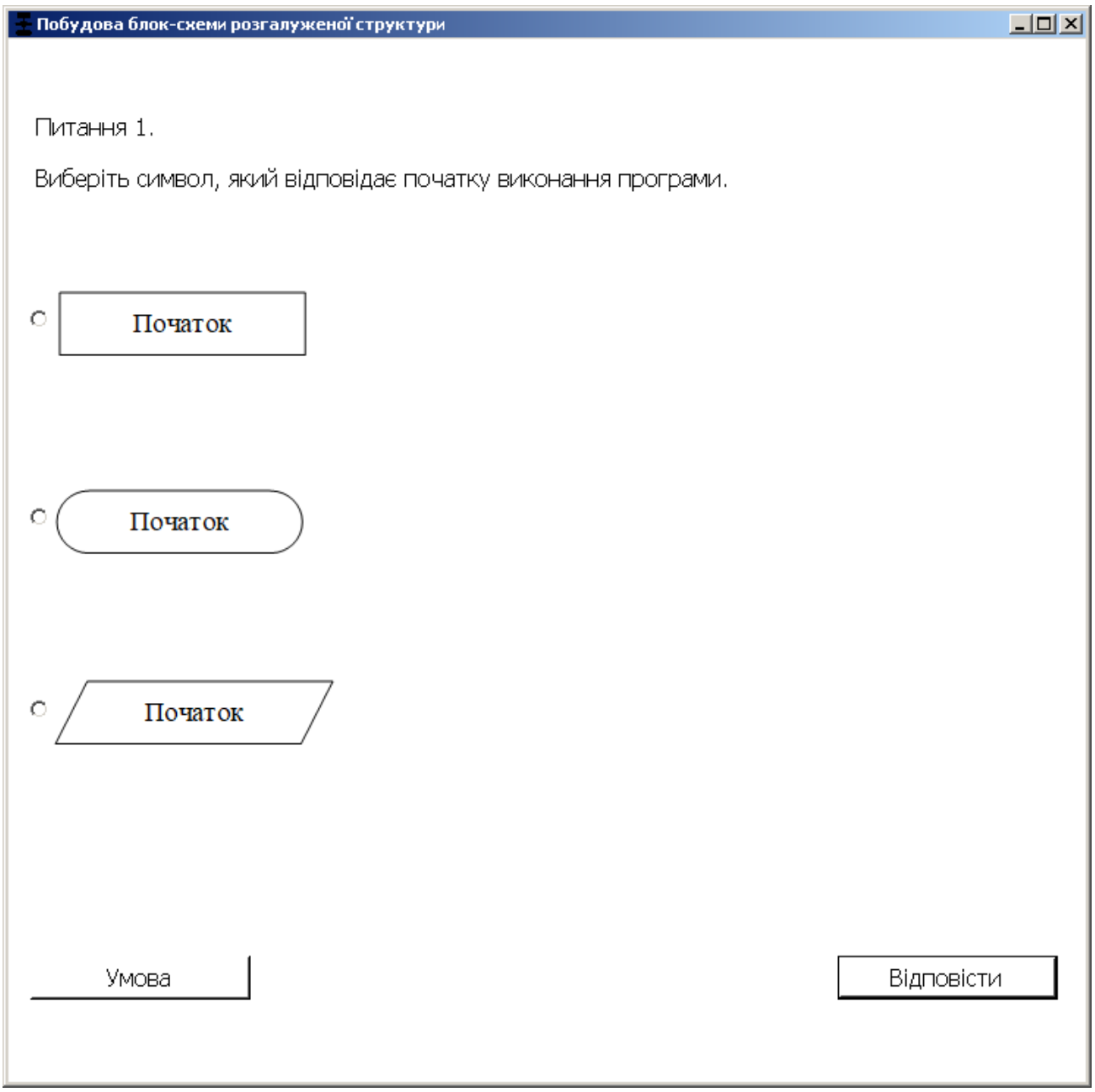


Рисунок 2.3 – Питання 1

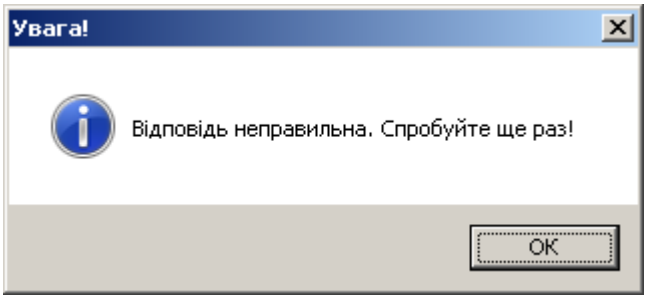


Рисунок 2.4 – Обробка невірної відповіді

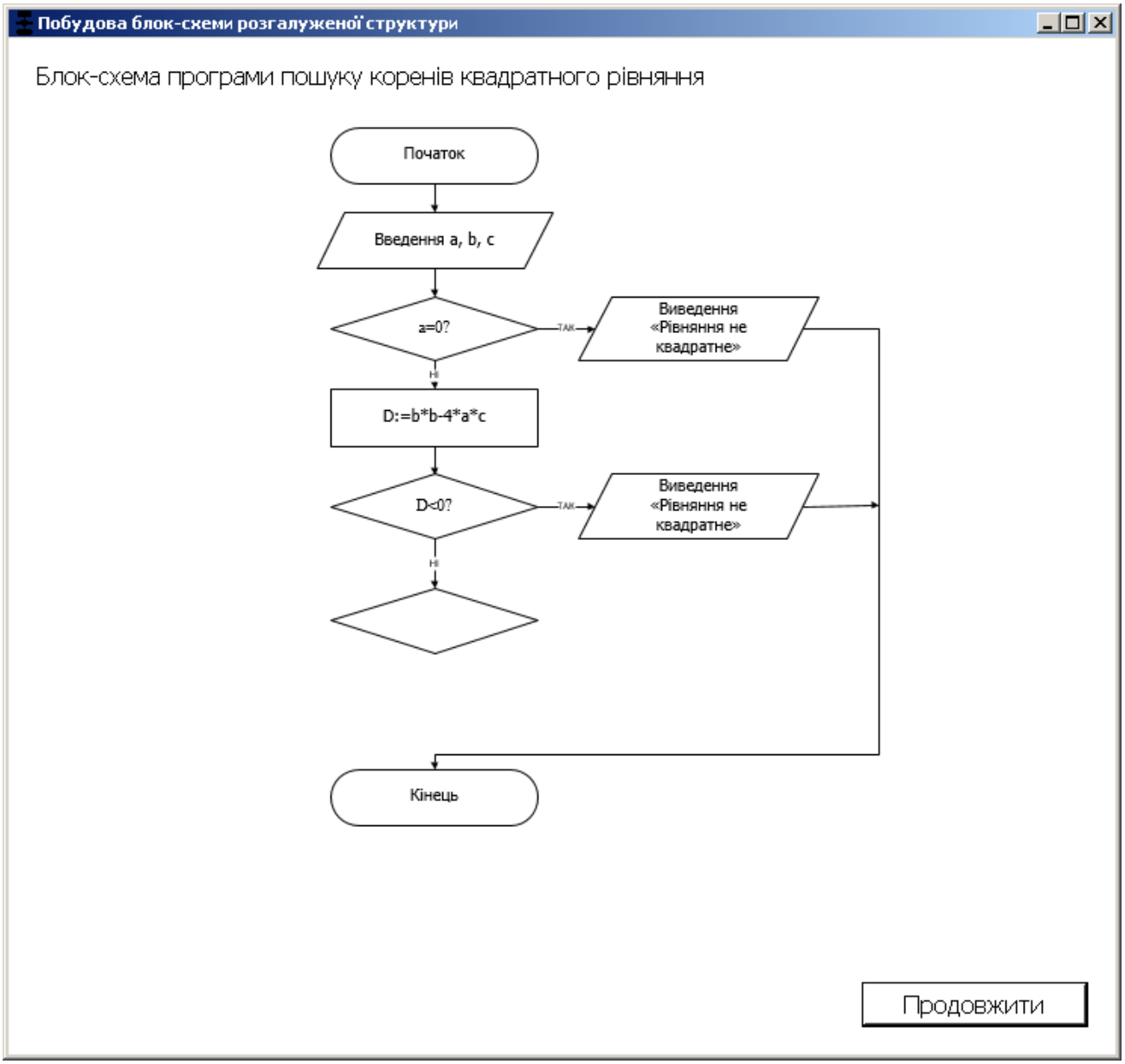


Рисунок 2.5 – Фрагмент побудованої після чергового кроку блок-схеми

В основі роботи цього симулятору (рис. 2.7-2.11) така ж ідея як і в першому. Є один приклад, в якому алгоритм задається кодом мови програмування Pascal (рис. 2.8). Слід побудувати блок-схему.

На рис. 2.9 показано одне з питань тренінгу. Після натиснення кнопки «Перевірити», кольором вказується вірні та невірні відповіді (рис. 2.10). Червоним – хибні, зеленим – вірні. Кнопка «Перевірити» змінює назву на «Наступне питання». Натиснувши кнопку вдруге, відбувається перехід до наступного кроку.

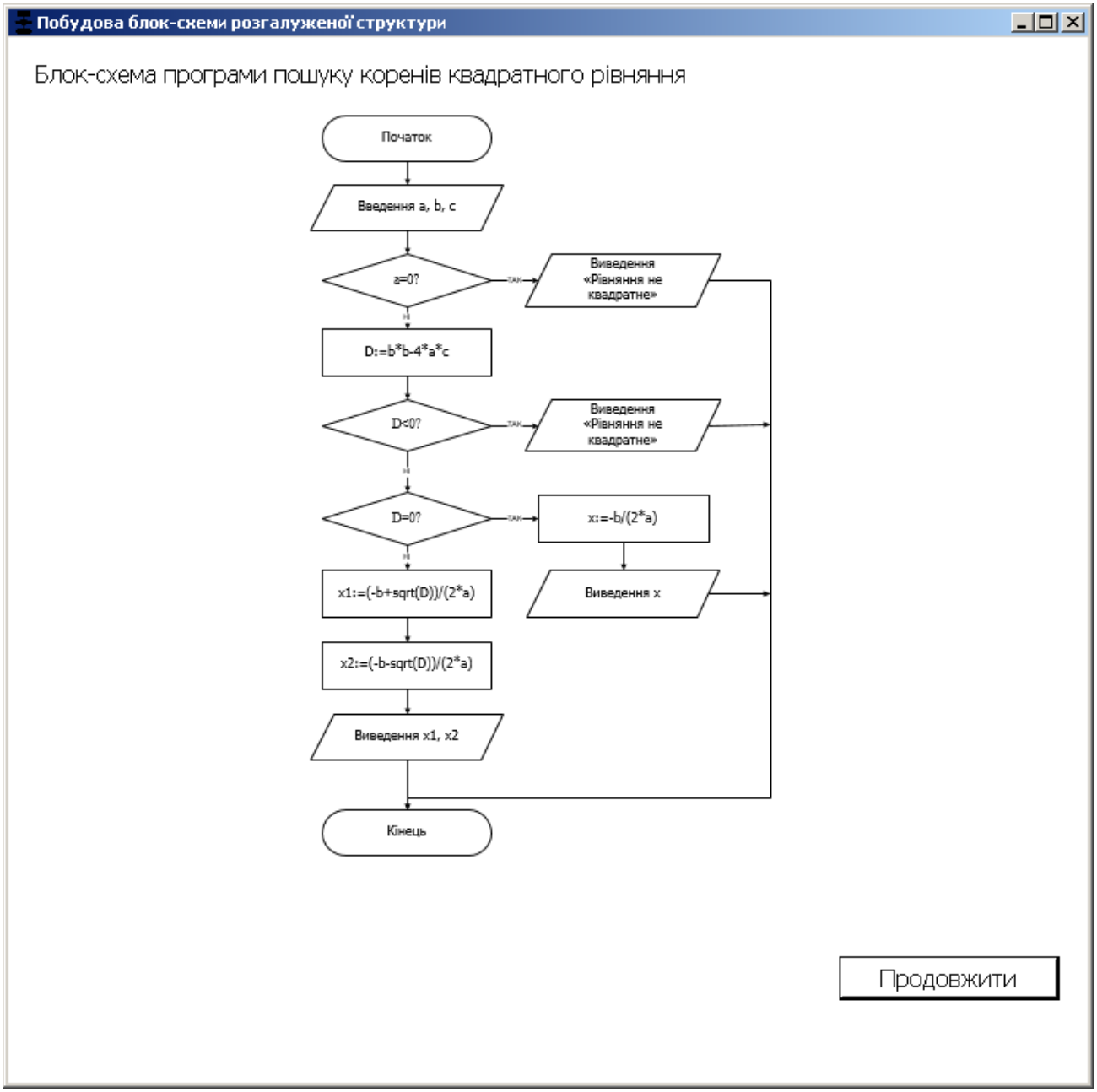


Рисунок 2.6 – Результат

Побудований фрагмент блок-схеми тут не з’являється, лише у кінці тренінгу показується повна блок-схеми, створена за умовою прикладу (рис. 2.11).

Третя робота **«БлокСхемник»** (рис. 2.12-2.19) була знайдена в мережі інтернет і є візуалізатором, який за кодом будує блок-схему. Програма може намалювати блок-схему за кодом трьох мов: Pascal, C, C++.

На рис. 2.12-2.13 показано робота додатку для двох прикладів, на базі яких в цій випусковій роботі буде будуватися тренажер. Але побудовані БлокСхемником рисунки не відповідають стандарту.

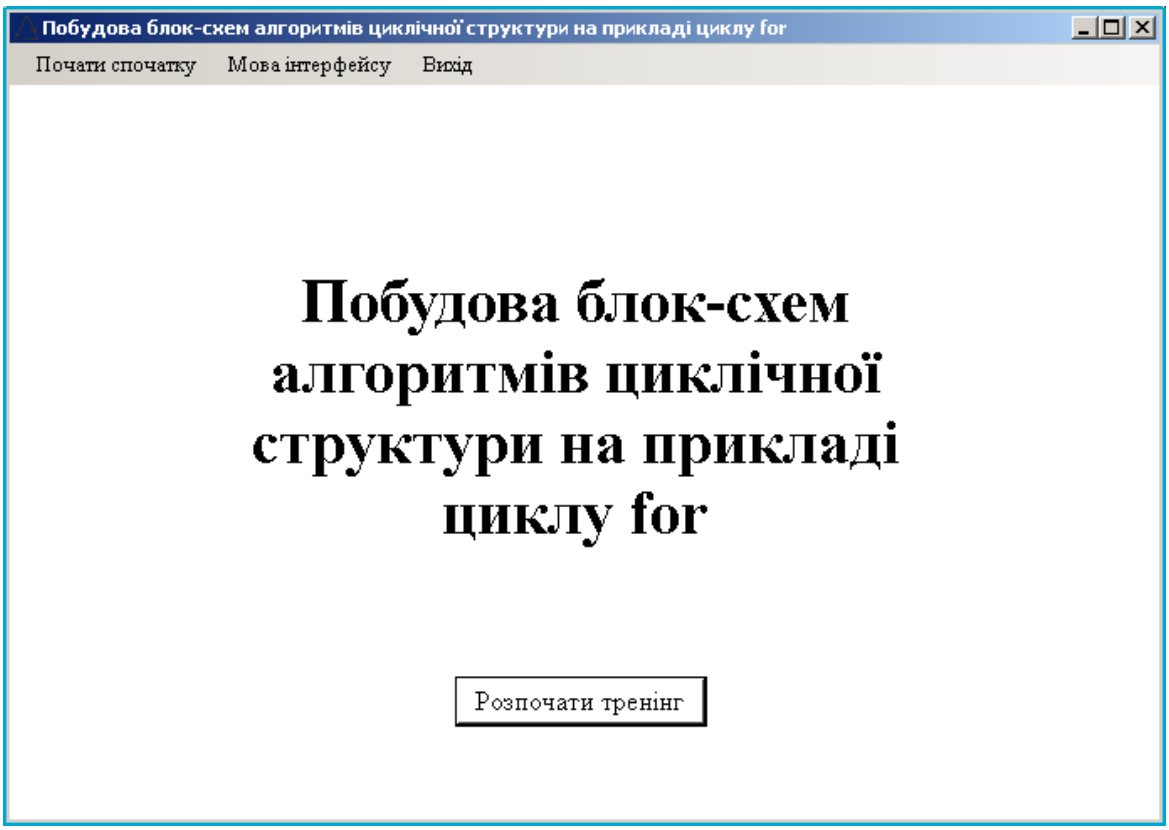


Рисунок 2.7 – Тренінг з побудови блок-схем алгоритмів з for

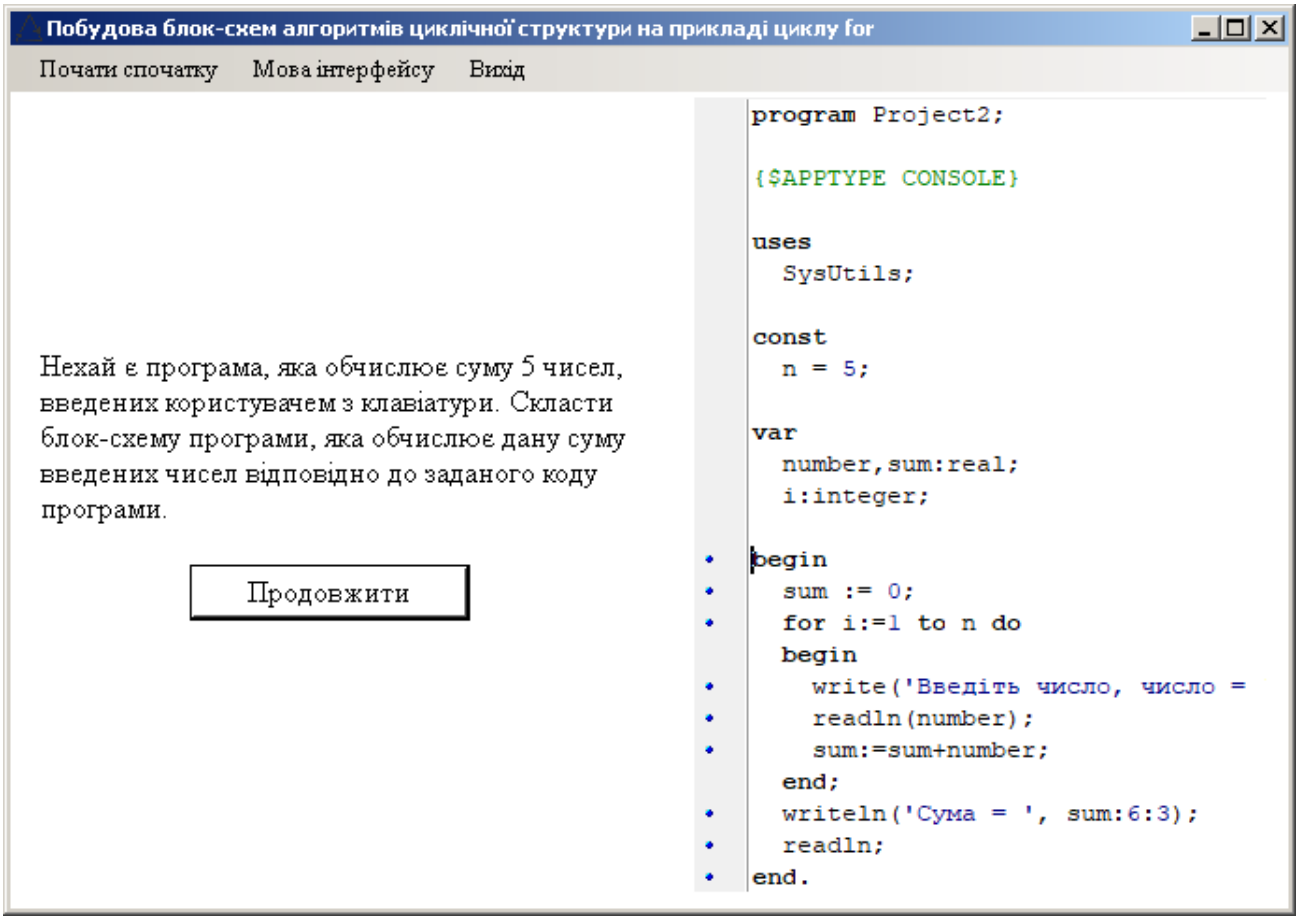


Рисунок 2.8 – Умова прикладу

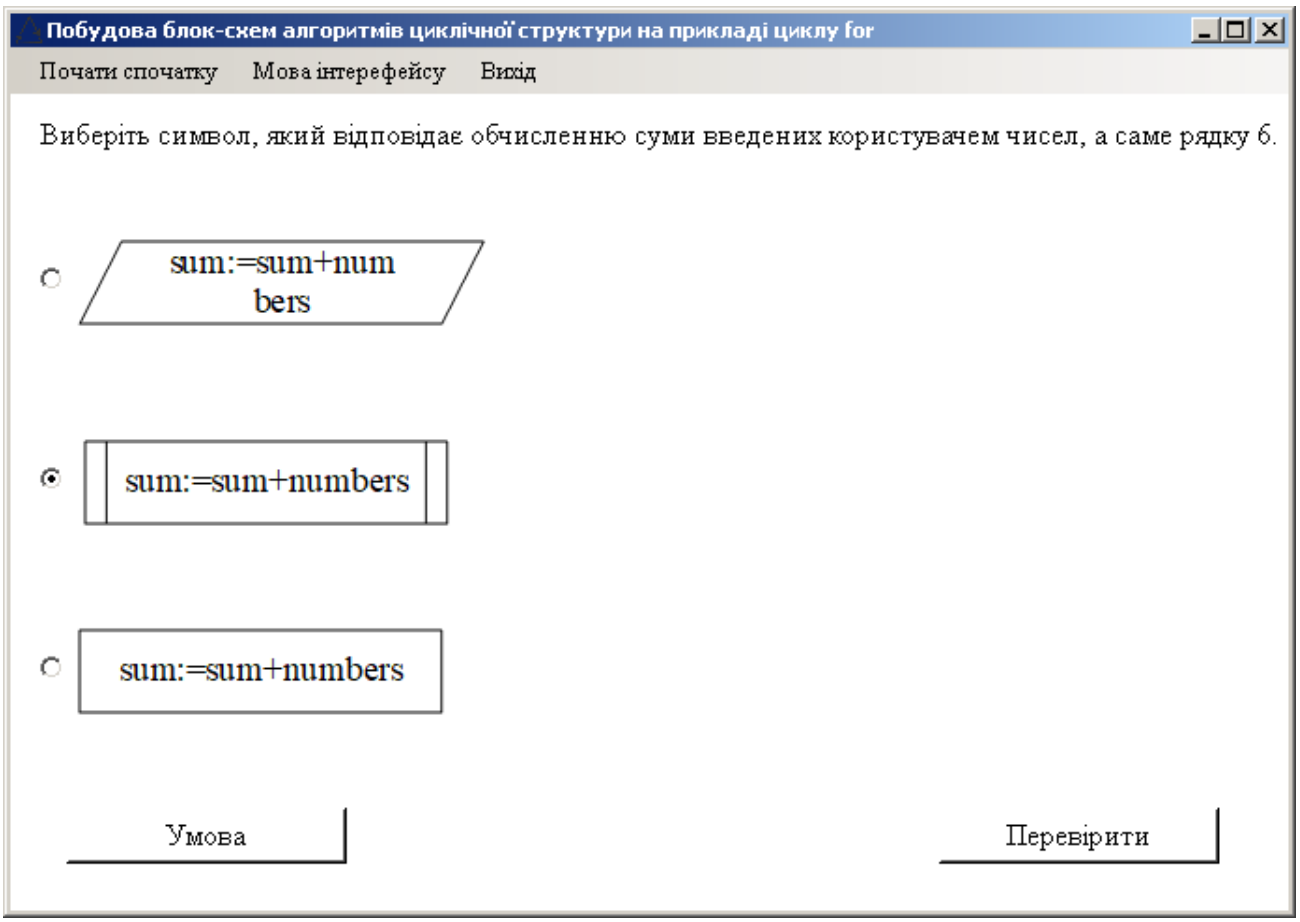


Рисунок 2.9 – Питання тренінгу

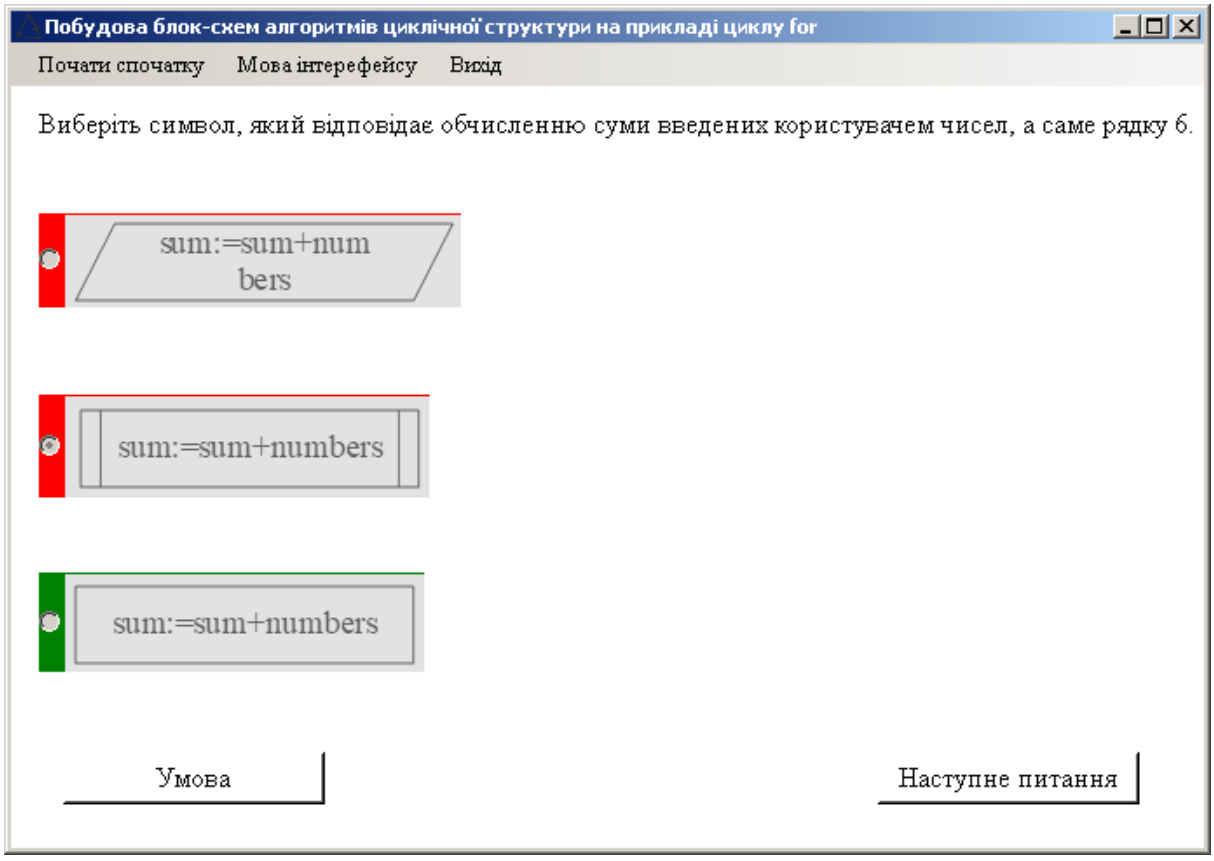


Рисунок 2.10 – Перевірка наданої відповіді

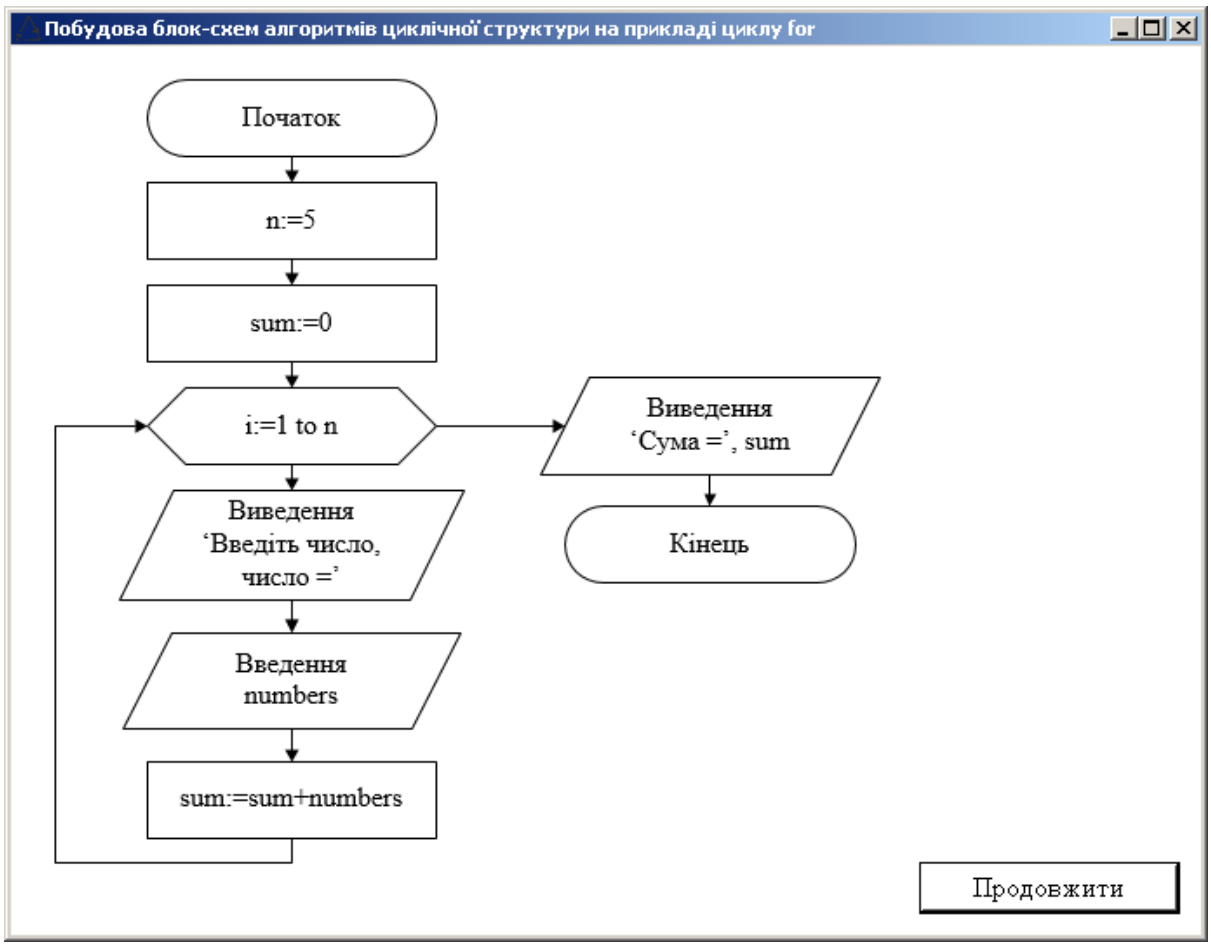


Рисунок 2.11 – Результат

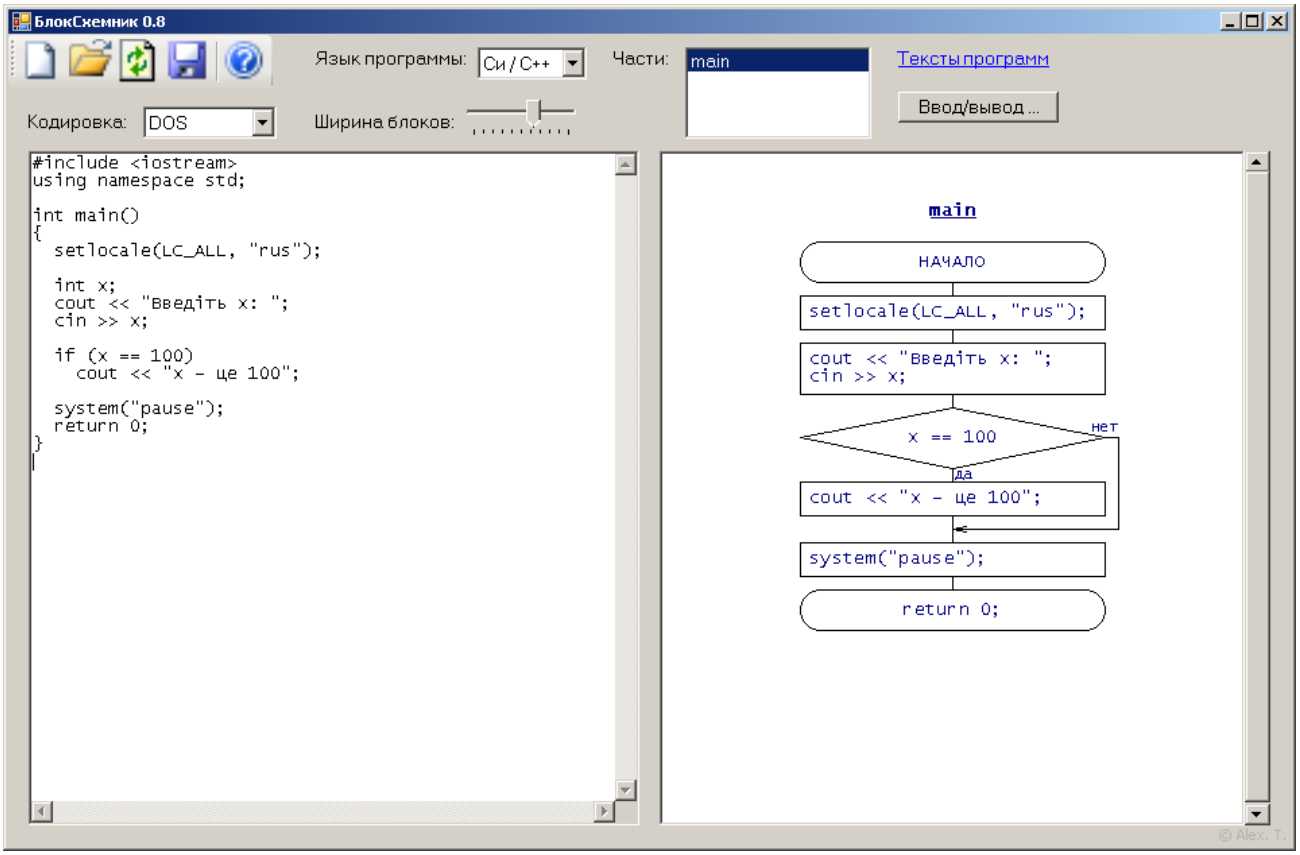


Рисунок 2.12 – Приклад зі структурою if

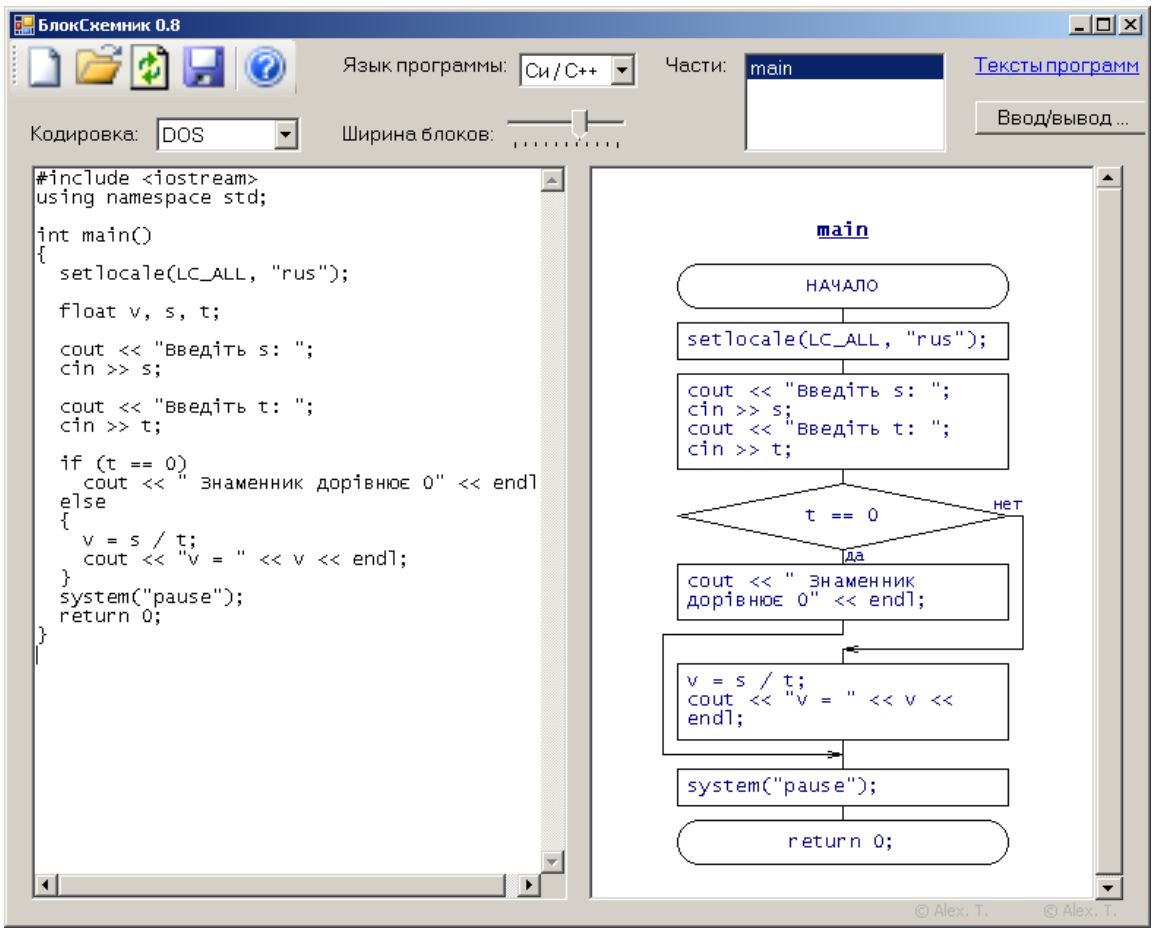


Рисунок 2.13 – Приклад зі структурою if/else

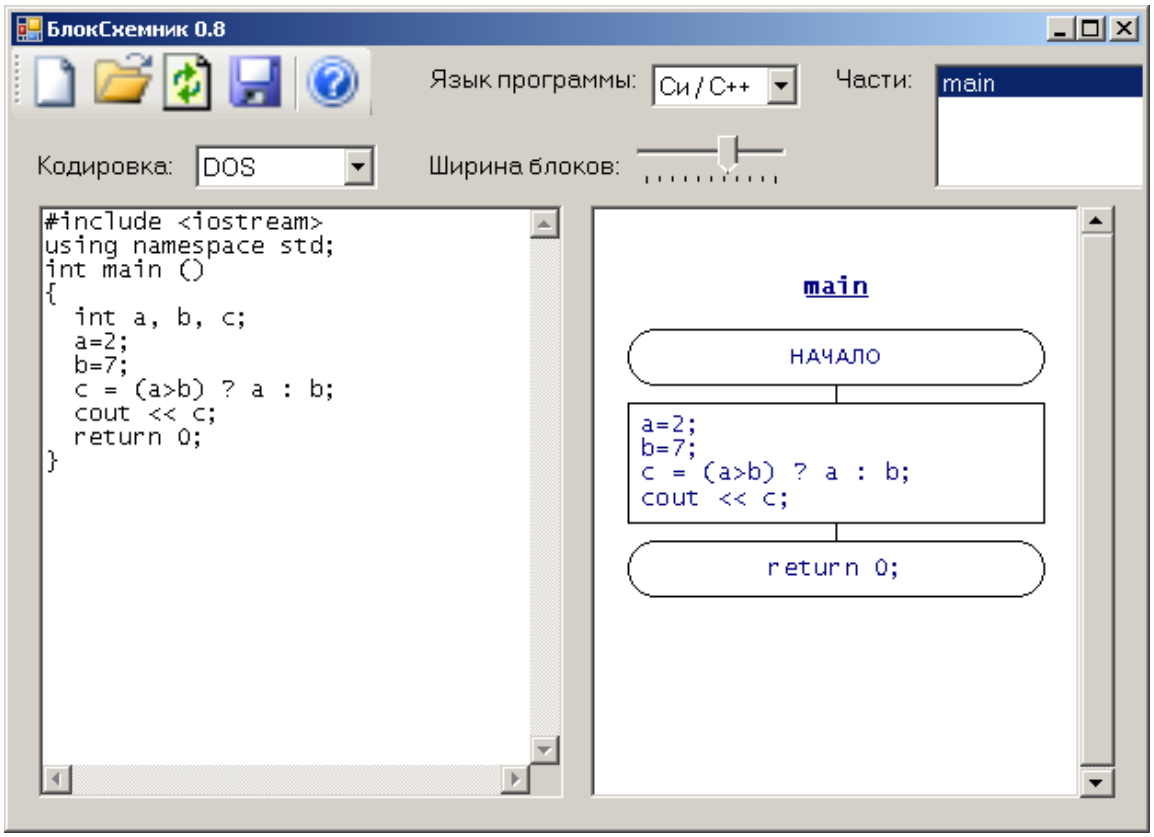


Рисунок 2.14 – Приклад з тринарним умовний оператором

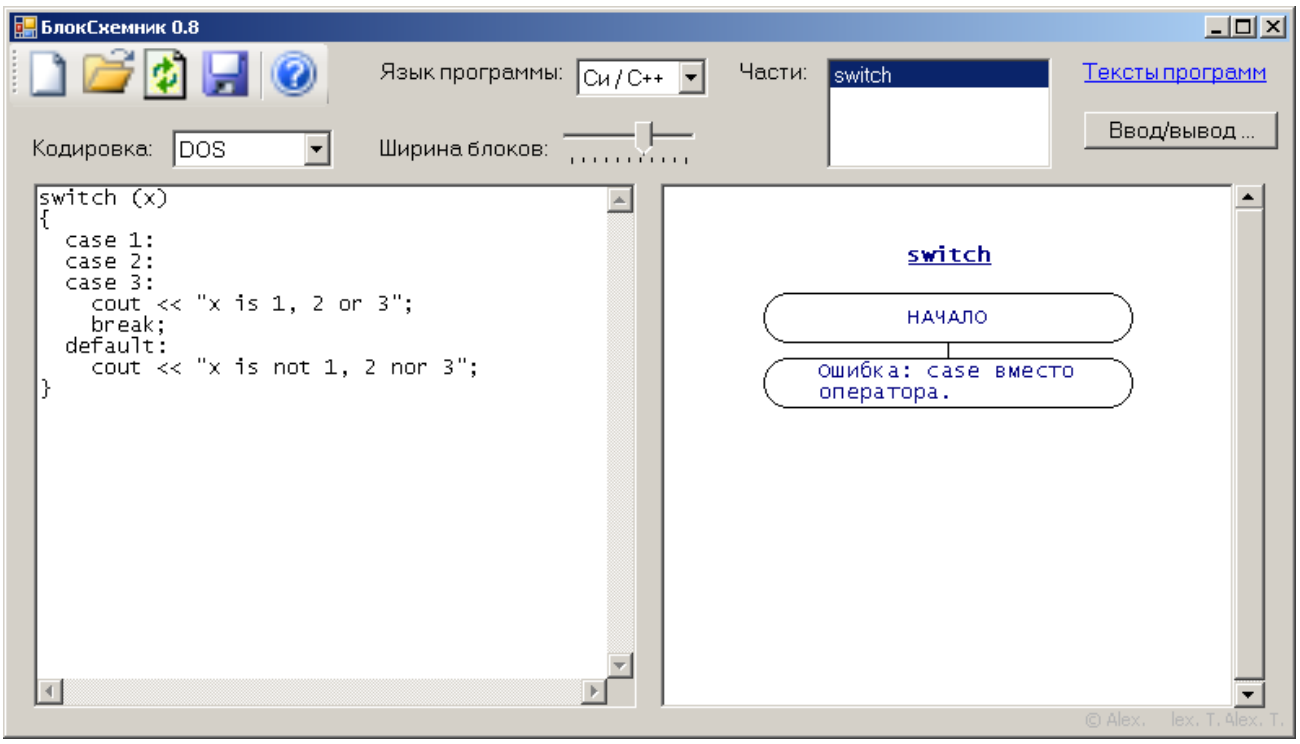


Рисунок 2.15 – Приклад зі структурою switch

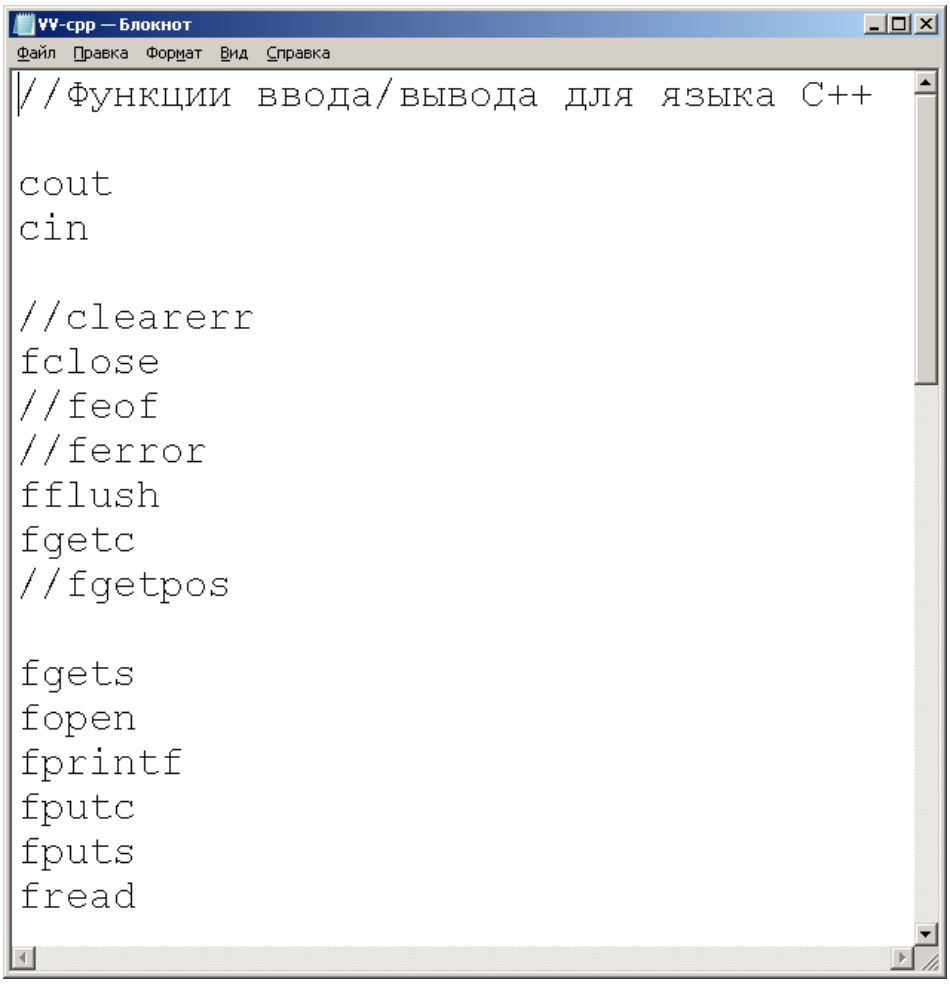


Рисунок 2.16 – Команди мови C++

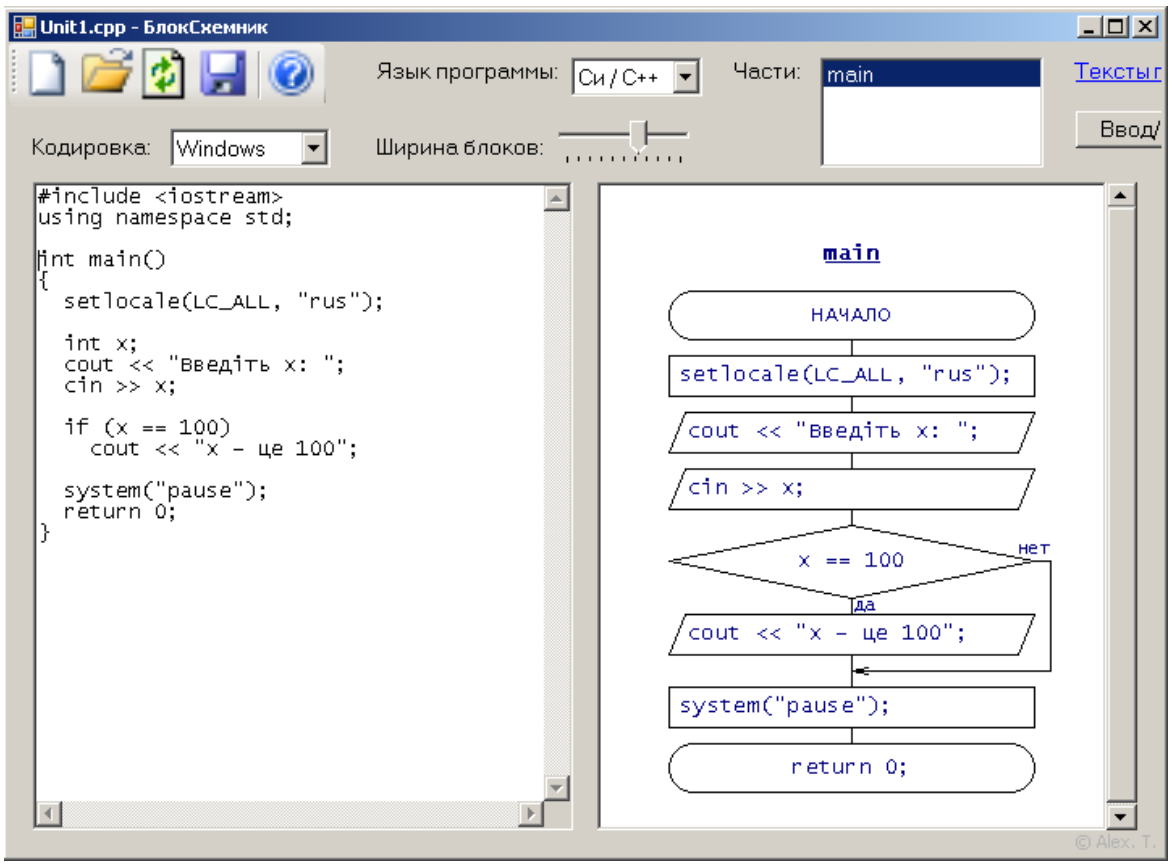


Рисунок 2.17 – Приклад зі структурою if після зміни налаштувань

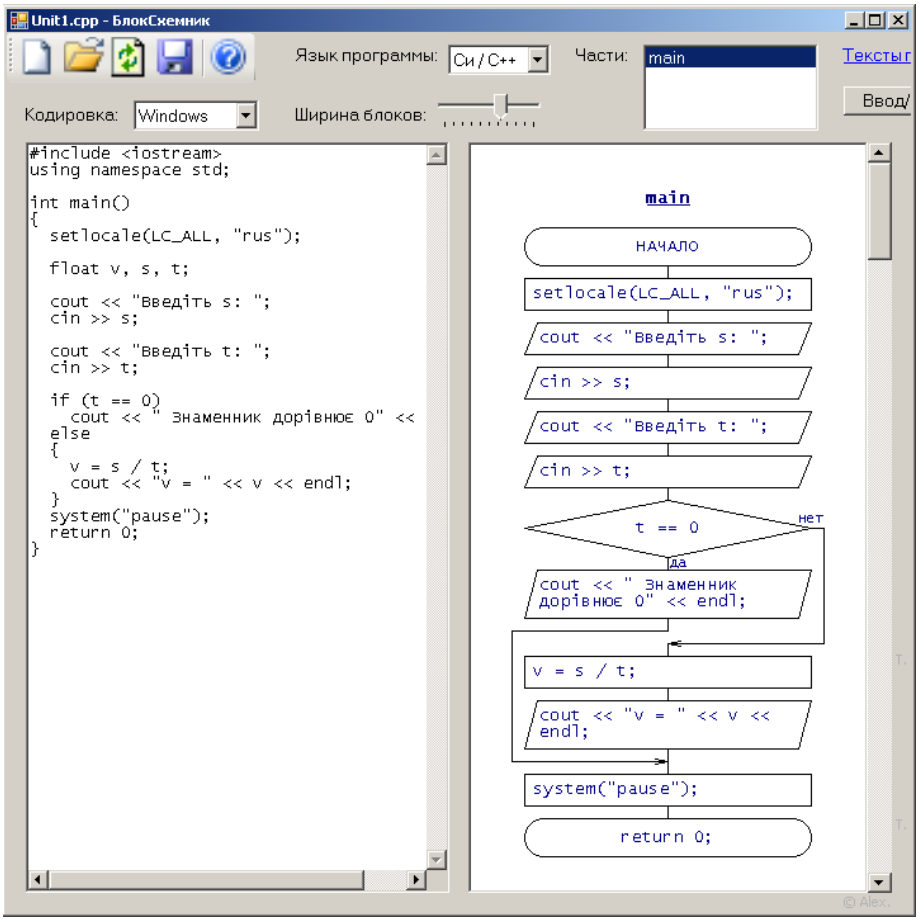


Рисунок 2.18 – Приклад зі структурою if/else після зміни налаштувань

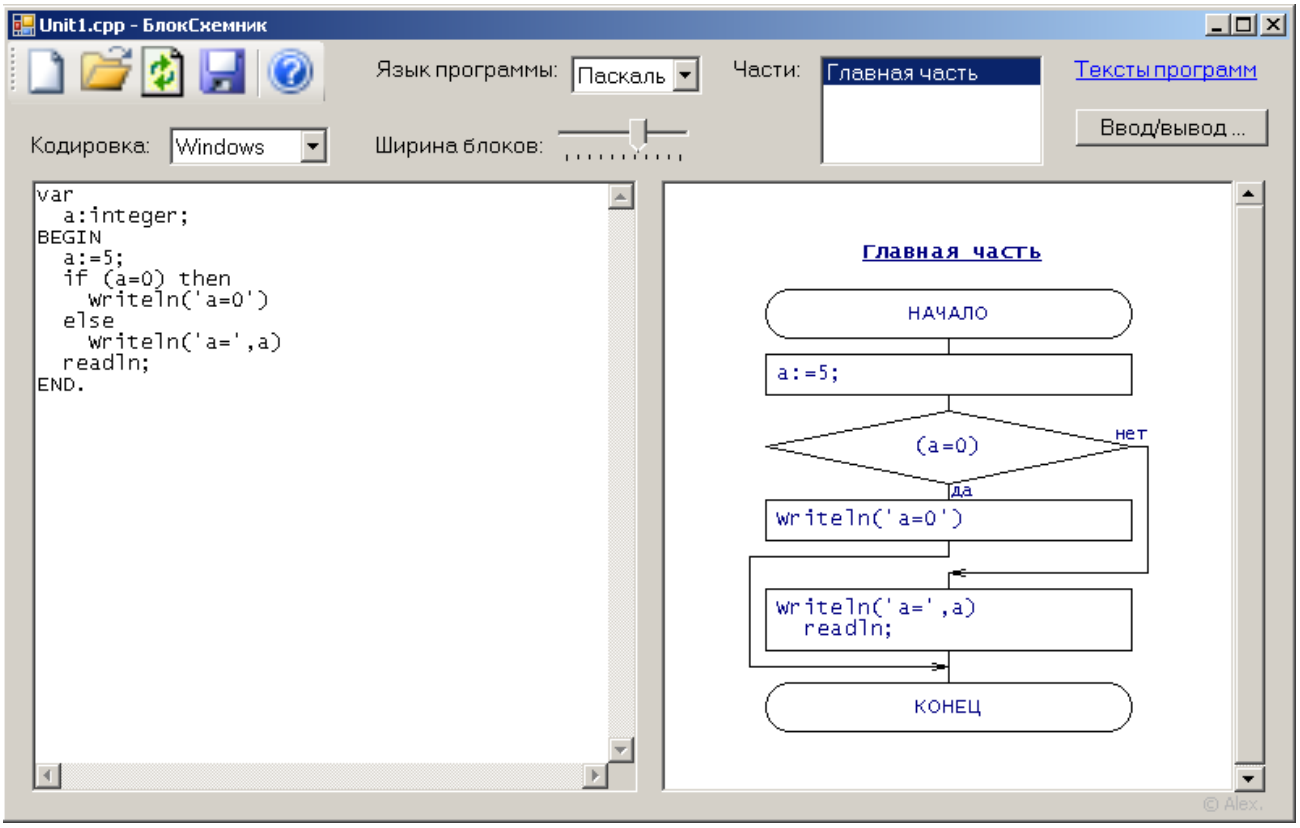


Рисунок 2.18 – Приклад на мові Pascal

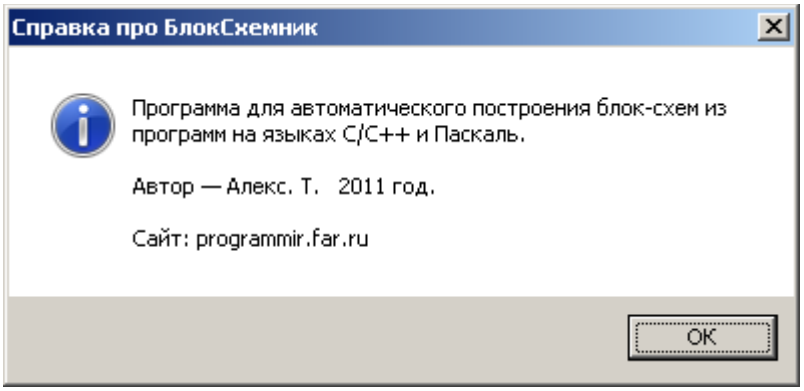


Рисунок 2.19 – Довідка про розробника

Разом з програмою йдуть два текстові файли, в яких перераховані команди вводу та виводу на мові C++ (перший файл) та Pascal (другий файл). Після зміни першого файлу (додано команди cout, cin, рис. 2.16) програма для прикладів з випускової роботи будує більш точні блок-схеми (рис. 2.17-2.18).

Інші можливості БлокСхемника: зміна ширину символів блок-схеми; оновлення рисунку; зберігання рисунку (блок-схеми) у окремому файлі;

розпізнавання функцій та побудова окремих блок-схем для різних функцій; зміна кодування.

2.2. Позитивні аспекти переглянутих робіт

Тренажер Мордасової І.:

1) Після кожного кроку з’являється фрагмент побудованої блок-схеми.
2) Є інформації про розробника.

Тренажер Гмизи Б.

1) Правильна відповідь виділяється зеленим кольором, не вірна – червоним.
2) Програма самостійно виправляє помилки та йде на наступний крок.

Блок-схемник

1) Можливість зчитування коду з файл cpp тощо.
2) Розпізнавання трьох мов: C, C++, Pascal.
3) Розпізнавання функцій у коді та малювання окремних блок-схем для кожної функції.
4) Можливість збереження як картинки блок-схеми.
5) Можливість зміни ширини символів та оновлення картинки.

2.3. Недоліки переглянутих робіт

Тренажер Мордасової І.:

1) Немає пояснення помилки, тільки фіксується, що помилка є.
2) Немає пояснення, як з трьох (або більше) наданих відповідей вірна. Слід вгадувати.
3) Немає фіксації, що користувач не надав відповіді.

4) Немає нумерація рядків ні в завданні, ні протягом тренінгу. А отже, або неможливо відповідати на питання, або слід вгадувати з контексту питання та відповідей.
5) Питання 10 (вибір декількох відповідей з 4 наданих) – багато варіантів для вгадування.
6) Питання 10 (містить помилку) – немає дужок у знаменнику. 1 і 2 відповідь ідентичні, 3 і 4 теж однакові.
7) Помилка на останньому кроці – немає стрілки на лінії, що йде справа наліво, як цього вимагає стандарт.

Тренажер Гмизи Б.

1) Немає заявленого англійськомовного та російськомовного інтерфейсу.
2) Немає нумерація рядків ні в завданні, ні протягом тренінгу.
3) Немає пояснення помилки.
4) Немає фіксації, що користувач не надав жодної відповіді.
5) Після кожного кроку не з’являється фрагмент побудованої блок-схеми.
6) У ітоговій блок-схемі помилка – немає стрілки на лініях, що йдуть справа наліво та знизу догори, як цього вимагає стандарт.
7) Немає нумерації кроків.
8) Немає інформації про розробника.

Блок-схемник

1) Не розпізнає фрагмент коду (рис. 2.15).
2) Не розпізнає тринарний умовний оператор (рис. 2.14).
3) Є помилки у використанні символів (для команд cout та cin повинні відображатися паралелограми, рис. 2.22-2.13).
4) Є порушення вимог стандарту, зокрема, різна висота символів.
5) Є помилки для коду на Pascal. Не розпізнаються команди readln, writeln (рис. 2.18).

2.4. Необхідність та актуальність теми

Переглянуті додатки, містять суттєві помилки, тому не можуть бути використані у навчальному процесі. Крім того, перші дві роботи побудовані на прикладах з кодом іншої мови програмування.

Отже, є необхідність в розробці тренажеру з побудови блок-схем алгоритмів розгалуженої структури на прикладах програм мовою С++. Ця задача є доцільною та актуальною.

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Алгоритм програми

Умова прикладу (код програми) видима впродовж тренінгу.

Якщо користувач надав вірну відповідь, то виводиться підтвердження «Правильно!», з'являється побудована частина блок-схеми, відбувається перехід до наступного кроку алгоритму.

Якщо користувач надав хибну відповідь, то виводиться «Помилка!» і пояснення помилки. Користувач знову відповідає на питання.

Приклад 1.

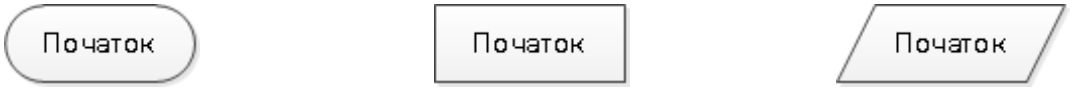
Створити блок-схему програми:

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

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

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

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

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

Рисунок 3.1 – Умова прикладу №1

1. Блок-схема починається символом

1) 2) 3)



Правильна відповідь – 1.

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

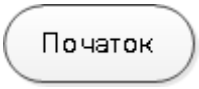
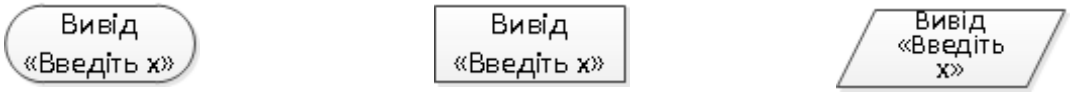


Рисунок 3.2 – Блок-схема, що створюється

2. Виводу фрази (рядок 7) відповідає символ:

1) 2) 3)



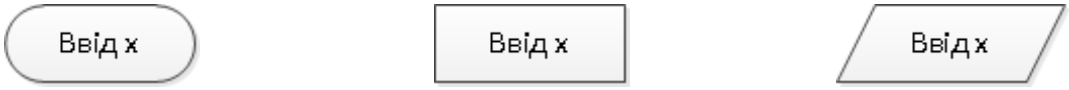
Правильна відповідь – 3.

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

Рисунок 3.3 – Блок-схема, що створюється

3. Введенню *x* (рядок 8) відповідає символ:

1) 2) 3)



Правильна відповідь – 3.

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

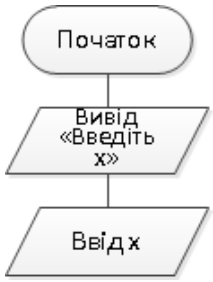
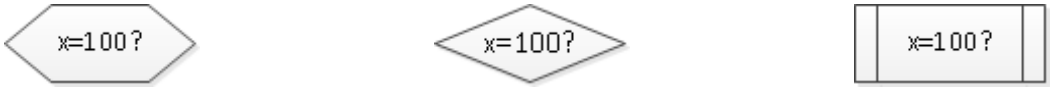


Рисунок 3.4 – Блок-схема, що створюється

4. Перевірці умови (рядок 9) відповідає символ:

1) 2) 3)



Правильна відповідь – 2.

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

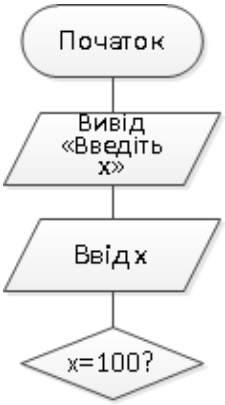
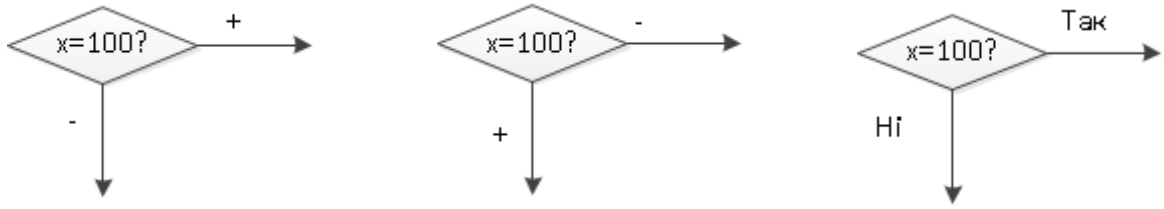


Рисунок 3.5 – Блок-схема, що створюється

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

1) 2) 3)



Правильна відповідь – 1, 2, 3.

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

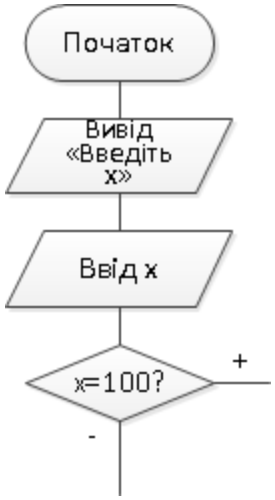
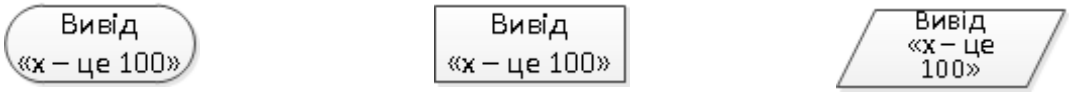


Рисунок 3.6 – Блок-схема, що створюється

6. Виводу фрази (рядок 10) відповідає символ:

1) 2) 3)



Правильна відповідь – 3.

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

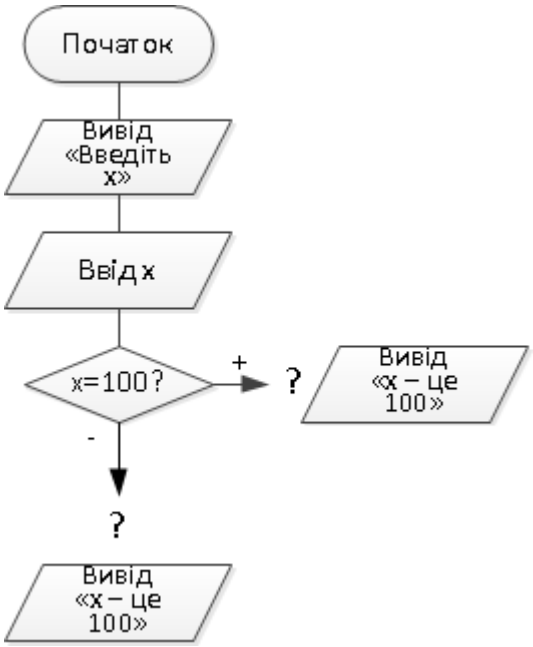
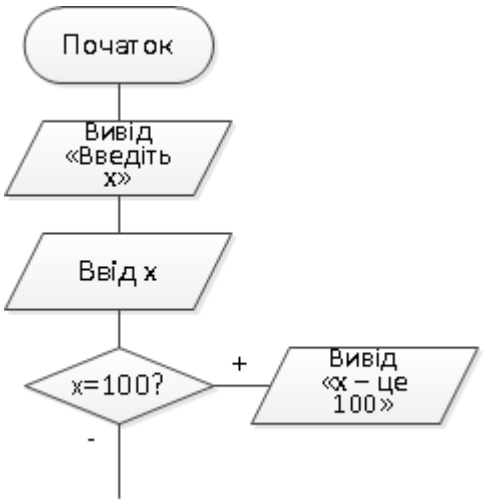


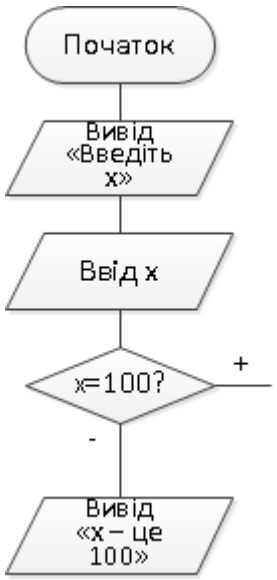
Рисунок 3.7 – Блок-схема, що створюється

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

1)



2)



Правильна відповідь – 1.

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

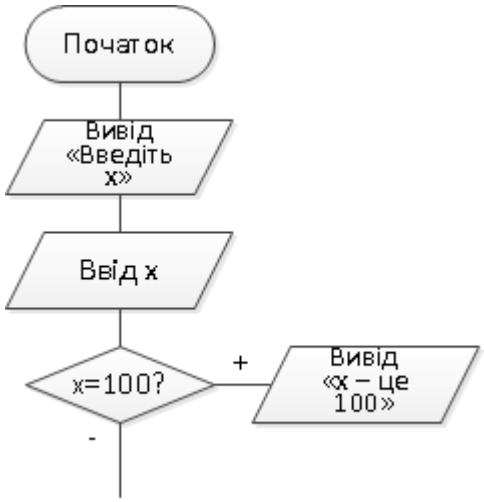
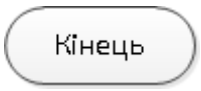


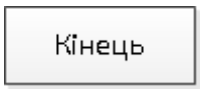
Рисунок 3.8 – Блок-схема, що створюється

8. Блок-схема закінчується символом

1)



2)



3)

Правильна відповідь – 1.

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

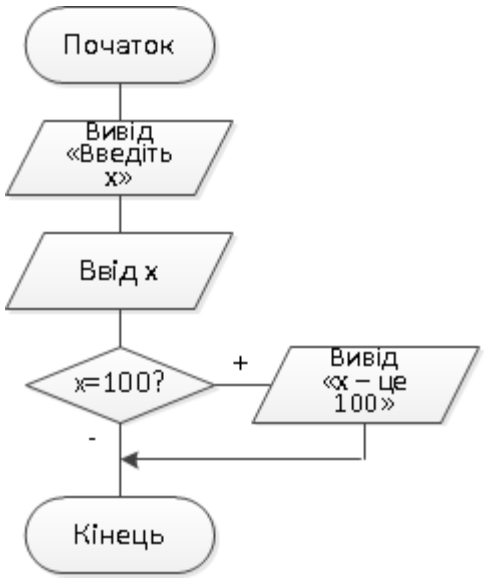


Рисунок 3.9 – Побудована блок-схема

Алгоритм другого прикладу представлено у додатку А.

3.2. Блок-схема алгоритму

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

4. ПРАКТИЧНА ЧАСТИНА

4.1. Опис роботи програмного продукту

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

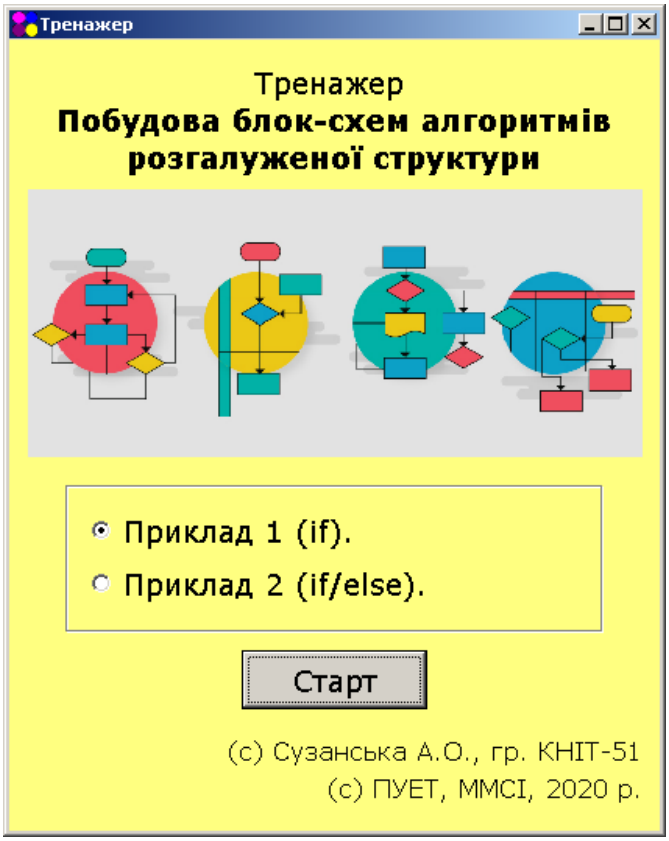


Рисунок 4.1 – Головне вікно тренажеру

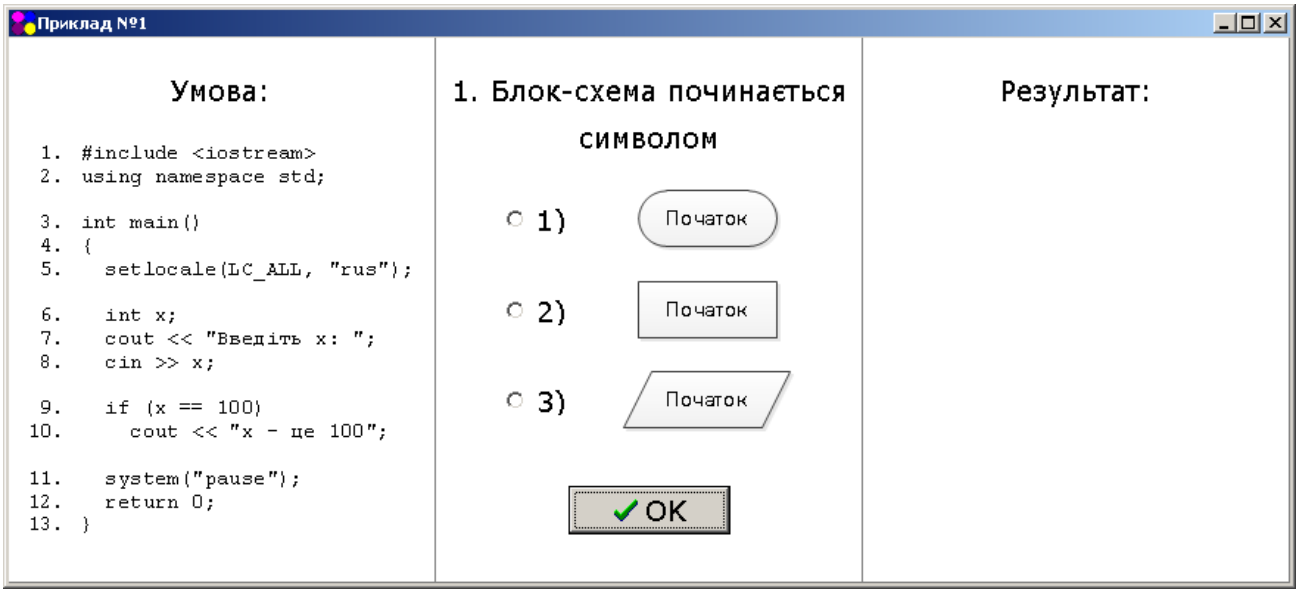


Рисунок 4.2 – Питання №1

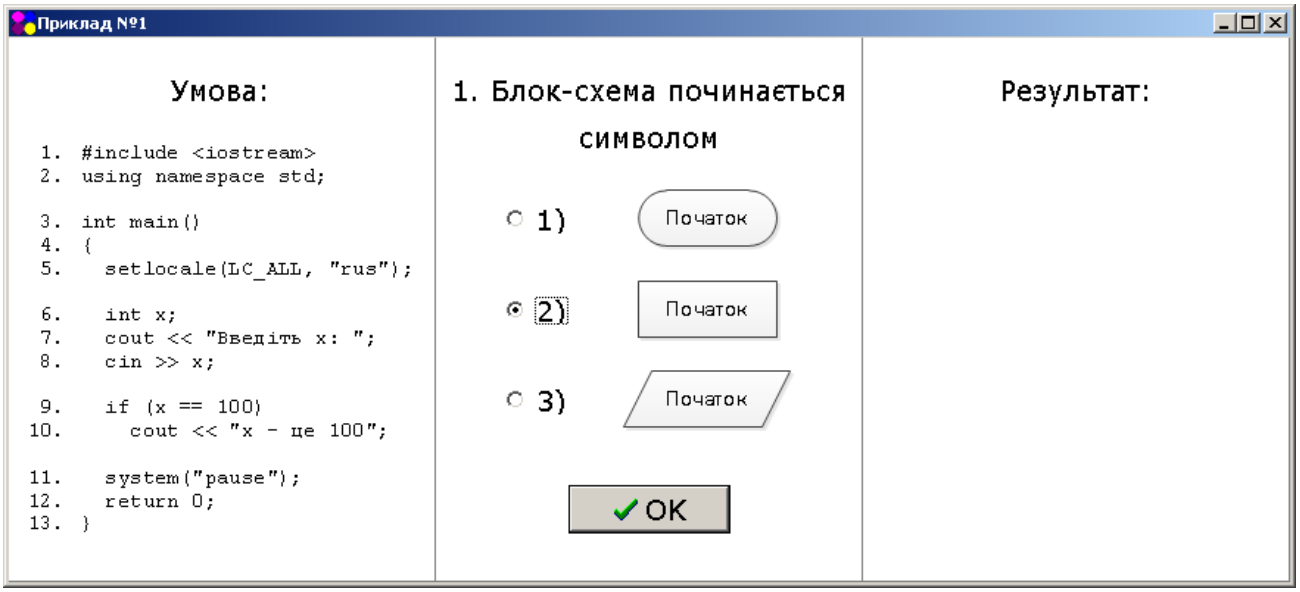


Рисунок 4.3 – Питання №1 (з помилкою)

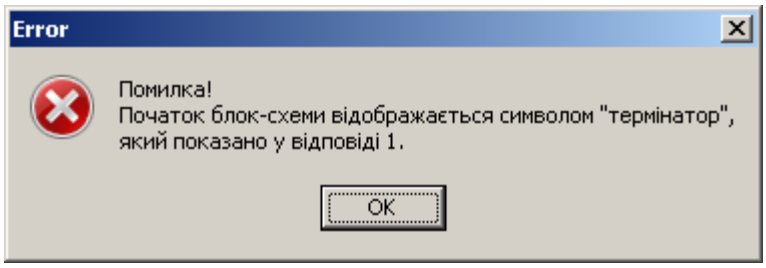


Рисунок 4.4 – Пояснення похибки

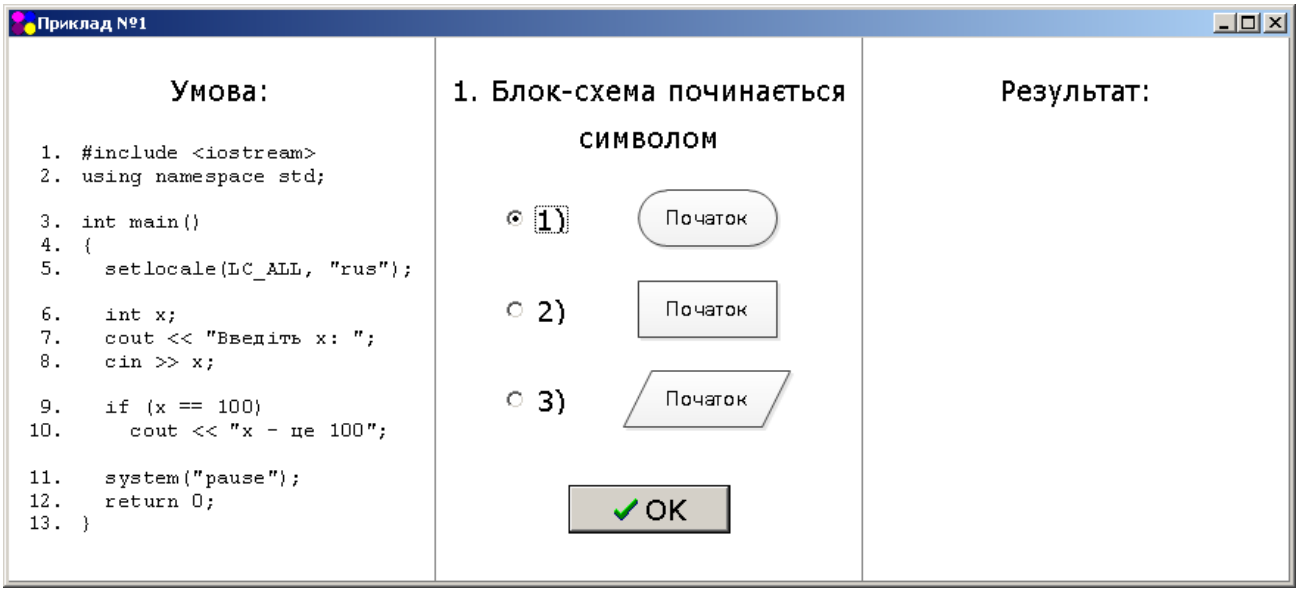


Рисунок 4.5 – Питання №1 (з вірною відповіддю)

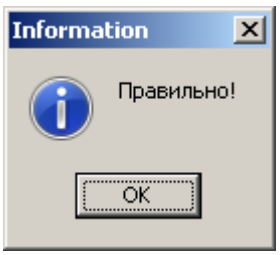


Рисунок 4.6 – У випадку вірності відповіді

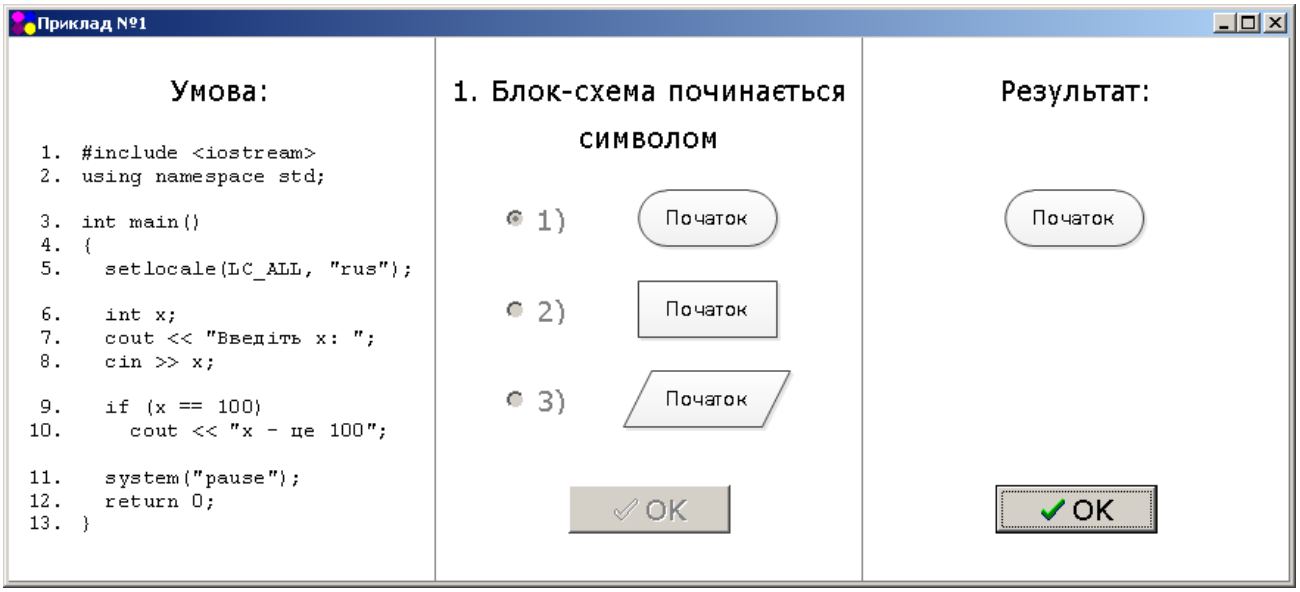


Рисунок 4.7 – Питання №1 (з побудованим фрагментом блок-схеми)

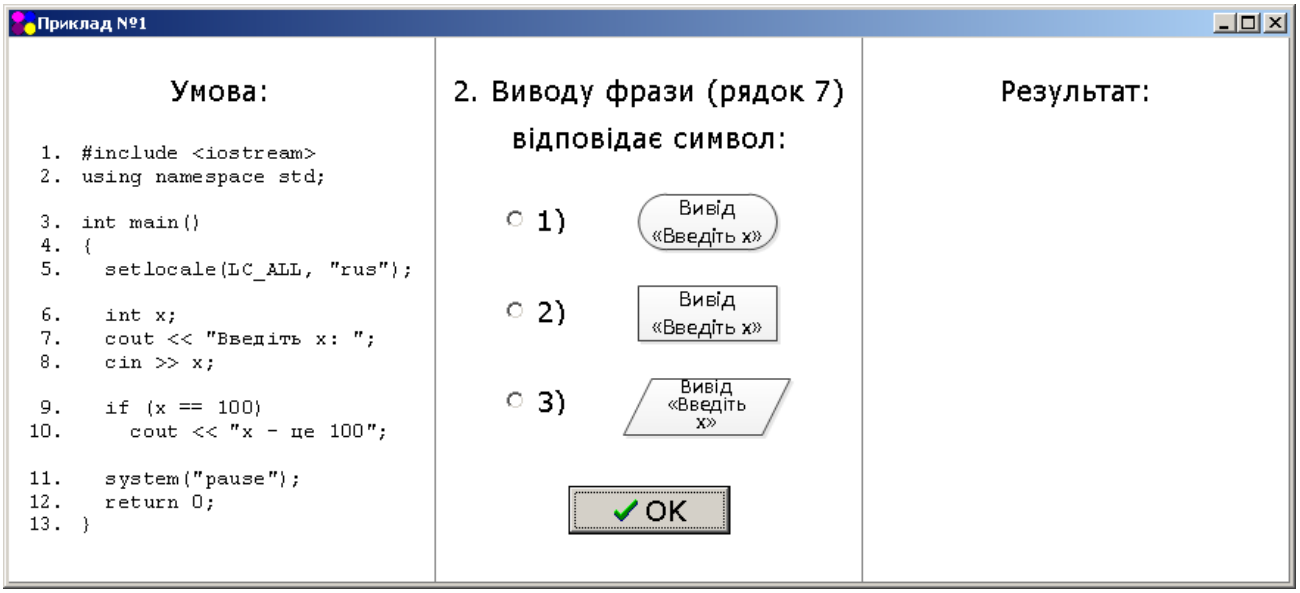


Рисунок 4.8 – Питання №2

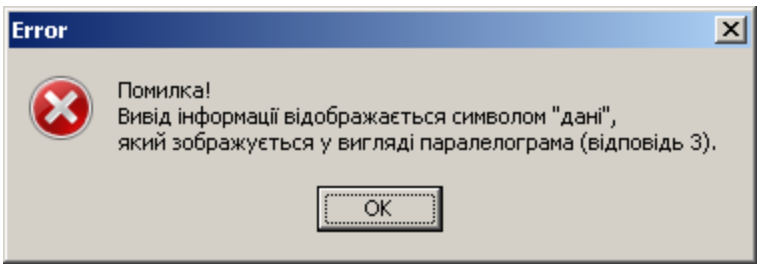


Рисунок 4.9 – Пояснення похибки

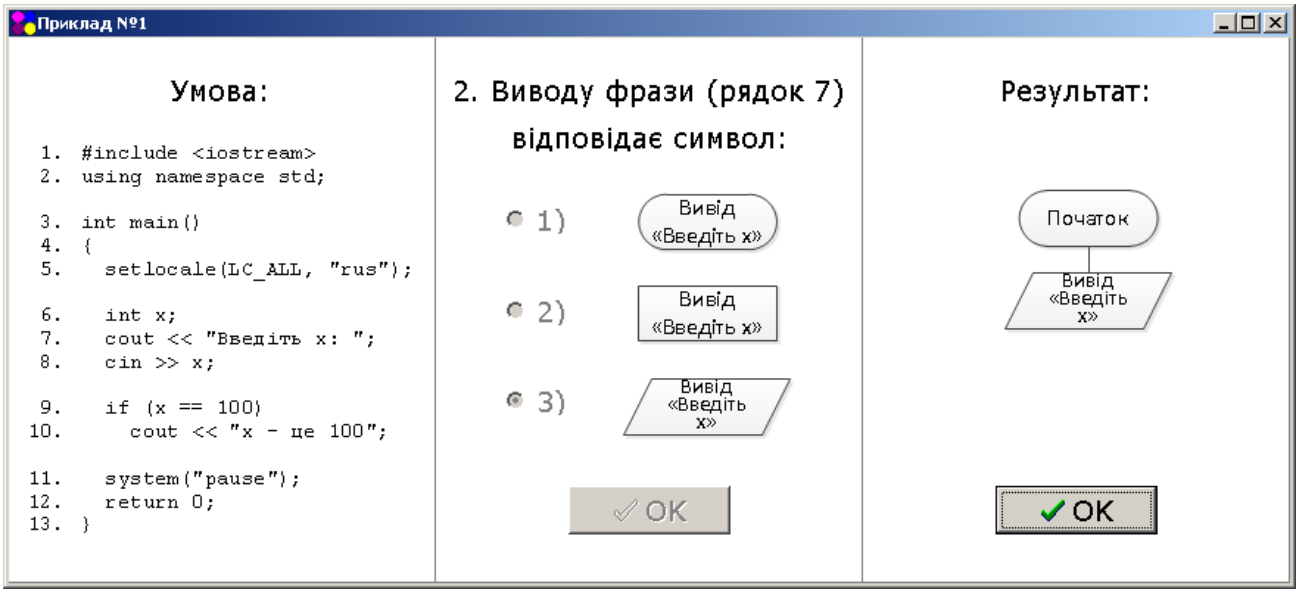


Рисунок 4.10 – Питання №2 (з побудованим фрагментом блок-схеми)

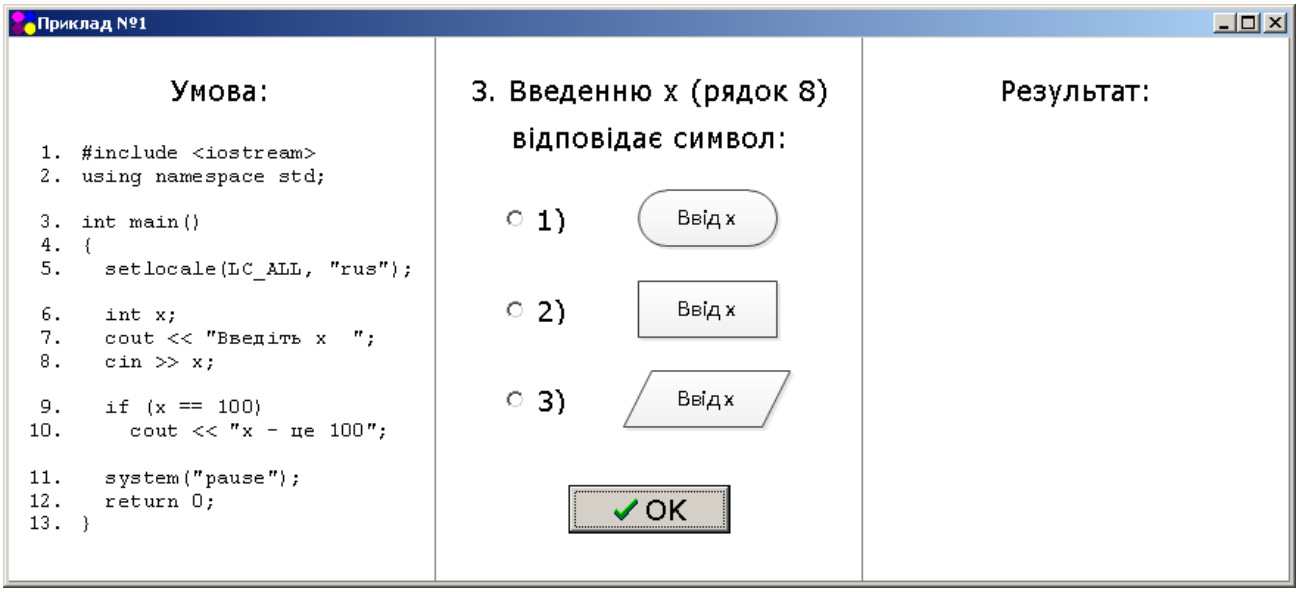


Рисунок 4.11 – Питання №3

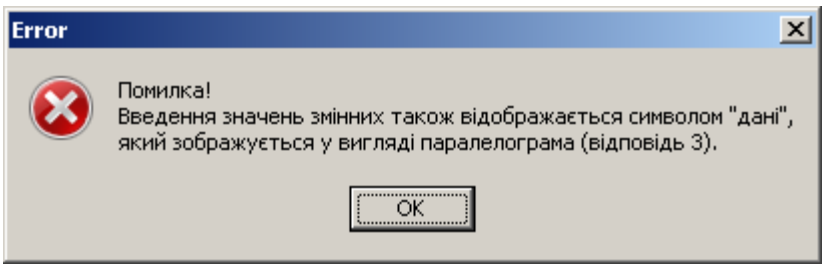


Рисунок 4.12 – Пояснення похибки

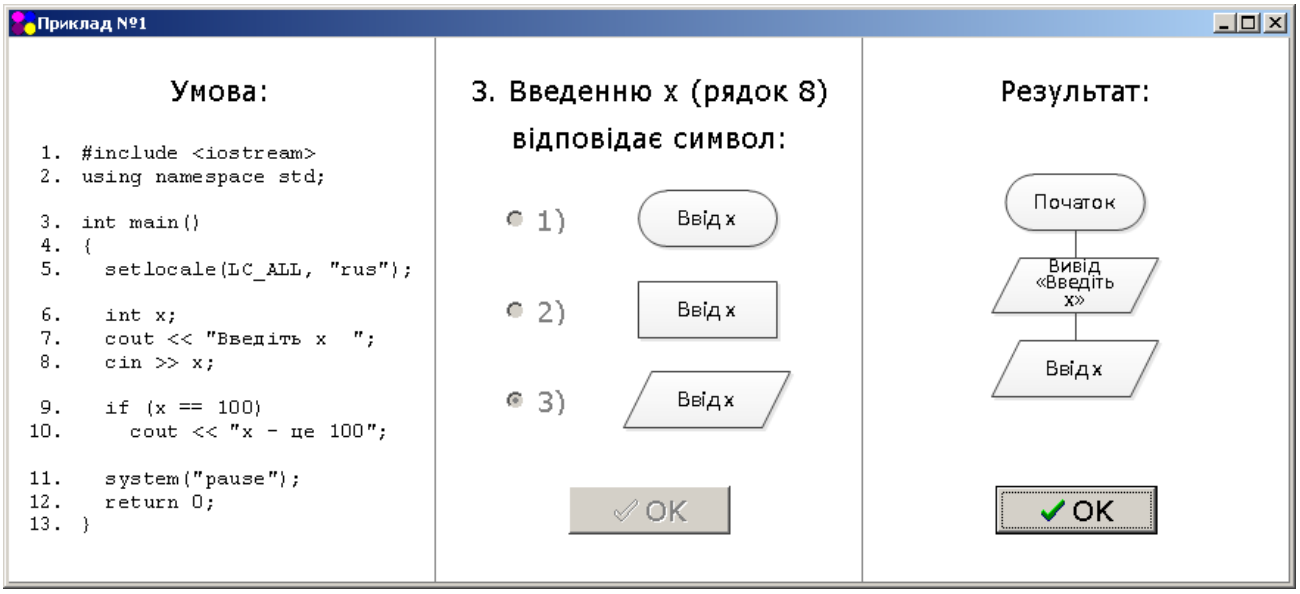


Рисунок 4.13 – Питання №3 (з побудованим фрагментом блок-схеми)

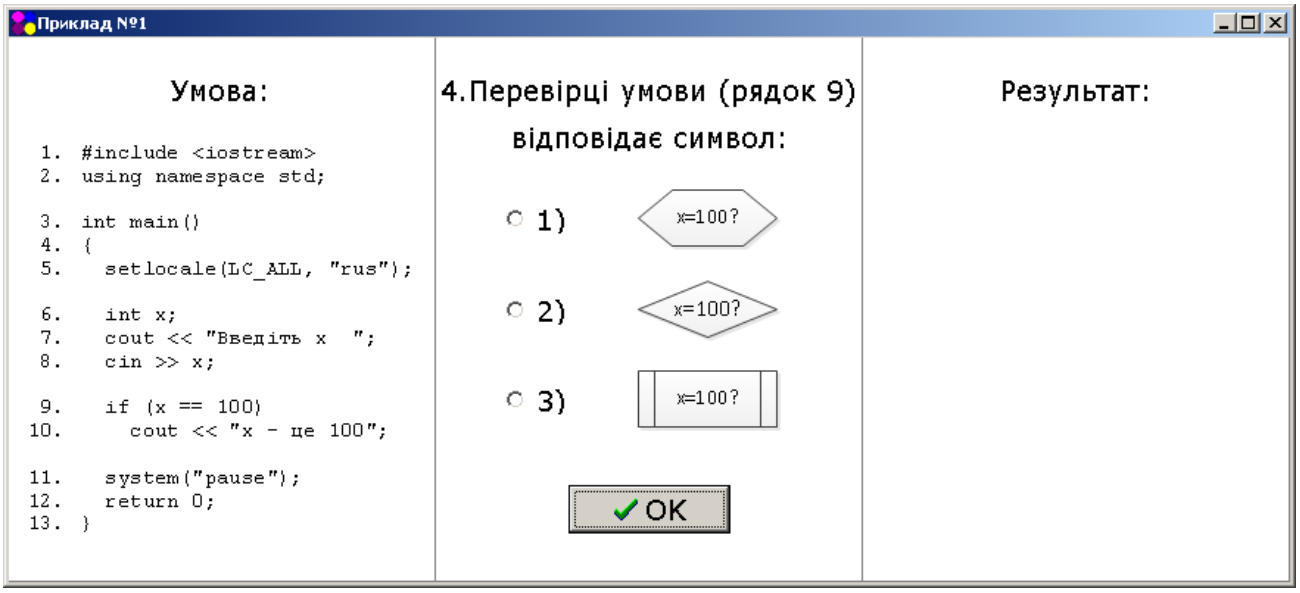


Рисунок 4.14 – Питання №4

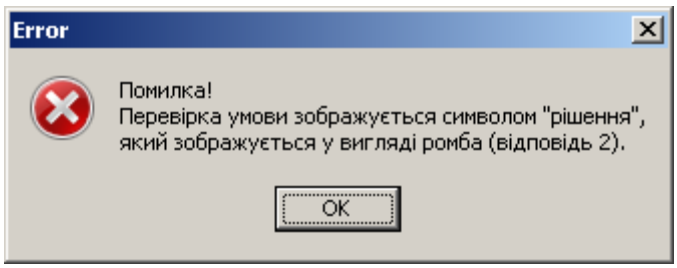


Рисунок 4.15 – Пояснення похибки

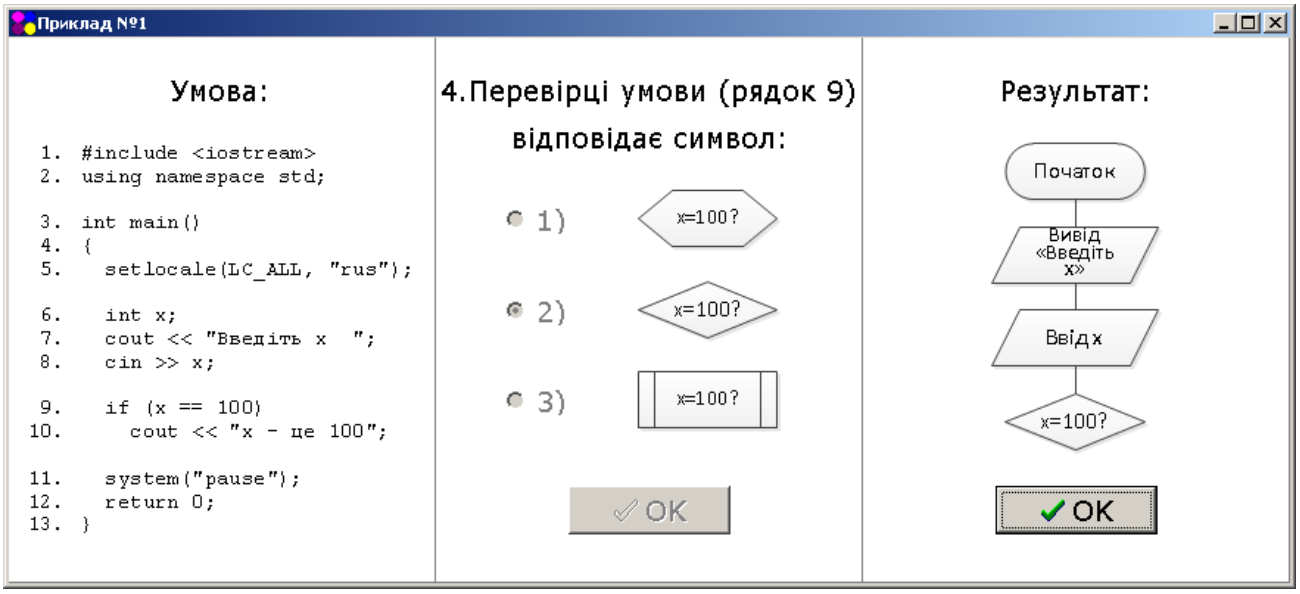


Рисунок 4.16 – Питання №4 (з побудованим фрагментом блок-схеми)

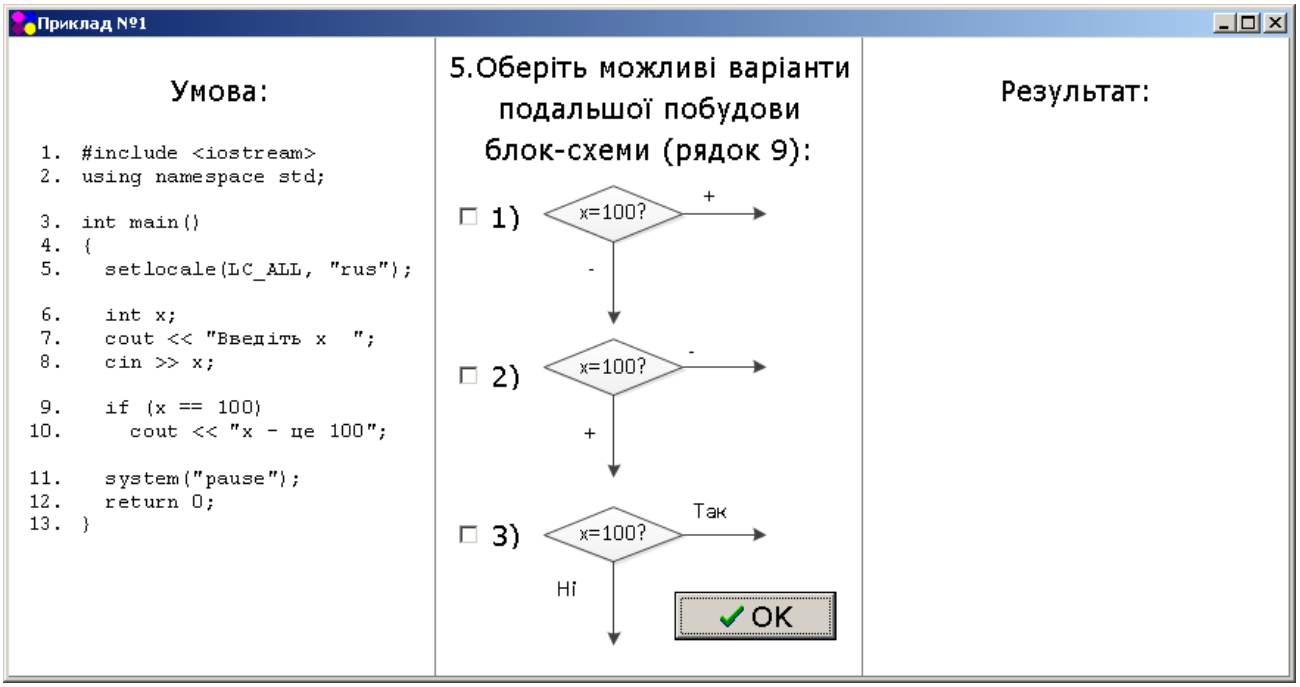


Рисунок 4.17 – Питання №5

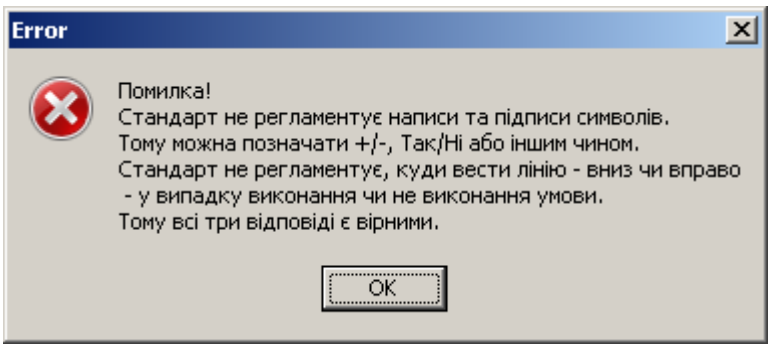


Рисунок 4.18 – Пояснення похибки

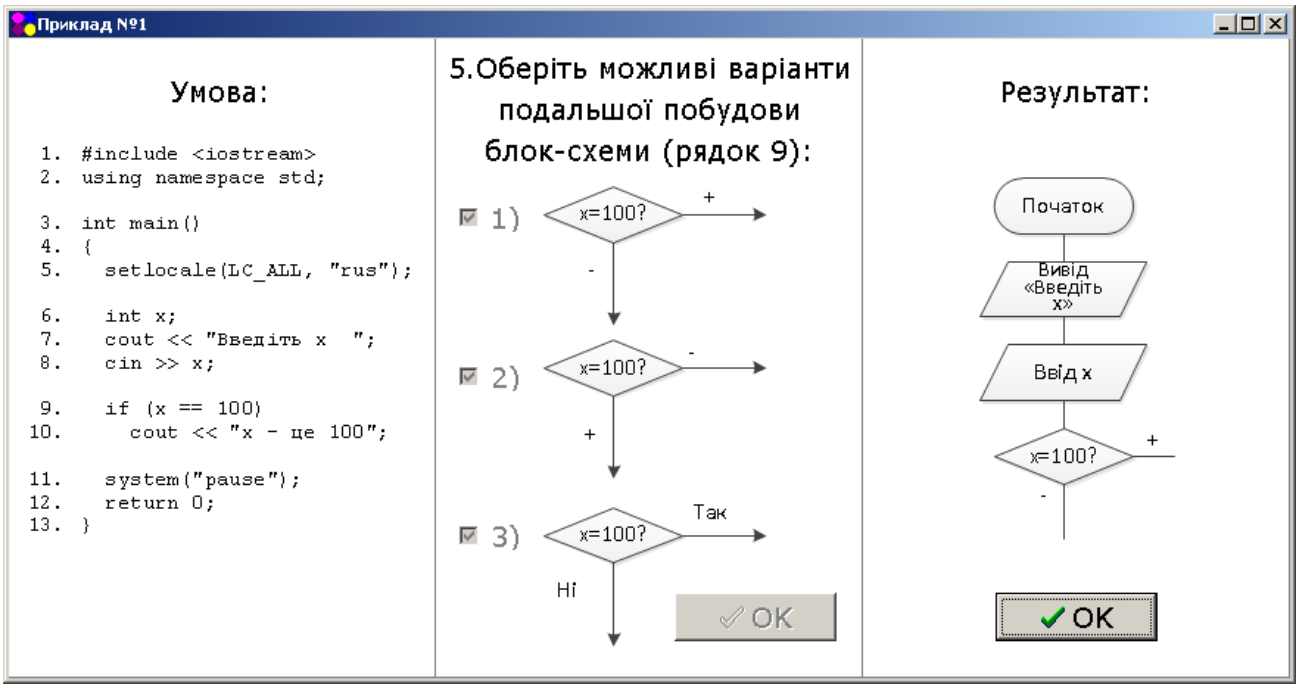


Рисунок 4.19 – Питання №5 (з побудованим фрагментом блок-схеми)

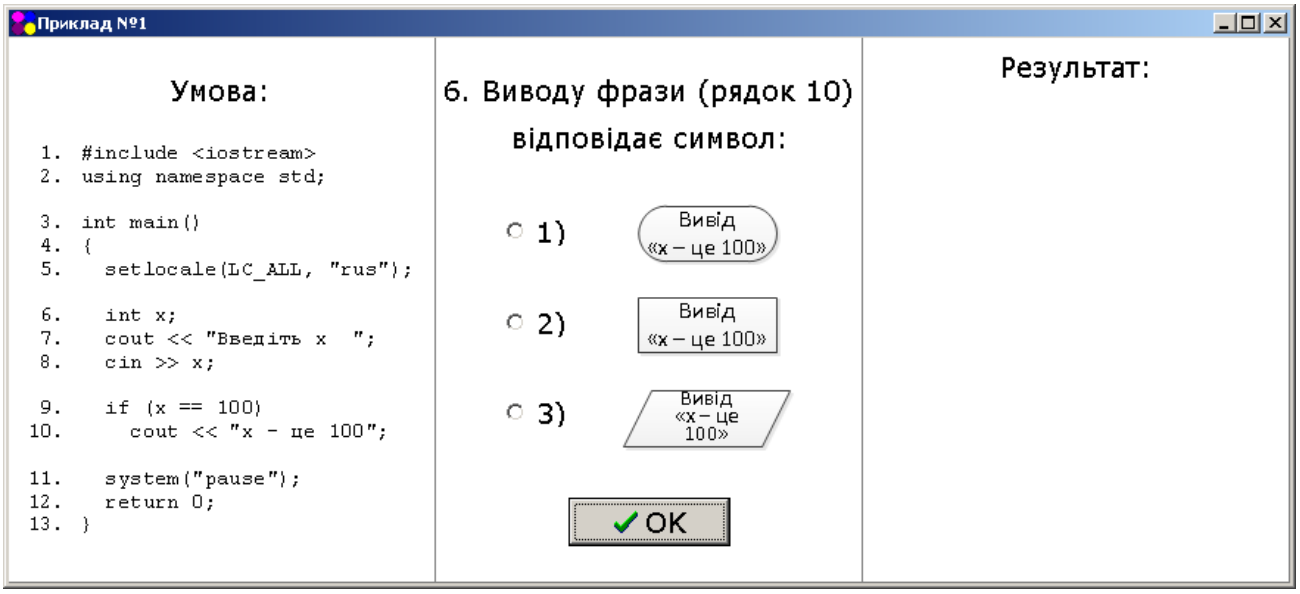


Рисунок 4.20 – Питання №6

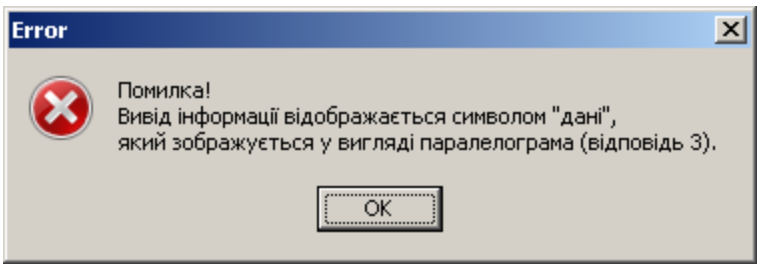


Рисунок 4.21 – Пояснення похибки

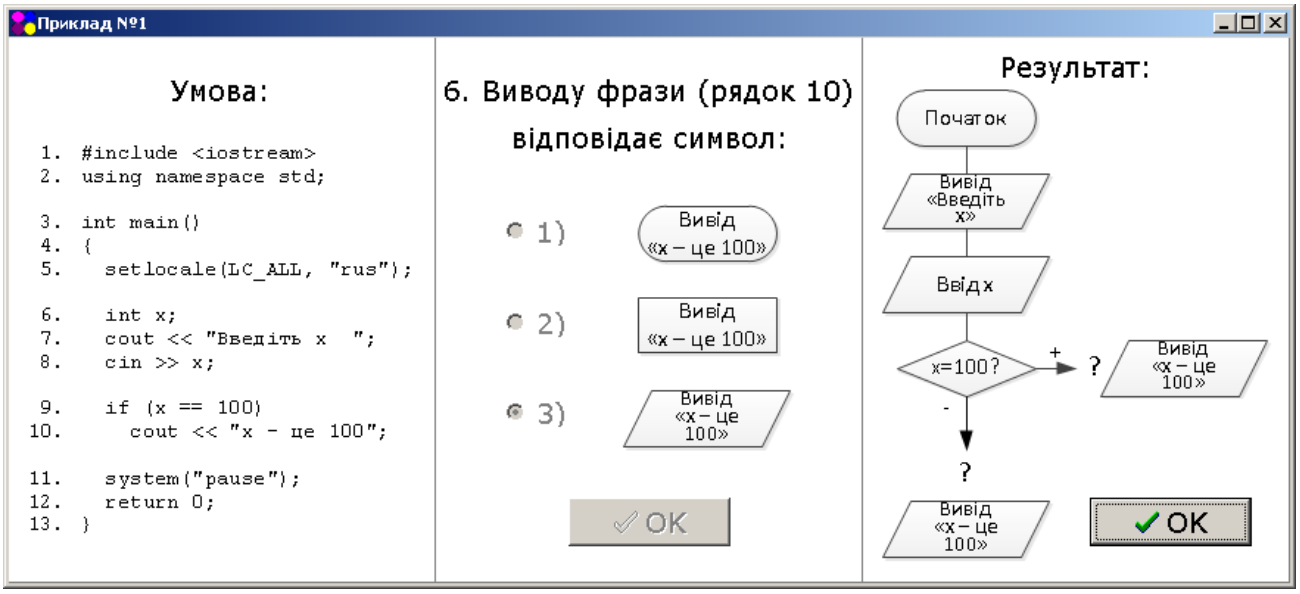


Рисунок 4.22 – Питання №6 (з побудованим фрагментом блок-схеми)

Рисунок 4.23 – Питання №7

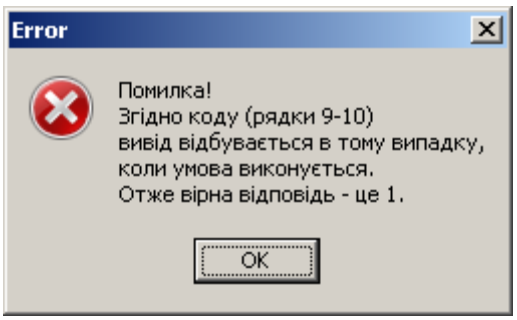


Рисунок 4.24 – Пояснення похибки

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

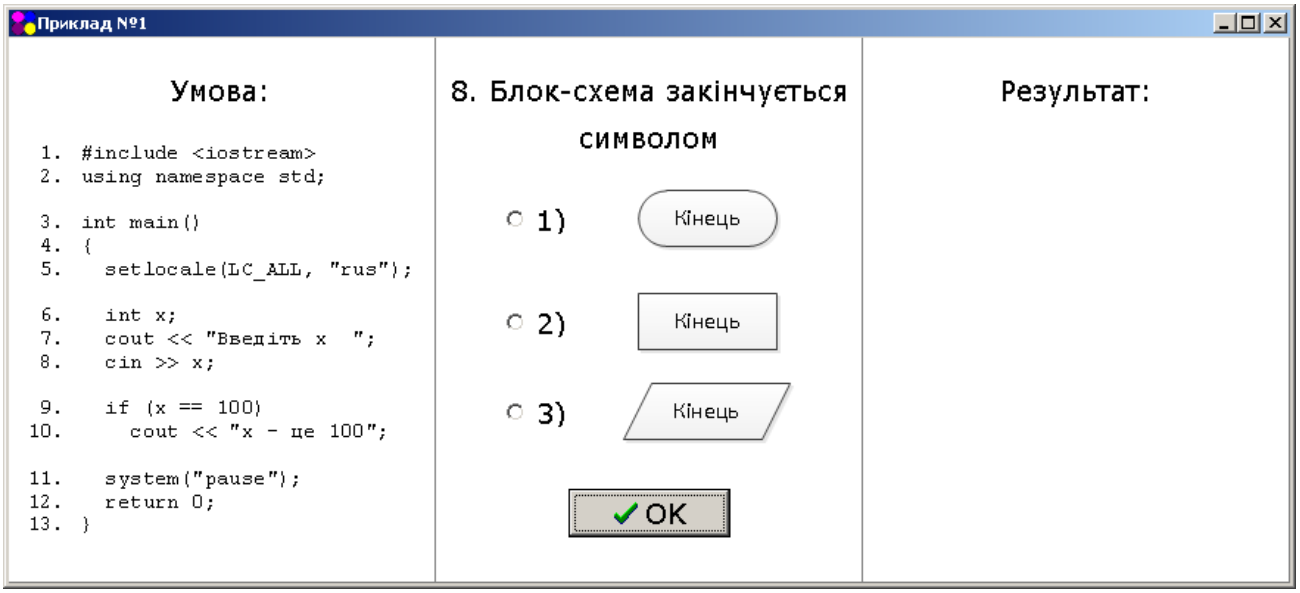


Рисунок 4.26 – Питання №8

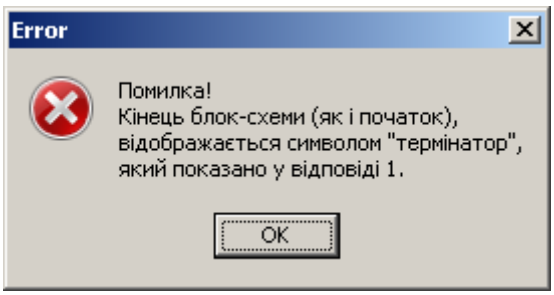


Рисунок 4.27 – Пояснення похибки

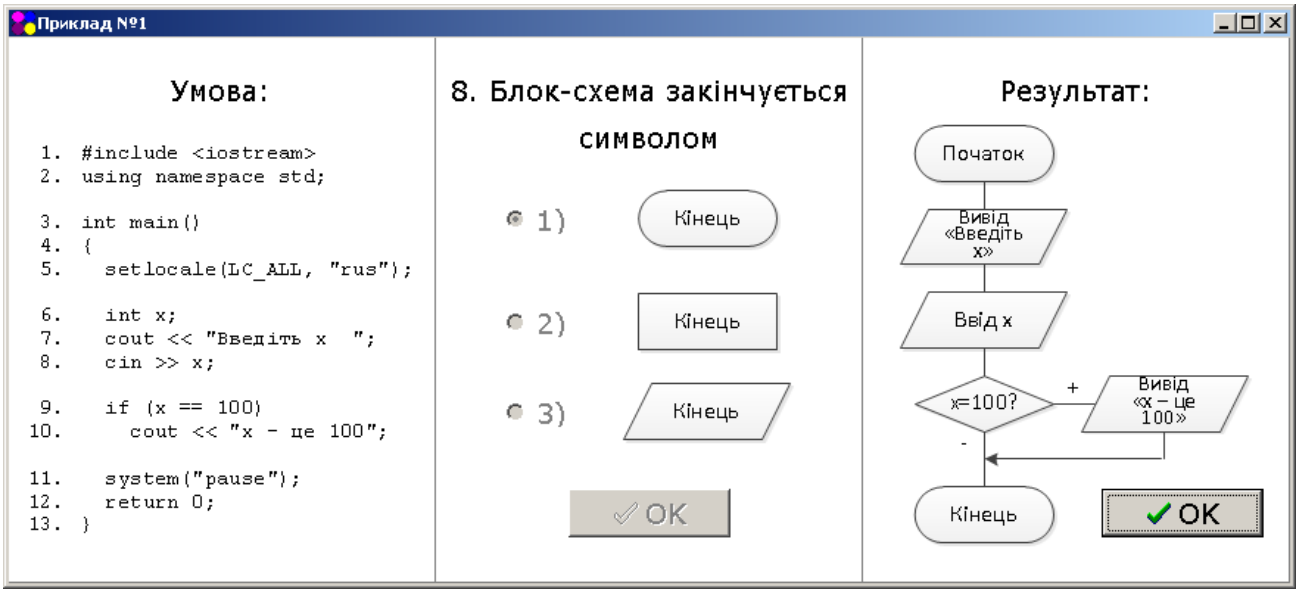


Рисунок 4.28 – Питання №8 (з побудованою блок-схемою)

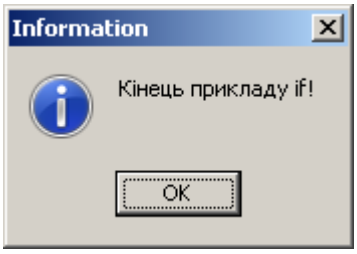


Рисунок 4.29 – Останнє повідомлення

4.2. Опис процесу створення програми

Для створення програми була обрана мова C++ та середовище програмування Borland Builder.

Розглянемо роботу програми для першого питання.

Форма для цього кроку (як і для всіх наступних) поділена на три частини.

В першій частині (лівій) представлена умова завдання (код програми).

В другій частині (центральній) подані питання та варіанти відповідей.

В третій частині (правій) показано фрагмент блок-схеми, що створюється. Рисунок стає видимим після надання правильної відповіді, і показує новий символ або лінії, що додались до блок-схеми на цьому кроці.

Користувач спочатку працює з центральної частиною, де надає свою відповідь і натискає кнопку «ОК». Якщо відповідь помилкова, то з'являється пояснення помилки. Користувач повинен виправити відповідь на вірну та знову натиснути кнопку «ОК». Якщо відповідь вірна, то з'являється повідомлення про це, центральна частина форми блокується від змін, стає активною права частина.

У правій частині з'являється рисунок та кнопку. Після перегляду та аналізу рисунку користувач натискає кнопку «ОК». Відбувається перехід до наступного кроку.

Для першої кнопки «ОК» був створений код:

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

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

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

Якщо відповідь вірна (тут це перша надана альтернатива) *RadioButton1->Checked==true,* то з'являється повідомлення.

Повідомлення створено за допомогою стандартної процедури *MessageDlg*: *MessageDlg("Правильно!", mtInformation, TMsgDlgButtons() << mbOK, 0).*

Далі стає видимою картинка у правій частині: *Image2->Visible=true.*

Стає видимою картинка у правій частині: *BitBtn2->Visible=true.*

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

Усі компоненти центральної частини блокуються (стають неактивними для будь-яких дій користувача): *BitBtn1->Enabled=false; RadioButton1->Enabled=false; RadioButton2->Enabled=false; RadioButton3->Enabled=false.*

Якщо відповідь користувача невірна (обрано другу, третю альтернативи або не обраної жодної з трьох), то з'являється повідомлення про помилки та її пояснення: *MessageDlg("Помилка!\nПочаток блок-схеми відображається символом \"термінатор\",\nякий показано у відповіді 1.", mtError, TMsgDlgButtons() << mbOK, 0).*

Тут символ *\n* означає перехід на наступний рядок, а символ *\"* – вивід лапок у повідомленні.

Для другої кнопки «ОК» був створений код:

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

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

ВИСНОВКИ

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

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

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

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

Для них створено блок-схему та алгоритм тренажеру. Алгоритм складається з восьми кроків для прикладу 1 та дванадцяти кроків для прикладу 2.

Алгоритм тренажеру закодовано мовою С++. Програма перевірена на помилки та описки. Продукт повністю готовий до використання.

Процес створення та роботи програмного додатку задокументовано та викладено у пояснювальній записці до дипломного проекту.

Тренажер передано на впровадження у дистанційний курс «Програмування ІІ» Полтавського університету економіки і торгівлі, що підтверджує акт впровадження.

Результати роботи пройшли апробацію на науково-практичному семінарі «Комп’ютерні науки і прикладна математика».

Були опубліковані тези: 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.

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

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

2. Ємець О. О. Дистанційний курс Полтавського університету економіки та торгівлі «Програмування ІІ. Частина 1» для студентів спеціальностей «Комп'ютерні науки та інформаційні технології», «Комп'ютерні науки» / О. О. Ємець. – [Електронний ресурс].

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

4. Гмиза Б. Ю. Тренажер з теми «Побудова блок-схем алгоритмів циклічної структури на прикладі циклу for» дистанційного навчального курсу «Інформатика» та розробка його програмного забезпечення / Б. Ю. Гмиза, 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. 38-39. – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/7036.

5. Мордасова І. В. Тренажер з теми «Побудова блок-схем алгоритмів розгалуженої структури» дистанційного навчального курсу «Інформатика» та розробка його програмного забезпечення / І. В. Мордасова, 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. 35-37. – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/7037.

6. 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.