ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

ІНСТИТУТ ЗАОЧНО-ДИСТАНЦІЙНОГО НАВЧАННЯ
ФОРМА НАВЧАННЯ ЗАОЧНА

КАФЕДРА МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ ТА СОЦІАЛЬНОЇ ІНФОРМАТИКИ

Допускається до захисту

Завідувач кафедри _____О.О. Ємець
(підпис)

«_____»_____2019 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА ДО ДИПЛОМНОЇ РОБОТИ

на тему

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТРЕНАЖЕРУ З ТЕМИ «РЕКУРСИВНІ АЛГОРИТМИ» ДИСТАНЦІЙНОГО НАВЧАЛЬНОГО КУРСУ «АЛГОРИТМИ ТА СТРУКТУРИ ДАНИХ»

з спеціальності 122 «Комп'ютерні науки»

Виконавець роботи Чуб Олександр Іванович _____ «___»_____2019 р.
(підпис)

Науковий керівник к.ф.-м.н., доц., Ємець Олександра Олегівна
_____ «___»_____2019 р.
(підпис)

ПОЛТАВА 2019 р.

РЕФЕРАТ

Записка: 60 с., 70 рис., 4 таблиці, 2 додатки (на 18 сторінках), 12 джерел.

Предмет розробки – комп'ютерний тренажер з теми «Рекурсивні алгоритми».

Мета роботи – створення тренажеру з теми «Рекурсивні алгоритми» для дистанційного курсу «Алгоритми і структури даних».

Методи розробки – мова С++, середовище програмування Borland Builder.

Досліджено тему «Рекурсія. Рекурсивні алгоритми».

Здійснено огляд візуалізаторів рекурсивних алгоритмів.

Розроблено алгоритм тренажеру, блок-схему алгоритму.

Створено програму-тренажер з теми «Рекурсивні алгоритми», яка складається з трьох прикладів.

Ключові слова: РЕКУРСІЯ, РЕКУРСИВНІ АЛГОРИТМИ, ТРЕНАЖЕР.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні позначення, символи, одиниці, скорочення, терміни	**Пояснення умовних позначень, символів, одиниць, скорочень, термінів**
Непряма рекурсія	Рекурсія називається непрямою, якщо одне поняття визначається через інше, яке, у свою чергу, визначене через вихідне.
Пряма рекурсія	Рекурсія називається прямою, якщо поняття визначається через те саме поняття, яке, у свою чергу, визначене через вихідне.
Рекурентні співвідношення	Це рівняння або нерівності, які описують функції із використанням самих себе, але тільки з меншими аргументами.
Рекурсивні алгоритми	Рекурсивні алгоритми – ті, що застосовують для розв’язання задач, що використовують рекурсивні поняття. Алгоритм називається рекурсивним, якщо він використовує як допоміжний сам себе (пряма рекурсія) або інший алгоритм, який використовує всередині себе вихідний (непряма рекурсія).
Рекурсія	Поняття називається рекурсивним, якщо воно визначається через те саме поняття
Трасування алгоритму	Покрокове виконання алгоритму (програми) на певних вхідних даних.

ВСТУП

Рекурсивною підпрограмою (алгоритмом) називають підпрограму (алгоритм), яка викликає сама себе.

У програмуванні рекурсія зустрічається часто. З одного боку рекурсивні алгоритмі швидші, з іншого боку вони потребують більше оперативної пам’яті, а у випадку надмірної кількості викликів існує ймовірність нестачі пам’яті для обчислень. В зв’язку із специфікою такого класу алгоритмів їм приділяється значна увагу при вивченні програмування.

Отже, розробка тренажеру, який би навчав або поглиблював знання з теми «Рекурсивні алгоритми» є актуальною та важливою задачею.

Метою дипломного проектування є створення тренажеру з теми «Рекурсивні алгоритми» для дистанційного курсу «Алгоритми і структури даних».

Задачі дипломного проектування:

1) дослідити тему «Рекурсія. Рекурсивні алгоритми»;

2) виконати огляд тренажерів, візуалізаторів, інших аналогічних розробок з теми «Рекурсивні алгоритми»;

3) підібрати декілька прикладів на рекурсивні алгоритми для розробки тренажеру;

4) розробити алгоритм тренажеру;

5) створити блок-схему алгоритму тренажеру;

6) розробити програму;

7) описати створену програму;

8) передати програму на впровадження.

Об’єкт дослідження – рекурсія (пряма та непряма), рекурсивні функції, рекурсивні алгоритми, рекурентні співвідношення.

Предмет дослідження – рекурсивні алгоритми.

Методи, використані для розробки програми, – мова С++, середовище програмування Borland Builder.

Особистий вклад – самостійно створені алгоритм тренажеру, блок-схема алгоритму, програма.

Практична новизна роботи – створено новий, унікальний тренажер з теми «Рекурсивні алгоритми». Тренажер використовую пряму та непряму рекурсію.

Ступінь готовності, готовність до впровадження результатів – програма повністю готова, передана на провадження у дистанційний курс «Алгоритми і структури даних» Полтавського університету економіки і торгівлі.

Магістерська робота складається з чотирьох частин – постановки задачі, інформаційного огляду, теоретичної та практичної частин. В першій частині викладено технічне завдання; в другій – здійснено огляд візуалізаторів рекурсивних алгоритмів; в третій – подано алгоритм тренажеру та блок-схему; в четвертій – показано, як працює програма, та пояснено, як вона створювалась.

Обсяг пояснювальної записки без додатків – 60 сторінок.

1. ПОСТАНОВКА ЗАДАЧІ

Завдання дипломного проектування полягає в тому, щоб створити комп'ютерний тренажер з теми «Рекурсивні алгоритми» для дистанційного курсу «Алгоритми і структури даних» Полтавського університету економіки і торгівлі [1-2].

Для виконання завдання взяти подані нижче алгоритми (приклади, задачі) рекурсивного характеру.

Тут приклади 1 та 3 на використовують пряму рекурсію, приклад 2 – непряму [3-4].

Приклад 1.

Алгоритм обчислення значення функції $F(n)$*, де* n *– натуральне число, задано співвідношеннями:*

$F(1)=1,\ F(2)=1,$

$F(n)=F(n-2)\cdot(n-1)+2$, *якщо* $n>2$.

Чому дорівнює значення функції $F(8)$ *?*

Приклад 2.

Алгоритм обчислення значення функцій $F(w)$ *і* $Q(w)$*, де* w *– натуральне число, задано співвідношеннями:*

$F(1)=1,\ Q(1)=1,$

$F(w)=F(w-1)+2\cdot Q(w-1)$, *якщо* $w>1,$

$Q(w)=Q(w-1)-2\cdot F(w-1)$, *якщо* $w>1.$

Чому дорівнює значення $F(5)+Q(5)$ *?*

Приклад 3.

Є рекурсивний алгоритм:

```
procedure F (n: integer);
begin
   writeln (n);
   if (n < 6) then
   begin
     F (n + 2);
     F (n * 3);
   end;
end;
```

Знайти суму чисел, які будуть виведені при виклику $F(2)$.

Розробити алгоритм тренажеру, виконавши трасування взятих алгоритмів або використавши інші методи знаходження відповіді.

За алгоритмом тренажеру створити блок-схему та програму.

Програму перевірити на працездатність, протестувати.

Створити документацію для програми.

Передати програму на впровадження у відповідний дистанційний курс.

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Огляд візуалізаторів, які демонструють роботу рекурсивних алгоритмів

На сайті університету Сан-Франциско http://www.cs.usfca.edu/~galles/visualization/Algorithms.html представлена візуалізація трьох алгоритмів, які використовують рекурсію: 1) запис рядка у зворотному порядку; 2) знаходження факторіалу числа; 3) задача про вісім ферзів.

Візуалізація рекурсивного **алгоритму, що записує рядок у зворотному порядку,** розташовується за адресою http://www.cs.usfca.edu/~galles/visualization/RecReverse.html.

Рядок записують у поле та натискають кнопку «Reverse» (записати у зворотному порядку). Далі йде трасування алгоритму, а справа від нього показується у картинках (рис. 2.1-2.2), як алгоритм працює.

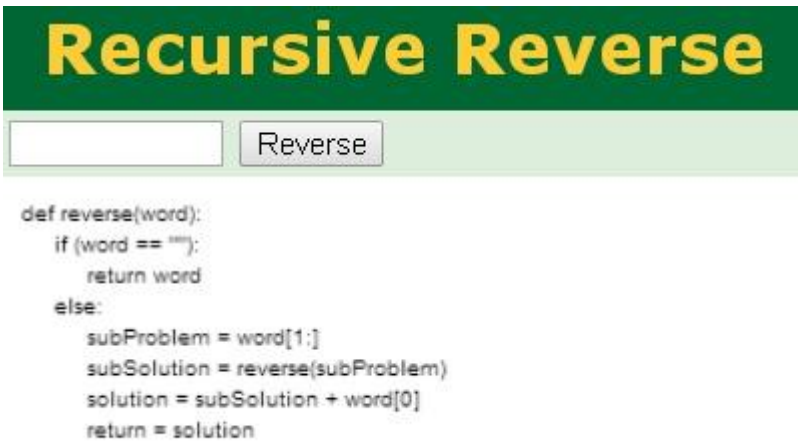

Рисунок 2.1 – Запис рядка у зворотному порядку (початок візуалізації)

Візуалізація рекурсивного **алгоритму, що обчислює факторіал числа,** розташовується за адресою http://www.cs.usfca.edu/~galles/visualization/RecFact.html.

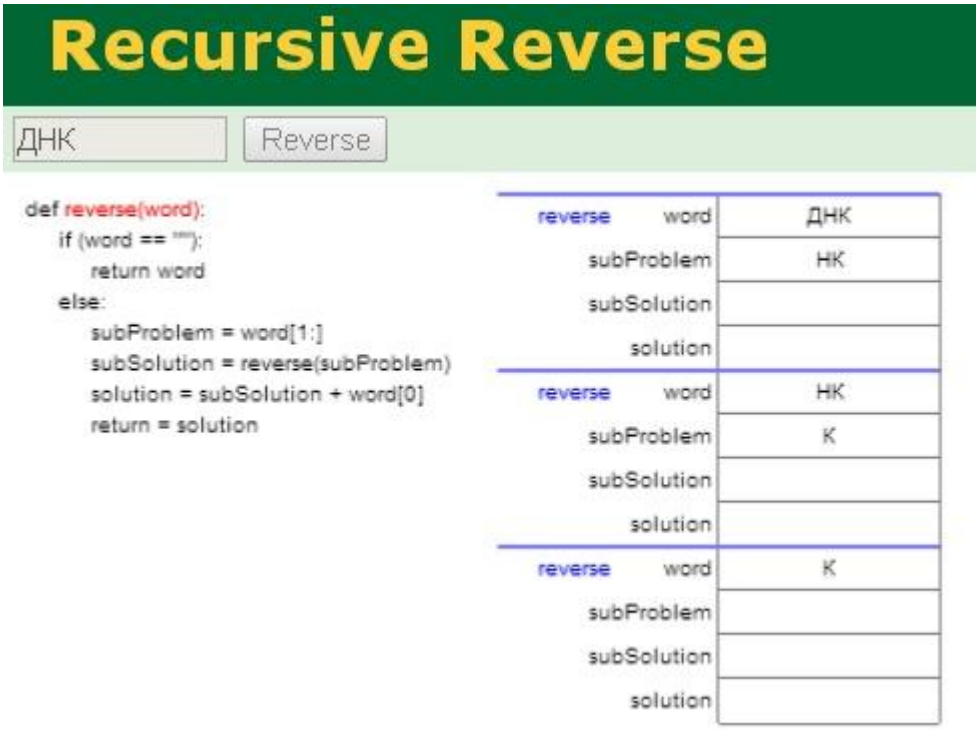

Рисунок 2.2 – Запис рядка у зворотному порядку (робота візуалізації)

У поле вводять число, факторіал якого слід знайти (рис. 2.3). Після цього натискають кнопку «Factorial» (рис. 2.3-2.4).

Рисунок 2.3 – Обчислення факторіалу числа (початок візуалізації)

Візуалізація рекурсивного **алгоритму, що розв'язує задачу про вісім ферзів,** розташовується за адресою https://www.cs.usfca.edu/~galles/visualization/RecQueens.html.

Вхідними даними (рис. 2.5) є розмір сторони шахової дошки (від 1 до 8). Далі натискається кнопка «Queens» (рис. 2.6-2.7).

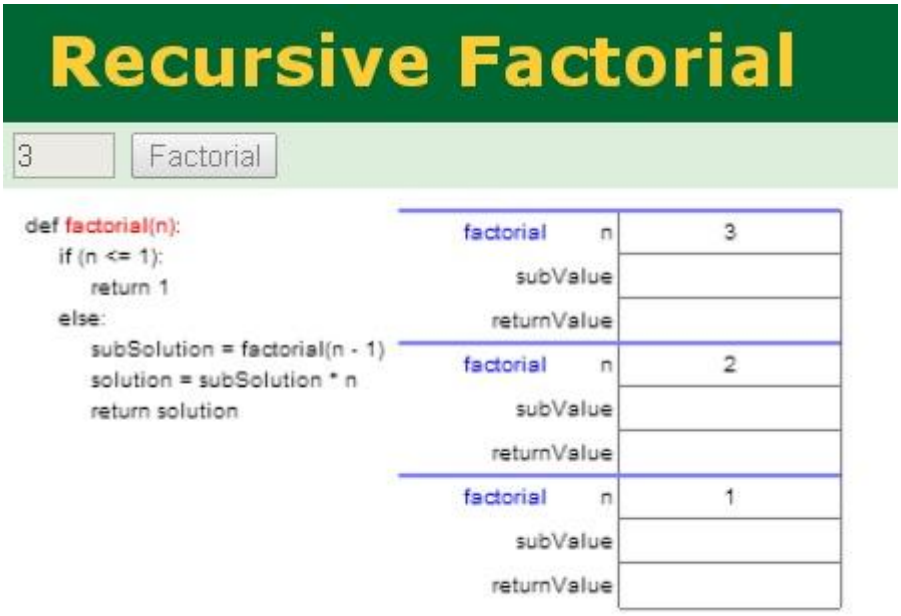

Рисунок 2.4 – Обчислення факторіалу числа (робота візуалізації)

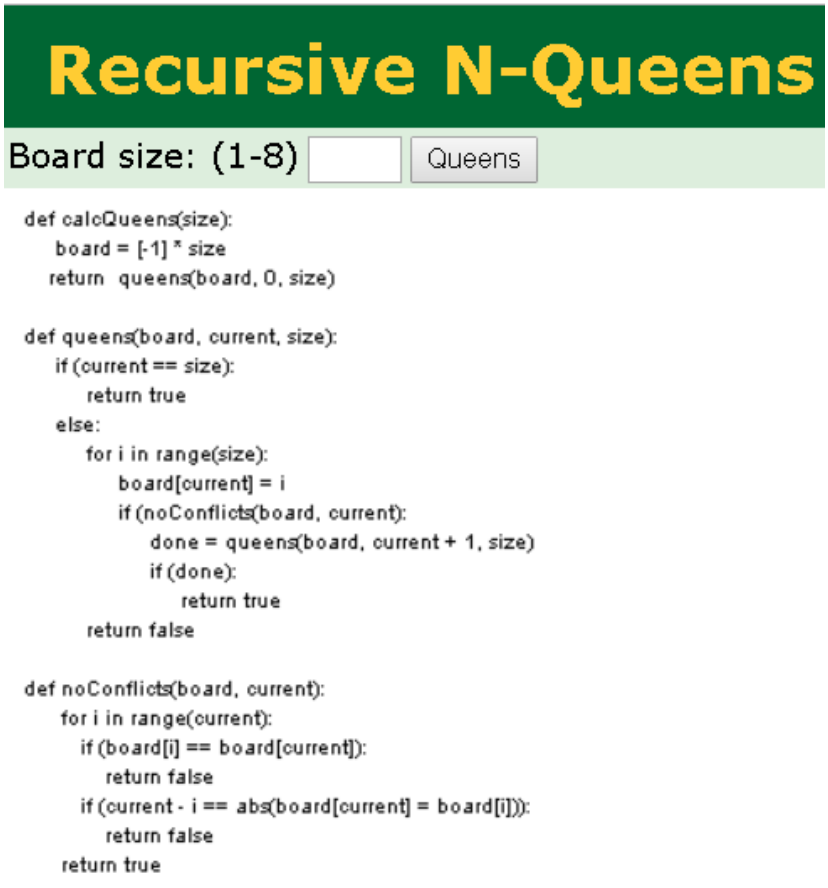

Рисунок 2.5 – Задача про вісім ферзів (початок візуалізації)

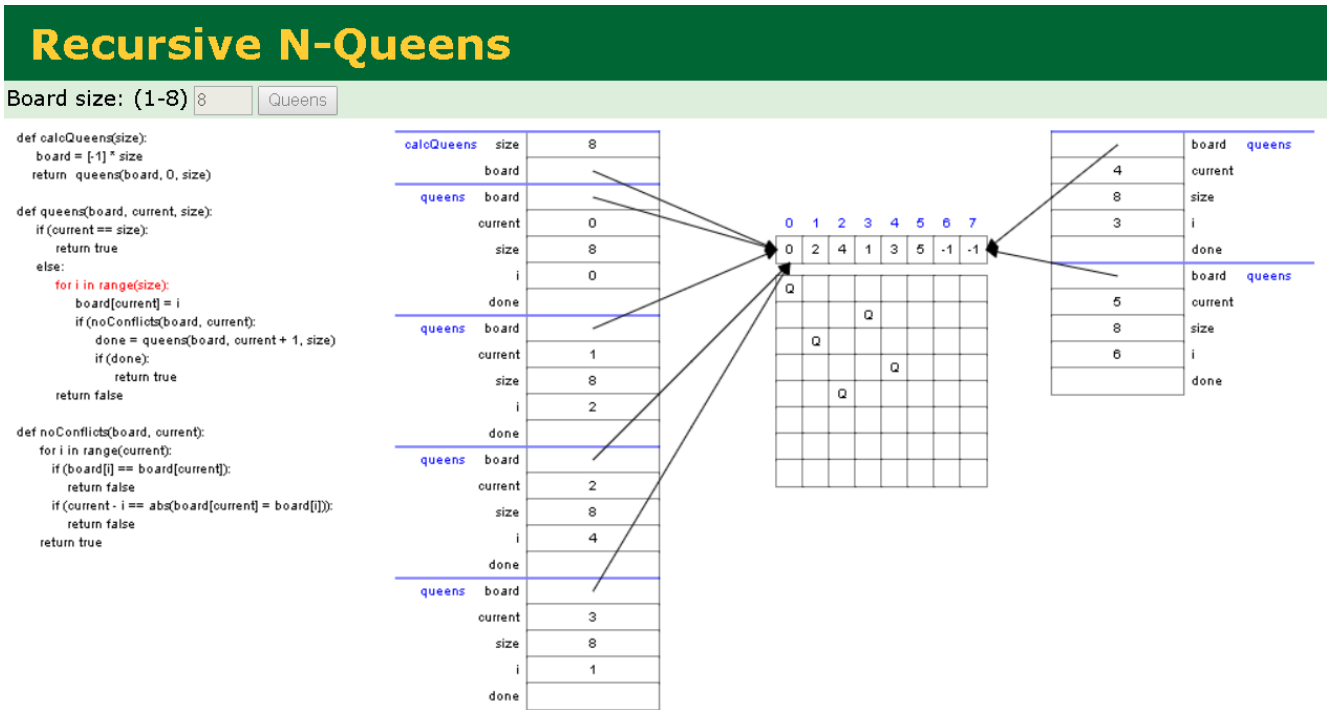

Рисунок 2.6 – Задача про вісім ферзів (робота візуалізації)

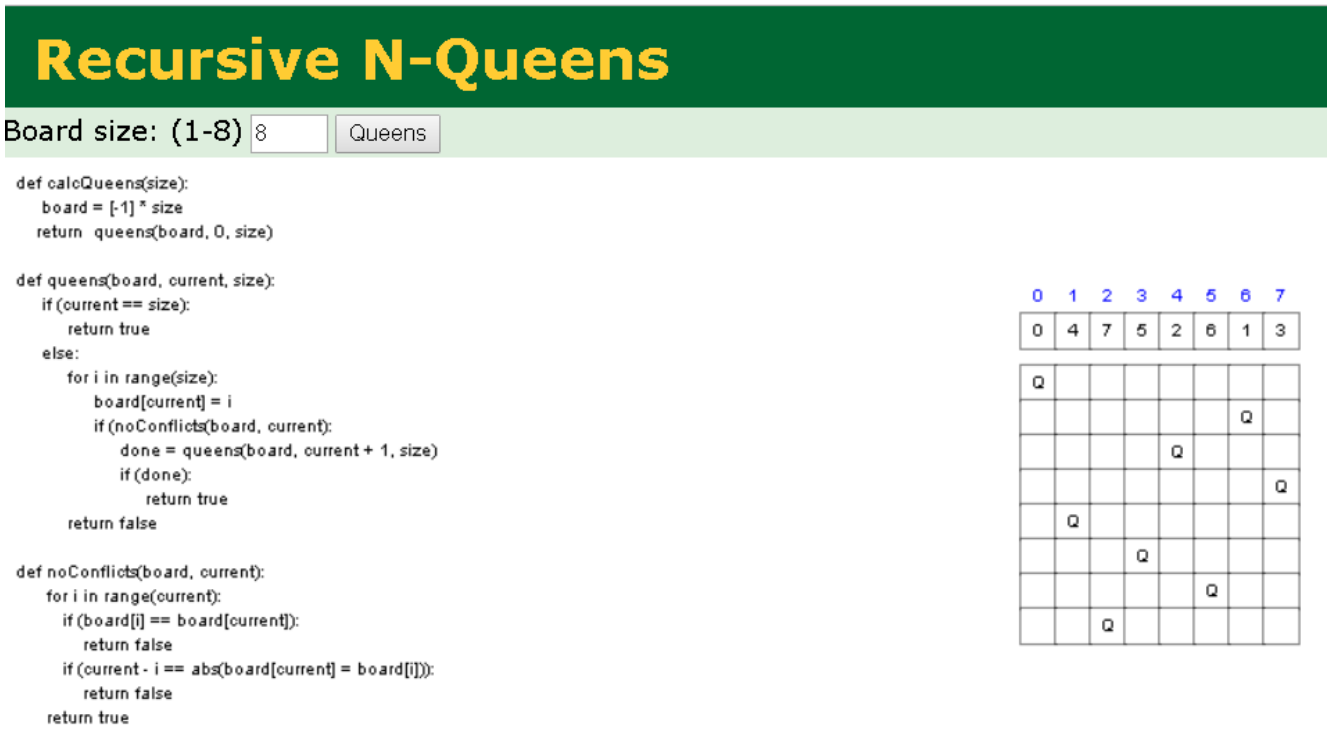

Рисунок 2.7 – Задача про вісім ферзів (відповідь)

Ресурс https://visualgo.net/ru/recursion демонструє візуалізацію таких рекурсивних алгоритмів:

1) числа Фібоначчі (рис. 2.8);

2) факторіал числа (рис. 2.9);

3) каталанські числа;

4) розв’язок задачі комівояжера (рис. 2.10);

5) знаходження найбільшого спільного дільника (рис. 2.11);

6) знаходження біноміального коефіцієнту тощо.

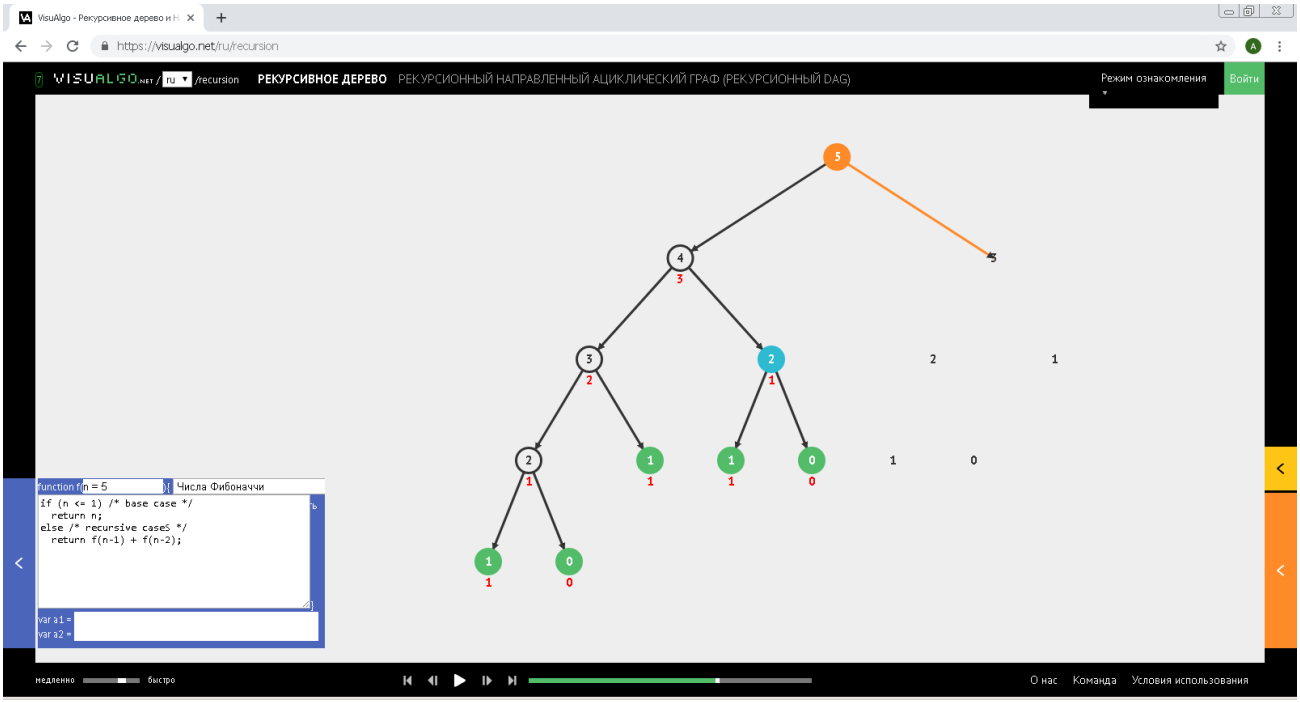

Рисунок 2.8 – Числа Фібоначчі (робота анімації)

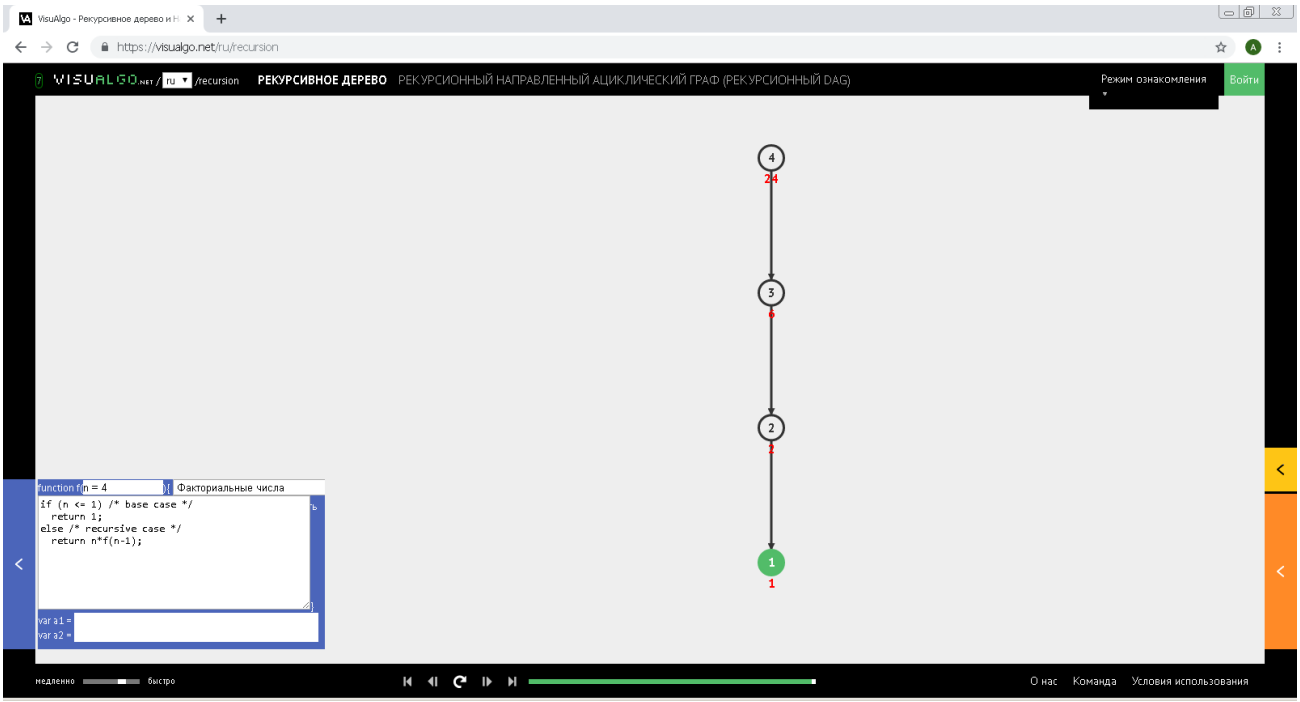

Рисунок 2.9 – Факторіал числа (відповідь)

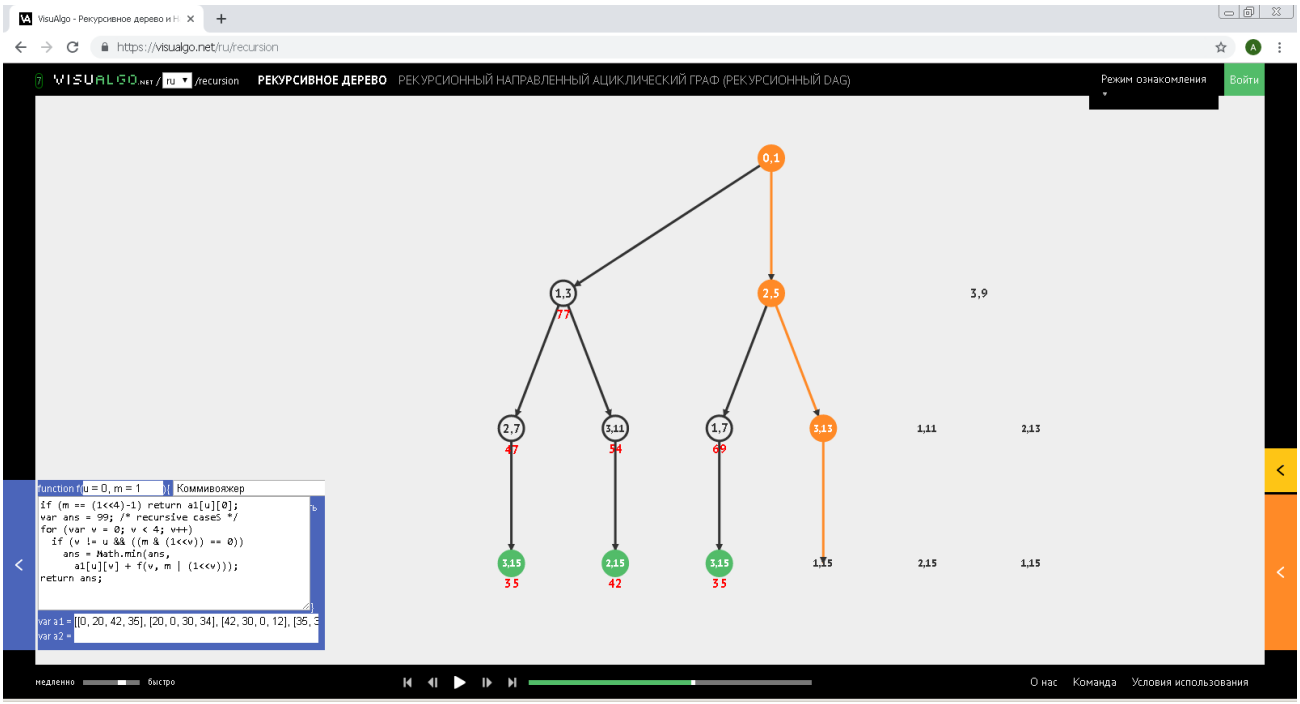

Рисунок 2.10 – Задача комівояжера (робота анімації)

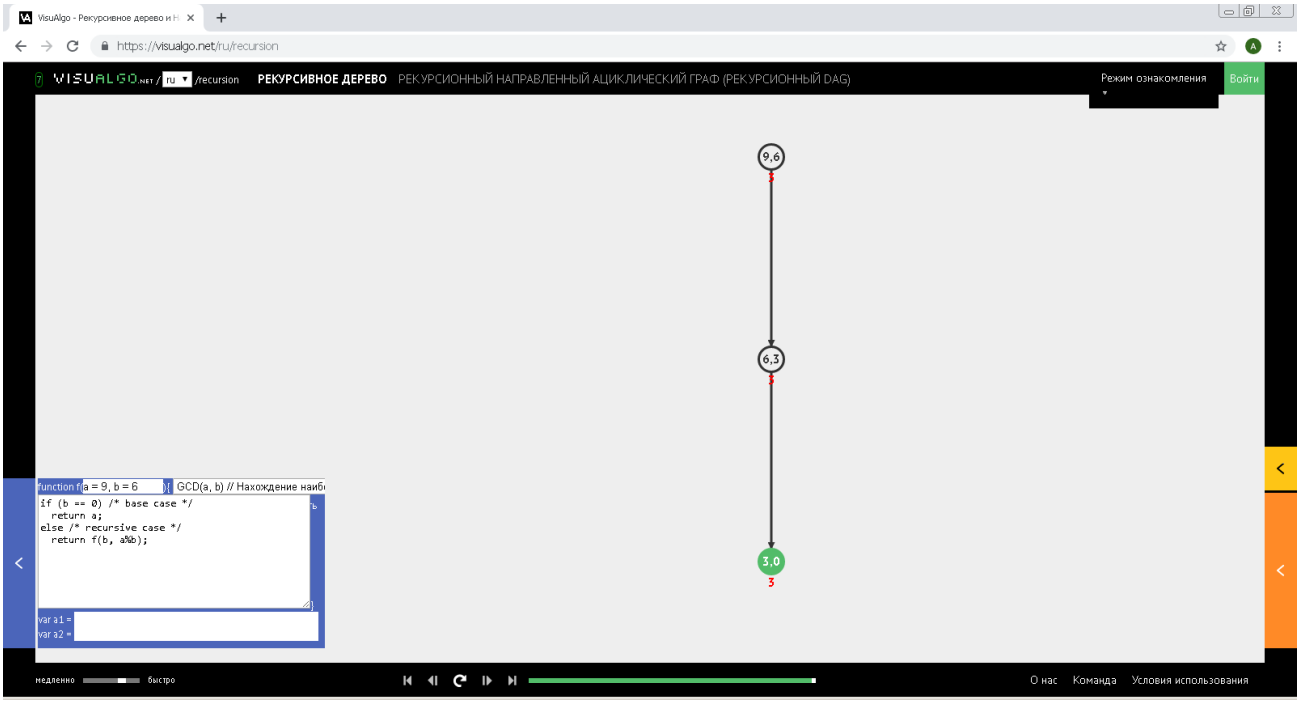

Рисунок 2.11 – Знаходження найбільшого спільного дільника (робота анімації)

2.2. Позитивні аспекти розглянутих візуалізаторів

Анімації університету Сан-Франциско:

1) Є запис алгоритму на псевдокоді.

2) Рядок, що виконується, виділяється червоним кольором.

3) Є ілюстрація роботи алгоритму, що відповідає поточному моменту виконання коду.

4) Можливість зміни швидкості роботи анімації (повільніше, швидше, рис. 2.12).

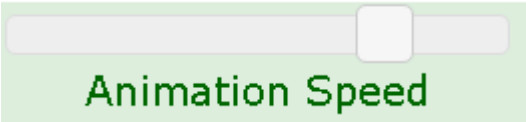

Рисунок 2.12 – Повзунок, що регулює швидкість

5) Є кнопка «Пауза», яка зупиняє анімацію (рис. 2.13).

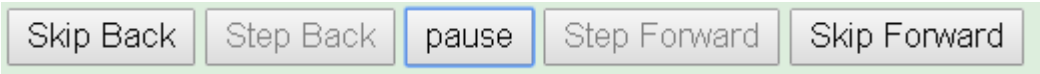

Рисунок 2.13 – Кнопки для керування анімацією

6) Є можливість покрокової прогонки анімації, з можливістю повернутися на крок назад або вперед (рис. 2.13).

7) Анімації універсальні, розраховані на будь-які коректні вхідні дані (можна ввести будь-який рядок; будь-яке числа; розмір сторони дошки від 1 до 8).

8) Фіксується стан анімації (на паузі, завершена, працює тощо, рис. 2.14).

Рисунок 2.14 – Позначки для стану анімації

Анімації ресурсу https://visualgo.net/ru/recursion:

1) Є запис алгоритму на мові C++.

2) Великий вибір рекурсивних алгоритмів (рис. 2.15).

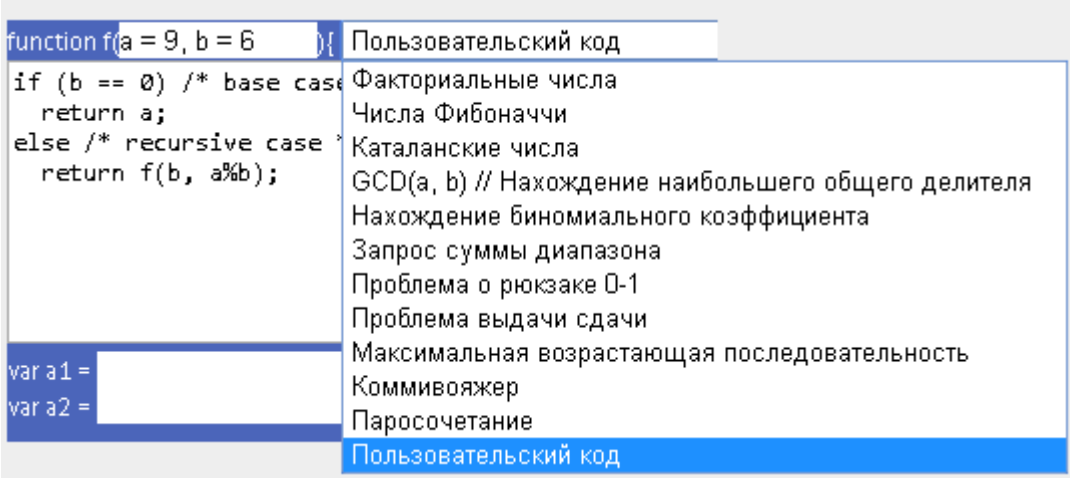

Рисунок 2.15 – Перелік алгоритмів

3) Є можливість вводу інших вхідних даних (рис. 2.15).

4) Є можливість написання свого рекурсивного алгоритму (рис. 2.15).

5) Є можливість зміни швидкості роботи анімації (використовуючи відповідний повзунок, рис. 2.16).

Рисунок 2.16 – Повзунок, що регулює швидкість

6) Є можливість покрокової прогонки анімації, з можливістю повернутися на крок назад або вперед, є кнопка «Пауза» (рис. 2.17).

Рисунок 2.17 – Кнопки для керування анімацією

7) Є можливість вибору іншої мови (рис. 2.18-2.19).

8) Можна перейти до ознайомчою лекції з цієї теми (рис. 2.20).

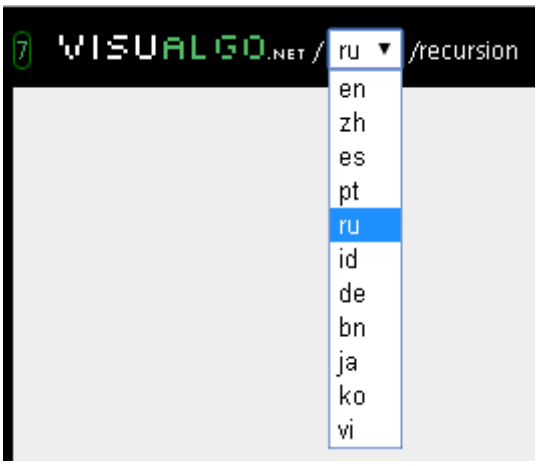

Рисунок 2.18 – Вибір мови

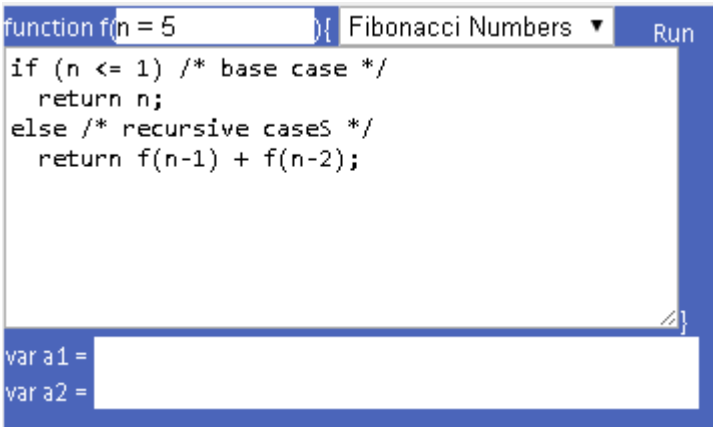

Рисунок 2.19 – Алгоритм з використанням англійської мови

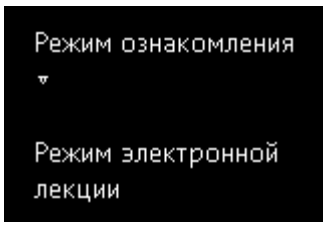

Рисунок 2.20 – Перехід до лекції

2.3. Недоліки розглянутих візуалізаторів

Оскільки анімації розташовані інтернеті, то доступ до них може бути припинений в будь-який момент часу.

На ресурсі *https://visualgo.net/ru/recursion* не показується, який рядок алгоритму виконується в даний момент часу і як це пов’язано з картинкою.

2.4. Необхідність та актуальність теми

Хоча в мережі інтернет представлені візуалізатори рекурсивних алгоритмів, вони не повністю задіють користувача. Візуалізатори лише демонструють, як рекурсивний алгоритм працює. Якщо ми хочемо занурити студента в цю тему більш активно, то ми хочемо від нього більш активних дій, ніж пасивний перегляд анімації.

Цю задачу вирішують тренажери, які навчають певній темі та вимагають від користувача активних дій. При роботі з тренажером студент змушений активніше залучатися до вивчення теми. І як результат, у мозку студента спрацьовує більше розумових процесів (читання, осмислення, введення відповідей тощо). І тема опрацьовується на більш глибокому рівні.

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Алгоритм тренажеру

Тренажер містить декілька прикладів. Приклади (умови завдань) для тренажеру були взяти з ресурсів [3-4] з теми «Рекурсія. Рекурсивні функції».

Для кожного питання у випадку невірної відповіді виникає повідомлення «Помилка!», далі йде пояснення помилки. У випадку вірності здійснюється перехід на наступний крок прикладу.

Для кожного прикладу умова видима під час усіх питань.

Розглянемо алгоритми для кожного прикладу.

Приклад 1.

Алгоритм обчислення значення функції $F(n)$, де n – натуральне число, задано співвідношеннями:

$F(1)=1,\ F(2)=1,$

$F(n)=F(n-2)\cdot(n-1)+2$, якщо $n>2$.

Чому дорівнює значення функції $F(8)$?

Крок 1.

Введіть числа у клітинки:

$F(8)=F(8-2)\cdot(8-1)+2=F(\square)\cdot\square+\square$.

Правильна відповідь: $F(8)=F(8-2)\cdot(8-1)+2=F(6)\cdot 7+2$.

Крок 2.

Введіть числа у клітинки:

$F(6)=F(\square-\square)\cdot(\square-\square)+\square=F(\square)\cdot\square+\square$.

Правильна відповідь: $F(6)=F(6-2)\cdot(6-1)+2=F(4)\cdot 5+2$.

Крок 3.

Введіть числа у клітинки:

$F(4)=F(\square-\square)\cdot(\square-\square)+\square=F(\square)\cdot\square+\square=$

$=\square\cdot\square+\square=\square+\square=\square$.

Правильна відповідь: $F(4)=F(4-2)\cdot(4-1)+2=F(2)\cdot 3+2=1\cdot 3+2=$

$=3+2=5$.

Крок 4.

Введіть числа у клітинки, враховуючи, що $F(4)=5$:

$F(6)=F(4)\cdot 5+2=\square\cdot 5+2=\square$.

Правильна відповідь: $F(6)=F(4)\cdot 5+2=5\cdot 5+2=27$.

Крок 5.

Введіть числа у клітинки, враховуючи, що $F(6)=27$:

$F(8)=F(6)\cdot 7+2=\square\cdot 7+2=\square$.

Правильна відповідь: $F(8)=F(6)\cdot 7+2=27\cdot 7+2=56$.

<u>Приклад 2.</u>

Алгоритм обчислення значення функцій $F(w)$ і $Q(w)$, де w – натуральне число, задано співвідношеннями:

$F(1)=1,\ Q(1)=1,$

$F(w)=F(w-1)+2\cdot Q(w-1)$, якщо $w>1$,

$Q(w)=Q(w-1)-2\cdot F(w-1)$, якщо $w>1$.

Чому дорівнює значення $F(5)+Q(5)$?

Крок 1.

Введіть числа у клітинки:

$F(5)=F(5-1)+2\cdot Q(5-1)=F(\square)+2\cdot Q(\square)$.

Правильна відповідь: $F(5)=F(5-1)+2\cdot Q(5-1)=F(4)+2\cdot Q(4)$.

Крок 2.

Введіть числа у клітинки:

$F(4)=F(\square-\square)+2\cdot Q(\square-\square)=F(\square)+2\cdot Q(\square)$.

Правильна відповідь: $F(4)=F(4-1)+2\cdot Q(4-1)=F(3)+2\cdot Q(3)$.

Крок 3.

Введіть числа у клітинки:

$F(3)=F(\square-\square)+2\cdot Q(\square-\square)=F(\square)+2\cdot Q(\square)$

Правильна відповідь: $F(3)=F(3-1)+2\cdot Q(3-1)=F(2)+2\cdot Q(2)$.

Крок 4.

Введіть числа у клітинки:

$F(2)=F(\square-\square)+2\cdot Q(\square-\square)=F(\square)+2\cdot Q(\square)=$

$=\square+\square\cdot\square=\square$.

Правильна відповідь: $F(2)=F(2-1)+2\cdot Q(2-1)=F(1)+2\cdot Q(1)=$

$=1+2\cdot 1=3$

Крок 5.

Введіть числа у клітинки:

$Q(5)=Q(\square-\square)-2\cdot F(\square-\square)=Q(\square)-2\cdot F(\square)$.

Правильна відповідь: $Q(5)=Q(5-1)-2\cdot F(5-1)=Q(4)-2\cdot F(4)$.

Крок 6.

Введіть числа у клітинки:

$Q(4)=Q(\square-\square)-2\cdot F(\square-\square)=Q(\square)-2\cdot F(\square).$

Правильна відповідь: $Q(4)=Q(4-1)-2\cdot F(4-1)=Q(3)-2\cdot F(3).$

Крок 7.

Введіть числа у клітинки:

$Q(3)=Q(\square-\square)-2\cdot F(\square-\square)=Q(\square)-2\cdot F(\square).$

Правильна відповідь: $Q(3)=Q(3-1)-2\cdot F(3-1)=Q(2)-2\cdot F(2).$

Крок 8.

Введіть числа у клітинки:

$Q(2)=Q(\square-\square)-2\cdot F(\square-\square)=Q(\square)-2\cdot F(\square)=$

$=\square-\square\cdot\square=\square\ .$

Правильна відповідь: $Q(2)=Q(2-1)-2\cdot F(2-1)=Q(1)-2\cdot F(1)=1-2\cdot 1=$

$=-1.$

Крок 9.

Введіть числа у клітинки:

$F(3)=F(2)+2\cdot Q(2)=\square+2\cdot\square=\square-\square=\square;$

$Q(3)=Q(2)-2\cdot F(2)=\square-2\cdot\square=\square-\square=\square.$

Правильна відповідь: $F(3)=F(2)+2\cdot Q(2)=3+2\cdot(-1)=3-2=1;$

$Q(3)=Q(2)-2\cdot F(2)=-1-2\cdot 3=-1-6=-7.$

Крок 10.

Введіть числа у клітинки:

$F(4)=F(3)+2\cdot Q(3)=\square+2\cdot\square=\square-\square=\square;$

$Q(4)=Q(3)-2\cdot F(3)=\square-2\cdot\square=\square-\square=\square.$

Правильна відповідь: $F(4)=F(3)+2\cdot Q(3)=1+2\cdot(-7)=1-14=-13;$

$Q(4)=Q(3)-2\cdot F(3)=-7-2\cdot 1=-7-2=-9.$

Крок 11.

Введіть числа у клітинки:

$F(5)=F(4)+2\cdot Q(4)=\square+2\cdot\square=\square-\square=\square;$

$Q(5)=Q(4)-2\cdot F(4)=\square-2\cdot\square=\square+\square=\square.$

Правильна відповідь: $F(5)=F(4)+2\cdot Q(4)=-13+2\cdot(-9)=-13-18=-31;$

$Q(5)=Q(4)-2\cdot F(4)=-9-2\cdot(-13)=-9+26=17.$

Крок 12.

Введіть числа у клітинки:

$F(5)+Q(5)=\square+\square=\square.$

Правильна відповідь: $F(5)+Q(5)=-31+17=-14.$

Приклад 3.

Є рекурсивний алгоритм:

```
procedure F (n: integer);
begin
  writeln (n);
  if (n < 6) then
  begin
    F (n + 2);
    F (n * 3);
  end;
end;
```

Знайти суму чисел, які будуть виведені при виклику $F(2)$.

Крок 1.

Запишемо рекурентне співвідношення для загального випадку.

Позначимо через $F(n)$ суму чисел, яка буде виводитися при виклику $F(n)$.

$F(n) = n$ при $n \geq 6$;

$F(n) = n + F(n+2) + F(3n)$ при $n < 6$.

Введіть числа у клітинки:

$F(2) = \square + F(\square + \square) + F(\square \cdot \square) = \square + F(\square) + F(\square)$.

Правильна відповідь: $F(2) = 2 + F(2+2) + F(3 \cdot 2) = 2 + F(4) + F(6)$.

Крок 2.

Введіть числа у клітинки:

$F(4) = \square + F(\square + \square) + F(\square \cdot \square) = \square + F(\square) + F(\square)$.

Правильна відповідь: $F(4) = 4 + F(4+2) + F(3 \cdot 4) = 4 + F(6) + F(12)$.

Крок 3.

Введіть числа у клітинки:

$F(6) = \square$.

Правильна відповідь: $F(6) = 6$.

Крок 4.

Введіть числа у клітинки:

$F(12) = \square$.

Правильна відповідь: $F(12) = 12$.

Крок 5.

Введіть числа у клітинки:

$F(4) = 4 + F(6) + F(12) = \square + \square + \square = \square$.

Правильна відповідь: $F(4) = 4 + F(6) + F(12) = 4 + 6 + 12 = 22$.

Крок 6.

Введіть числа у клітинки:

$F(2) = 2 + F(4) + F(6) = \square + \square + \square = \square$.

Правильна відповідь: $F(2) = 2 + F(4) + F(6) = 2 + 22 + 6 = 30$.

3.2. Блок-схема алгоритму

На рис. 3.1-3.6 зображена блок-схема алгоритму (перших дев’яти кроків для прикладу 2).

Рисунок 3.1 – Блок-схема алгоритму

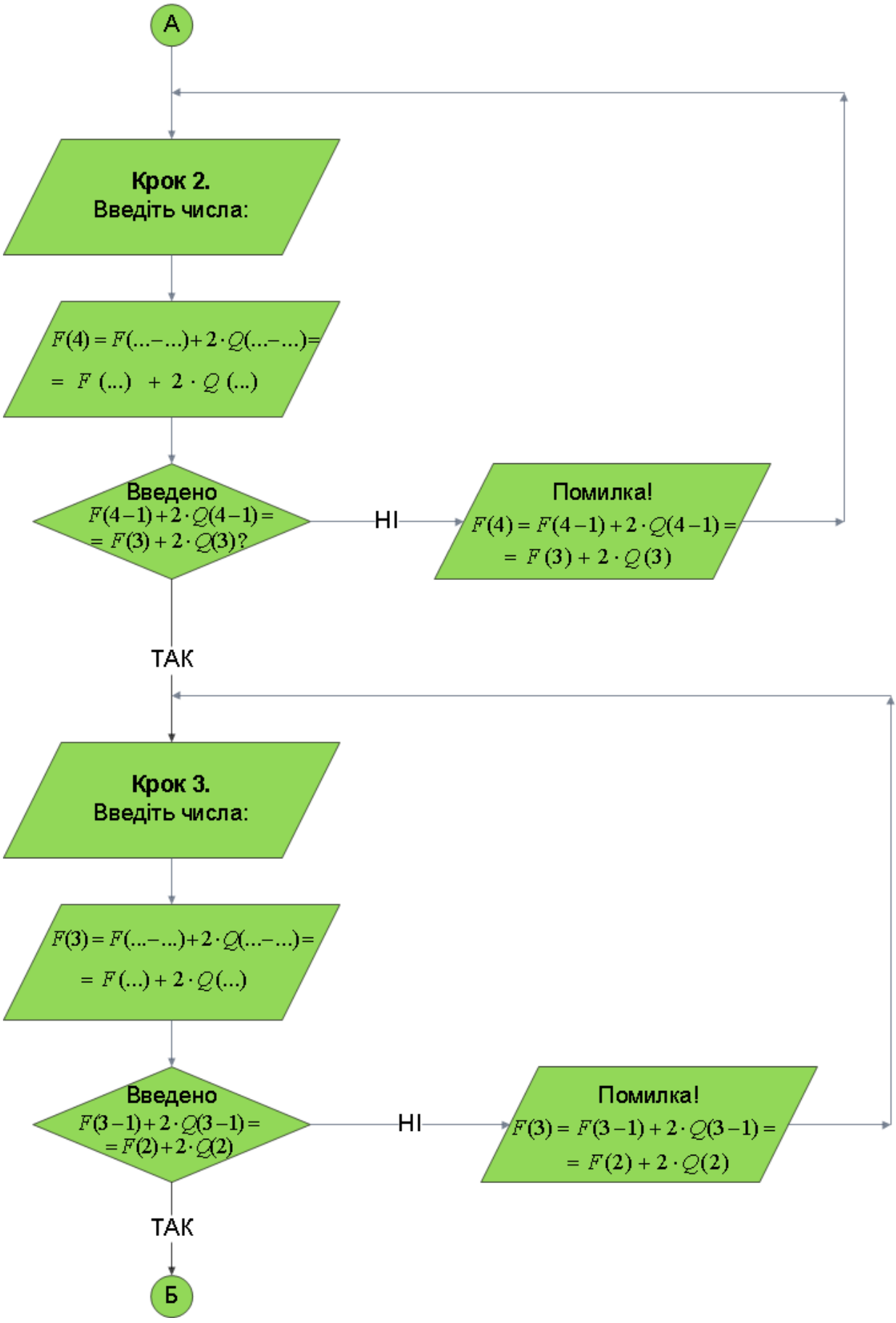

Рисунок 3.2 – Продовження блок-схеми алгоритму

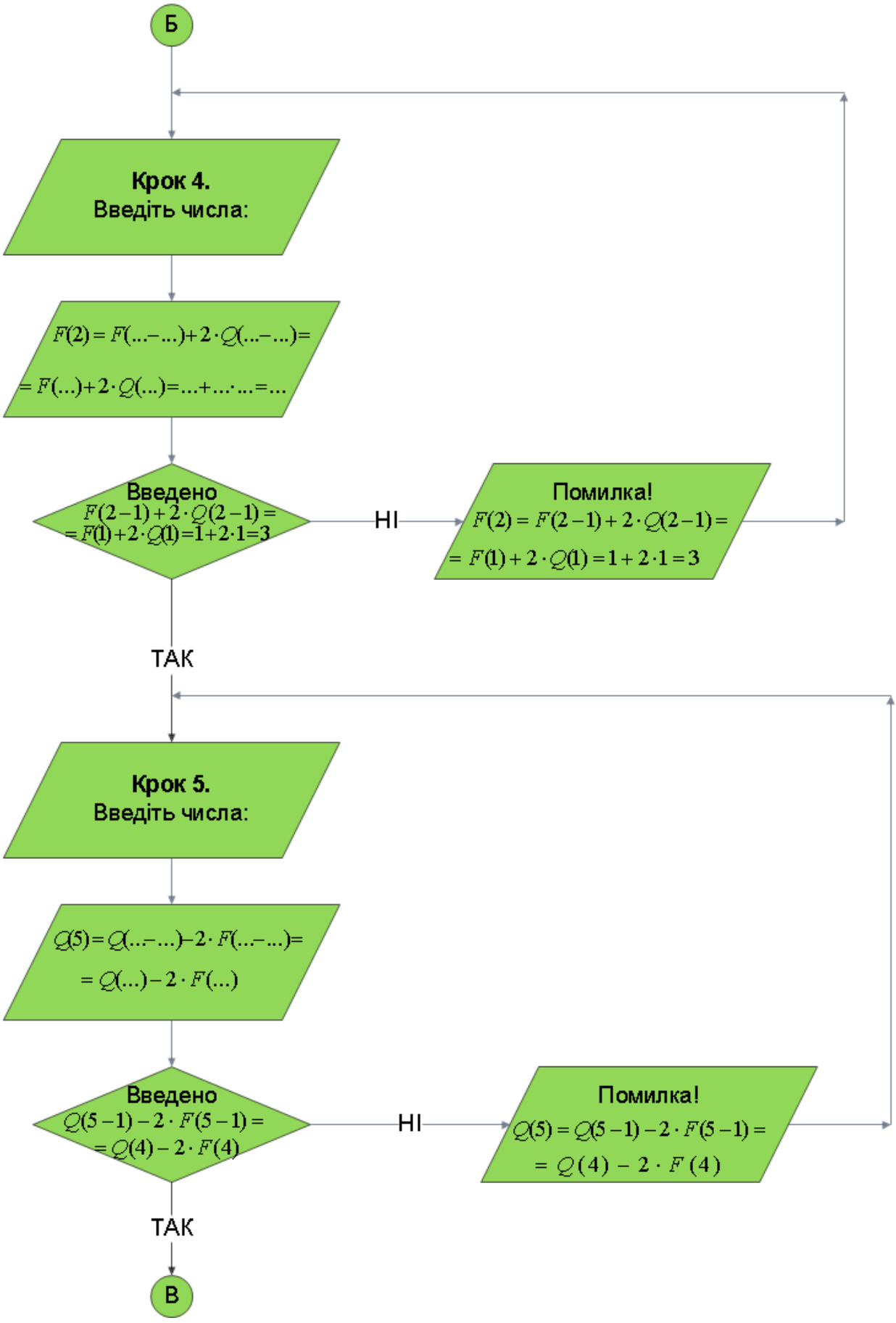

Рисунок 3.3 – Продовження блок-схеми алгоритму

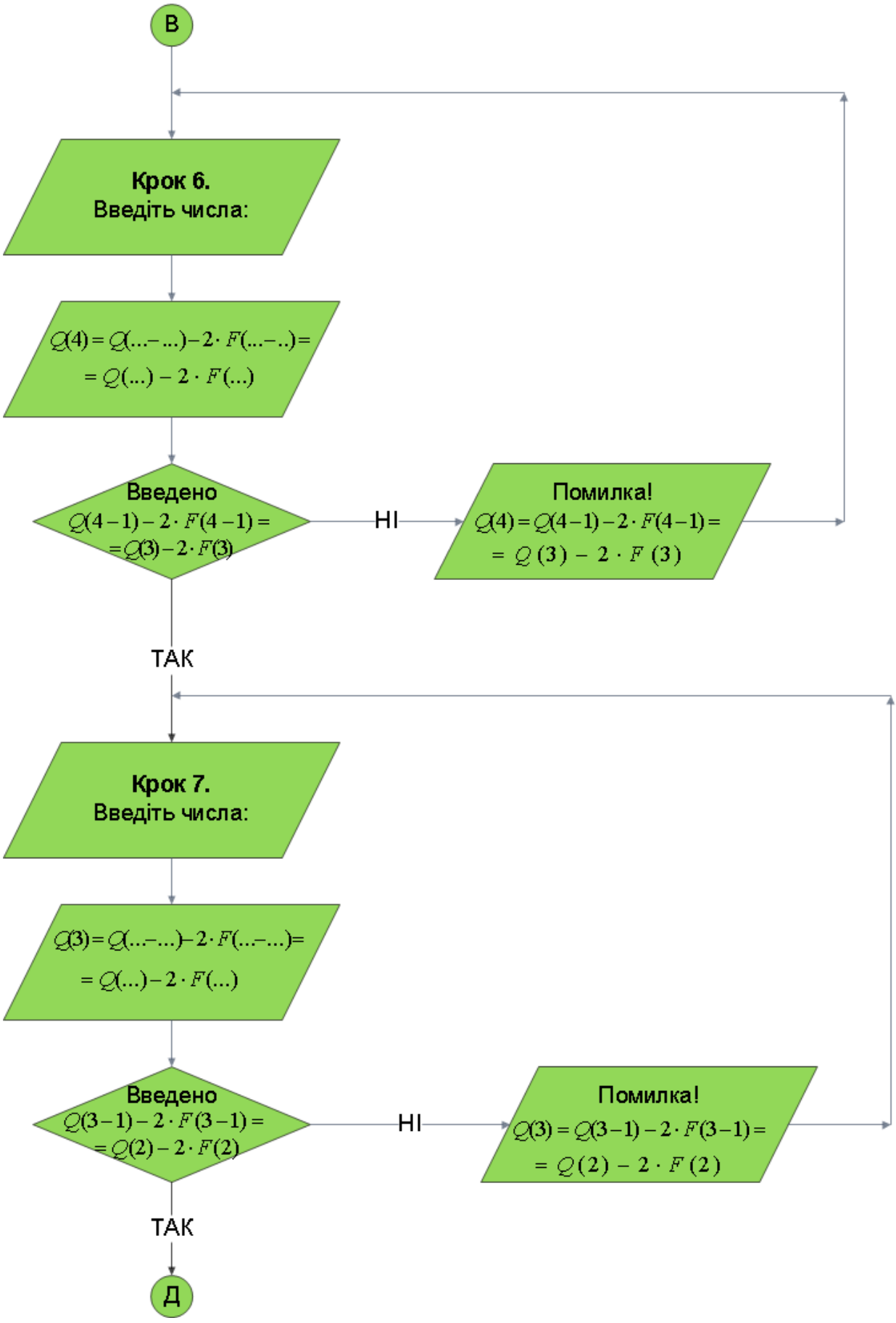

Рисунок 3.4 – Продовження блок-схеми алгоритму

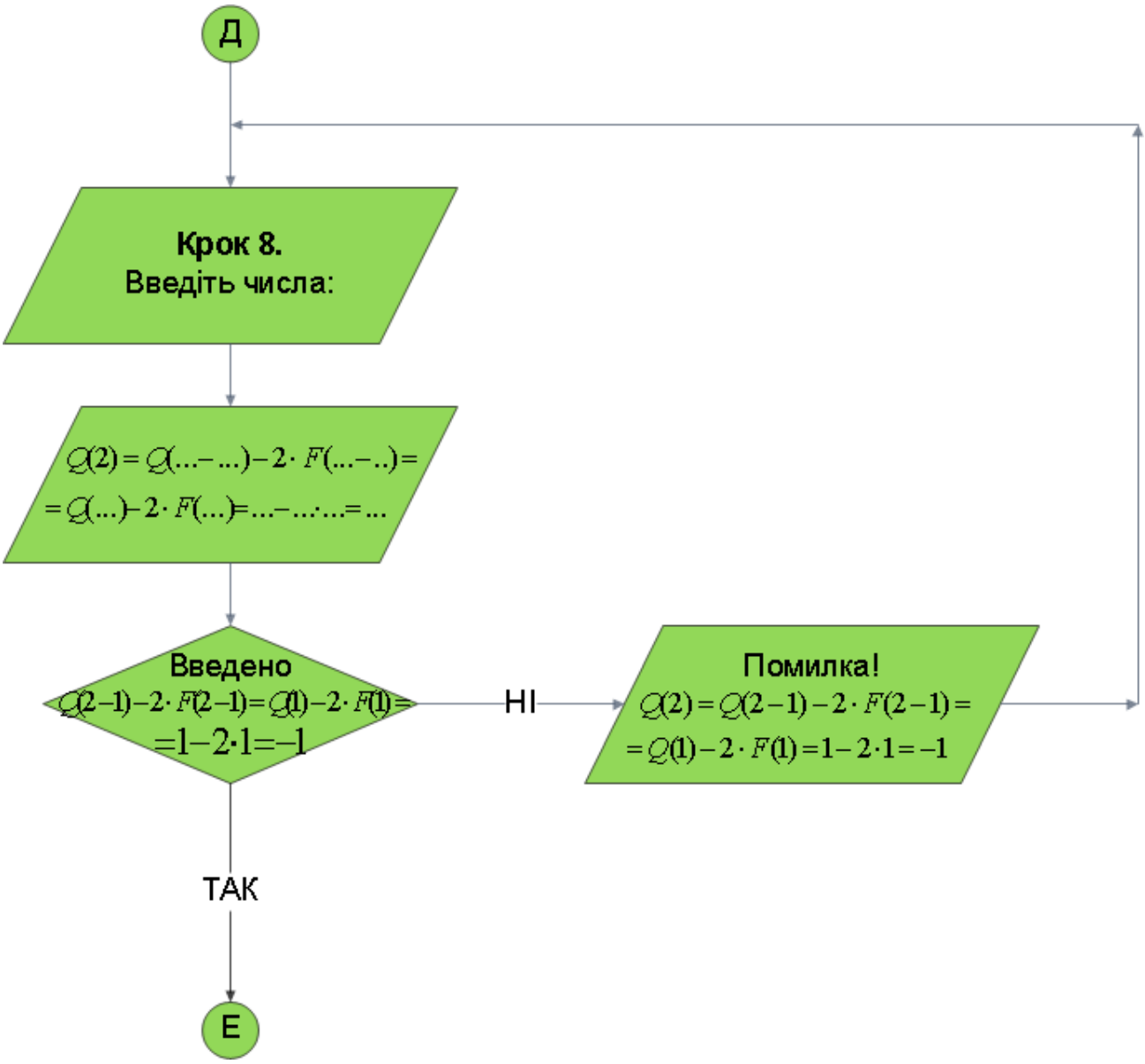

Рисунок 3.5 – Продовження блок-схеми алгоритму

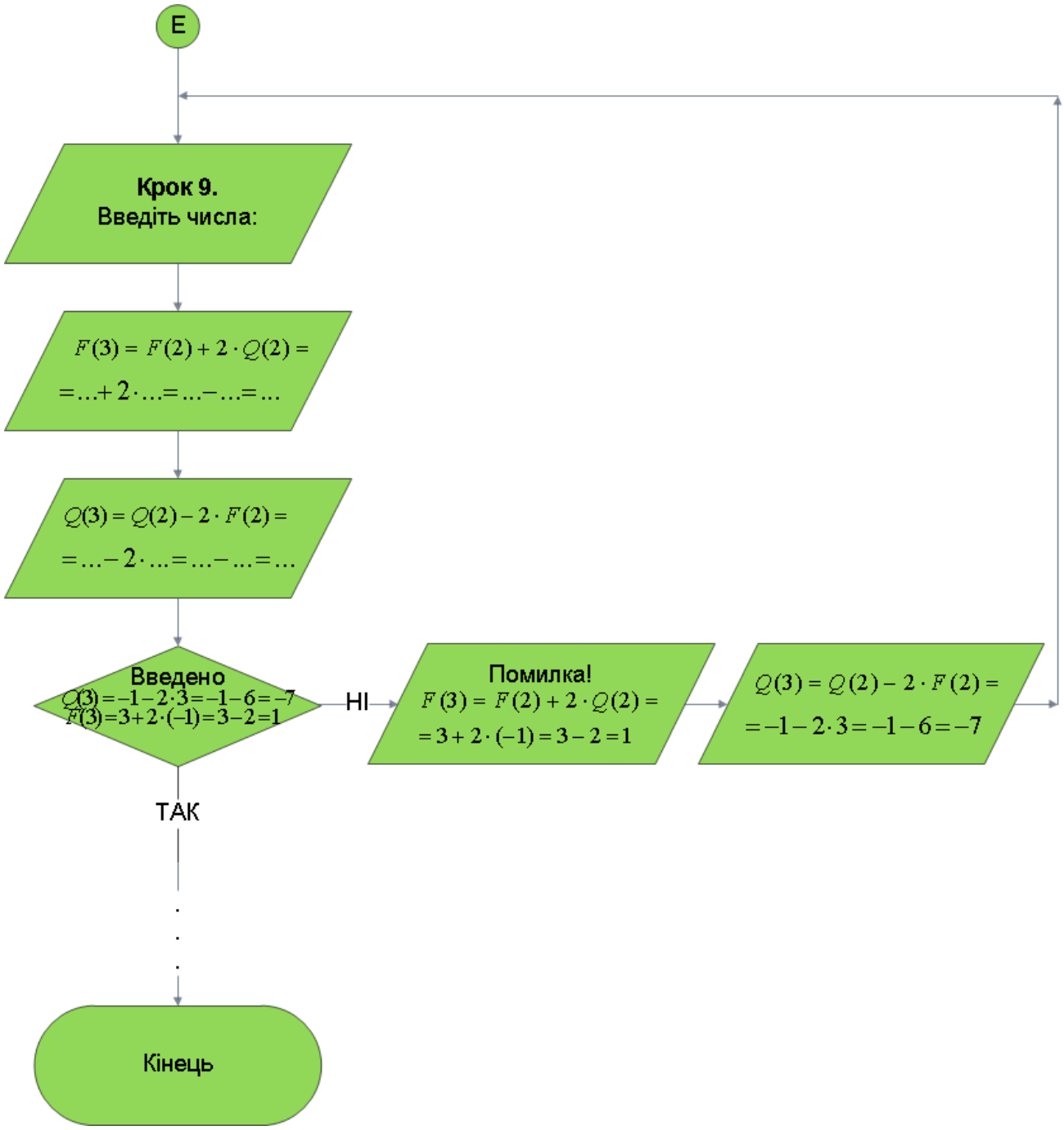

Рисунок 3.6 – Продовження блок-схеми алгоритму

4. ПРАКТИЧНА ЧАСТИНА

4.1. Інструкція по роботі з програмою

На рис. 4.1-4.35, А.1-А.12, продемонстровано, як працює програма.

Рисунок 4.1 – Вибір прикладів

Програма складається з 3 прикладів. Користувач обирає будь-який з них і розпочинає тренінг (рис. 4.1).

Ввівши відповідь, студент натискає кнопку «Перевірка». При помилці посередині вікна, яка виділена іншим кольором, з’являється пояснення помилки (див., наприклад, рис. 4.3). Помилку слід виправити вручну. Якщо відповідь вірна, відбувається перехід на наступний крок тренінгу.

Після проходження тренінгу з якогось з трьох прикладів користувач повертається на перше вікно програми (рис. 4.1). Далі він може пройти тренінг за іншим прикладом або знову за тим самим.

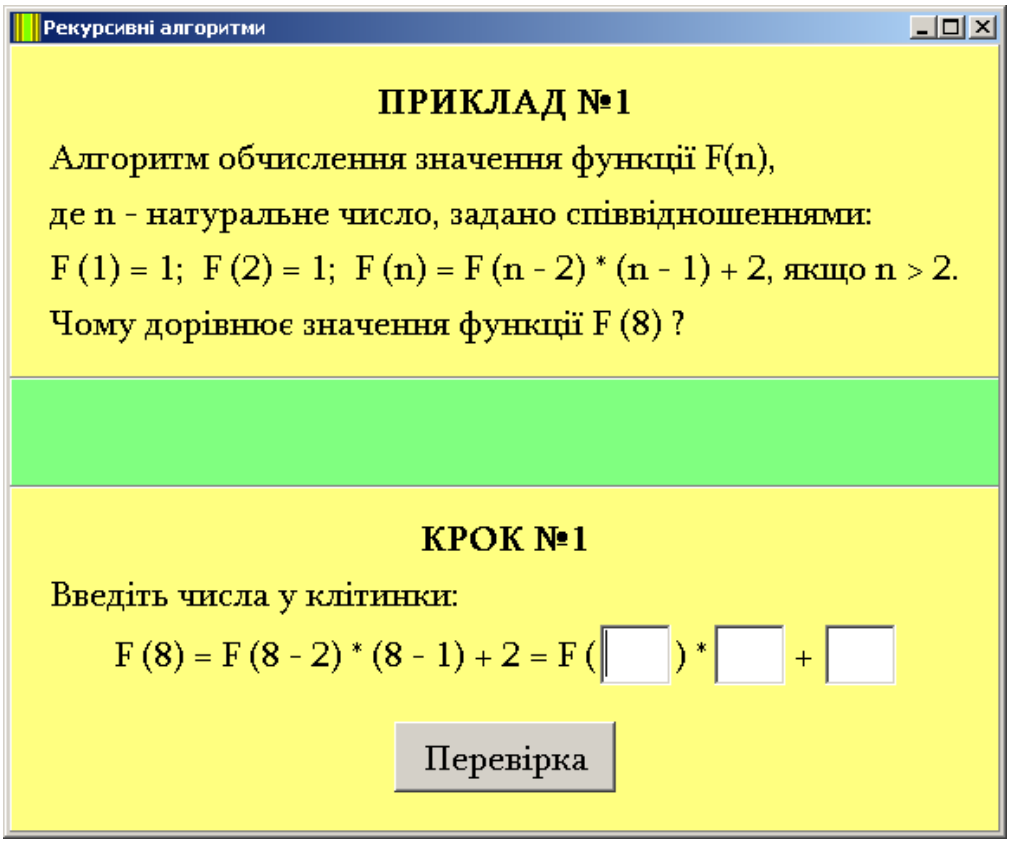

Рисунок 4.2 – Крок 1

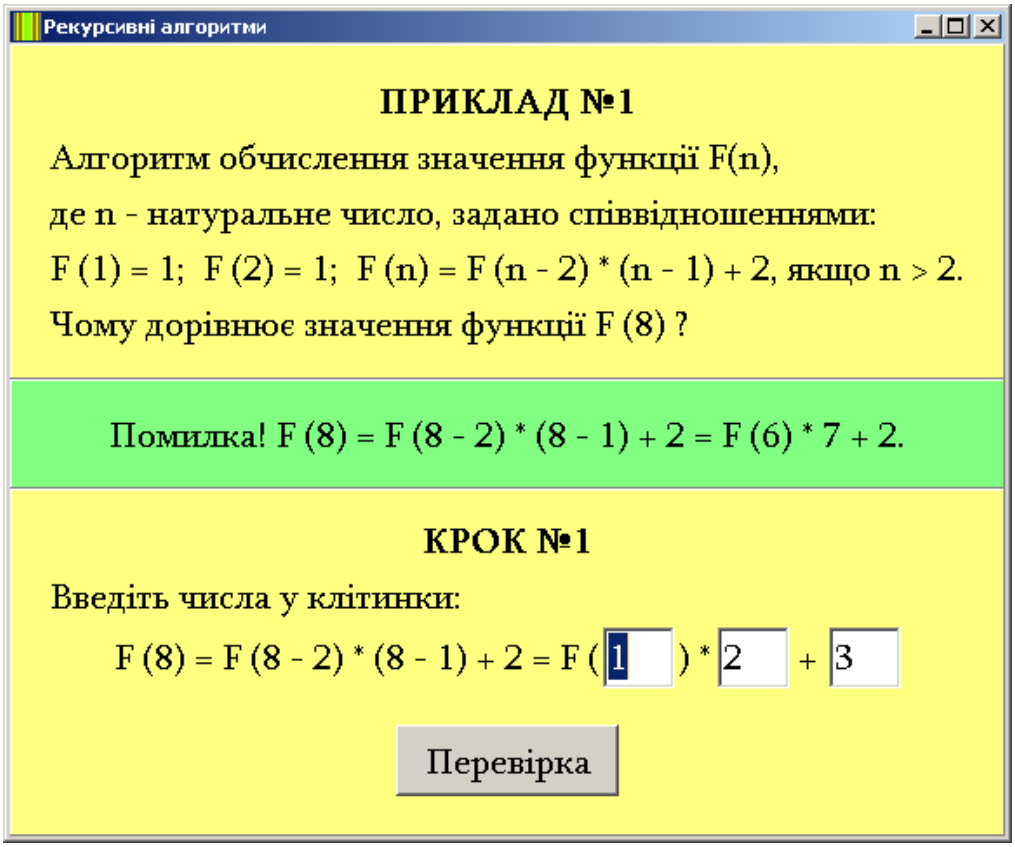

Рисунок 4.3 – Обробка помилки кроку 1

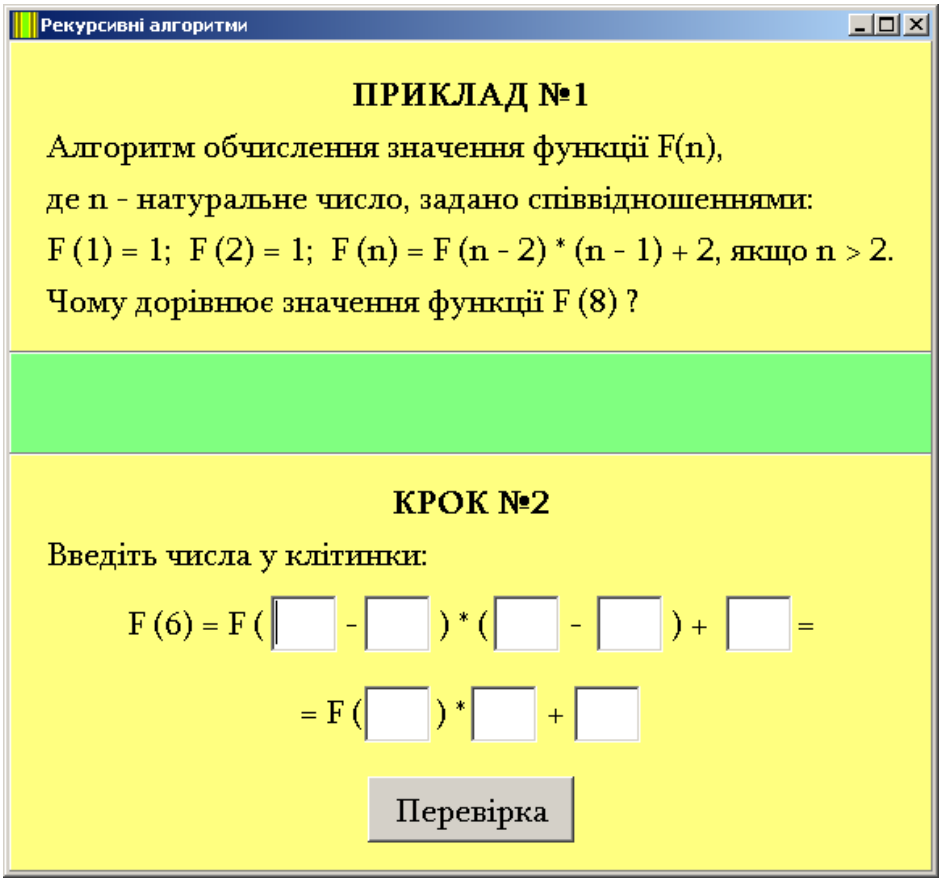

Рисунок 4.4 – Крок 2

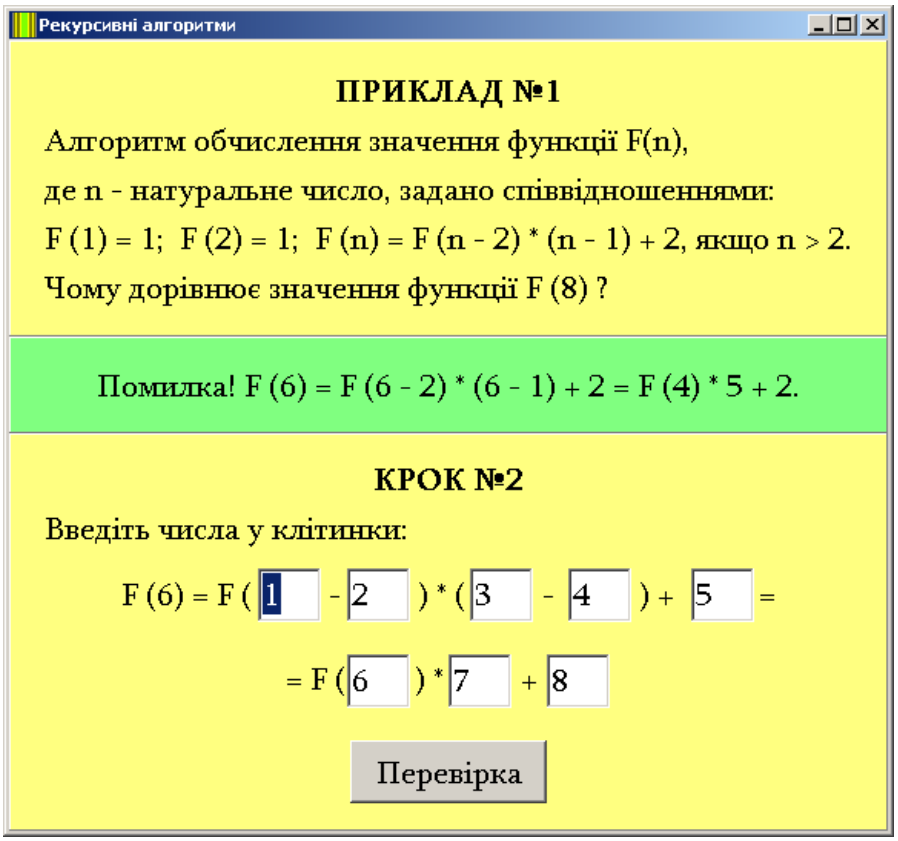

Рисунок 4.5 – Обробка помилки кроку 2

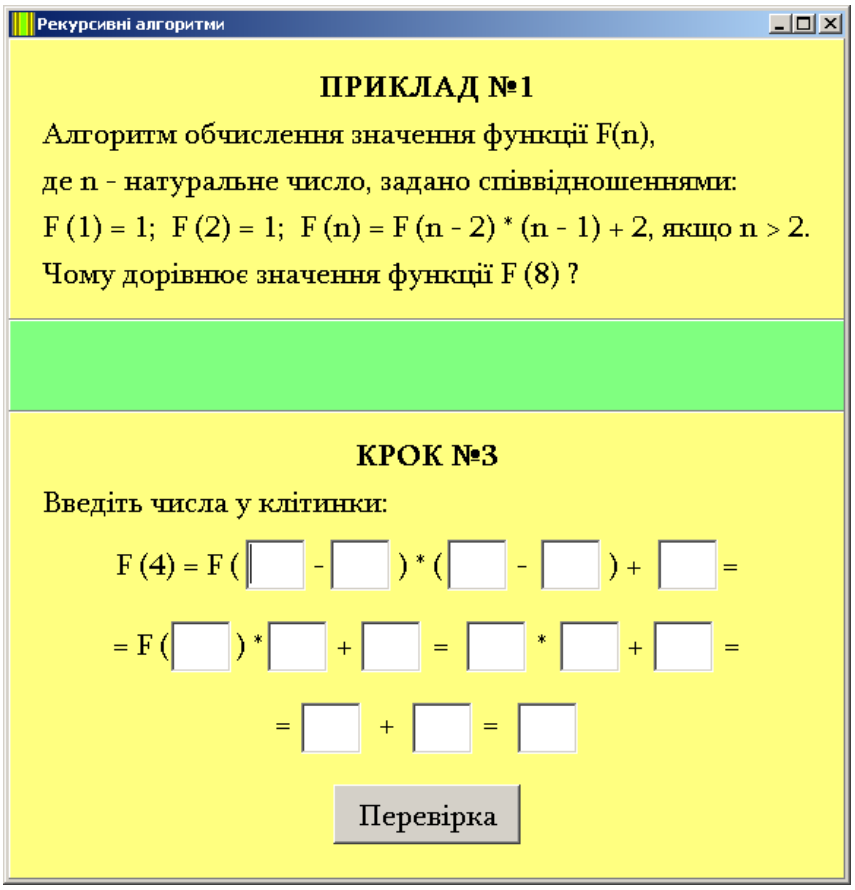

Рисунок 4.6 – Крок 3

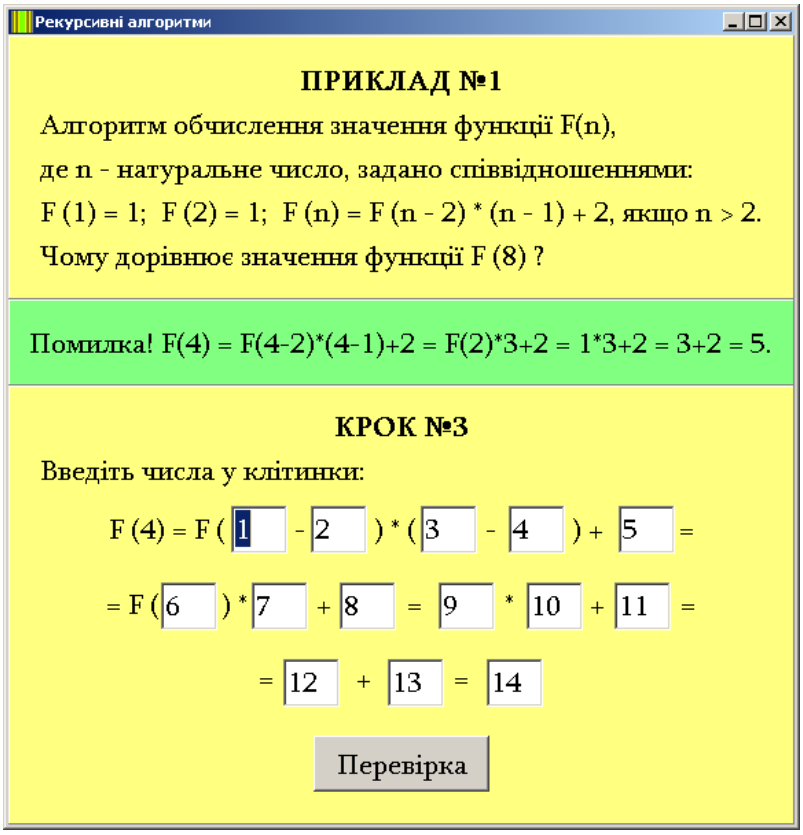

Рисунок 4.7 – Обробка помилки кроку 3

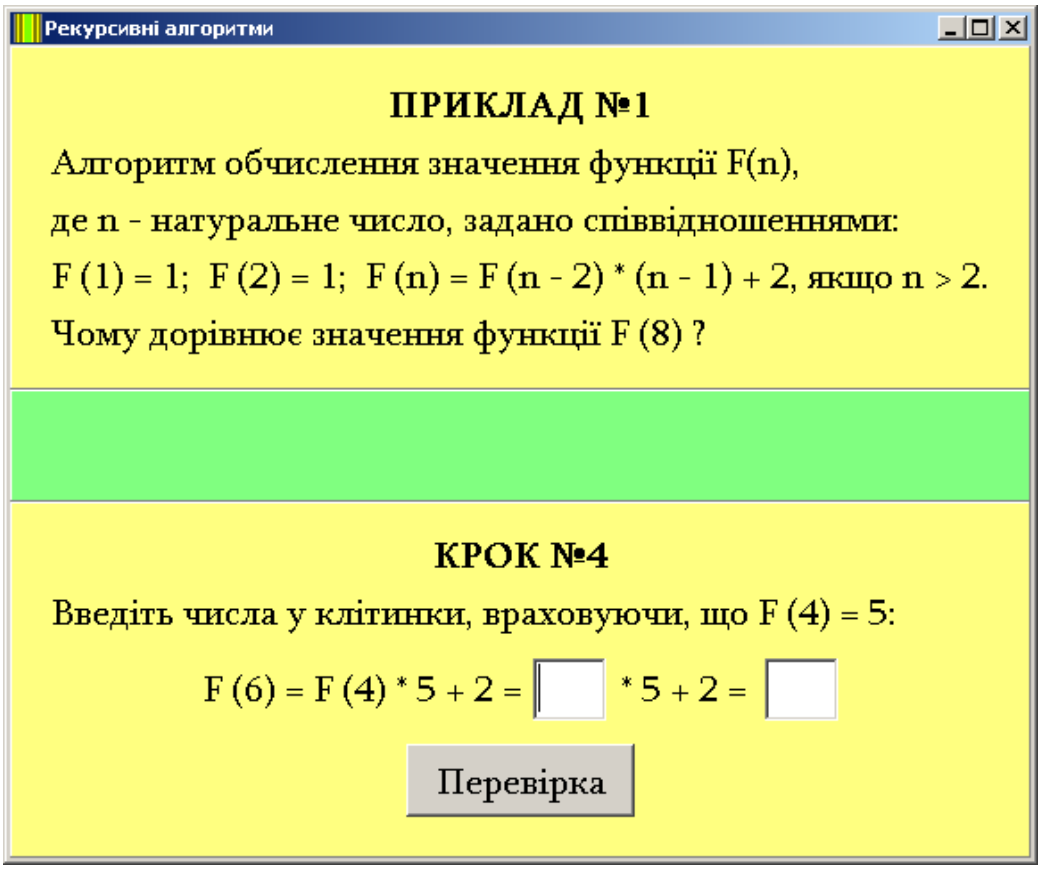

Рисунок 4.8 – Крок 4

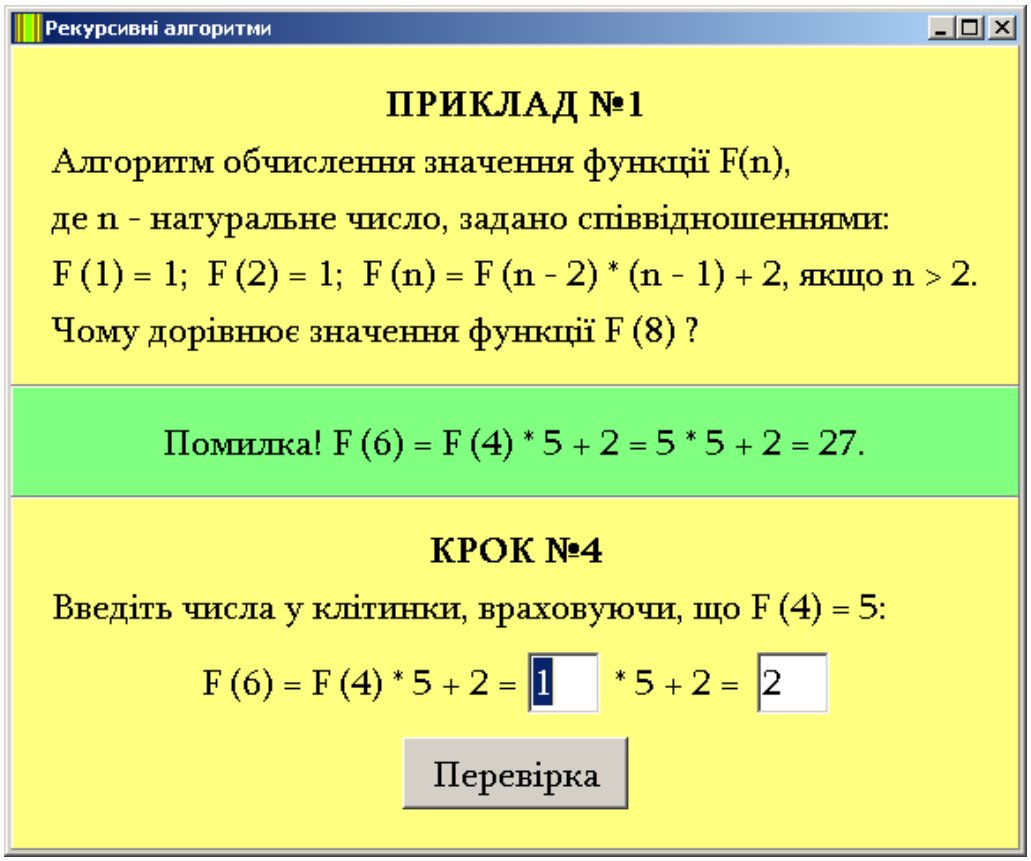

Рисунок 4.9 – Обробка помилки кроку 4

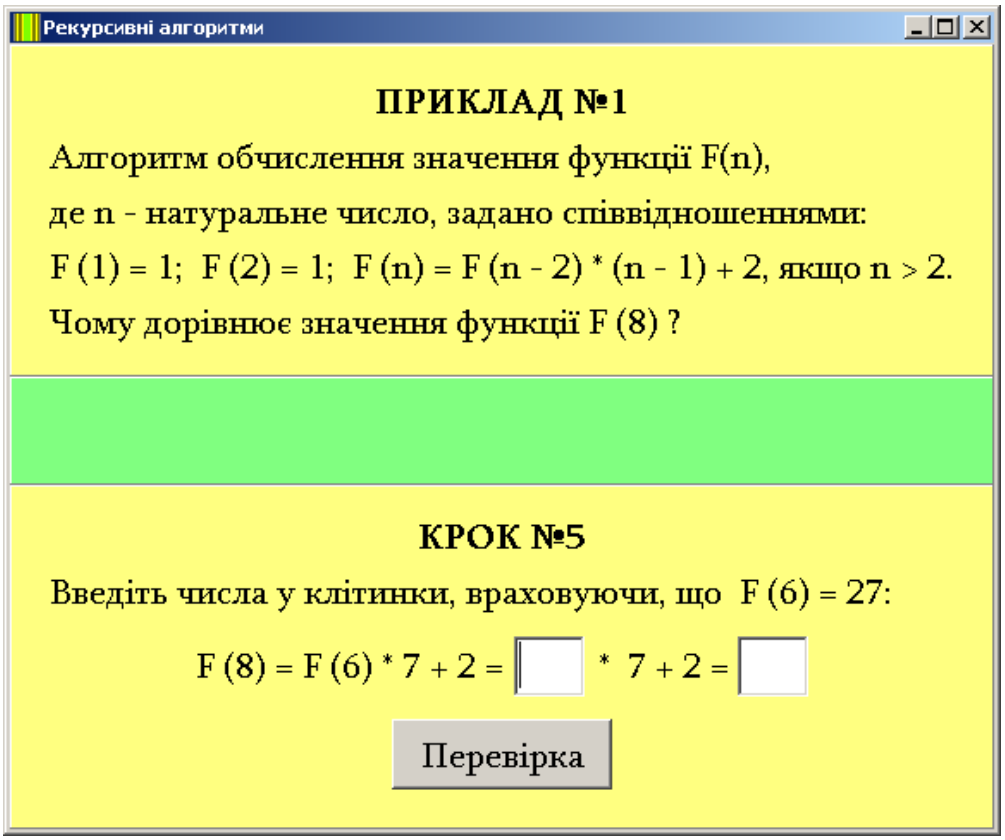

Рисунок 4.10 – Крок 5

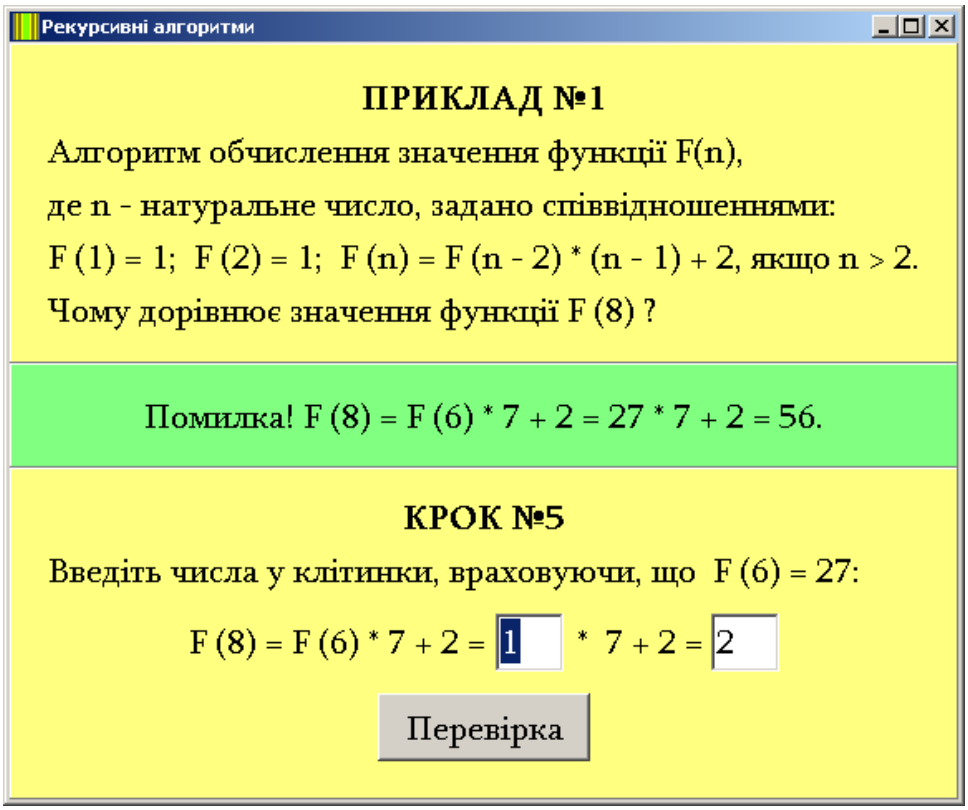

Рисунок 4.11 – Обробка помилки кроку 5

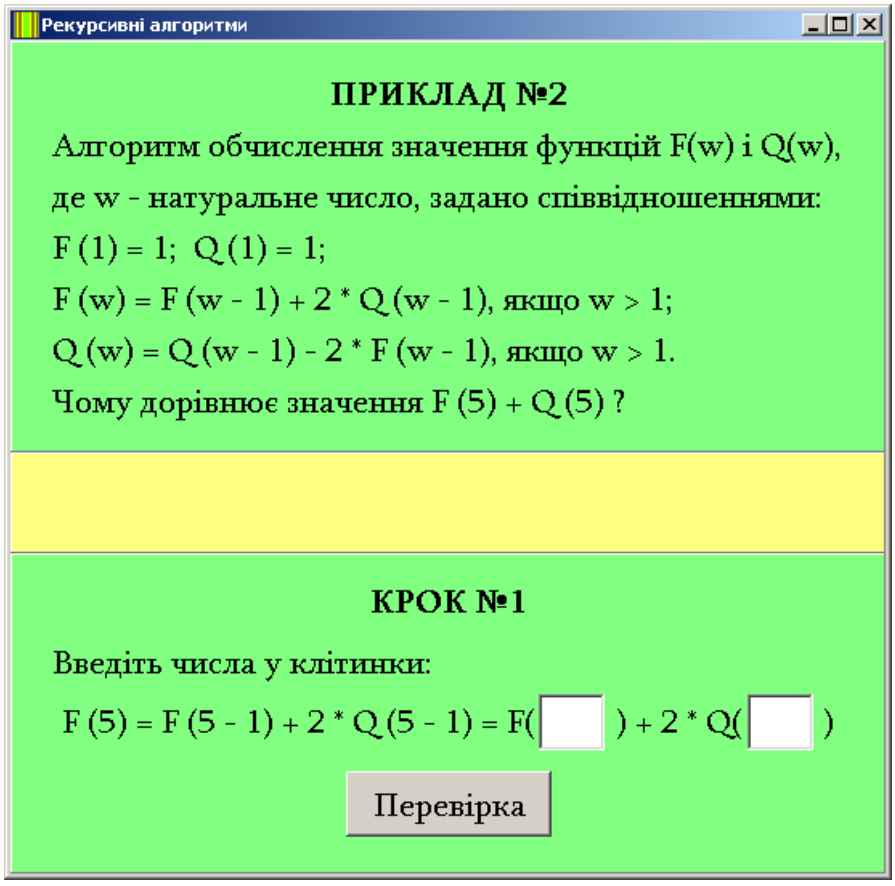

Рисунок 4.12 – Крок 1 (приклад 2)

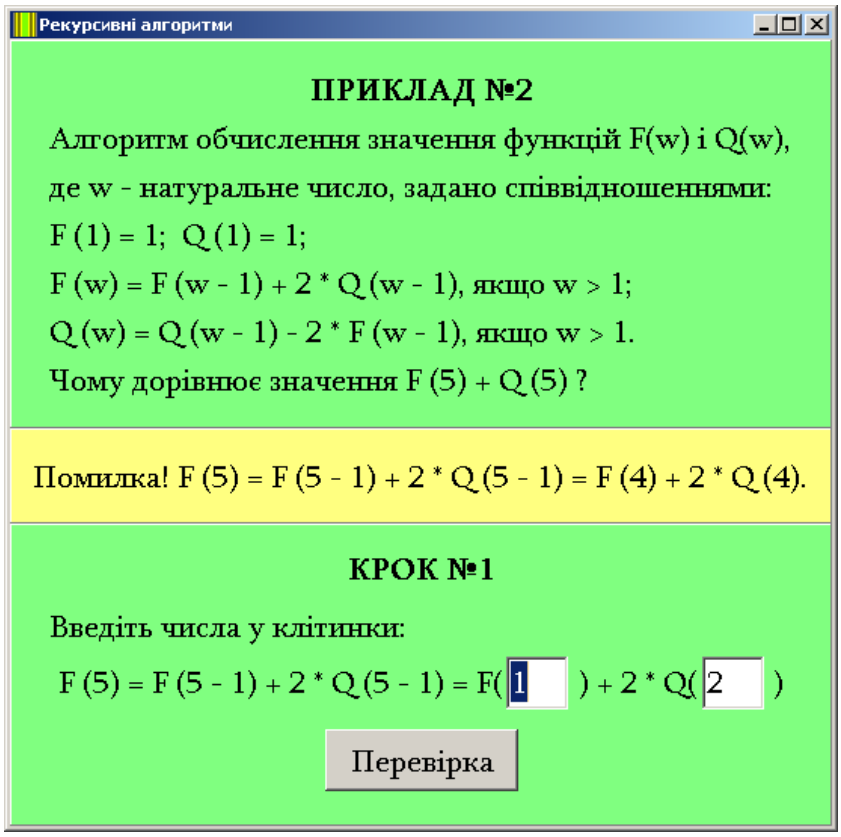

Рисунок 4.13 – Обробка помилки кроку 1 (приклад 2)

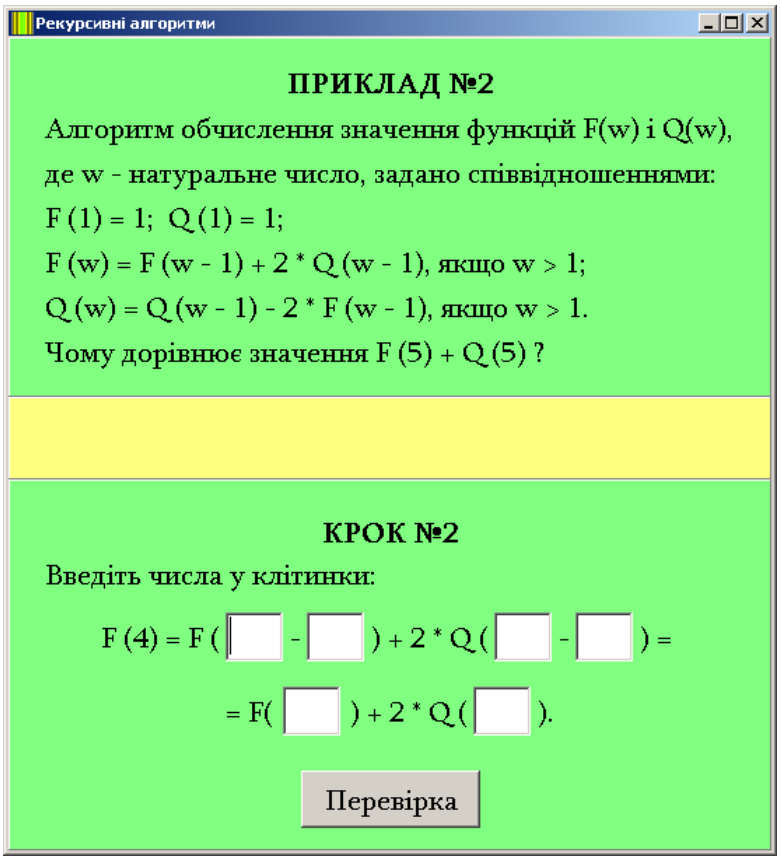

Рисунок 4.14 – Крок 2 (приклад 2)

Рисунок 4.15 – Обробка помилки кроку 2 (приклад 2)

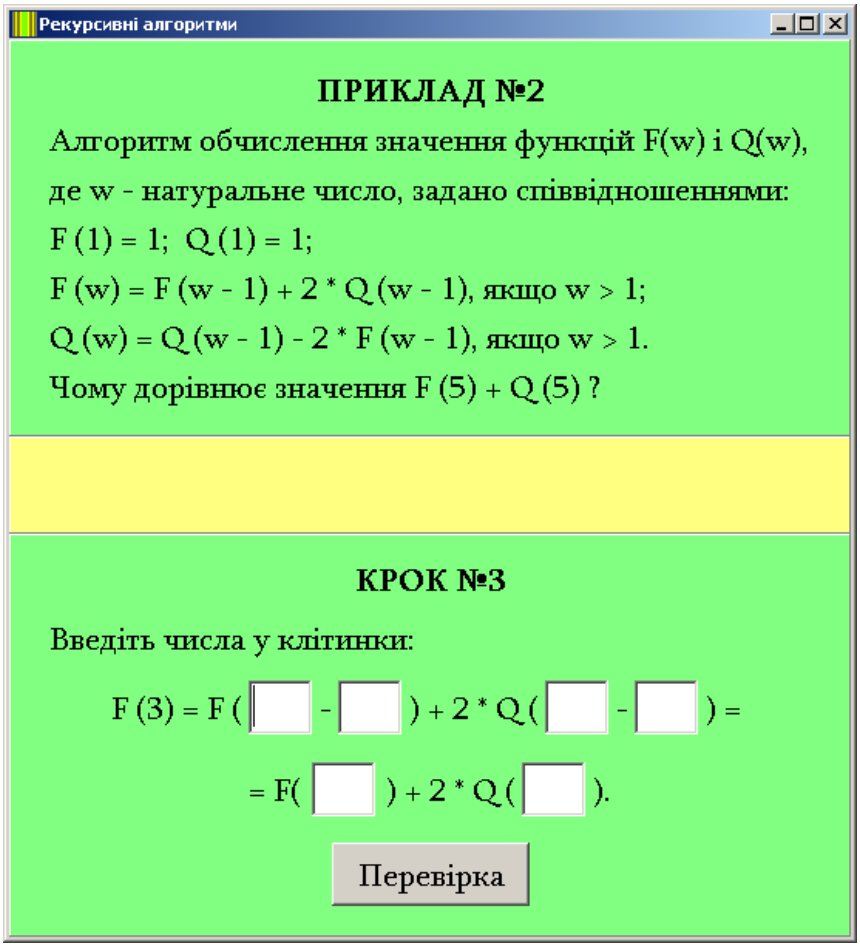

Рисунок 4.16 – Крок 3 (приклад 2)

Рисунок 4.17 – Обробка помилки кроку 3 (приклад 2)

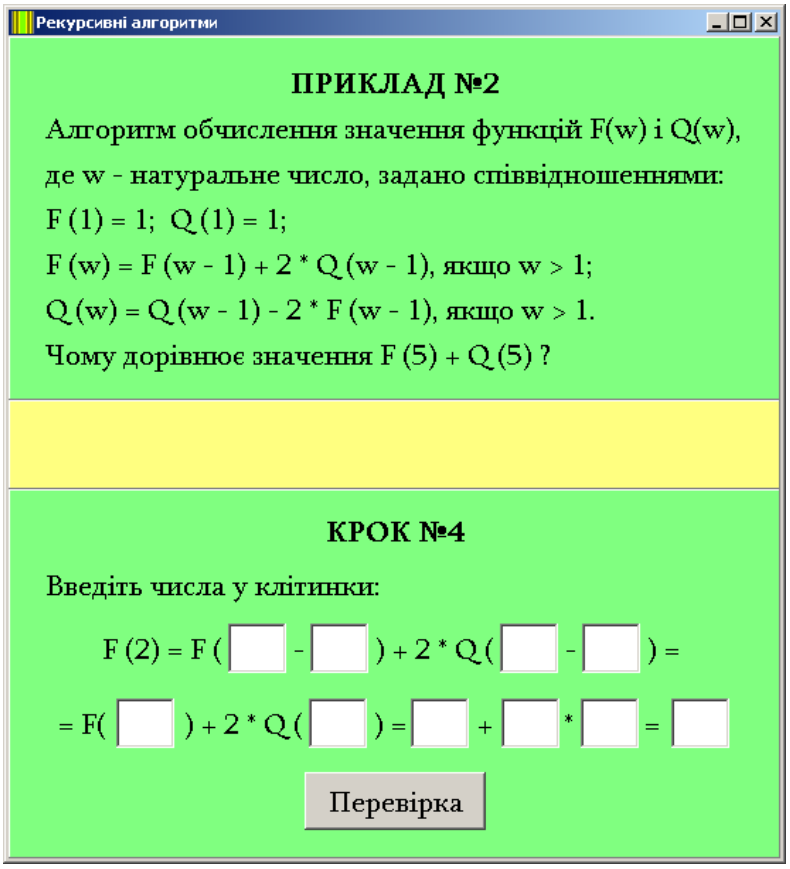

Рисунок 4.18 – Крок 4 (приклад 2)

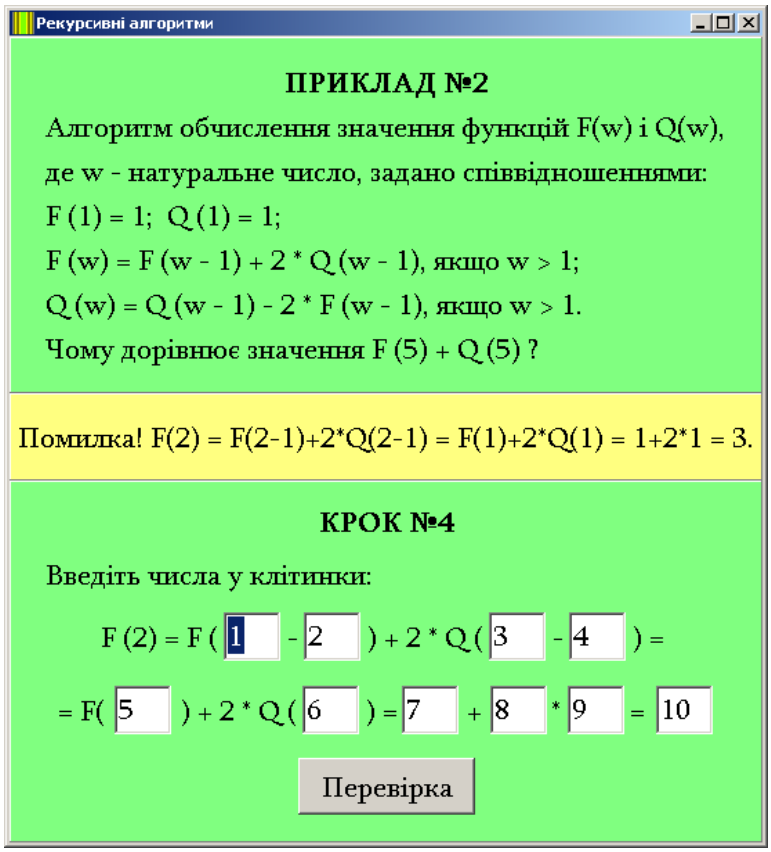

Рисунок 4.19 – Обробка помилки кроку 4 (приклад 2)

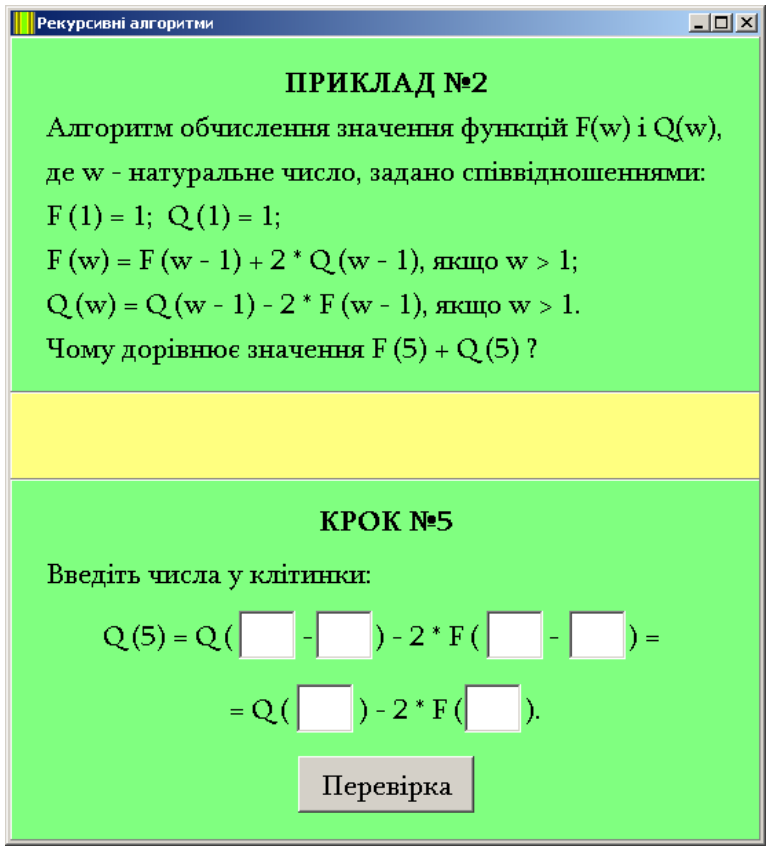

Рисунок 4.20 – Крок 5 (приклад 2)

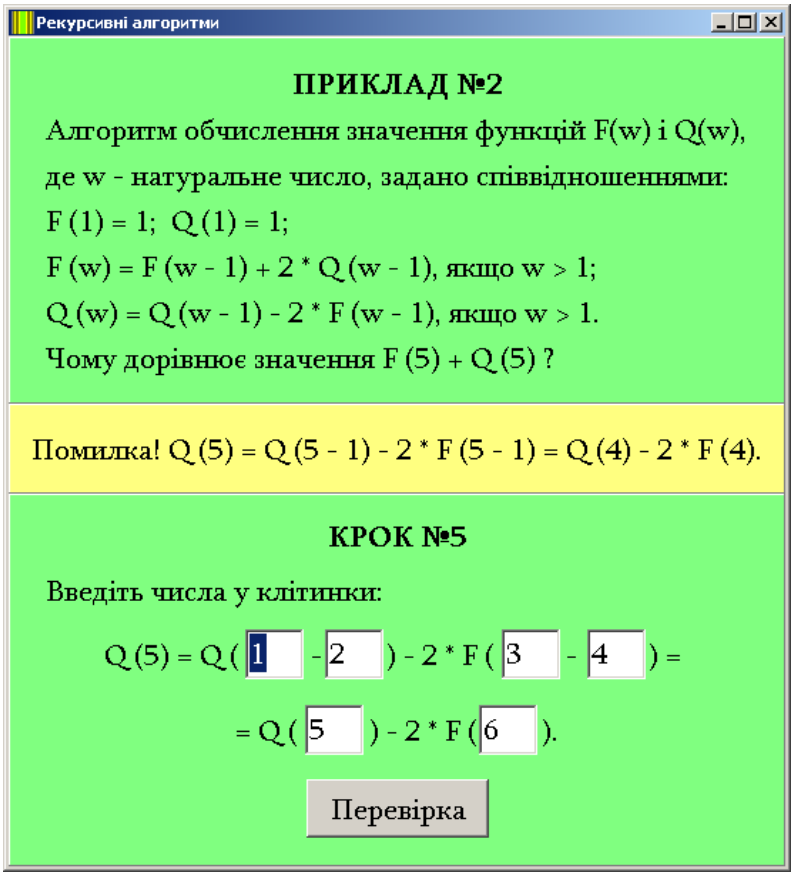

Рисунок 4.21 – Обробка помилки кроку 5 (приклад 2)

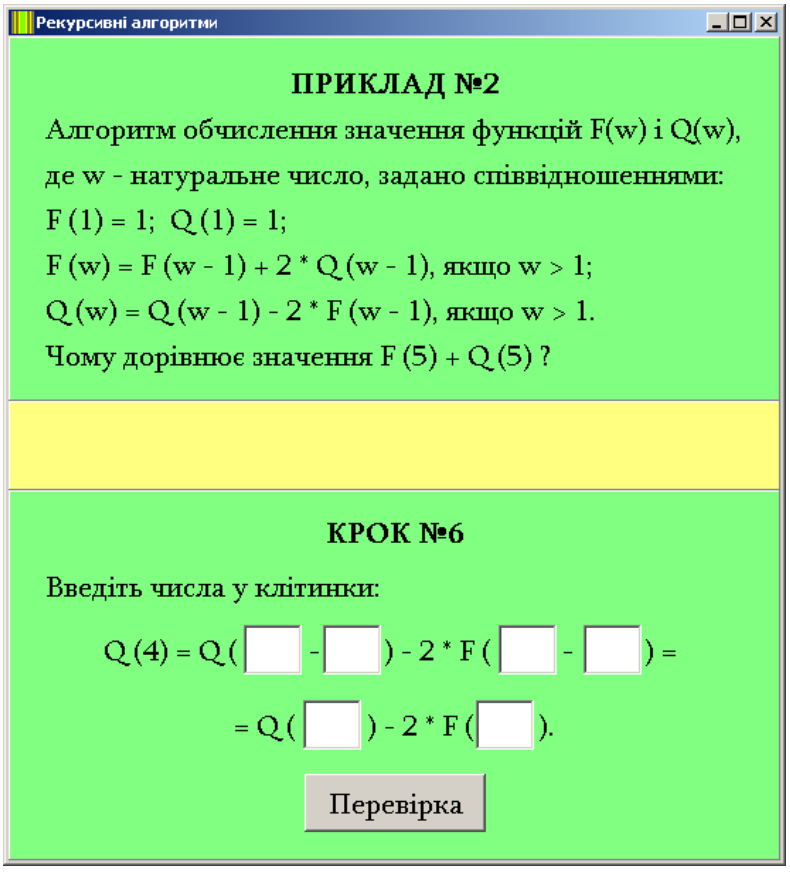

Рисунок 4.22 – Крок 6 (приклад 2)

Рисунок 4.23 – Обробка помилки кроку 6 (приклад 2)

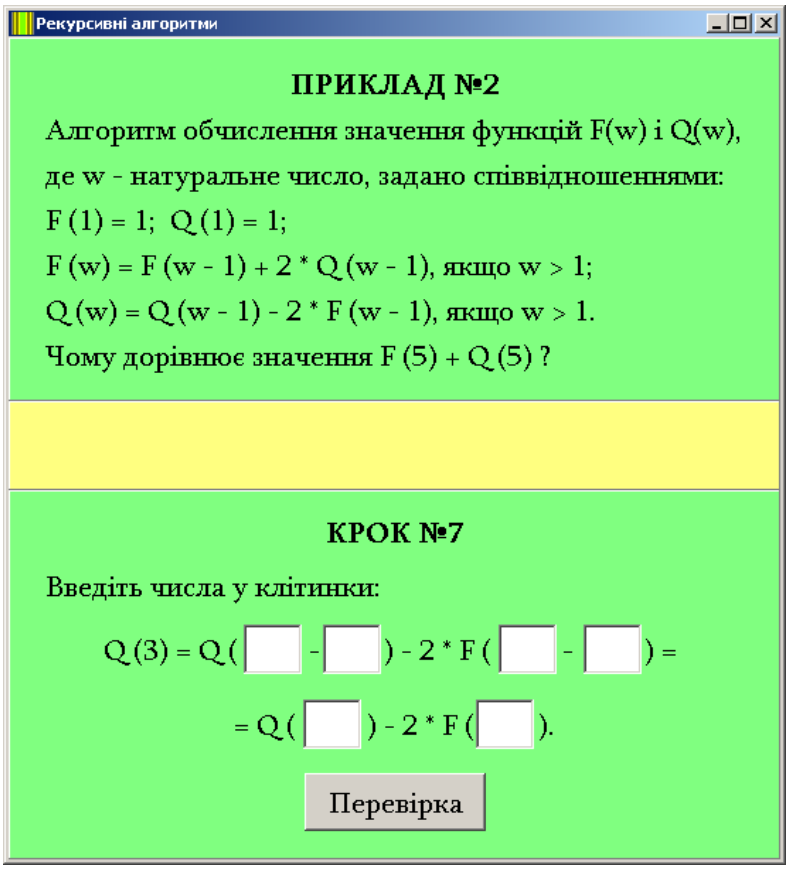

Рисунок 4.24 – Крок 7 (приклад 2)

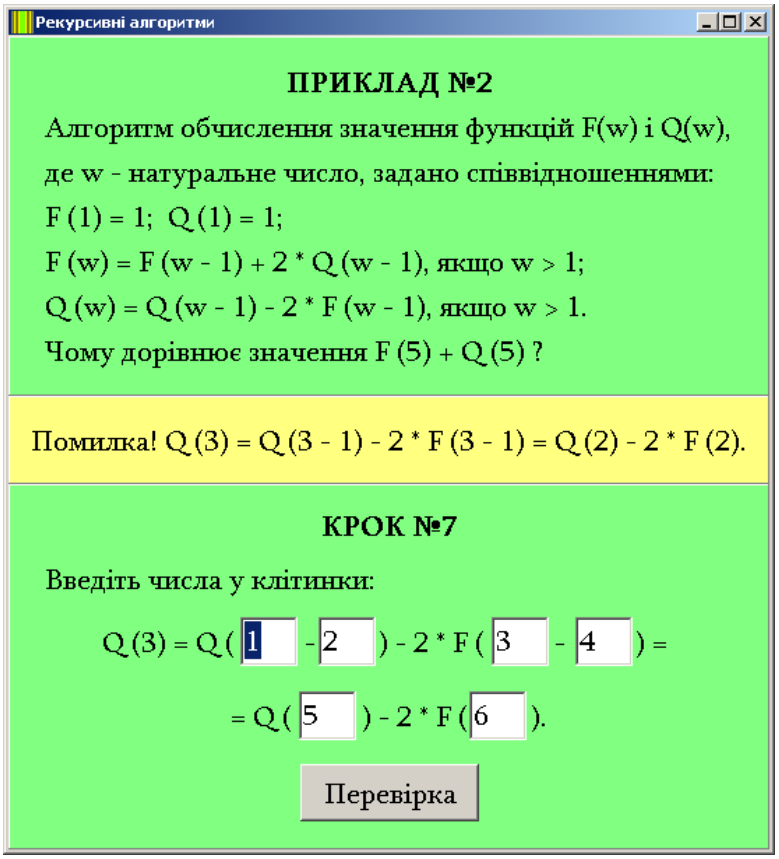

Рисунок 4.25 – Обробка помилки кроку 7 (приклад 2)

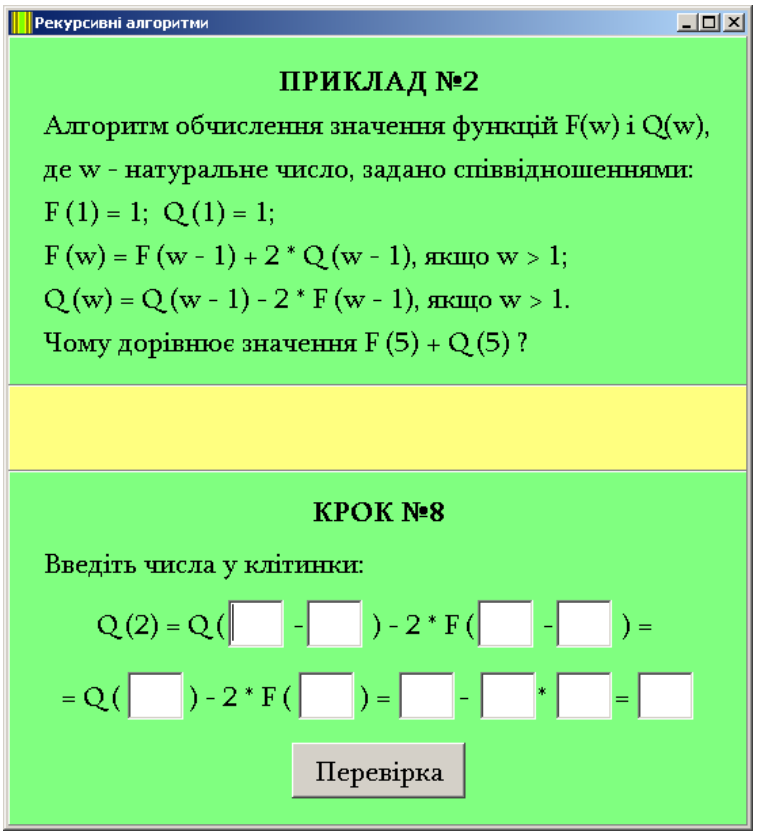

Рисунок 4.26 – Крок 8 (приклад 2)

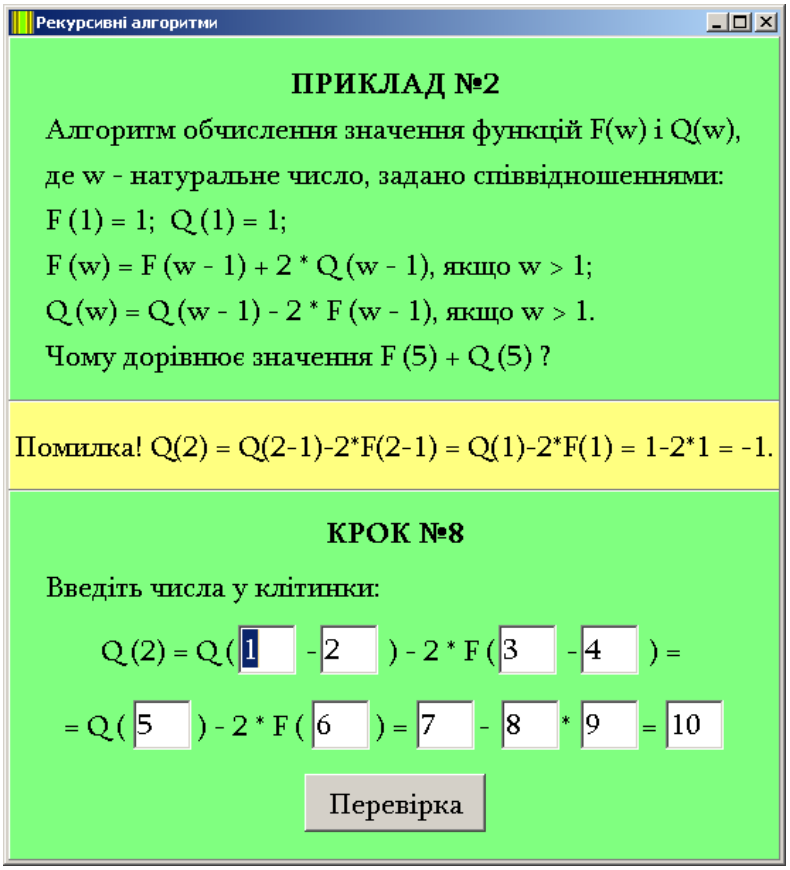

Рисунок 4.27 – Обробка помилки кроку 8 (приклад 2)

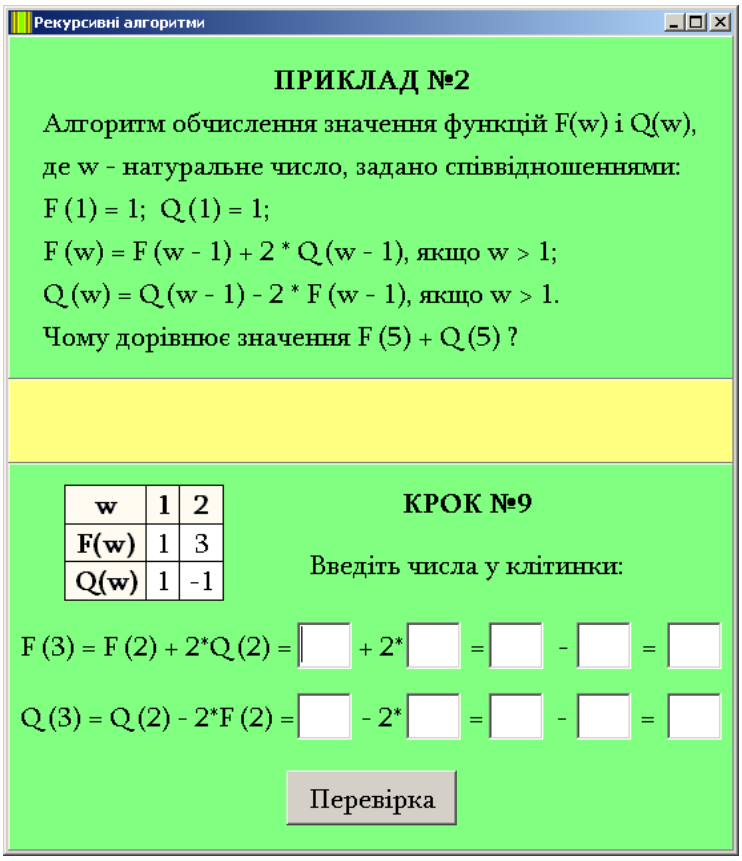

Рисунок 4.28 – Крок 9 (приклад 2)

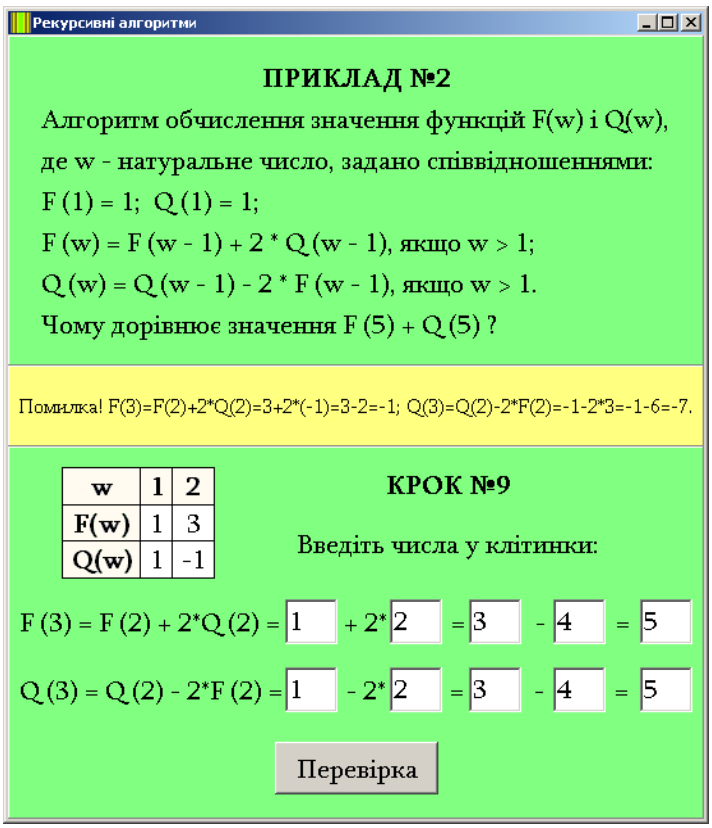

Рисунок 4.29 – Обробка помилки кроку 9 (приклад 2)

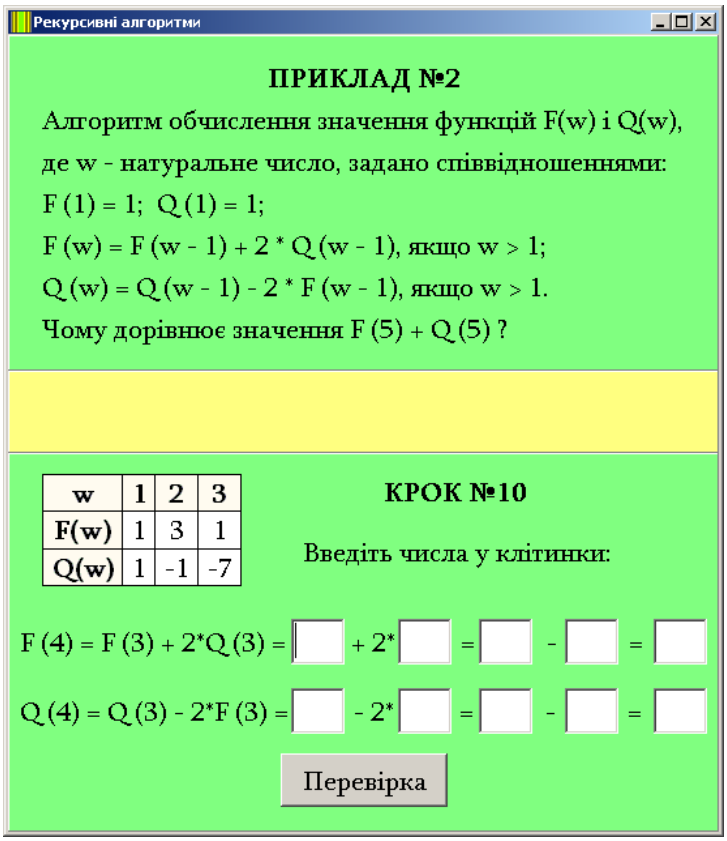

Рисунок 4.30 – Крок 10 (приклад 2)

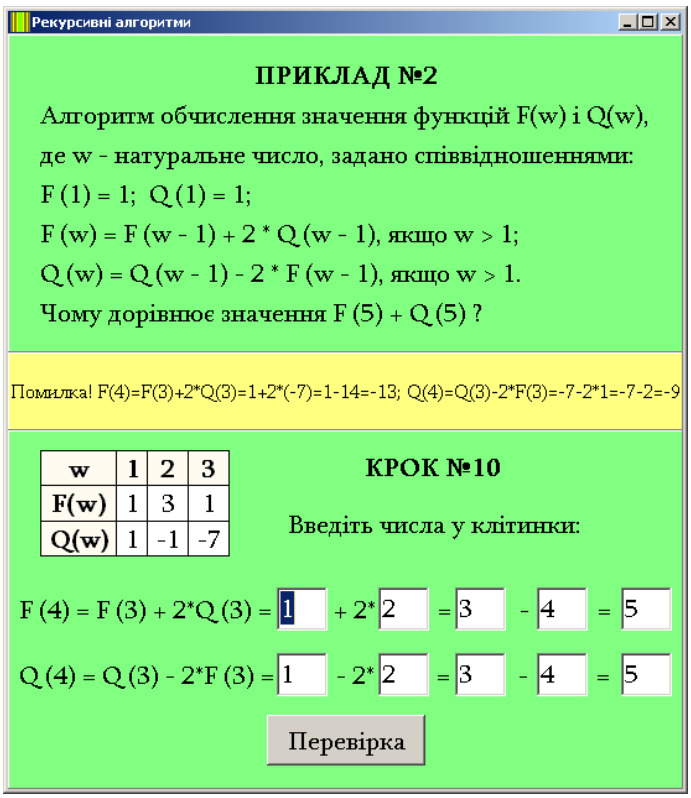

Рисунок 4.31 – Обробка помилки кроку 10 (приклад 2)

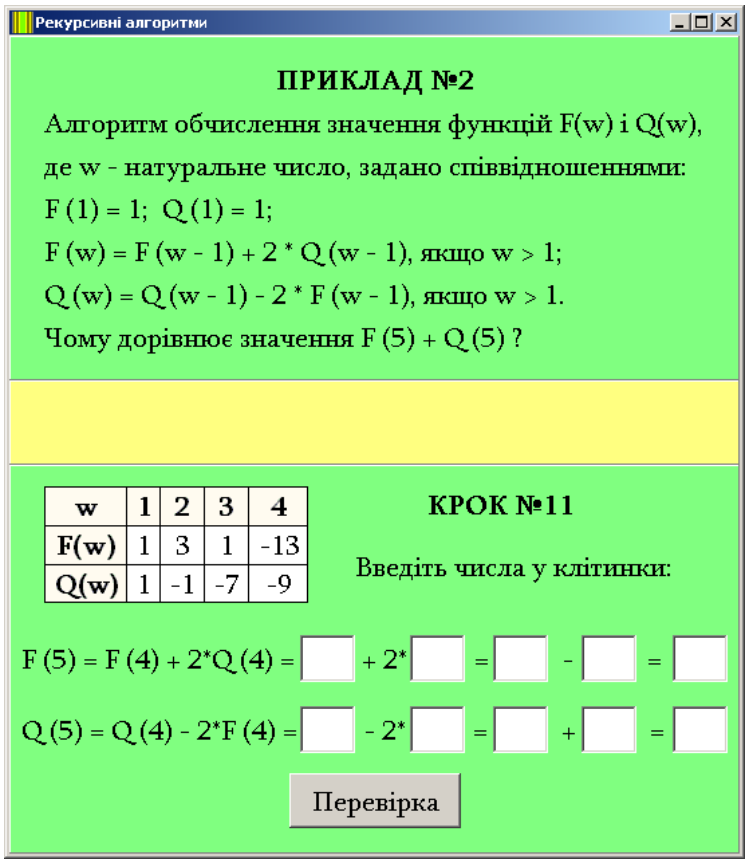

Рисунок 4.32 – Крок 11 (приклад 2)

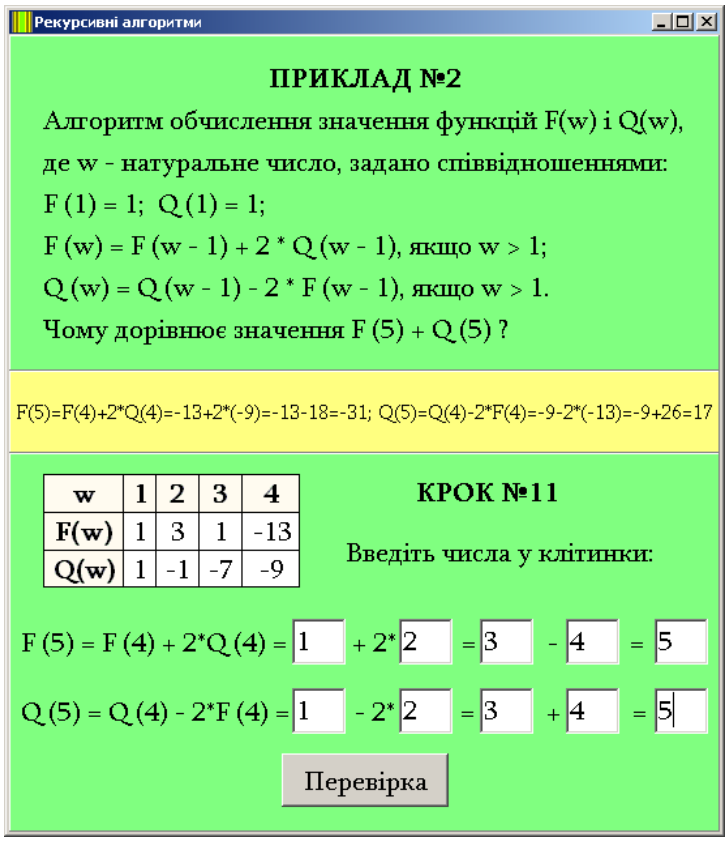

Рисунок 4.33 – Обробка помилки кроку 11 (приклад 2)

Рисунок 4.34 – Крок 12 (приклад 2)

Рисунок 4.35 – Обробка помилки кроку 12 (приклад 2)

4.2. Опис програми

Для переводу алгоритму в програму було обрано мову *C++*. Програма створювалась в середовищі *Borland Builder* [8-10].

Структура програми подана у табл. 4.1-4.3.

Таблиця 4.1 – Структура програми для прикладу 1

Номер кроку	Ім’я форми
1	Form2
2	Form3
3	Form4
4	Form5
5	Form6

Таблиця 4.2 – Структура програми для прикладу 2

Номер кроку	Ім’я форми	Номер кроку	Ім’я форми
1	Form7	7	Form13

2	Form8	8	Form14
3	Form9	9	Form15
4	Form10	10	Form16
5	Form11	11	Form17
6	Form12	12	Form18

Таблиця 4.3 – Структура програми для прикладу 3

Номер кроку	**Ім’я форми**
1	Form219
2	Form20
3	Form21
4	Form22
5	Form23
6	Form24

Form1 – це перше вікно програми (рис. 4.1).

Всі кроки трьох прикладів полягають у тому, що користувач вводить відповіді. Тому розглянемо, як створювалась програма, на прикладі одного кроку. Всі інші кроки були створені подібним чином.

Наведемо, як працює перший крок прикладу 1 (табл. 4.4):

Таблиця 4.4 – Код кроку 1 для прикладу 1

№ рядка	**Код**
1	// приклад 1, крок 1
2	void __fastcall TForm2::BitBtn1Click(TObject *Sender)
3	{
4	if ((Edit1->Text=="6") && (Edit2->Text=="7") && (Edit3-> Text=="2"))
5	{
6	// наступний крок
7	Form3->Show();
8	Form3->Edit1->SetFocus();
9	Form2->Hide();

```
10
11    // слід очищати цей крок
12    Form2->Panel3->Caption="";
13    Form2->Edit1->Clear();
14    Form2->Edit2->Clear();
15    Form2->Edit3->Clear();
16   }
17   else
18   {
19    Panel3->Caption="Помилка! F (8) = F (8 - 2) * (8 - 1) + 2 = F (6) * 7 +
20 2.";
21    Edit1->SetFocus();
22   }
23 }
```

Пояснимо код.

У рядку 4 перевіряється, чи користувач ввів вірні значення.

Якщо значення, введені користувачем, помилкові, то виконуються рядки 19-21. На панель, що знаходиться посередині, виводиться пояснення помилки – рядки 19-20, і курсор знову розміщується у першому полі для вводу – рядок 21.

Якщо значення, введені користувачем, вірні, то виконуються рядки 7-15. А саме: усі текстові поля очищаються – рядки 13-15; очищається панель з текстом, що пояснював помилку – рядок 12; поточне вікно закривається – рядок 9; відкривається наступне вікно – рядок 7; у новому вікні курсор розміщується у першому текстовому полі – рядок 8.

Очищення текстових полів – рядки 13-15, панелі – рядок 12 – та розміщення курсору в першому текстовому полі наступного вікна – рядок 8 – потрібно здійснювати для того. щоб тренінг з цього прикладу можна було пройти ще раз.

Код програми (для прикладу 1) представлено у додатках.

Для програми була створена унікальна піктограма. Опишемо, процес її створення та розміщення в програмі.

З середовища Borland Builder було запущено програму Image Editor (рис. 4.36) за допомогою пунктів меню Tools -> Image Editor.

Image Editor – це графічний редактор, який дозволяє створювати графічні файли різного призначення.

Для створення піктограми було виконані команди File -> New -> Icon File (.ico) (рис. 4.36).

Далі було обрано файл з шириною в 32 пікселя і 16 кольорами (рис. 4.37).

Після цього у вікні, що відкрилось (рис. 4.38), було створено малюнок (рис. 4.39).

Рисунок було збережено як піктограму, тобто як файл з розширенням .ico (рис. 4.40).

Створену піктограму було розміщено в програму. Для цього в середовищі Borland Builder за допомогою пунктів меню Project -> Options було запущено вікно опцій програми (рис. 4.41). Далі була відкрита вкладка Application (рис. 4.41).

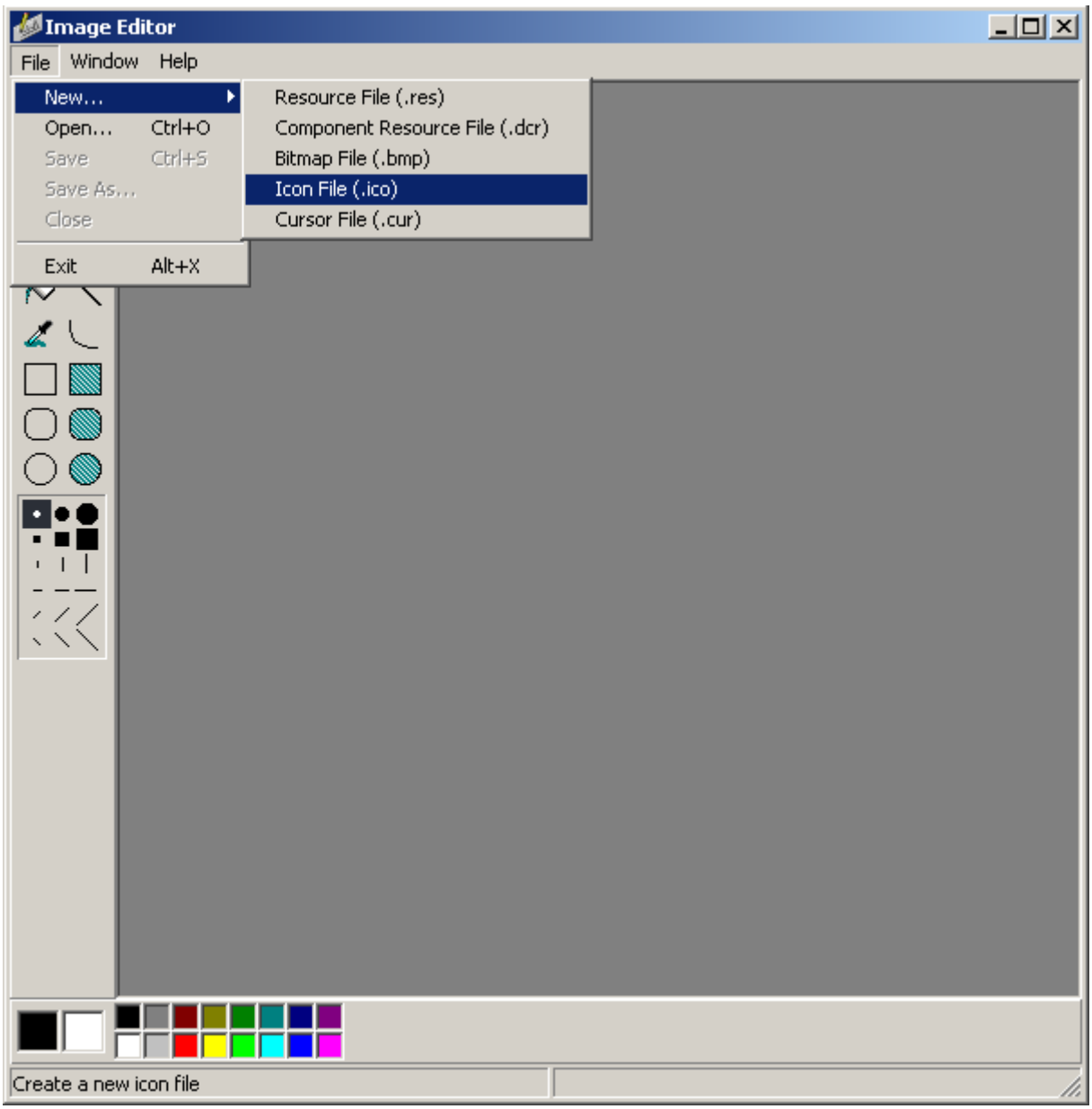

Рисунок 4.36 – Обробка помилки кроку 12 (приклад 2)

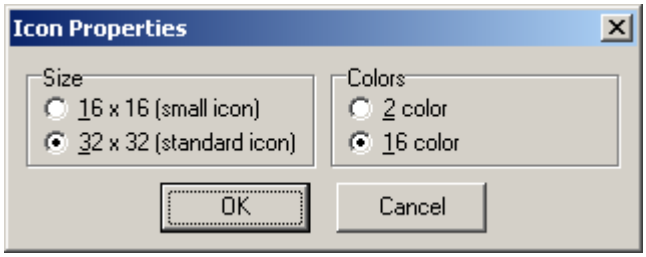

Рисунок 4.37 – Створення піктограми

У вікні опцій (рис. 4.41) поле Icon показує поточну піктограму. Для її зміни було натиснуто кнопку Load Icon та обрану новостворену піктограму (рис. 4.42).

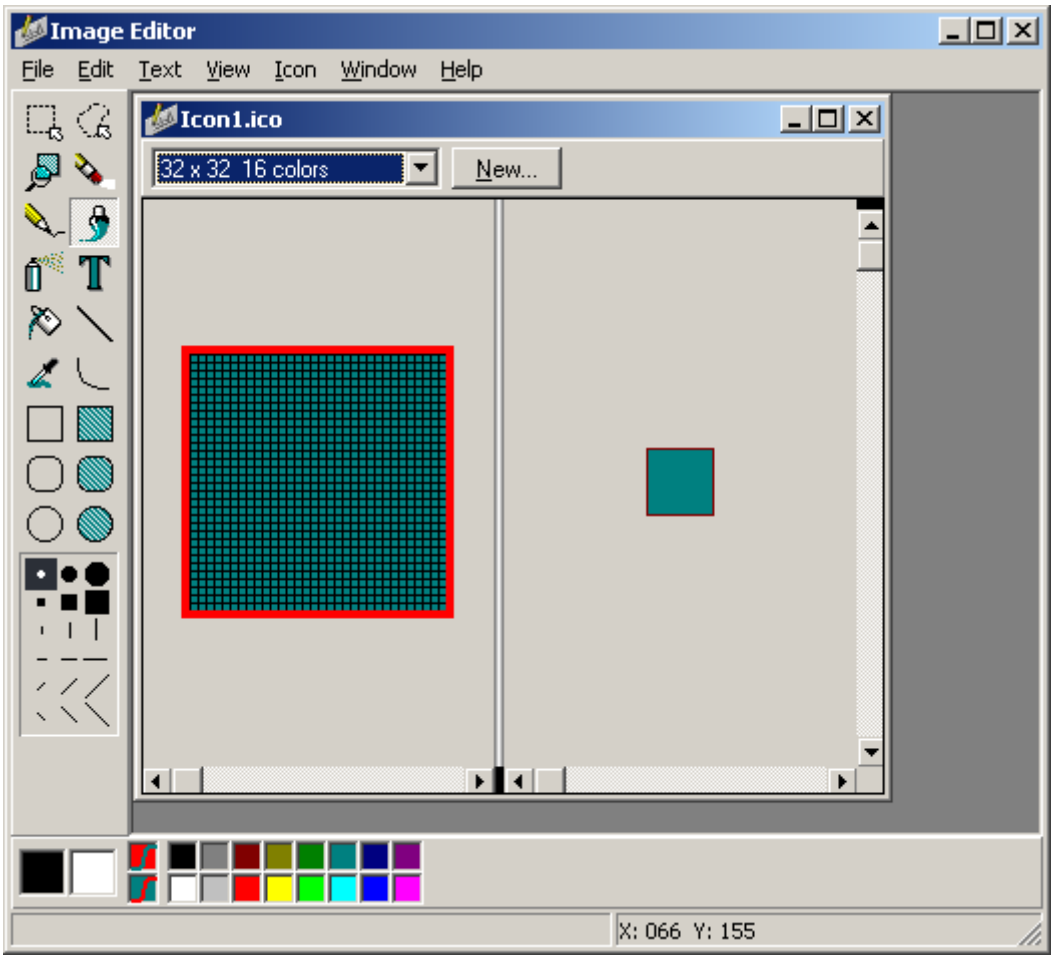

Рисунок 4.38 – Створення піктограми

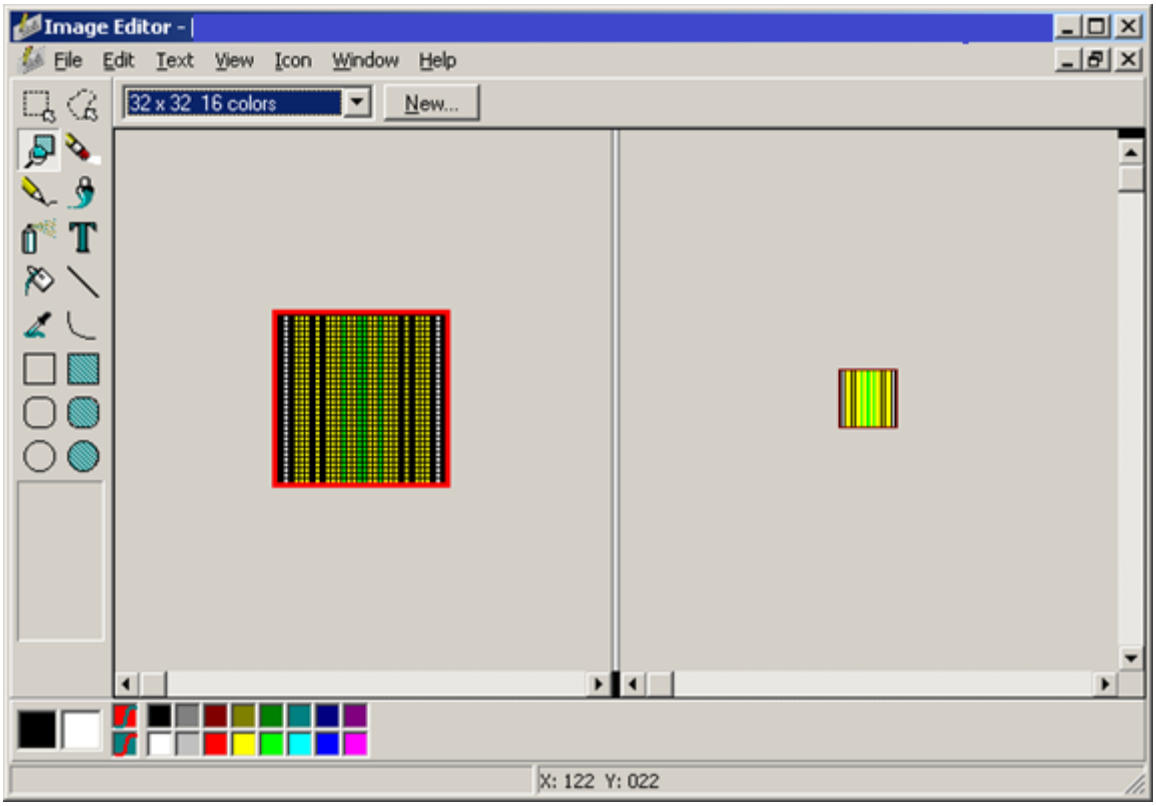

Рисунок 4.39 – Створений рисунок

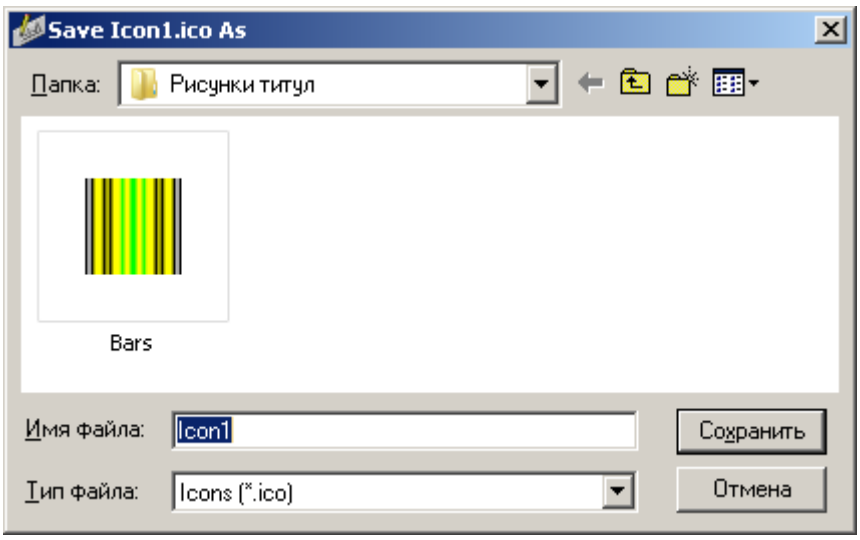

Рисунок 4.40 – Збереження піктограми

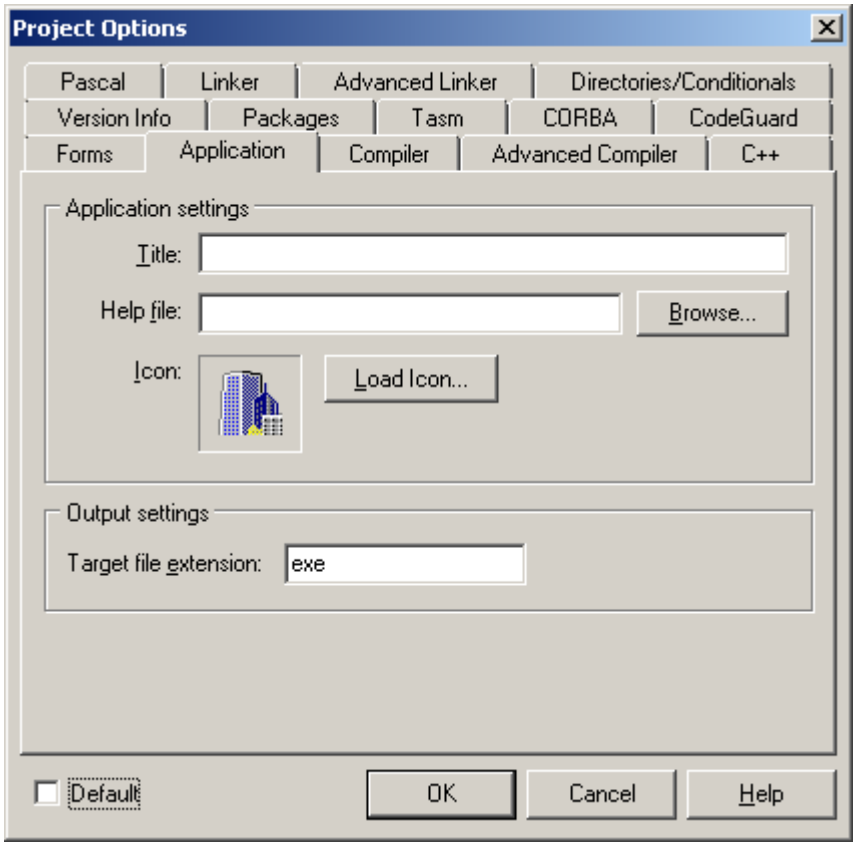

Рисунок 4.41 – Вікно опцій програми

Нова піктограма тепер відображається в заголовку вікон при виконанні програми (рис. 4.43). А також па панелі запущених у Windows вікон (рис. 4.44).

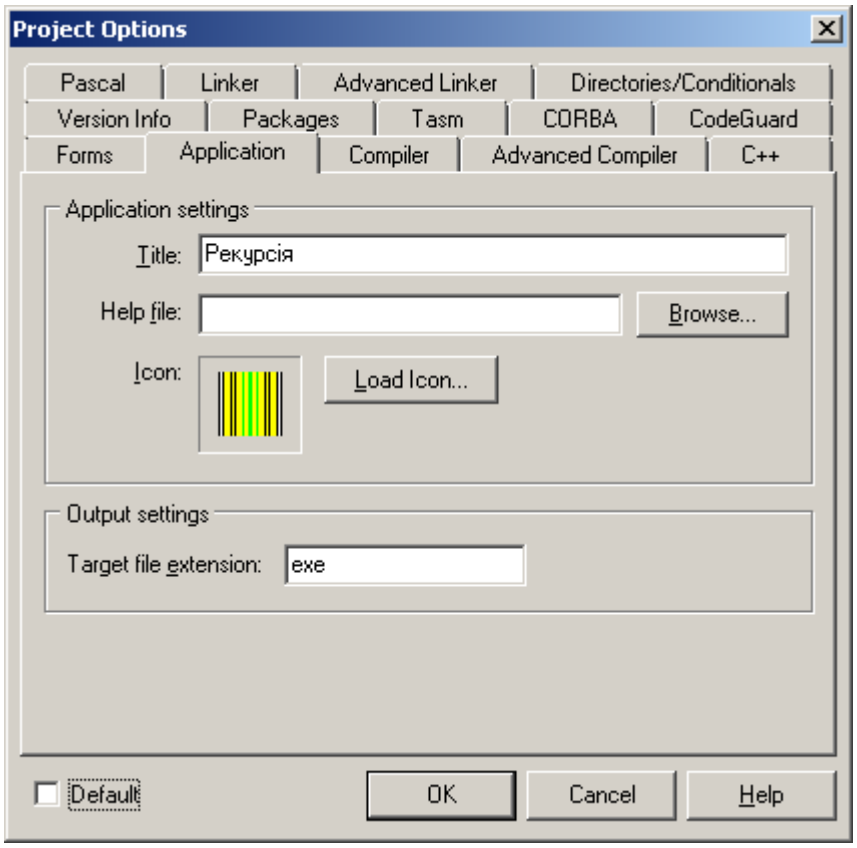

Рисунок 4.42 – Вікно опцій програми

Рисунок 4.43 – Заголовок вікна

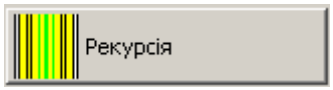

Рисунок 4.44 – Позначка запущеного тренажеру

ВИСНОВКИ

У магістерській роботі було досліджено тему «Рекурсивні алгоритми».

Було здійснено огляд візуалізаторів рекурсивних алгоритмів, представлених в глобальній мережі.

Були підібрано три приклади на тему «Рекурсивні алгоритми» та розроблено алгоритм тренажеру.

У перших двох прикладах алгоритми задані словесно, а третьому – у вигляді коду на мові Pascal.

Для першого та другого прикладу для розв’язування використано трасування алгоритмів, у третьому – за алгоритмом задано рекурентне співвідношення.

У першому та третьому прикладах використана пряма рекурсія, у другому – непряма рекурсія.

Була створена блок-схема алгоритму.

За алгоритмом написана програма. Програма має приємний різнокольоровий дизайн, зручна у користуванні. Програма перевірена, усі помилки були усунені.

Була створена інструкція по роботі з програмою, висвітлено код тренажеру.

Була проведена апробація результатів магістерської роботи на науково-практичному семінарі «Прикладна математика та інформаційні технології», який проходив на кафедрі математичного моделювання та соціальної інформатики Полтавського університету економіки і торгівлі.

Тренажер передано на впровадження у дистанційний курс «Алгоритми та структури даних» Полтавського університету економіки і торгівлі. Це підтверджує акт впровадження.

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

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

2. Соколов О. Ю. Інформатика для інженерів / О. Ю. Соколов, І. Т. Зарецька, Г. М. Жолткевич, О. В. Ярова. – Харків: Факт, 2006. – 424 с.

3. Поляков К. Методические материалы и программное обеспечение для школьников и учителей. Подготовка к ЭГЕ. – [Электронный ресурс]. – Режим доступа: http://kpolyakov.spb.ru/school/ege.htm.

4. Исламов Р. Г. Рекурсивные алгоритмы. Разбор заданий No11 ЕГЭ по информатике и ИКТ / Р. Г. Исламов. – [Электронный ресурс]. – Режим доступа: https://docplayer.ru/72362955-Rekursivnye-algoritmy-razbor-zadaniy-11-ege-po-informatike-i-ikt.html.

5. Использование рекурсивных алгоритмов для решения экономических задач. – [Электронный ресурс]. – Режим доступа: https://works.doklad.ru/view/ug96n8U8r40/all.html

6. Ульянов М. В. Теория рекурсии для программистов / М. В. Ульянов, В. А. Головешкин. – М.:Физматлит, 2016. – 200 с.

7. Кормен Т. Алгоритмы построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн. – М.: Вильямс, 2018 – 1328 с.

8. Винокуров Н. А. Практика и теория программирования. Книга 2. / Н. А. Винокуров Н.А., А.В. Ворожцов. – М: Физматкнига, 2008. – 288 с.

9. Страуструп Б.Программирование. Принципы и практика с использованием C++ / Б. Страуструп. – М.: Вильямс, 2016. – 1328 с.

10. Стивен П. Язык программирования C++. Лекции и упражнения / П. Стивен. – М.: Вильямс, 2017. – 1248 с.

11. Ємець О. О. Методичні рекомендації щодо оформлення пояснювальних записок до курсових проектів (робіт) 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. – 69 с.

12. 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.

ДОДАТОК А

ПРИКЛАД 3

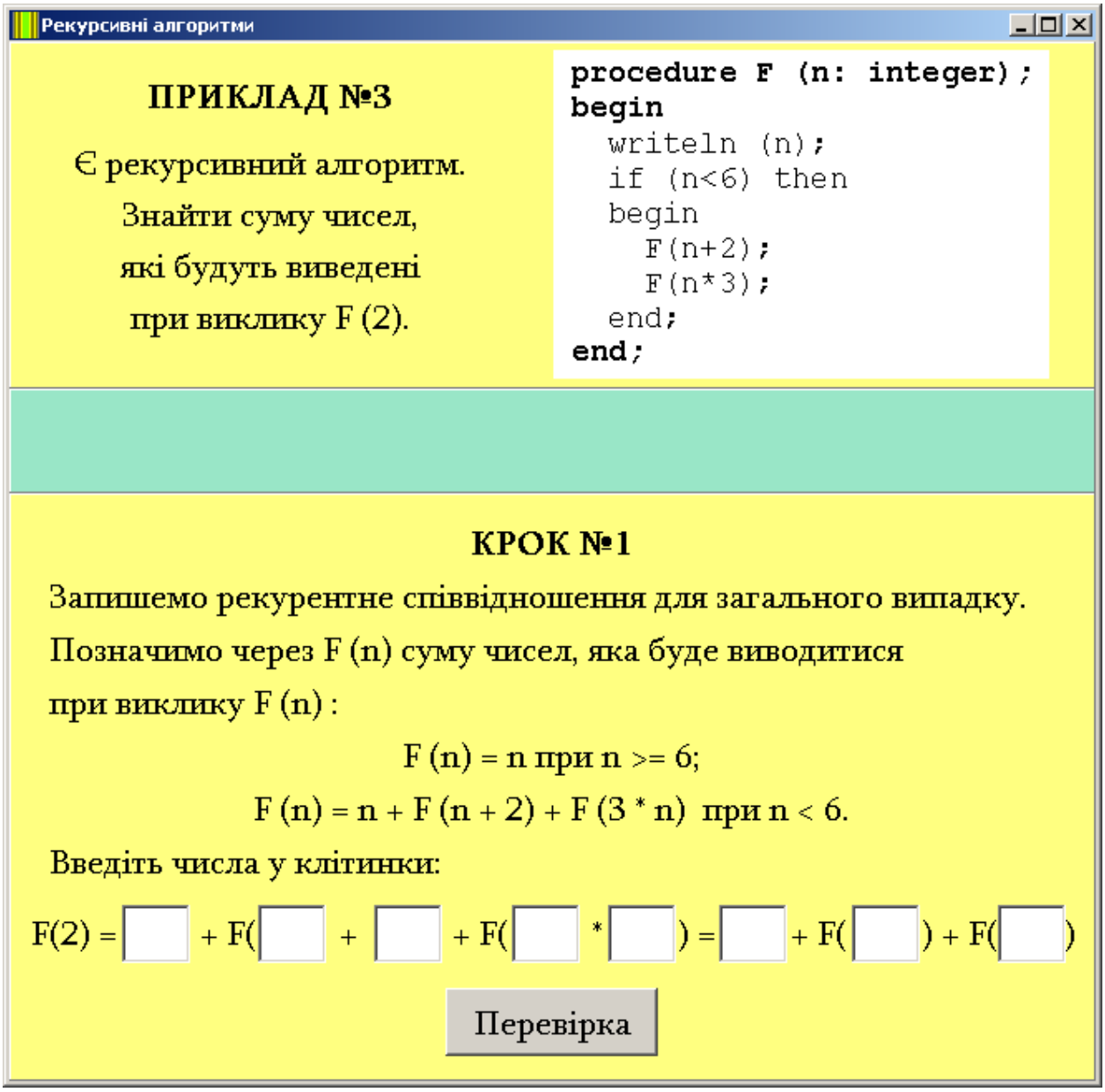

Рисунок А.1 – Крок 1 (приклад 3)

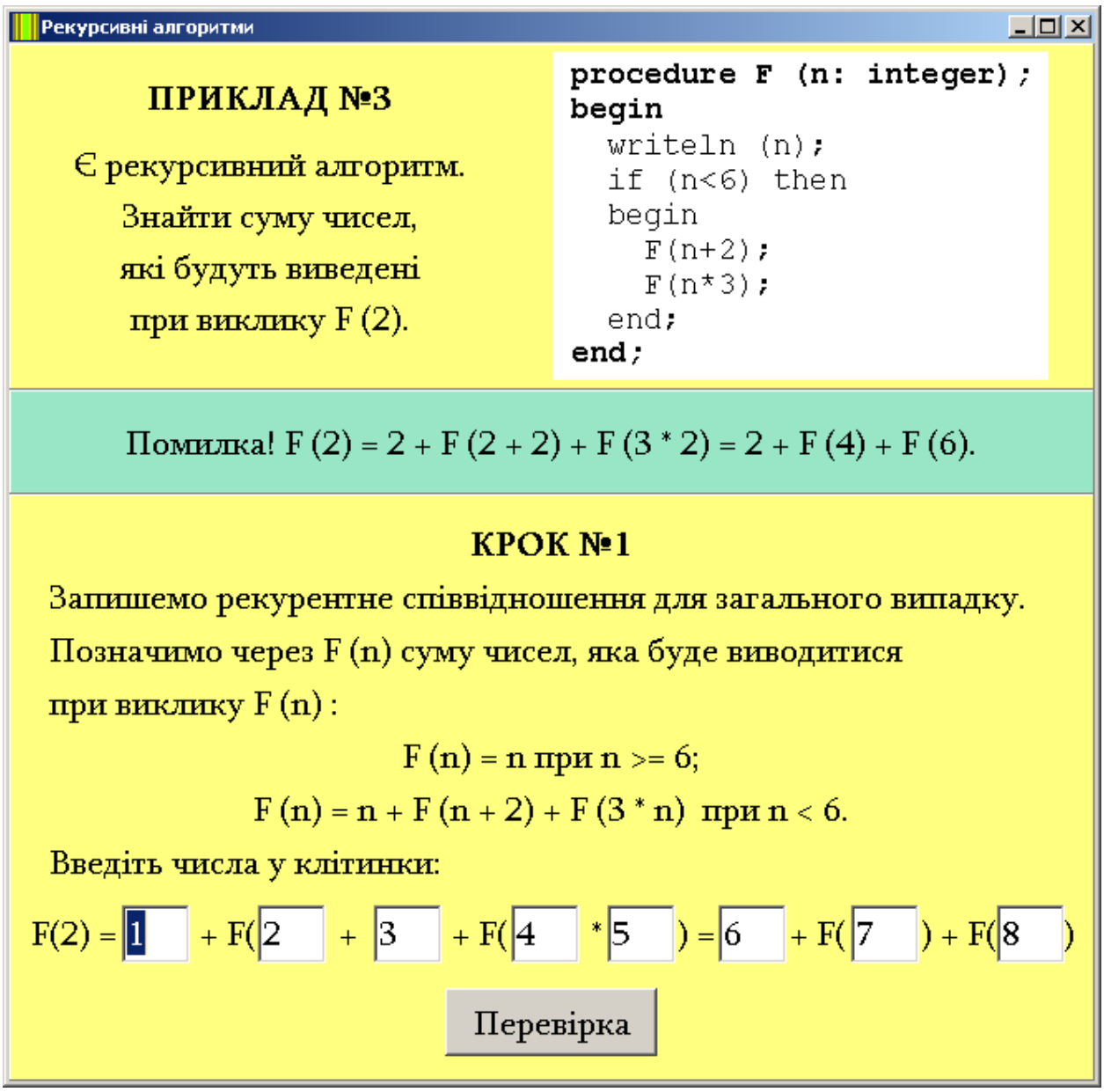

Рисунок А.2 – Обробка помилки кроку 1 (приклад 3)

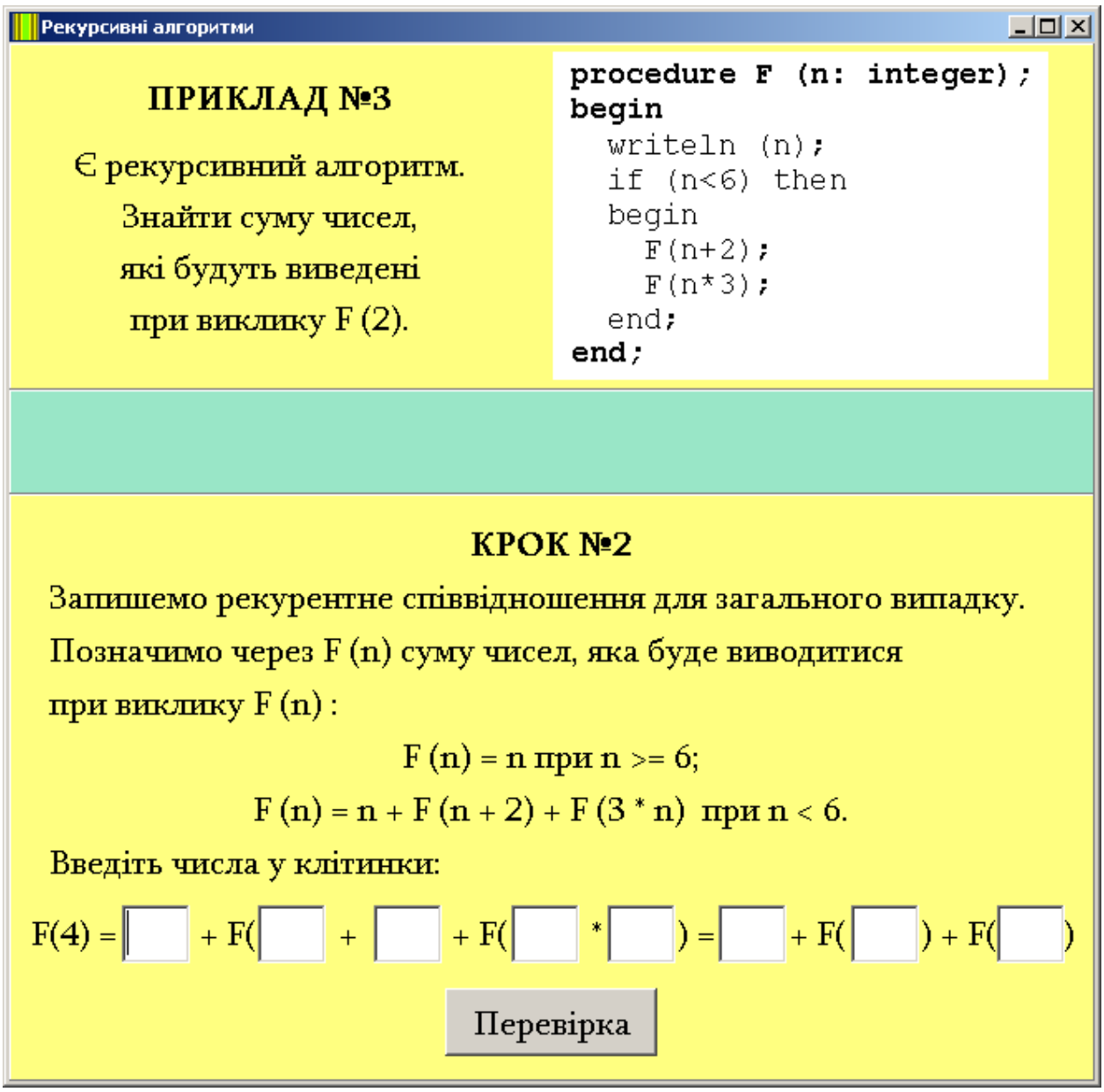

Рисунок А.3 – Крок 2 (приклад 3)

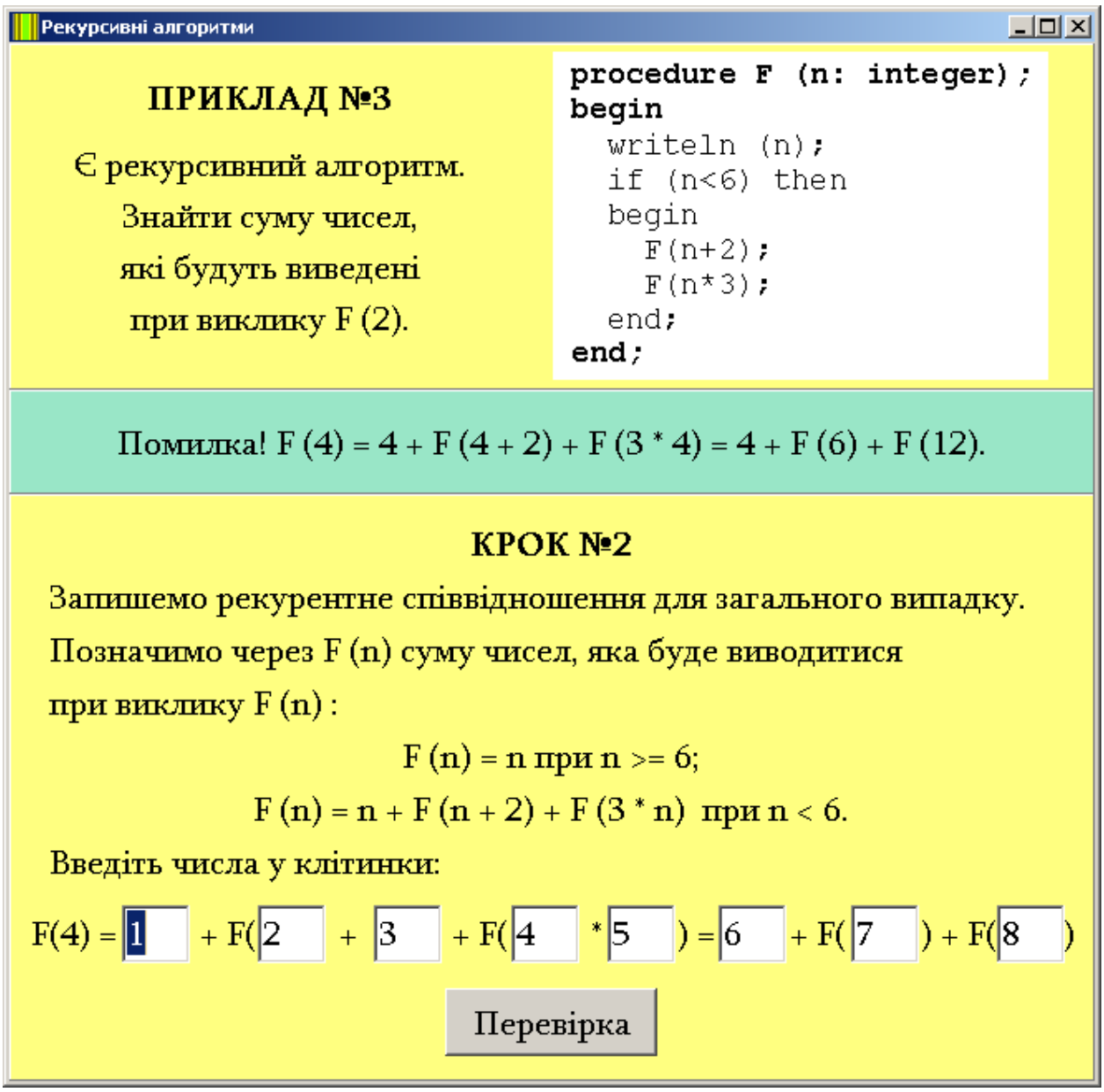

Рисунок А.4 – Обробка помилки кроку 2 (приклад 3)

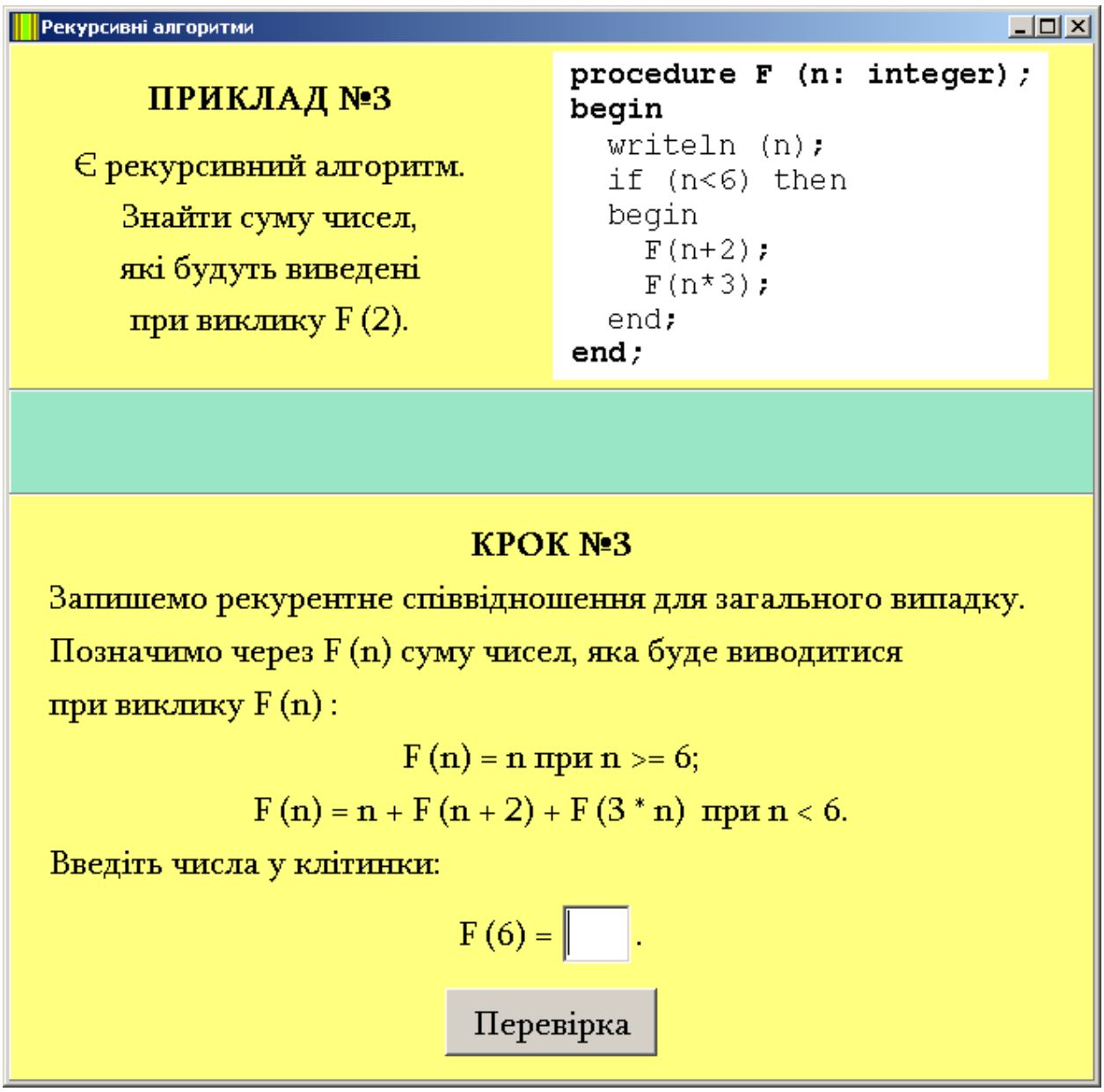

Рисунок А.5 – Крок 3 (приклад 3)

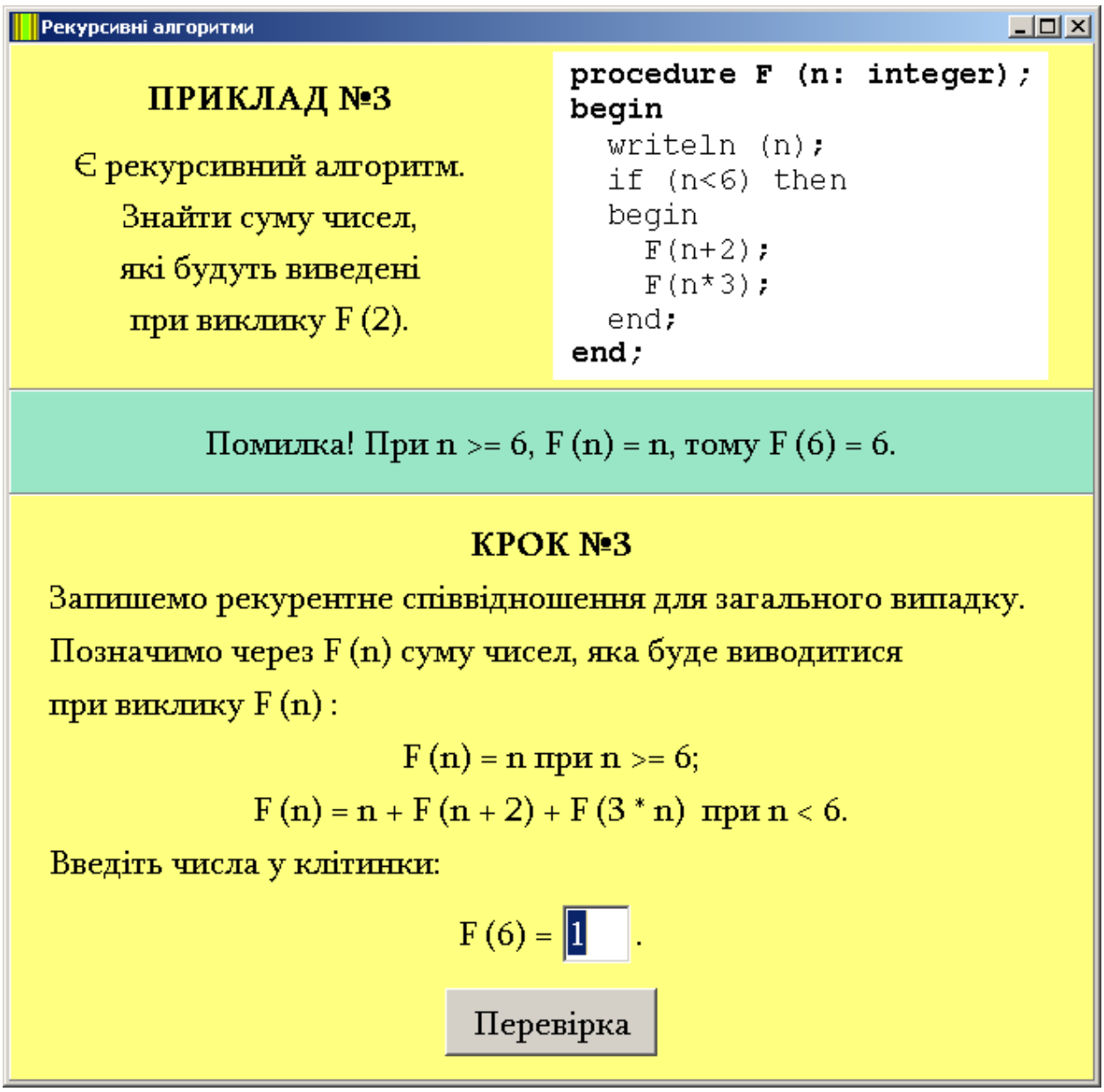

Рисунок А.6 – Обробка помилки кроку 3 (приклад 3)

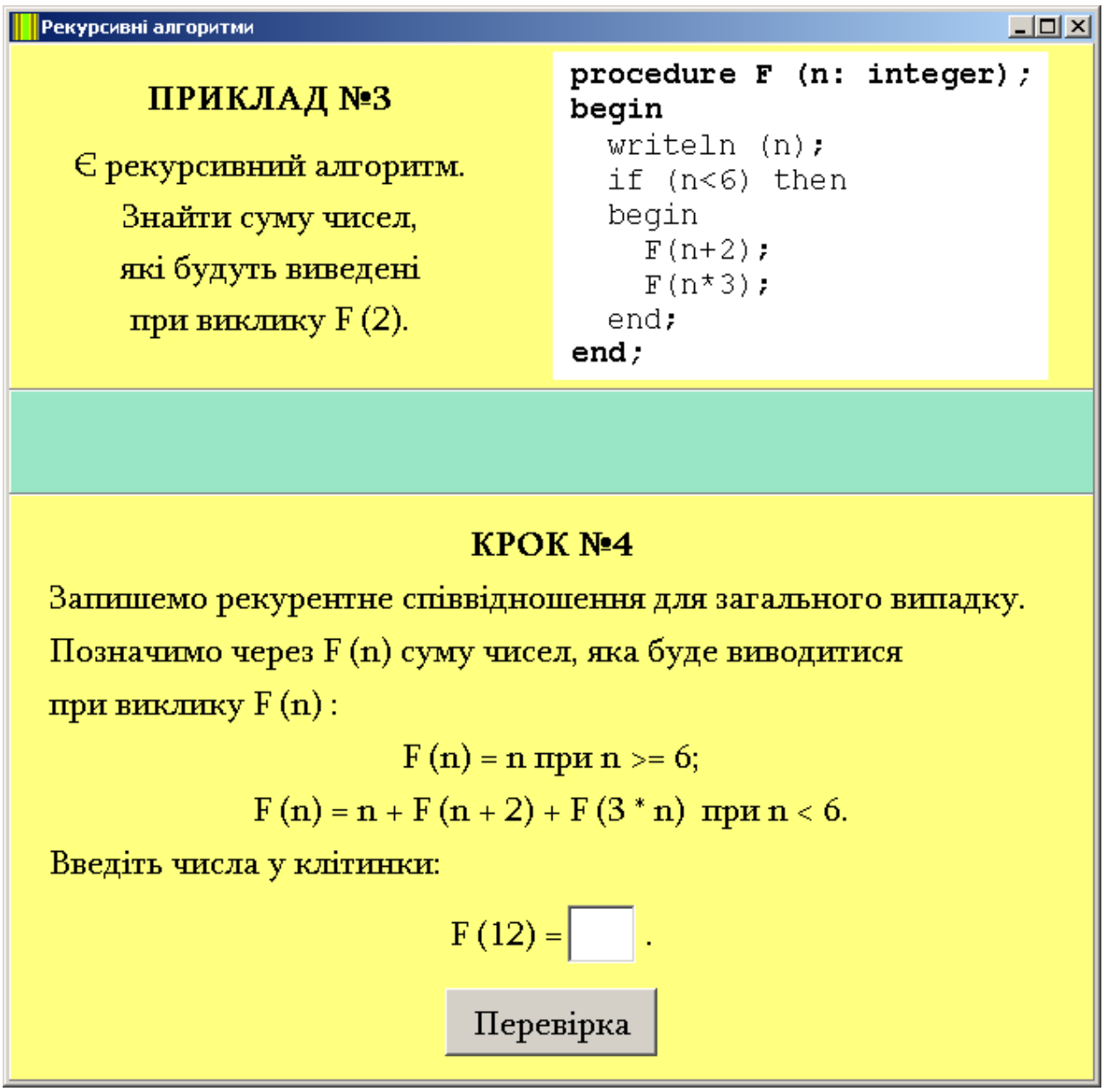

Рисунок А.7 – Крок 4 (приклад 3)

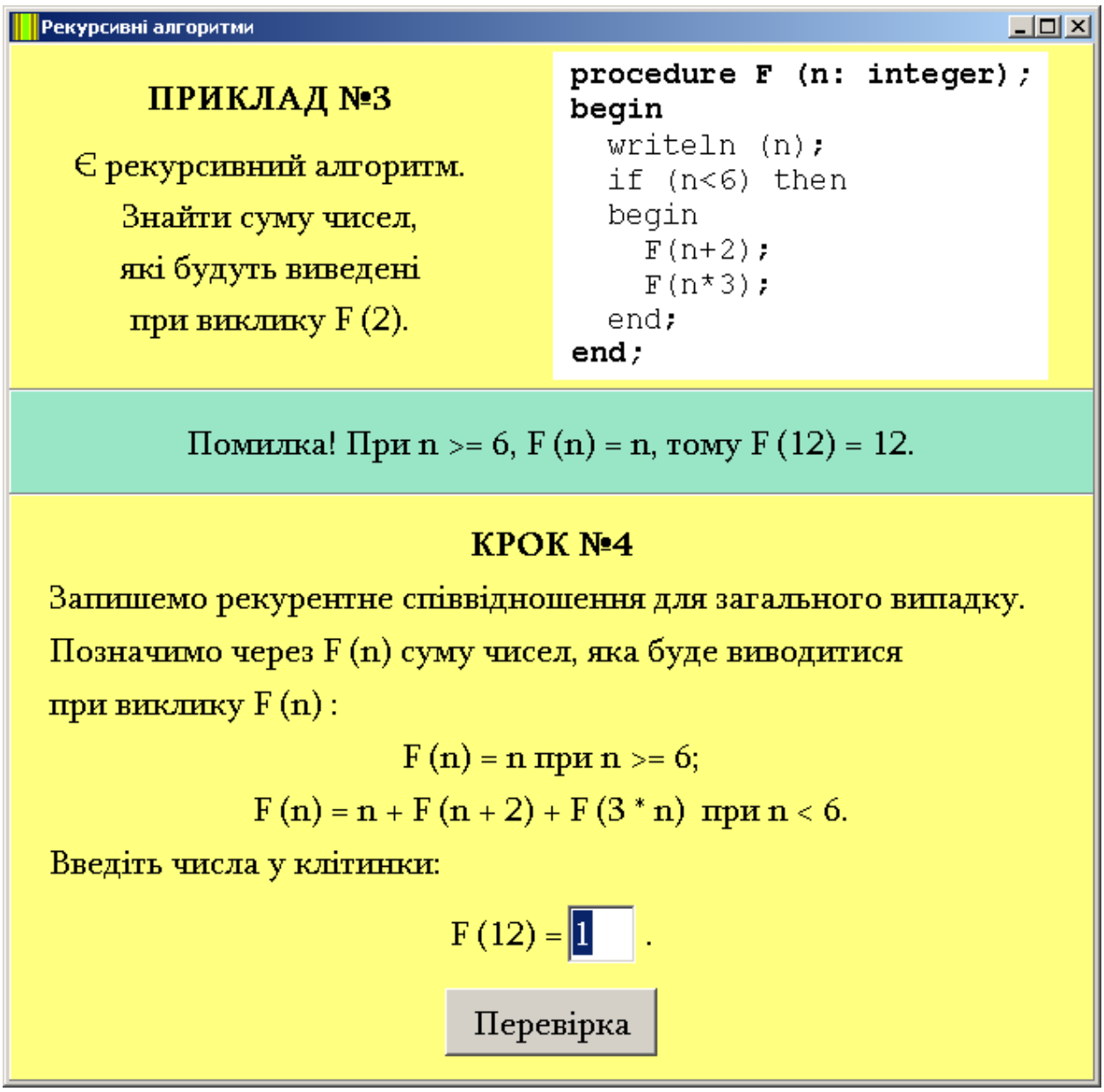

Рисунок А.8 – Обробка помилки кроку 4 (приклад 3)

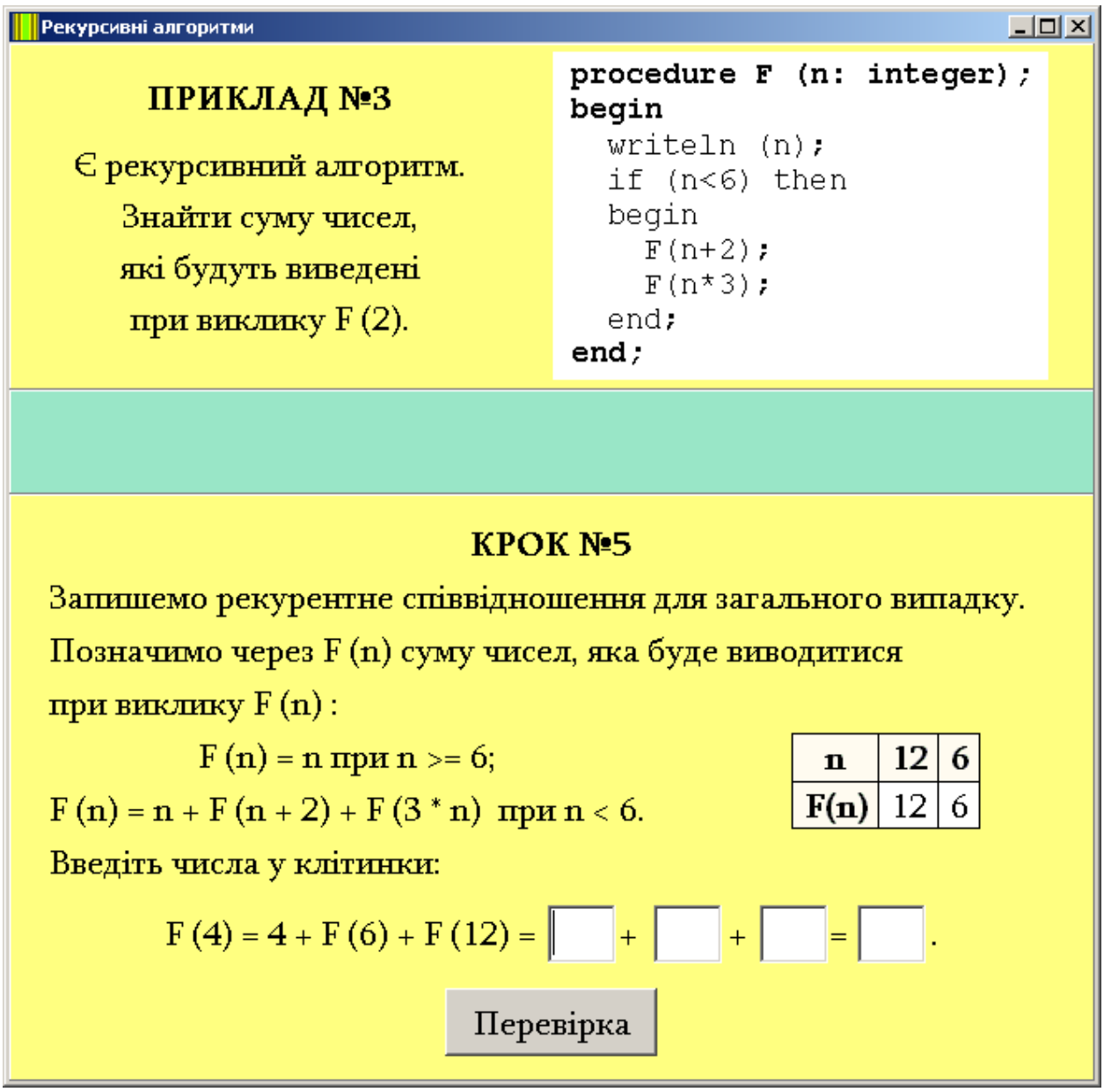

Рисунок А.9 – Крок 5 (приклад 3)

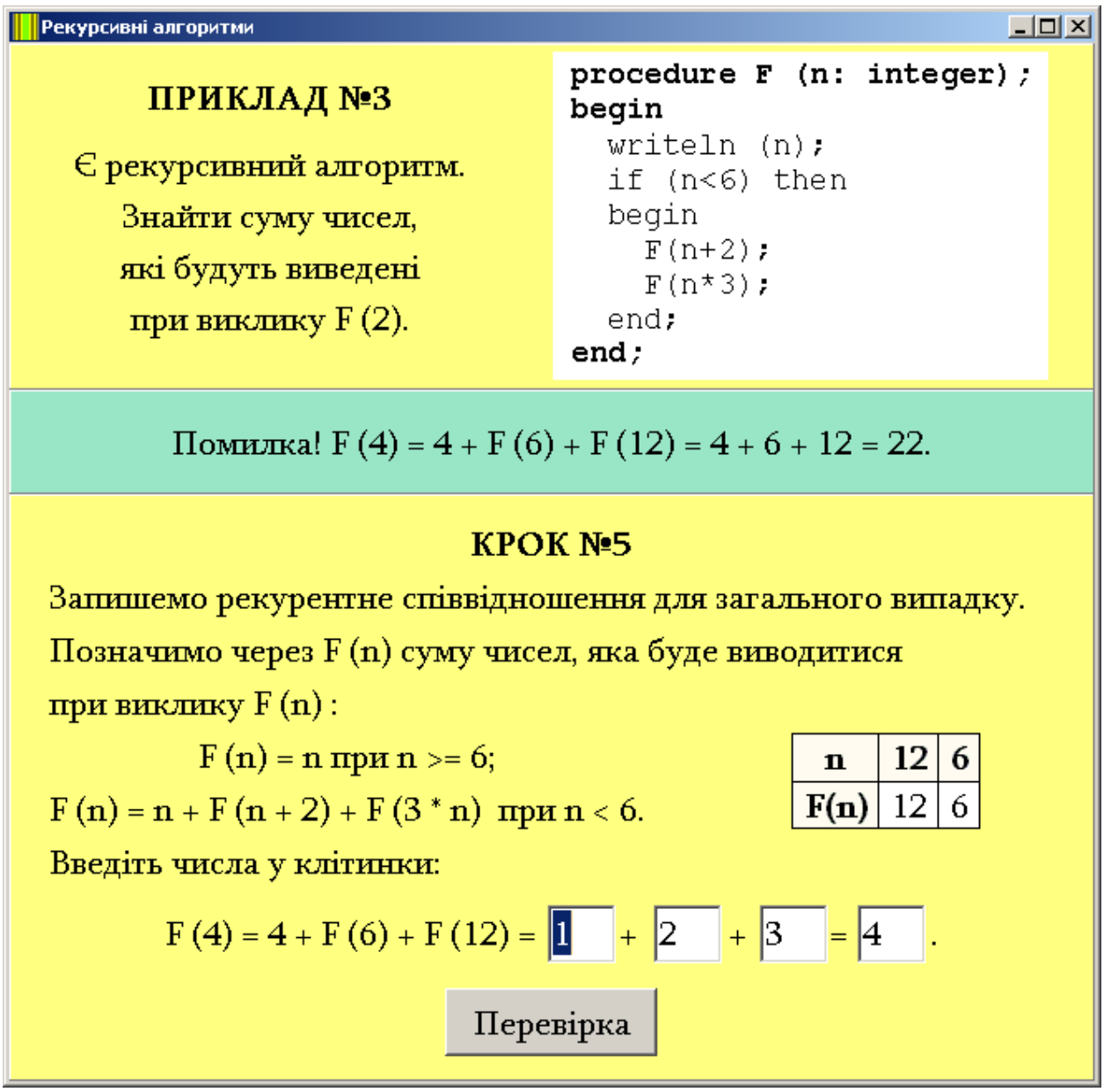

Рисунок А.10 – Обробка помилки кроку 5 (приклад 3)

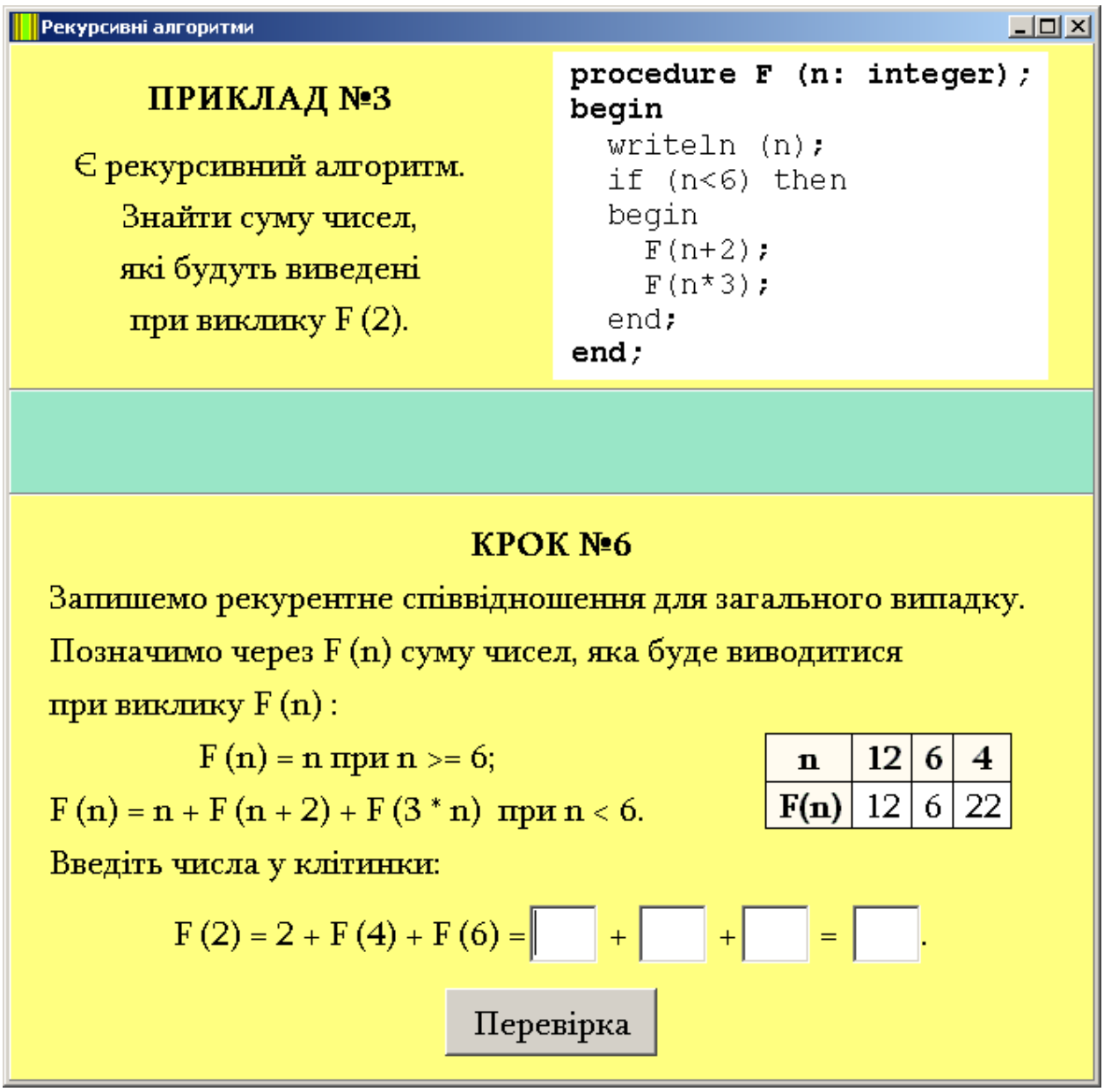

Рисунок А.11 – Крок 6 (приклад 3)

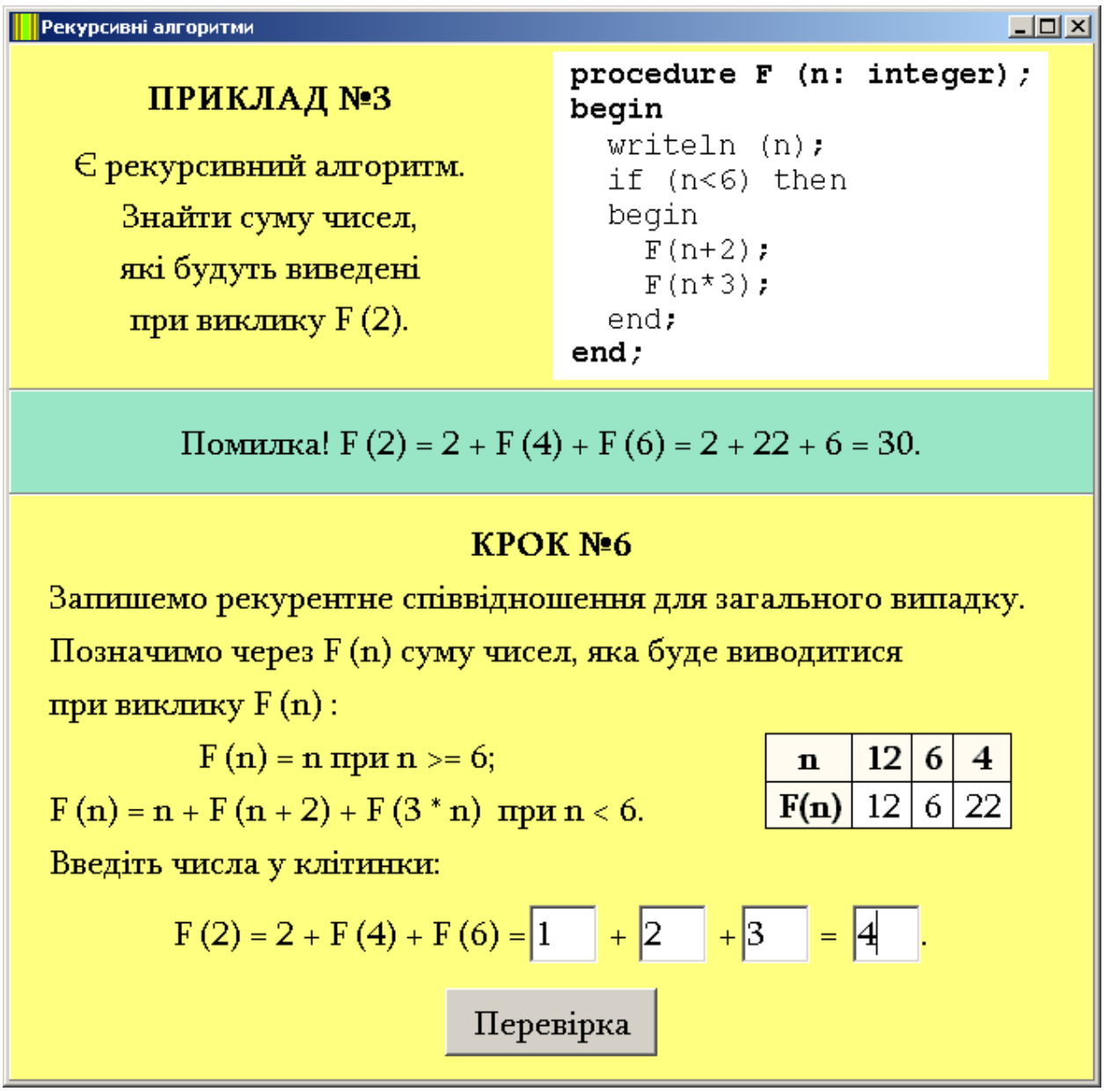

Рисунок А.12 – Обробка помилки кроку 6 (приклад 3)

ДОДАТОК Б
КОД ПРОГРАМИ

ПРИКЛАД 1

```
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "Unit2.h"
#include "Unit6.h"

#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{ }

// приклад 1
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
  Form2->Show();
  Form2->Edit1->SetFocus();
  Form1->Hide();
}

// приклад 2
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
  Form6->Show();
  Form6->Edit1->SetFocus();
  Form1->Hide();
}
```

```
#include <vcl.h>
#pragma hdrstop
#include "Unit2.h"
#include "Unit3.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;

__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{  }

// приклад 1,  крок 1
void __fastcall TForm2::BitBtn1Click(TObject *Sender)
{
  if ((Edit1->Text=="6") && (Edit2->Text=="7") && (Edit3->Text=="2"))
  {
    // наступний крок
    Form3->Show();
    Form3->Edit1->SetFocus();
    Form2->Hide();

    // слід очищати цей крок
    Form2->Panel3->Caption="";
    Form2->Edit1->Clear();
    Form2->Edit2->Clear();
    Form2->Edit3->Clear();
  }
  else
  {
    Panel3->Caption="Помилка! F (8) = F (8 - 2) * (8 - 1) + 2 = F (6) * 7 + 2.";
    Edit1->SetFocus();
  }
}

void __fastcall TForm2::FormClose(TObject *Sender, TCloseAction &Action)
{
   Application->Terminate();
```

```
}
#include <vcl.h>
#pragma hdrstop
#include "Unit3.h"
#include "Unit4.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm3 *Form3;

__fastcall TForm3::TForm3(TComponent* Owner) : TForm(Owner)
{  }

void __fastcall TForm3::FormClose(TObject *Sender, TCloseAction &Action)
{
  Application->Terminate();
}

// приклад 1,  крок 2
void __fastcall TForm3::BitBtn1Click(TObject *Sender)
{
  if ((Edit1->Text=="6") && (Edit2->Text=="2") && (Edit3->Text=="6") && (Edit4-
>Text=="1") && (Edit5->Text=="2") && (Edit6->Text=="4") && (Edit7-> Text=="5")
&& (Edit8->Text=="2"))
  {
    Form4->Show();
    Form4->Edit1->SetFocus();
    Form3->Hide();

    Form3->Panel3->Caption="";     // слід очищати цей крок
    Form3->Edit1->Clear();     Form3->Edit2->Clear();
    Form3->Edit3->Clear();     Form3->Edit4->Clear();
    Form3->Edit5->Clear();     Form3->Edit6->Clear();
    Form3->Edit7->Clear();     Form3->Edit8->Clear();
  }
  else
  {
    Panel3->Caption="Помилка! F (6) = F (6 - 2) * (6 - 1) + 2 = F (4) * 5 + 2.";
    Edit1->SetFocus();
  }}
```

```
#include <vcl.h>
#pragma hdrstop
#include "Unit4.h"
#include "Unit5.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm4 *Form4;

__fastcall TForm4::TForm4(TComponent* Owner) : TForm(Owner)
{ }

void __fastcall TForm4::FormClose(TObject *Sender, TCloseAction &Action)
{  Application->Terminate();                }

// приклад 1,  крок 3
void __fastcall TForm4::BitBtn1Click(TObject *Sender)
{
  if ((Edit1->Text=="4") && (Edit2->Text=="2") && (Edit3->Text=="4") && (Edit4-
>Text=="1") && (Edit5->Text=="2") && (Edit6->Text=="2") && (Edit7-> Text=="3")
&& (Edit8->Text=="2") && (Edit9->Text=="1") && (Edit10-> Text=="3") && (Edit11-
>Text=="2") && (Edit12->Text=="3") && (Edit13-> Text=="2") && (Edit14-
>Text=="5"))
  {
    Form5->Show();    Form5->Edit1->SetFocus();
    Form4->Hide();
    Form4->Panel3->Caption="";     // слід очищати цей крок
    Form4->Edit1->Clear();     Form4->Edit2->Clear();
    Form4->Edit3->Clear();     Form4->Edit4->Clear();
    Form4->Edit5->Clear();     Form4->Edit6->Clear();
    Form4->Edit7->Clear();     Form4->Edit8->Clear();
    Form4->Edit9->Clear();     Form4->Edit10->Clear();
    Form4->Edit11->Clear();     Form4->Edit12->Clear();
    Form4->Edit13->Clear();     Form4->Edit14->Clear();
  }
  else
  {  Panel3->Caption="Помилка! F(4)=F(4-2)*(4-1)+2=F(2)*3+2=1*3+2=3+2=5.";
     Edit1->SetFocus();
  }
```

```
}
#include <vcl.h>
#pragma hdrstop

#include "Unit5.h"
#include "Unit6.h"

#pragma package(smart_init)
#pragma resource "*.dfm"

TForm5 *Form5;
__fastcall TForm5::TForm5(TComponent* Owner) : TForm(Owner)
{ }

void __fastcall TForm5::FormClose(TObject *Sender, TCloseAction &Action)
{
  Application->Terminate();
}

// приклад 1,  крок 4
void __fastcall TForm5::BitBtn1Click(TObject *Sender)
{
  if ((Edit1->Text=="5")&&(Edit2->Text=="27"))
  {
    Form6->Show();
    Form5->Hide();

    // слід очищати цей крок
    Form5->Panel3->Caption="";
    Form5->Edit1->Clear();
    Form5->Edit2->Clear();
  }
  else
  {
    Panel3->Caption="Помилка! F (6) = F (4) * 5 + 2 = 5 * 5 + 2 = 27.";
    Edit1->SetFocus();
  }
}
```

```
#include <vcl.h>
#pragma hdrstop

#include "Unit6.h"
#include "Unit1.h"

#pragma package(smart_init)
#pragma resource "*.dfm"
TForm6 *Form6;

__fastcall TForm6::TForm6(TComponent* Owner)
        : TForm(Owner)
{ }

void __fastcall TForm6::FormClose(TObject *Sender, TCloseAction &Action)
{
  Application->Terminate();
}
// приклад 1
// крок 5
void __fastcall TForm6::BitBtn1Click(TObject *Sender)
{
  if ((Edit1->Text=="27")&&(Edit2->Text=="56"))
  {
    Form1->Show();
    Form6->Hide();
    // слід очищати цей крок
    Form6->Panel3->Caption="";
    Form6->Edit1->Clear();
    Form6->Edit2->Clear();
  }
  else
  {
    Panel3->Caption="Помилка! F (8) = F (6) * 7 + 2 = 27 * 7 + 2 = 56.";
    Edit1->SetFocus();
  }
}
```