ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ФОРМА НАВЧАННЯ ДЕННА

КАФЕДРА МАТЕМАТИЧНОГО МОДЕЛЮВАННЯТА СОЦІАЛЬНОЇ ІНФОРМАТИКИ

Допускається до захисту

Завідувач кафедри _____О.О. Ємець
(підпис)

«_____»_____ 2020 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА ДО БАКАЛАВРСЬКОЇ РОБОТИ

на тему

ТРЕНАЖЕР З ТЕМИ «РЕКУРСИВНЕ ПОРОДЖЕННЯ ПЕРЕСТАВЛЕНЬ» ДИСТАНЦІЙНОГО НАВЧАЛЬНОГО КУРСУ «АЛГОРИТМИ ТА СТРУКТУРИ ДАНИХ» ТА РОЗРОБКА ЙОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

зі спеціальності 122 «Комп'ютерні науки»

Виконавець роботи Шульга Ілля Ігорович _____ «___»_____2020 р.
(підпис)

Науковий керівник к.ф.-м.н., доц., Ємець Олександра Олегівна
_____ «___»_____ 2020 р.
(підпис)

ПОЛТАВА 2020 р.

РЕФЕРАТ

Записка: 45 с., 31 рис., 3 додатки (на 34 сторінках), 7 джерел.

Предмет розробки – тренажер з теми «Рекурсивне породження переставлень».

Мета роботи – створити тренажер з теми «Рекурсивне породження переставлень».

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

Створено алгоритм для тренажеру «Рекурсивне породження переставлень». Нарисовано блок-схему алгоритму. Створено програмну реалізацію тренажеру з теми «Рекурсивне породження переставлень».

Ключові слова: КОМБІНАТОРИКА, ПЕРЕСТАНОВКИ, РЕКУРСІЯ, РЕКУРСИВНЕ ПОРОДЖЕННЯ ПЕРЕСТАНОВОК, ТРЕНАЖЕР.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні позначення, символи, одиниці, скорочення, терміни	**Пояснення умовних позначень, символів, одиниць, скорочень, термінів**
n	Кількість елементів у перестановці.
$n!$	Факторіал числа n.
P_i	Перестановка з номером i
Рекурсивний алгоритм	Алгоритм, що використовує як допоміжний сам себе або інший алгоритм, який використовує всередині себе вихідний.
Рекурсивне поняття	Поняття, що воно визначається через те саме себе.

ВСТУП

Актуальність теми обумовлена тим, що на зміну класичній освіті (у навчальному закладі з викладачами) прийшли альтернативні способи навчання (наприклад, он-лайн через дистанційні ресурси). В зв'язку з цим допомагати особі, що навчається, буде не викладач, а програма-тренажер. І такі тренажери повинні існувати для кожного обчислювального методу або задачі.

Об'єкт розробки – комп'ютерний тренажер.

Предмет розробки – комп'ютерний тренажер з теми «Рекурсивне породження переставлень».

Мета випускової роботи – розробити навчальну програму, що навчає користувачів рекурсивному породженню переставлень.

Задачі випускової роботи – вивчити теоретичні відомості з теми «Породження переставлень»; переглянути тренажери схожої тематики та проаналізувати їх; створити алгоритм тренажеру та блок-схема алгоритму; написати програму.

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

Нові практичні розробки – створено новий тренажер з теми «Рекурсивне породження переставлень».

Особистий внесок – самостійно розроблено алгоритм, блок-схему та програму тренажеру.

Ступінь готовності до використання – програма повністю готова для роботи і передана для впровадження у дистанційний курс.

Структура пояснювальної записки: постановка задачі; інформаційний огляд; теоретична та практичні частини.

В першій частині (постановці) описані вимоги до роботи.

В другій (огляді) – здійснено огляд наявних в дистанційному курсі «Алгоритми і структури даних» тренажерів; викладено їх недоліки та переваги.

В третій (теоретичній) – викладено алгоритм програми та блок-схему.

В четвертій (практичній) – описано, як створювався програмний продукт, та подано інструкцію по роботі з ним.

Обсяг пояснювальної записки без додатків складає 45 сторінок.

1. ПОСТАНОВКА ЗАДАЧІ

Необхідно створити комп’ютерний тренажер, який навчає темі «Рекурсивне породження переставлень» дистанційного курсу «Алгоритми та структури даних».

Для створення програми розробити алгоритм тренажеру та блок-схему.

Тренажер слід створити на основі теорії з дистанційного курсу «Алгоритми та структури даних» Полтавського університету економіки і торгівлі.

Реалізувати навчання у програмі для кількості елементів $n = 3$ та/або $n = 4$. Приклад взяти з дистанційного курсу.

У тренажері умова прикладу (задачі) повинна бути видима на всіх кроках тренінгу.

У програмі повинні з’являтися пояснення помилок та підтвердження вірності.

Передбачити в тренажері рисунок (схему), яка б демонструвала, як генеруються перестановки.

Програму протестувати.

Протестовану програму описати.

Програму передати на впровадження у відповідний дистанційний курс Полтавського університету економіки і торгівлі.

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Огляд аналогічних розробок

Розглянемо вже існуючі тренажери дистанційного курсу «Алгоритми і структури даних».

1) Тренажер «Лексикографічне породження переставлень» (рис. 2.1-2.5) був створений у 2019 р. Цебою Миколою.

Містить як питання з теорії, так і практичні питання. Всього – 20 кроків.

Ідея алгоритму породження переставлень викладена у тренажері, та доступна під час тренінгу (рис. 2.4).

Є інформація про розробника програми (рис. 2.6).

Рисунок 2.1 – Перший тренажер

2) Тренажер «Рекурсивні алгоритми» (рис. 2.7-2.11) був створений у 2019 р. Чубом Олександром.

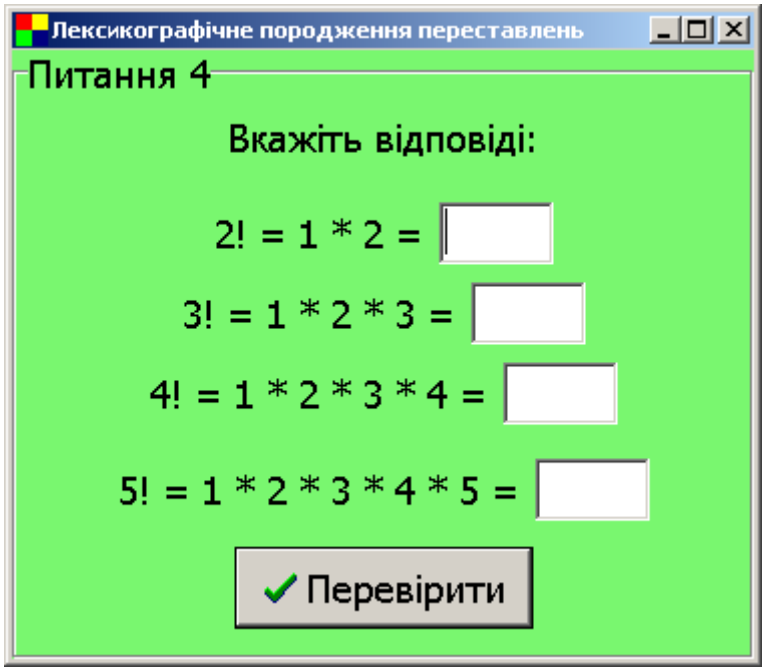


Рисунок 2.2 – Перший тренажер (питання №4)

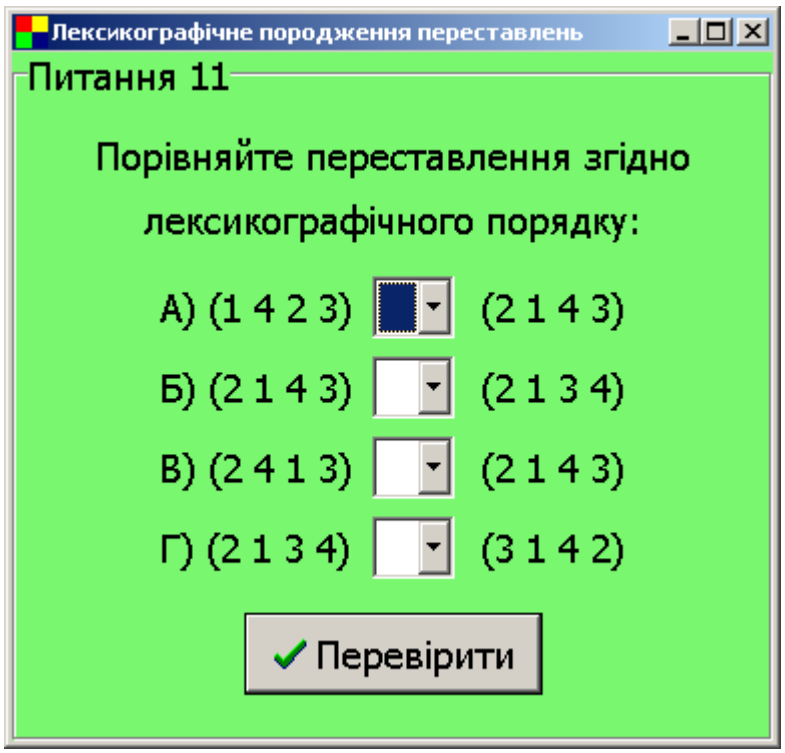


Рисунок 2.3 – Перший тренажер (питання №11)

Тренажер містить три різнопланових приклади на рекурсію. У двох прикладах рекурсія задана описом, у третьому через код програми. У двох прикладах використано пряму рекурсію, а в одному – не пряму.

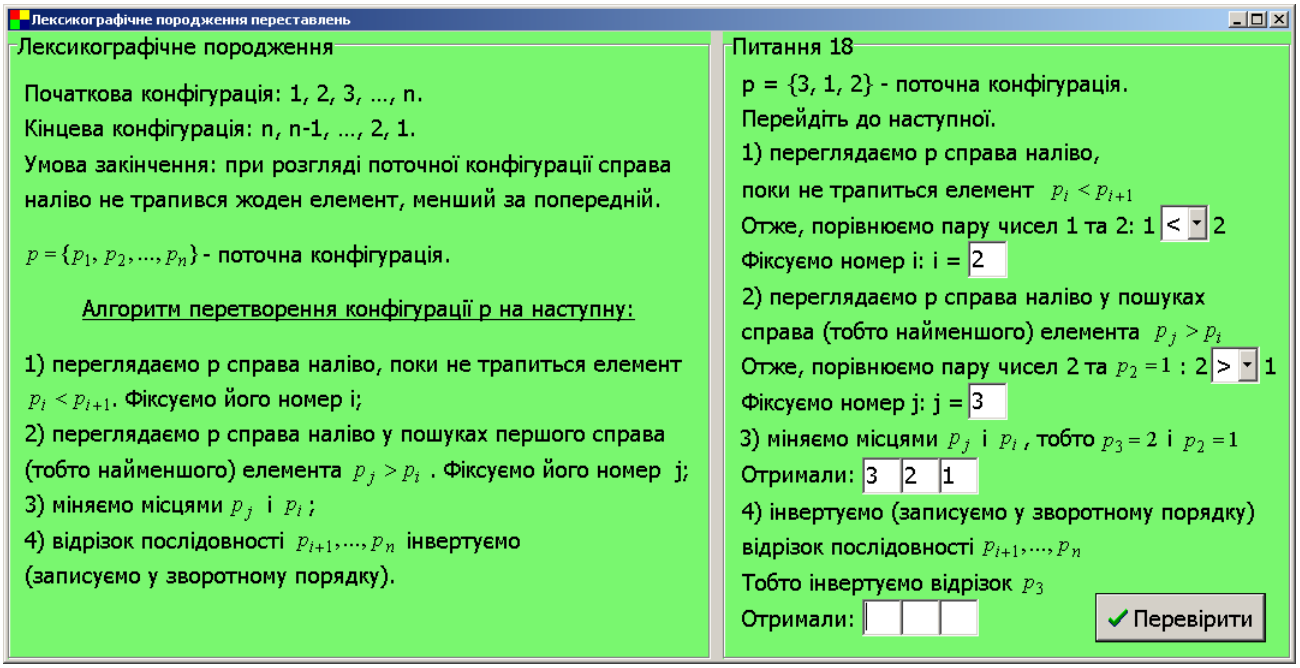


Рисунок 2.4 – Перший тренажер (питання №18)

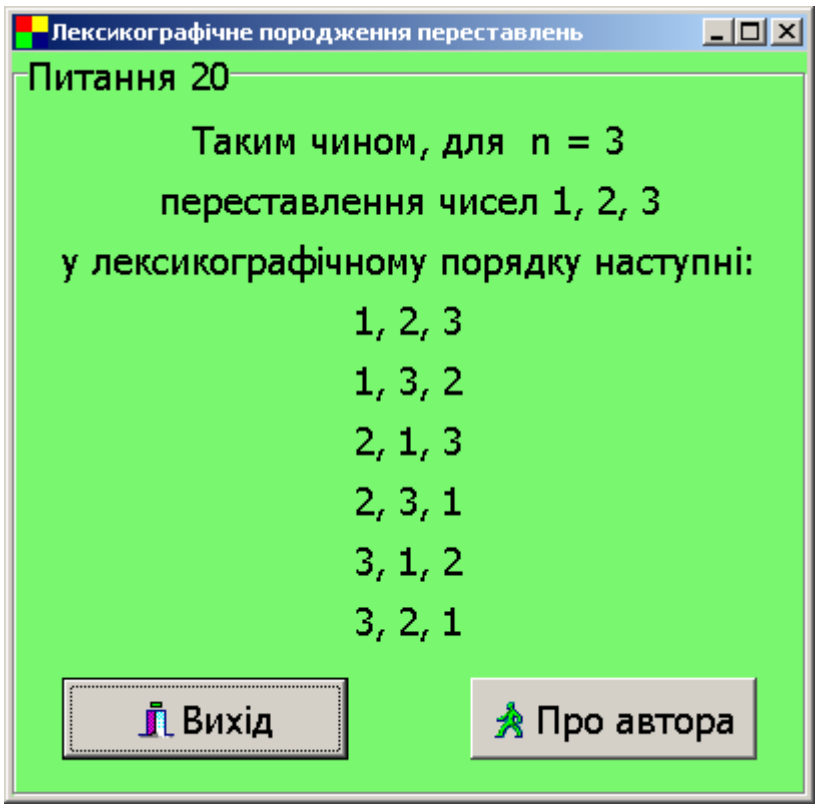


Рисунок 2.5 – Перший тренажер (питання №20)

Загалом в тренажері 26 кроків: 8 кроків у прикладі 1, 12 – у прикладі 2, 6 – у прикладі 3.

Цей тренажер також містить інформацію про автора (рис. 2.7).

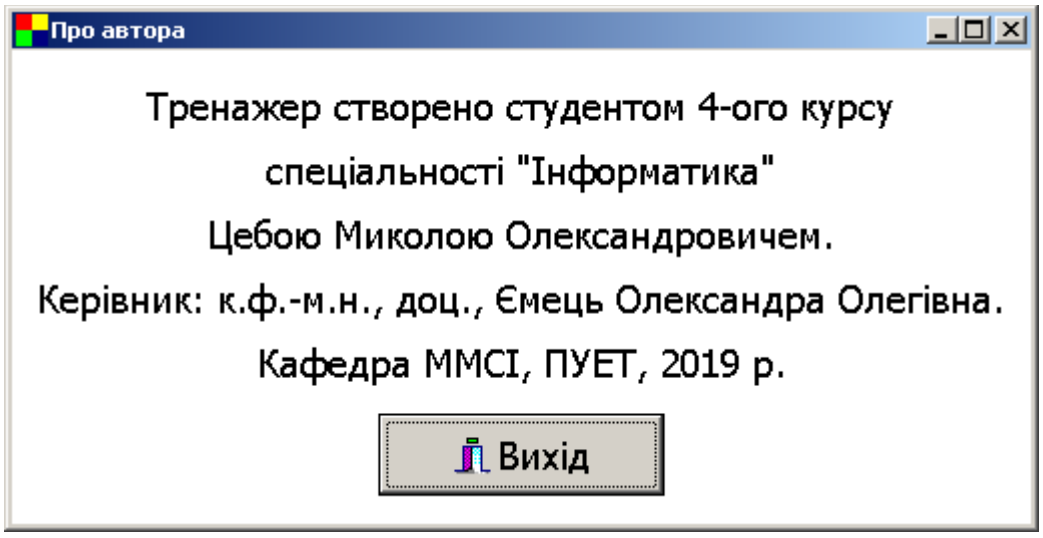


Рисунок 2.6 – Перший тренажер (інформація про розробника)

Рисунок 2.7 – Другий тренажер

3) Тренажер «Пірамідальне сортування» (рис. 2.12-2.16) був створений у 2019 р. Самборською Ксенією.

Пропонує тренінг українською (рис. 2.12) або англійською мовою (рис. 2.13). Містить близько 100 кроків в кожній версії.

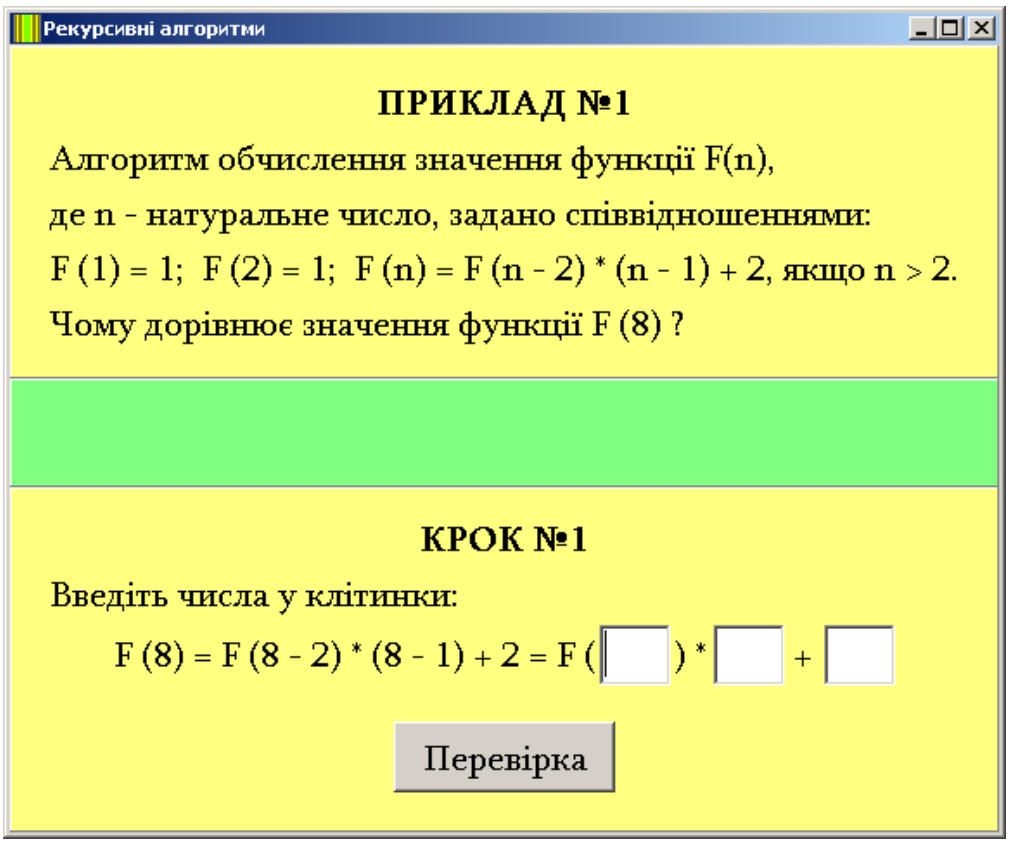


Рисунок 2.8 – Другий тренажер (приклад 1)

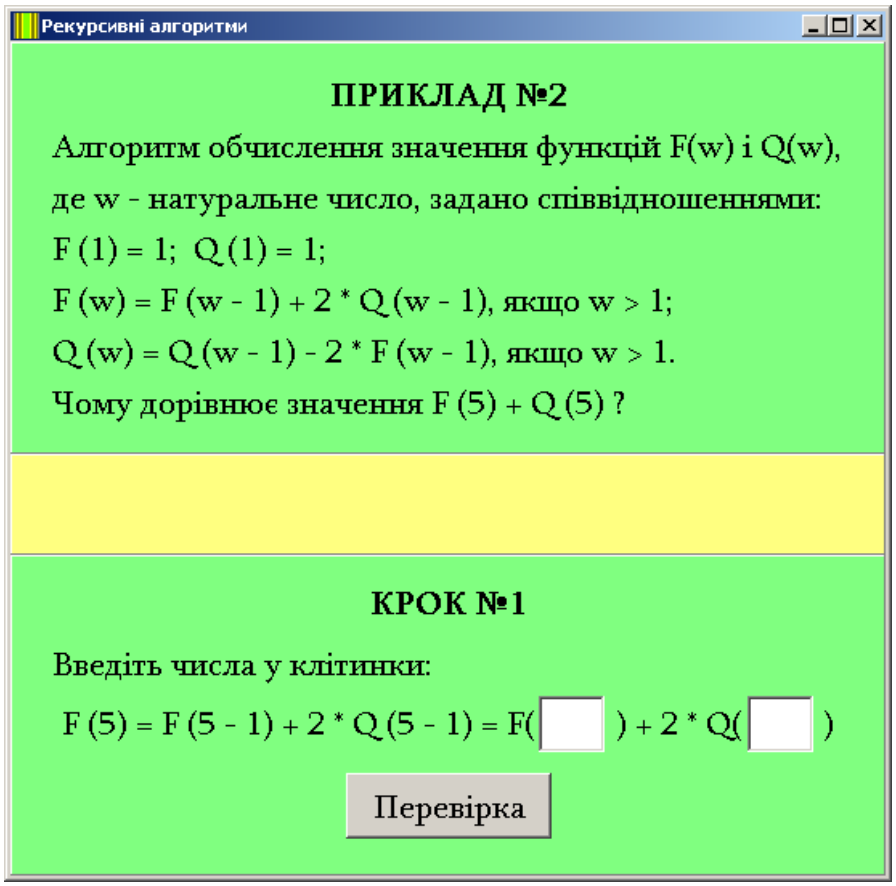


Рисунок 2.9 – Другий тренажер (приклад 2)

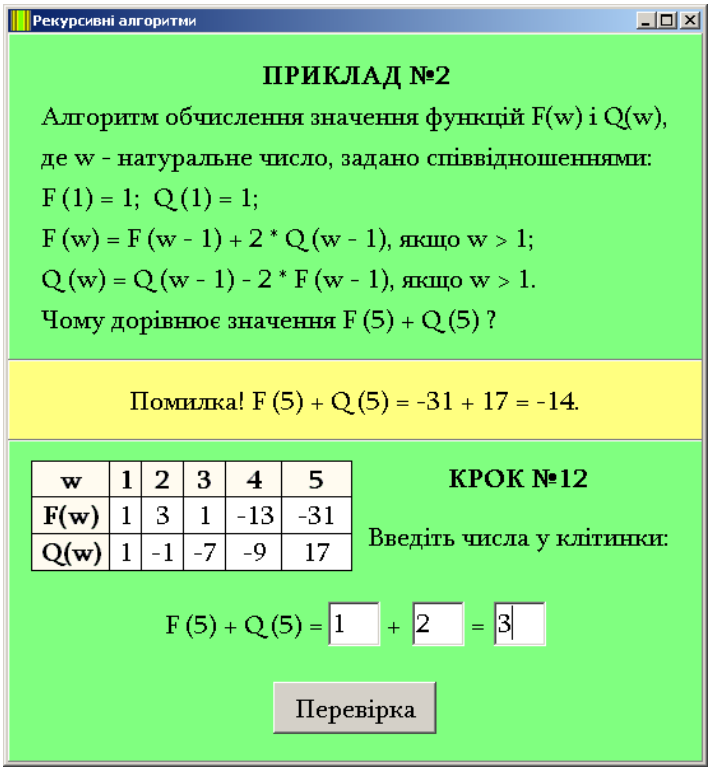


Рисунок 2.10 – Другий тренажер (приклад 2, крок 12)

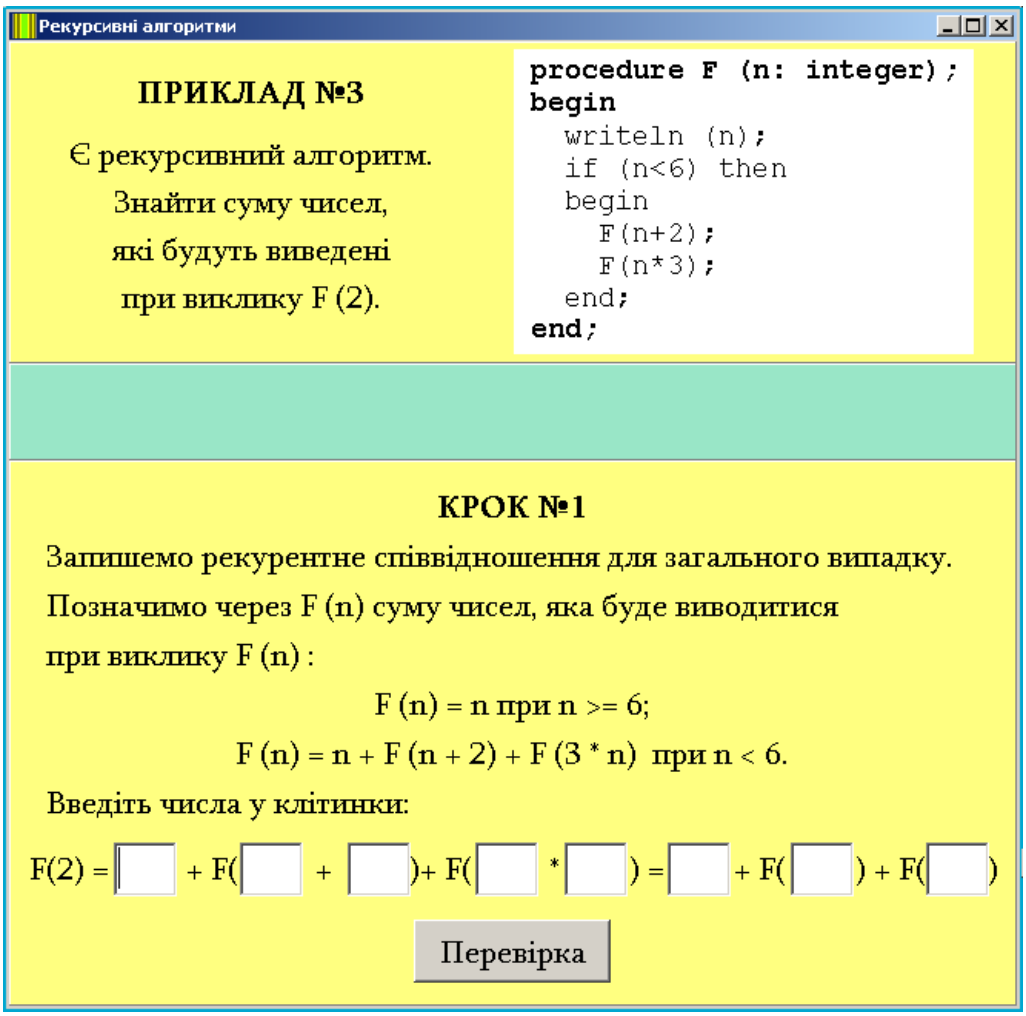


Рисунок 2.11 – Другий тренажер (приклад 3)

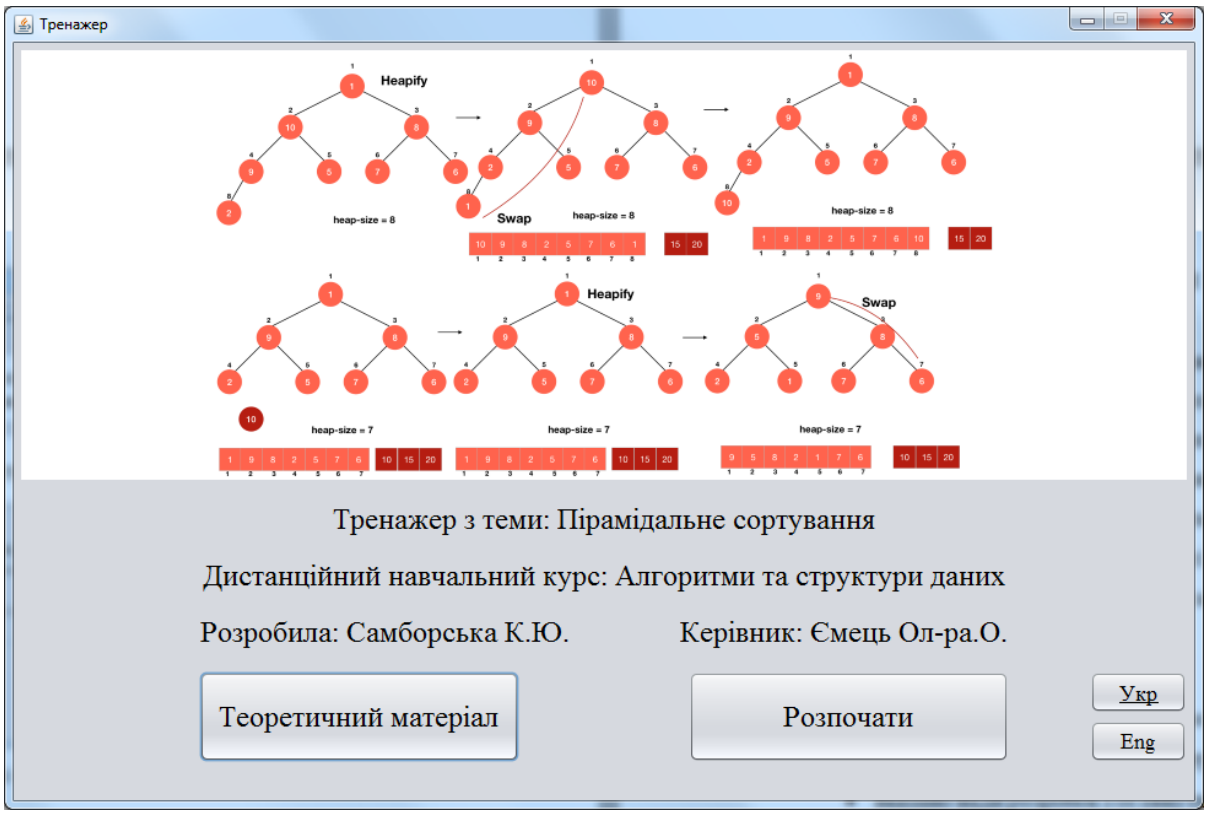


Рисунок 2.12 – Третій тренажер

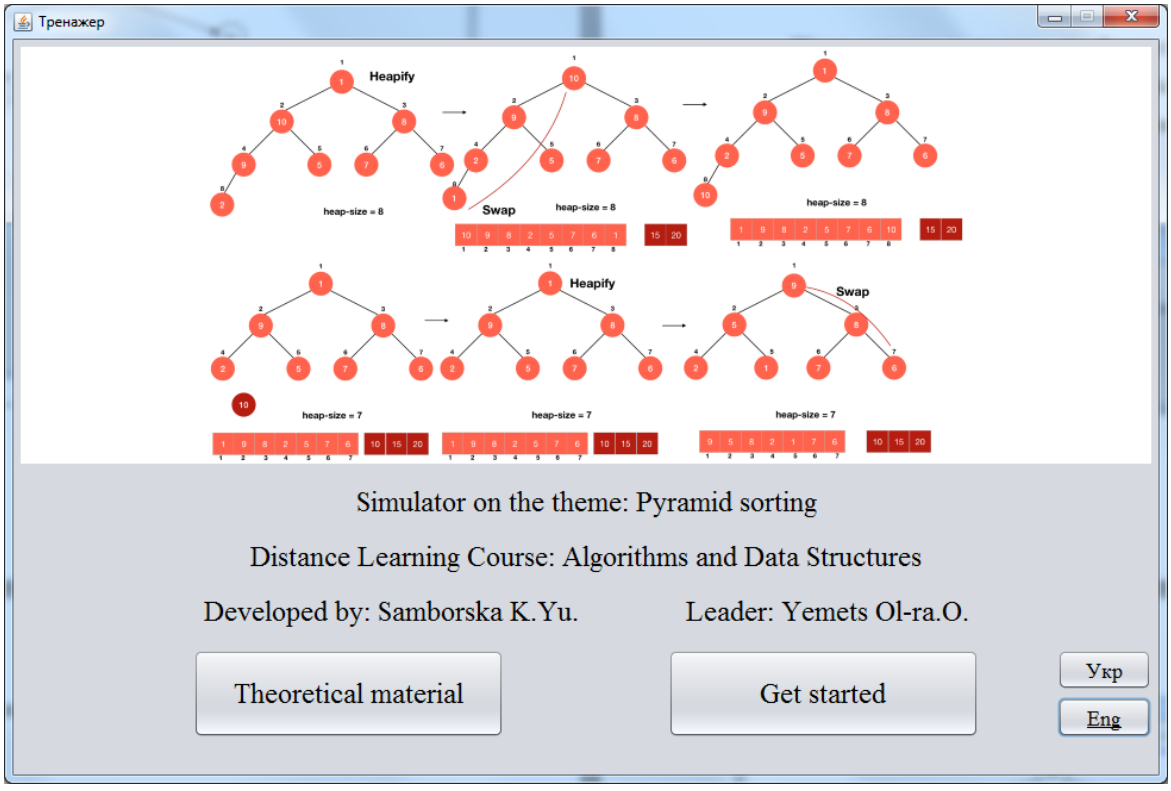


Рисунок 2.13 – Третій тренажер (англомовна версія)

Тренажер містить теоретичні відомості про метод (рис. 2.14).

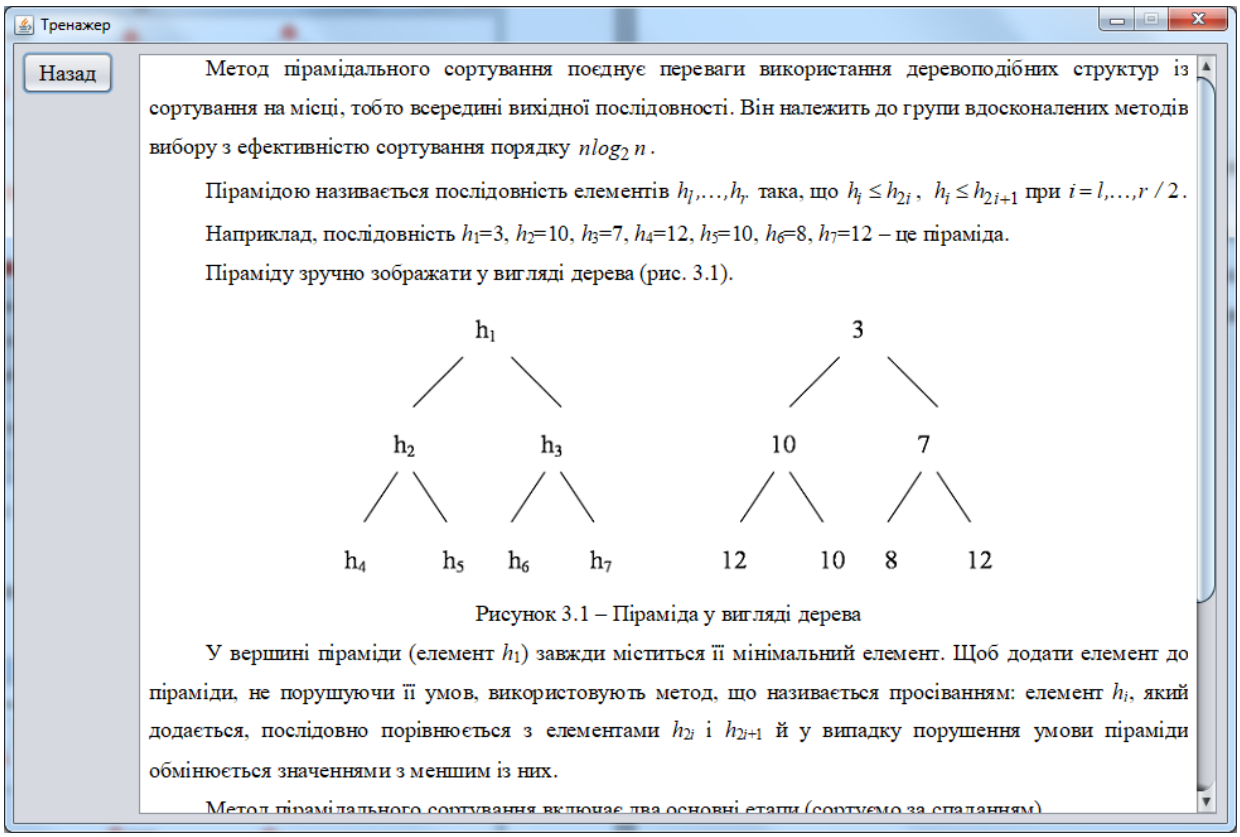


Рисунок 2.14 – Третій тренажер (теоретичні відомості)

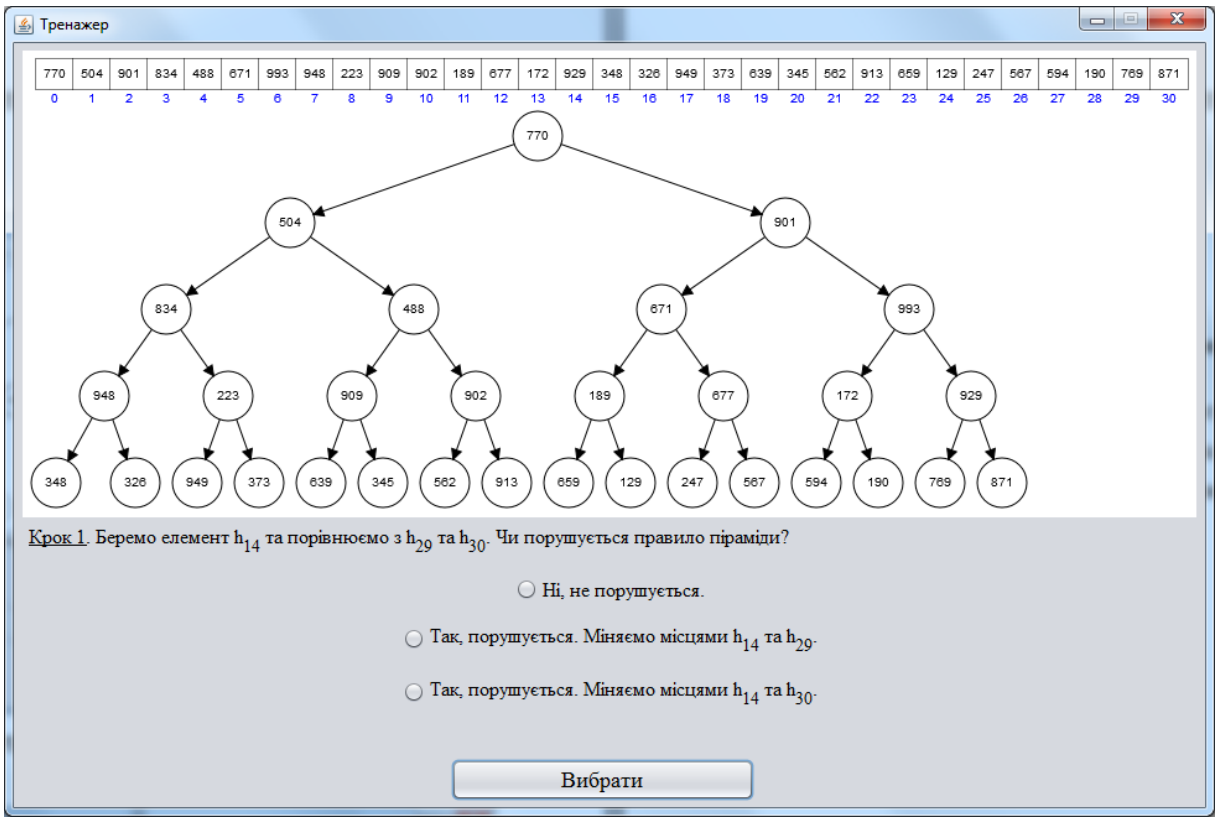


Рисунок 2.15 – Третій тренажер (приклад кроку)

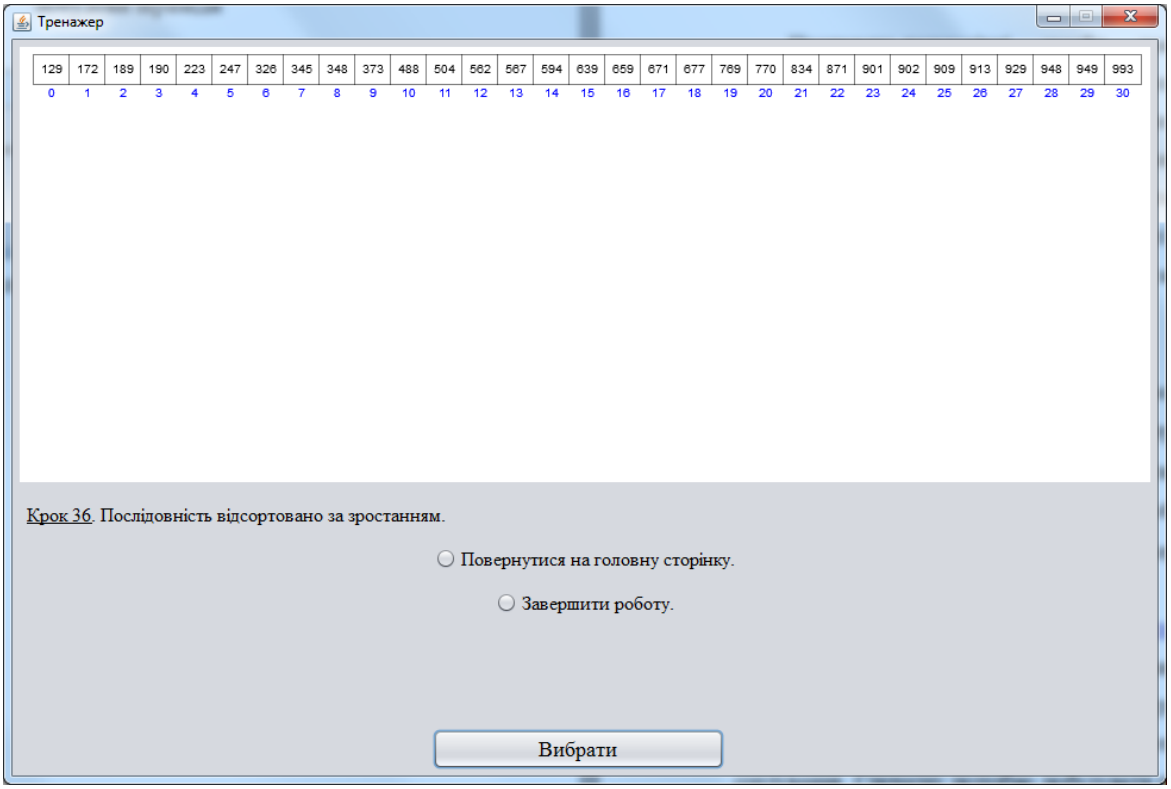


Рисунок 2.6 – Третій тренажер (завершення)

Також є інформація про автора (рис. 2.12-2.13).

2.2. Позитивні аспекти розглянутих розробок

1) Тренажер «Лексикографічне породження переставлень»:

- ✓ повно розрита тема;
- ✓ використано термінологію та позначки з дистанційного курсу;
- ✓ наявність на одній формі алгоритму та кроків;
- ✓ використання сплітеру для зміни ширини панелей;
- ✓ у дизайні використано кольори та картинки;
- ✓ наявна інформація про розробника програми.

2) Тренажер «Рекурсивні алгоритми»:

- ✓ підібрано різнопланові приклади;
- ✓ у дизайні використано кольори та картинки;
- ✓ використання різної кольорової гами для різних прикладів;
- ✓ гарно продумана структура кожного кроку;
- ✓ наявна інформація про розробника програми.

3) Тренажер «Пірамідальне сортування»:

- ✓ наявність двох мовних версій тренажеру;
- ✓ наявність теоретичних відомостей;
- ✓ наявність двох ілюстрацій пірамідального сортування – у вигляді масиву та у вигляді дерева;
- ✓ наявна інформація про розробника програми.

2.3. Негативні аспекти розглянутих розробок

1) Тренажер «Лексикографічне породження переставлень»:

- ✓ недоліків не виявлено.

2) Тренажер «Рекурсивні алгоритми»:

- ✓ недоліків не виявлено.

3) Тренажер «Пірамідальне сортування»:

- ✓ інші позначки, термінологія, напрямок сортування, ніж у дистанційному курсі;
- ✓ є помилки у методі;
- ✓ занадто громіздкий приклад для навчання – розглядається 31 елемент, а достатньо було б і 8 елементів.

2.4. Необхідність та актуальність теми

Перегляд та апробація тренажерів підтвердили ідею, що гарно розроблений тренажер (обов'язково із залученням викладача), допомагає засвоєнню матеріалу.

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Алгоритм тренажеру

Умова задачі завжди видима на екрані.

При вірній відповіді з’являється повідомлення «Правильно!» та відбувається перехід до наступного кроку.

При невірній відповіді з’являється повідомлення «Помилка!» та пояснення похибки. Користувачу потрібно виправити помилку.

Завдання. Утворити всі можливі перестановки чотирьох елементів, використовуючи метод рекурсивного породження перестановок.

Крок 1. Є множина елементів $a_1, ..., a_n$. Кількість всіх можливих перестановок – це величина

а) n;

б) $n!$**;**

в) $n!!$.

Правильна відповідь – «$n!$».

При помилці з’являється повідомлення «Кількість всіх можливих перестановок n елементів – це величина $n!$».

Крок 2. *Факторіал* числа n обчислюється за формулою

а) $n!=1\cdot 3\cdot 5\cdot \ldots \cdot n$;

б) $n!=2\cdot 4\cdot 6\cdot \ldots \cdot n$;

в) $n!=1\cdot 2\cdot 3\cdot \ldots \cdot n$**.**

Правильна відповідь – «$n!=1\cdot 2\cdot 3\cdot \ldots \cdot n$».

При помилці з'являється повідомлення «Факторіал числа n обчислюється за формулою $n!=1\cdot 2\cdot 3\cdot \ldots \cdot n$.».

Крок 3. *Поняття* називається *рекурсивним*, якщо воно визначається через те саме поняття.

а) Означення вірне;

б) Означення помилкове.

Правильна відповідь – «Означення вірне».

При помилці з'являється повідомлення «Означення вірне.».

Крок 4. *Алгоритм* називається *рекурсивним*, якщо він використовує як допоміжний сам себе або інший алгоритм, який використовує всередині себе вихідний.

а) Означення помилкове;

б) Означення вірне.

Правильна відповідь – «Означення вірне».

При помилці з'являється повідомлення «Означення вірне.».

Крок 5. *Алгоритм рекурсивного породження перестановок:*

1) Для $n=1$ – єдина перестановка: 1.

2) Нехай на множині з $n-1$ елемента вже побудовані всі можливі перестановки, $P_1, P_2, ..., P_{n-1}$, що задовольняють цій умові. Будемо розширювати кожну із цих перестановок P_i, вставляючи елемент n на кожне з можливих місць справа наліво, якщо i непарне, і зліва направо, якщо i парне.

а) Алгоритм вірний;

б) Алгоритм з помилками.

Правильна відповідь – «Алгоритм вірний».

При помилці з'являється повідомлення «Алгоритм рекурсивного породження перестановок вірний».

Крок 6. $n=1$. Кількість перестановок з n елементів – це

$$n!=$$

Правильна відповідь – «1».

При помилці з'являється повідомлення «1!=1».

Крок 7. $n=1$. Для цього n можна утворити лише одну перестановку

$$1.$$

Правильна відповідь – «1».

При помилці з’являється повідомлення «Для $n=1$ можна утворити лише одну перестановку $P_1:1.$».

Крок 8. Отримали:

$$P_1:\ \ 1$$

Крок 9. $n=2$. Кількість перестановок з n елементів – це

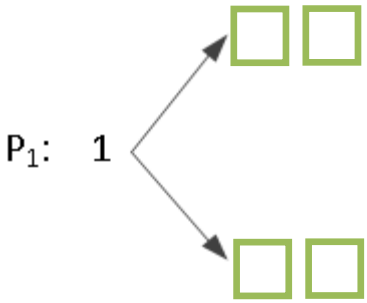
$$n!=2!=1\cdot 2=$$

Правильна відповідь – «2».

При помилці з’являється повідомлення «$2!=1\cdot 2=2.$».

Крок 10. $n=2.$

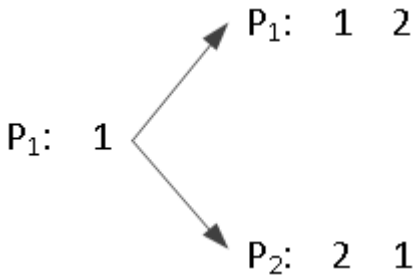
Розширимо перестановку $P_1: 1$ з кроку *7*, вставляючи елемент n на кожне з можливих місць справа наліво (оскільки, номер перестановки $i=1$ – це непарне число).



Правильна відповідь – «$P_1: 1\ 2,\ P_2: 2\ 1$».

При помилці з'являється повідомлення «Вставляючи $n=2$ на кожне з можливих місць справа наліво в перестановці $P_1:1$, отримуємо: $P_1: 1\ 2,\ P_2: 2\ 1$».

Крок 11. Отримали:



Крок 12. $n=3$. Кількість перестановок з n елементів це

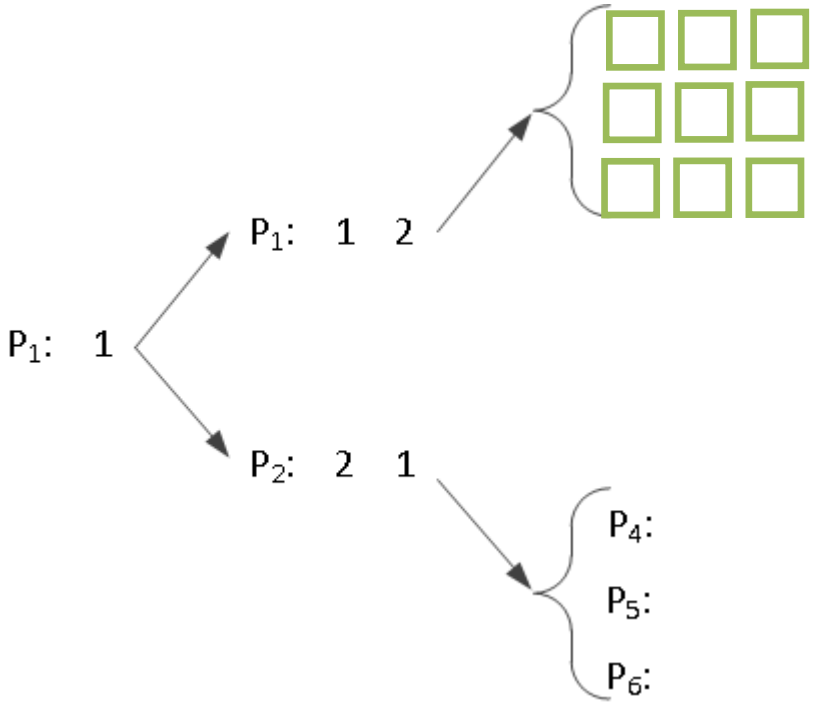
$$n!=3!=1\cdot 2\cdot 3=$$

Правильна відповідь – «6».

При помилці з'являється повідомлення «$3!=1\cdot 2\cdot 3=6$.».

Крок 13. $n=3$.

Розширимо перестановку P_1*: 1 2* з кроку *10*, вставляючи елемент n на кожне з можливих місць справа наліво (оскільки, номер перестановки $i=1$ – це непарне число).

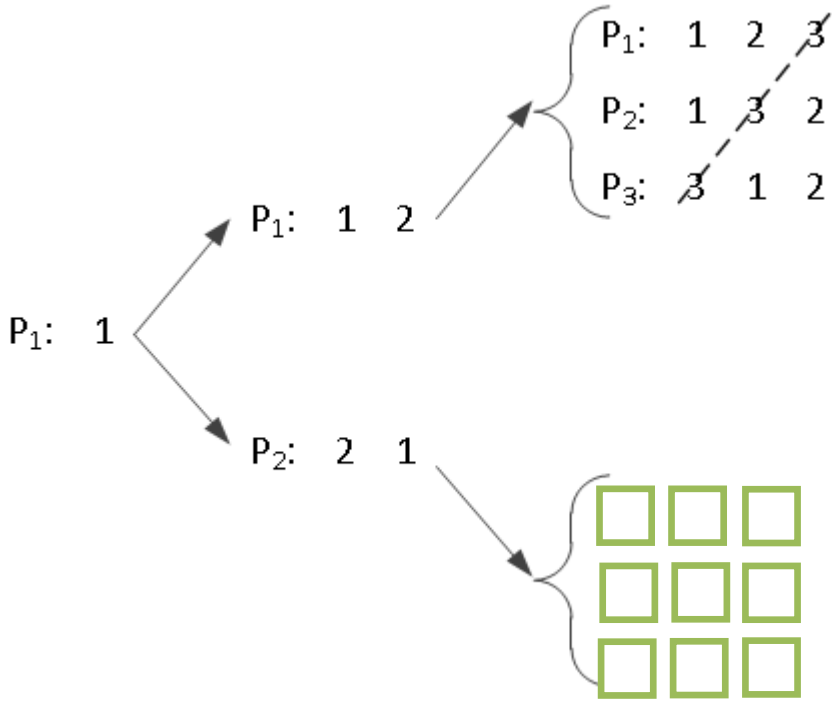


Правильна відповідь – «P_1: 1 2 3, P_2: 1 3 2, P_3: 3 1 2.».

При помилці з'являється повідомлення «Вставляючи $n=3$ на кожне з можливих місць справа наліво в перестановці P_1: 1 2, отримуємо: P_1: 1 2 3, P_2: 1 3 2, P_3: 3 1 2.».

Крок 14. $n=3$.

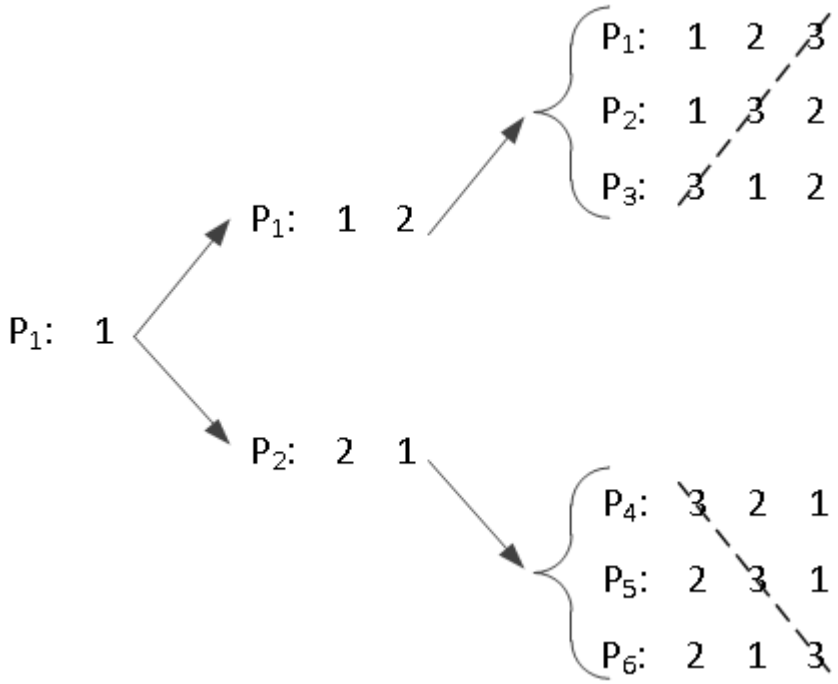
Розширимо перестановку P_2*: 2 1 з* кроку *10*, вставляючи елемент n на кожне з можливих місць зліва направо (оскільки, номер перестановки $i = 2$ – це парне число).



Правильна відповідь – «P_4 : 3 2 1, P_5 : 2 3 1, P_6 : 2 1 3.».

При помилці з'являється повідомлення «Вставляючи $n = 3$ на кожне з можливих місць зліва направо в перестановці P_2 : 2 1, отримуємо: P_4 : 3 2 1, P_5 : 2 3 1, P_6 : 2 1 3.».

Крок 15. Отримали:



Крок 16. $n=4$. Кількість перестановок з n елементів це

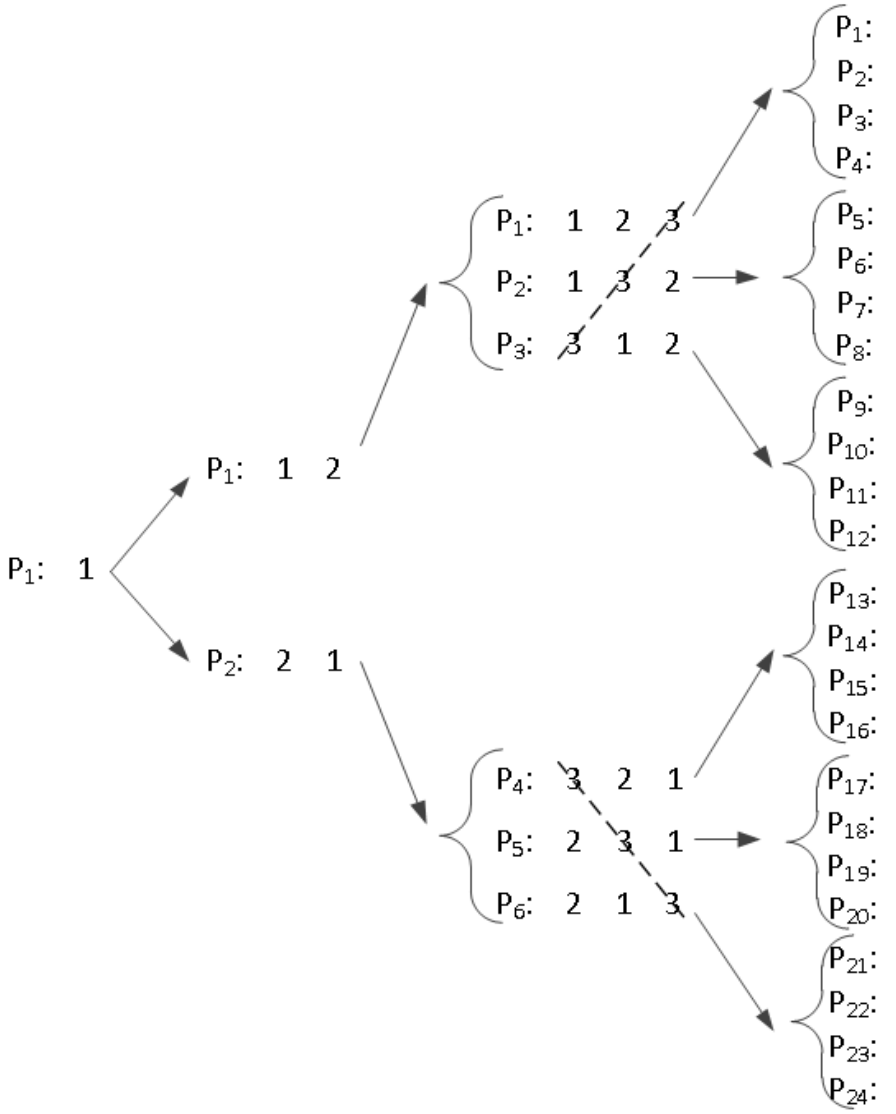
$$n!=4!=1\cdot 2\cdot 3\cdot 4=$$

Правильна відповідь – «24».

При помилці з'являється повідомлення «$4!=1\cdot 2\cdot 3\cdot 4=24$.».

Крок 17. $n=4$.

Розширимо перестановку P_1*: 1 2 3* з кроку *13*, вставляючи елемент n на кожне з можливих місць справа наліво (оскільки, номер перестановки $i=1$ – це непарне число).

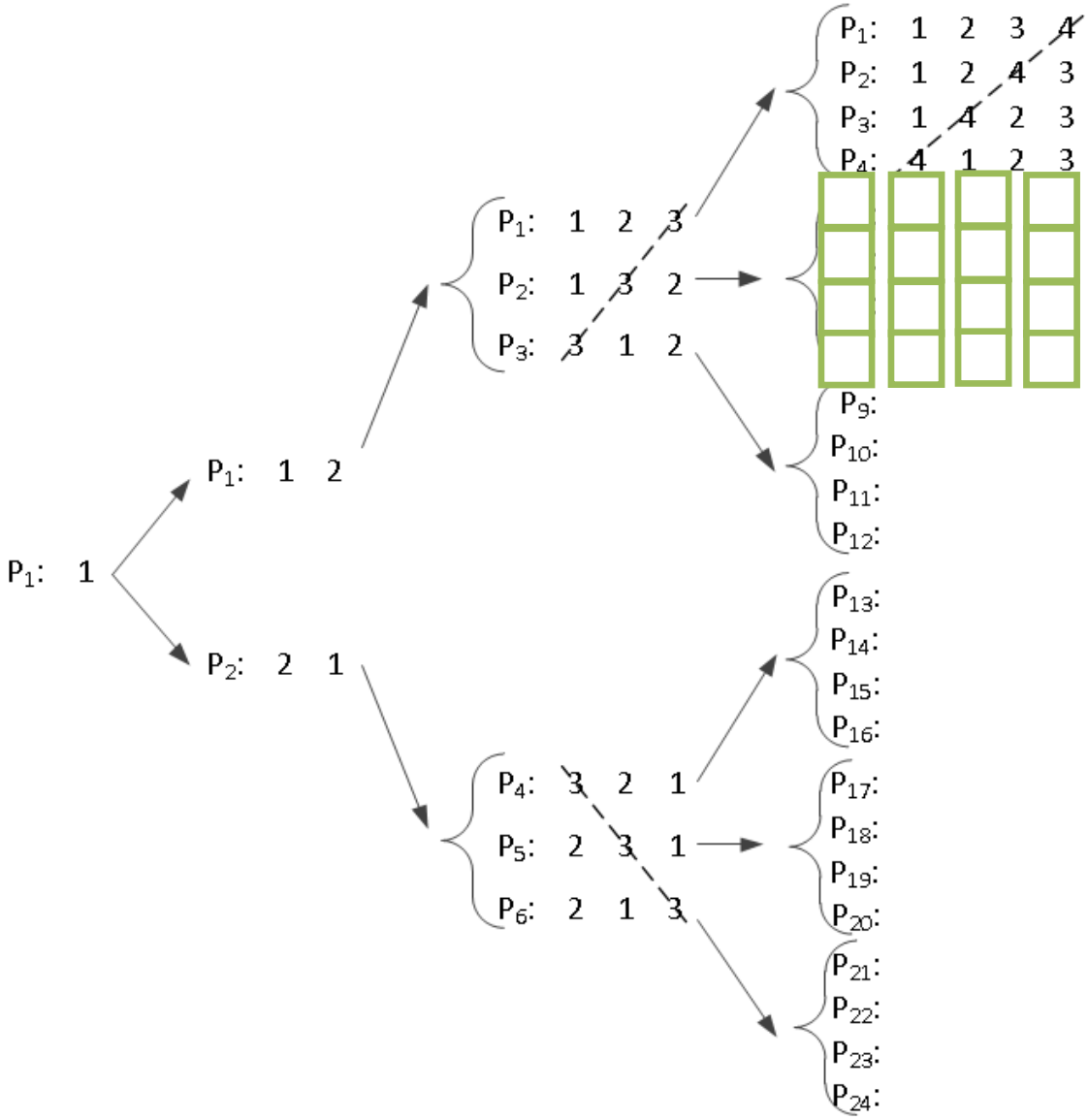


Правильна відповідь – «$P_1: 1\ 2\ 3\ 4,\ P_2: 1\ 2\ 4\ 3,\ P_3: 1\ 4\ 2\ 3,\ P_4: 4\ 1\ 2\ 3$.».

При помилці з’являється повідомлення «Вставляючи $n=4$ на кожне з можливих місць справа наліво в перестановці $P_1: 1\ 2\ 3$, отримуємо: $P_1: 1\ 2\ 3\ 4$, $P_2: 1\ 2\ 4\ 3, P_3: 1\ 4\ 2\ 3, P_4: 4\ 1\ 2\ 3$.».

Крок 18. $n=4$.

Розширимо перестановку P_2*: 1 3 2 з* кроку *13*, вставляючи елемент n на кожне з можливих місць зліва направо (оскільки, номер перестановки $i=2$ – це парне число).

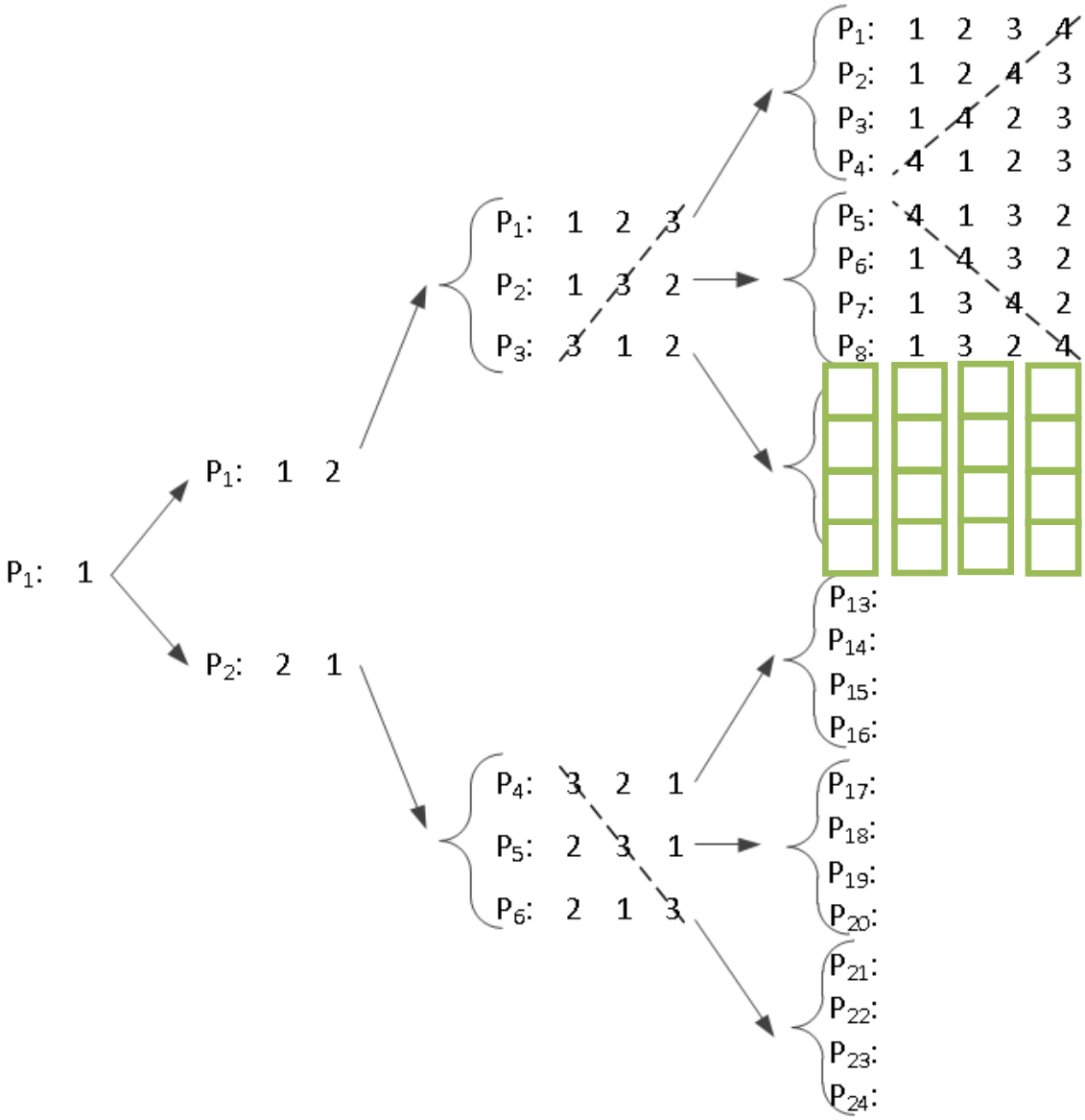


Правильна відповідь – «$P_5: 4\ 1\ 3\ 2$, $P_6: 1\ 4\ 3\ 2$, $P_7: 1\ 3\ 4\ 2$, $P_8: 1\ 3\ 2\ 4$ ».

При помилці з'являється повідомлення «Вставляючи $n=4$ на кожне з можливих місць зліва направо в перестановці $P_2: 1\ 3\ 2$, отримуємо: $P_5: 4\ 1\ 3\ 2$, $P_6: 1\ 4\ 3\ 2$, $P_7: 1\ 3\ 4\ 2$, $P_8: 1\ 3\ 2\ 4$.».

Крок 19. $n=4$.

Розширимо перестановку P_3: *3 1 2* з кроку *13*, вставляючи елемент n на кожне з можливих місць справа наліво (оскільки, номер перестановки $i = 3$ – це непарне число).

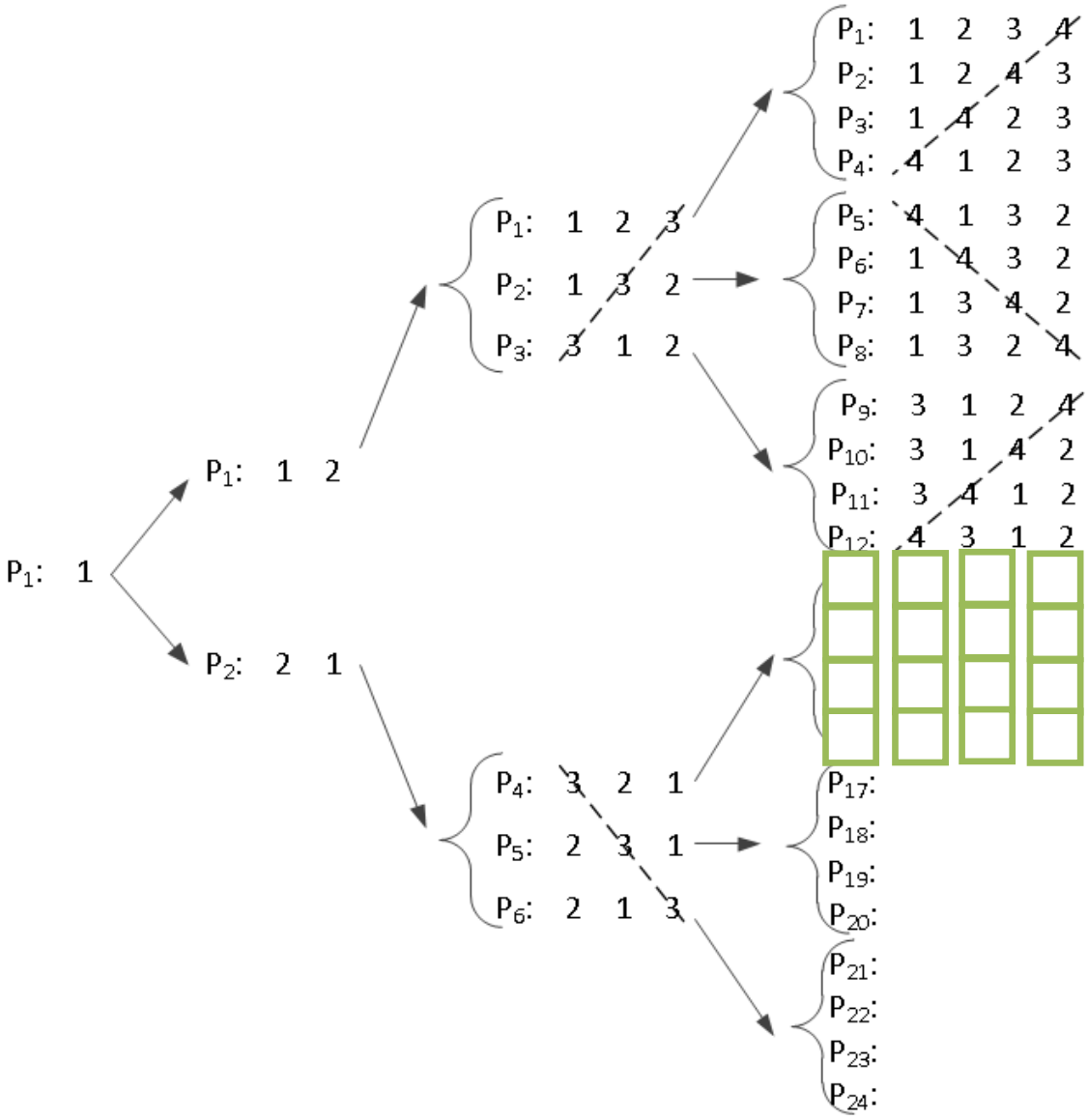


Правильна відповідь – «P_9 : 3 1 2 4, P_{10} : 3 1 4 2, P_{11} : 3 4 1 2, P_{12} : 4 3 1 2 .».

При помилці з'являється повідомлення «Вставляючи $n = 4$ на кожне з можливих місць справа наліво в перестановці P_3 : 3 1 2, отримуємо: P_9 : 3 1 2 4, P_{10} : 3 1 4 2, P_{11} : 3 4 1 2, P_{12} : 4 3 1 2.».

Крок 20. $n = 4$.

Розширимо перестановку P_4*: 3 2 1 з* кроку *14*, вставляючи елемент n на кожне з можливих місць зліва направо (оскільки, номер перестановки $i = 4$ – це парне число).

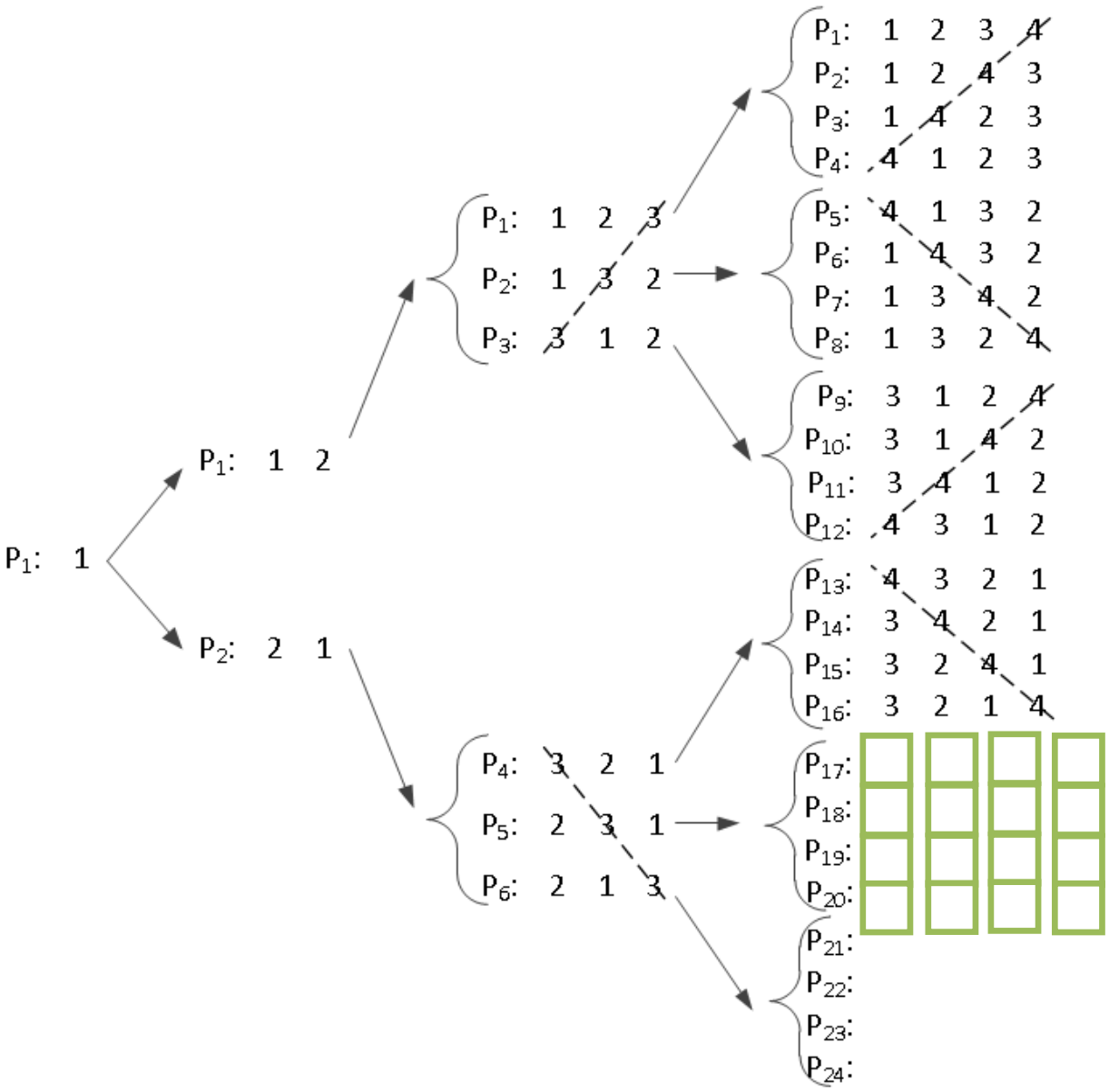


Правильна відповідь – «P_{13} : 4 3 2 1, P_{14} : 3 4 2 1, P_{15} : 3 2 4 1, P_{16} : 3 2 1 4.».

При помилці з’являється повідомлення «Вставляючи $n=4$ на кожне з можливих місць зліва направо в перестановці $P_4: 3\ 2\ 1$, отримуємо: $P_{13}: 4\ 3\ 2\ 1$, $P_{14}: 3\ 4\ 2\ 1$, $P_{15}: 3\ 2\ 4\ 1$, $P_{16}: 3\ 2\ 1\ 4$.».

Крок 21. $n=4$.

Розширимо перестановку $P_5: 2\ 3\ 1$ з кроку *14*, вставляючи елемент n на кожне з можливих місць справа наліво (оскільки, номер перестановки $i=3$ – це непарне число).

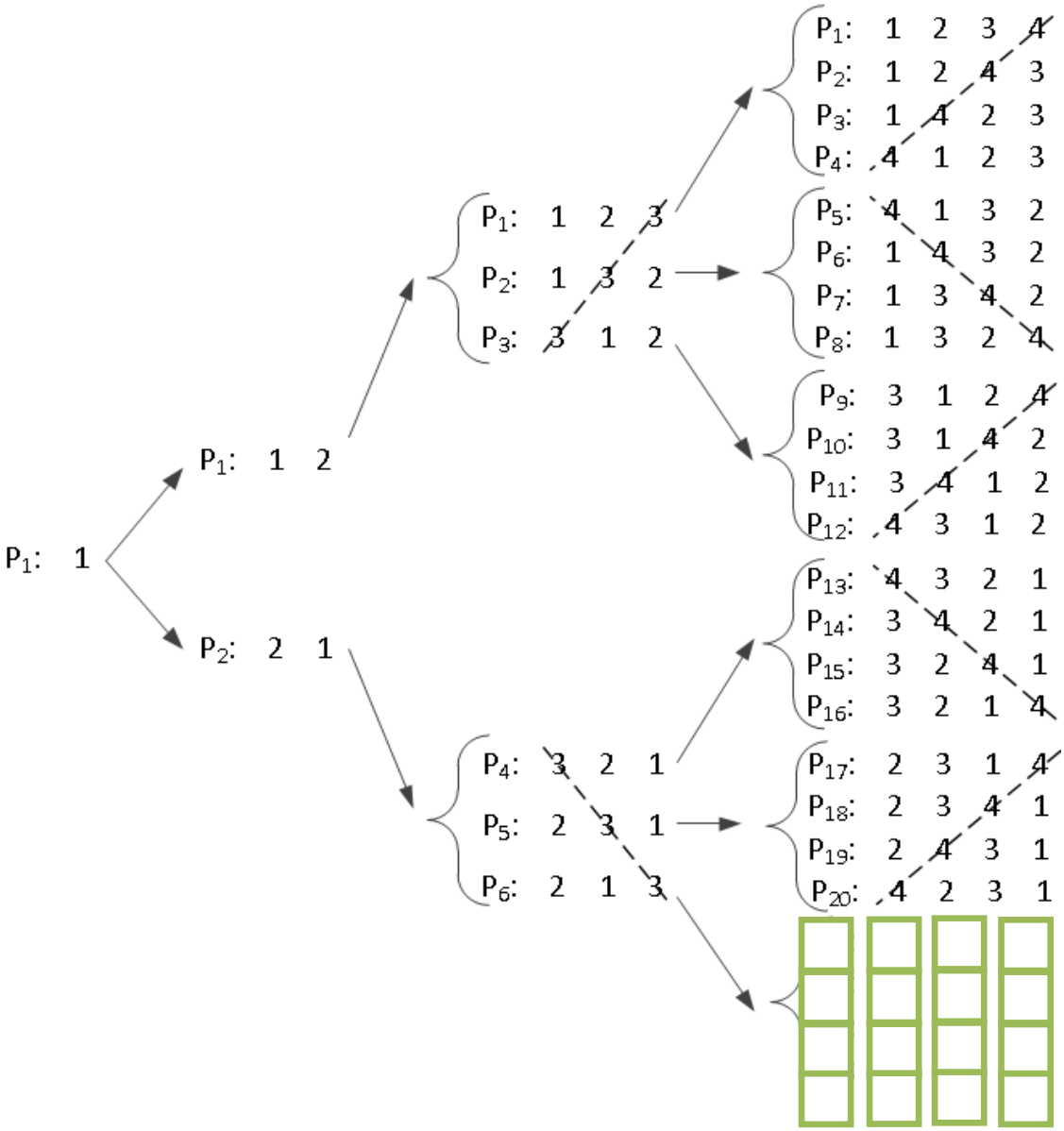


Правильна відповідь – «P_{17} : 2 3 1 4, P_{18} : 2 3 4 1, P_{19} : 2 4 3 1, P_{20} : 4 2 3 1.».

При помилці з'являється повідомлення «Вставляючи $n = 4$ на кожне з можливих місць справа наліво в перестановці P_5 : 2 3 1, отримуємо: P_{17} : 2 3 1 4, P_{18} : 2 3 4 1, P_{19} : 2 4 3 1, P_{20} : 4 2 3 1.».

Крок 22. $n = 4$.

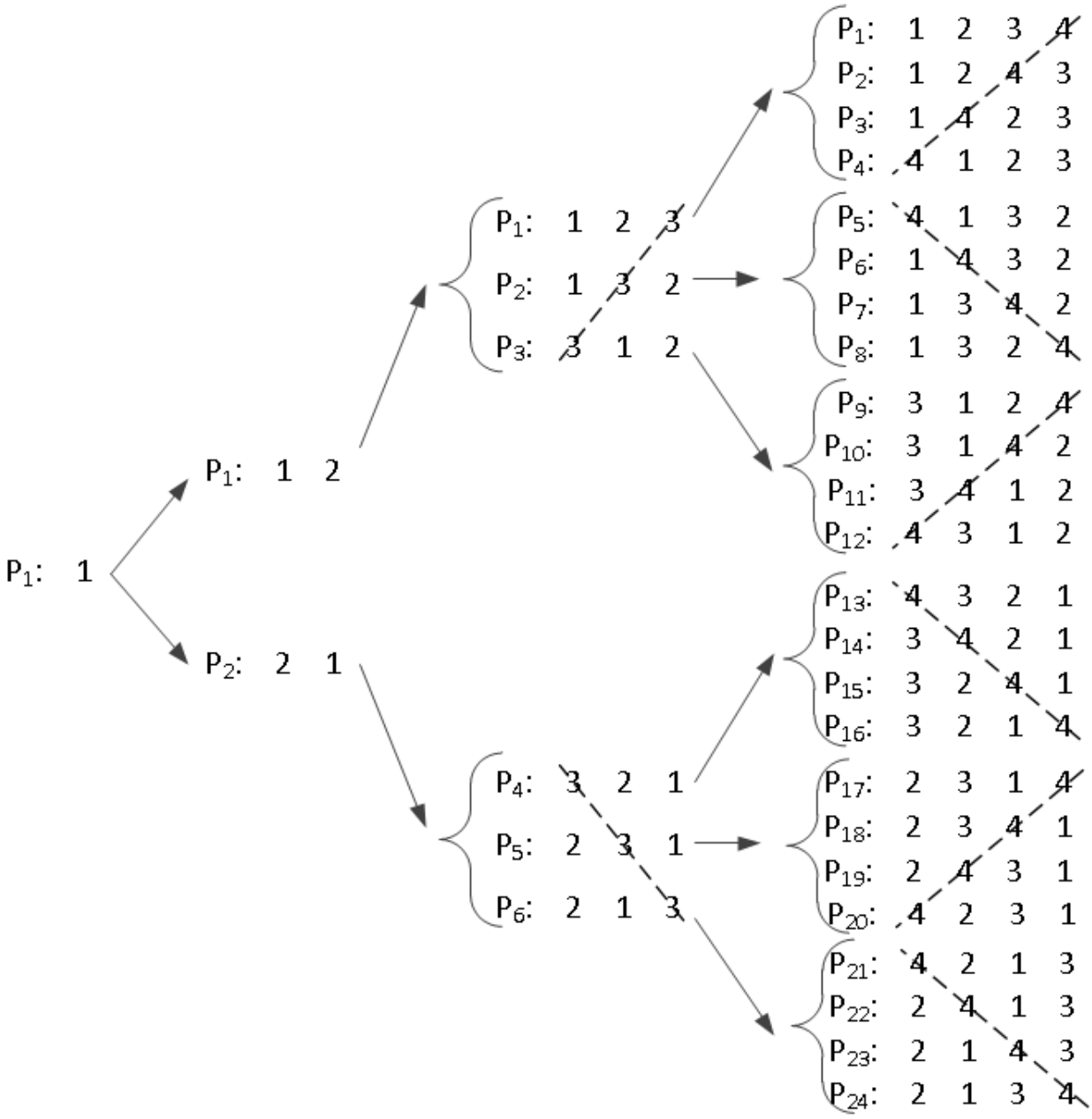
Розширимо перестановку P_6*: 2 1 3 з* кроку *14*, вставляючи елемент n на кожне з можливих місць зліва направо (оскільки, номер перестановки $i = 6$ – це парне число).



Правильна відповідь – «$P_{21}: 4\ 2\ 1\ 3$, $P_{22}: 2\ 4\ 1\ 3$, $P_{23}: 2\ 1\ 4\ 3$, $P_{24}: 2\ 1\ 3\ 4$.».

При помилці з'являється повідомлення «Вставляючи $n=4$ на кожне з можливих місць зліва направо в перестановці $P_6: 2\ 1\ 3$, отримуємо: $P_{21}: 4\ 2\ 1\ 3$, $P_{22}: 2\ 4\ 1\ 3$, $P_{23}: 2\ 1\ 4\ 3$, $P_{24}: 2\ 1\ 3\ 4$.».

Крок 23. Отримали:



З’являється повідомлення «Тренінг завершено!».

3.2. Блок-схема алгоритму

На рис. 3.1, Б.1-Б.3 зображена блок-схема алгоритму для першого кроку. Всі інші кроки аналогічні.

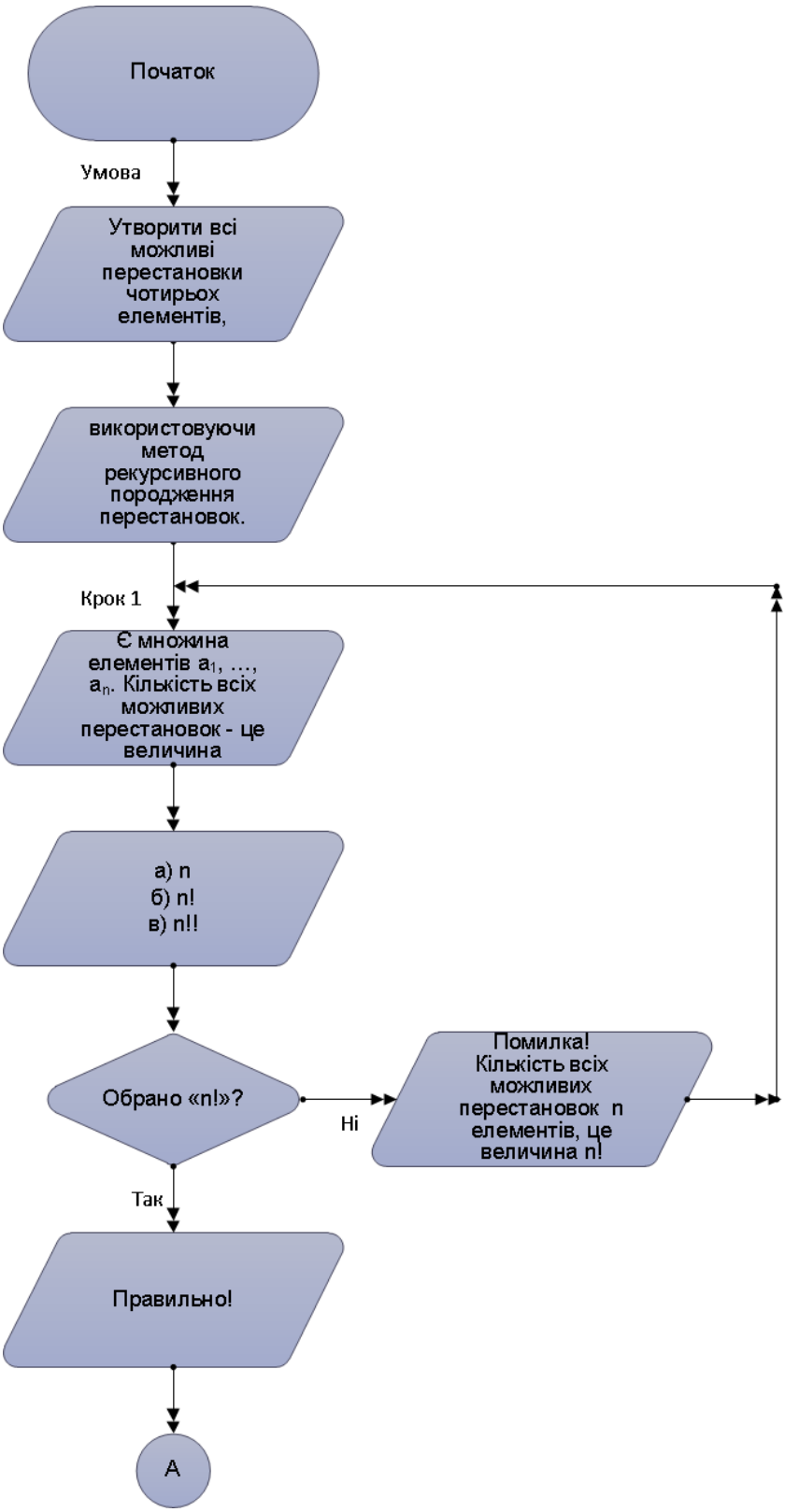


Рисунок 3.1 – Блок-схема алгоритму

4. ПРАКТИЧНА ЧАСТИНА

4.1. Опис програмної реалізації

Розроблений алгоритм було програмно реалізовано з використанням мов об’єктно-орієнтованого програмування С++ у середовищі візуального програмування Borland Builder.

Розглянемо, як створювався тренажер, на прикладі двох кроків, а саме: на прикладі десятого та тринадцятого кроків. Ці кроки є складними та об’ємними.

Спочатку були намальовані рисунки з використанням пакету MS Visio та збережені як точкові файли у форматі .bmp.

Для розміщення малюнків на формах було використано компонент Image з вкладки Additional палітри компонентів.

Використовуючи властивість Picture, малюнки були завантажені для відображення у компонентові Image (рис. 4.1).

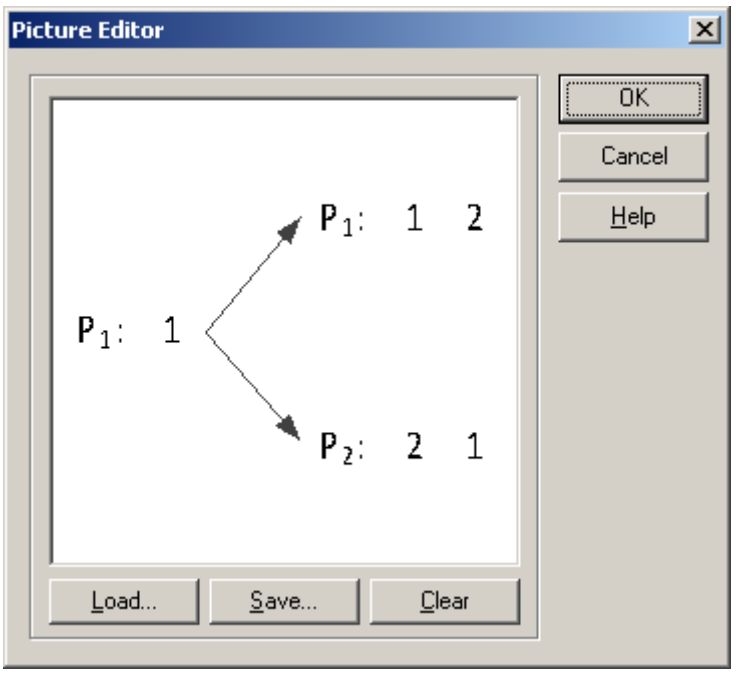


Рисунок 4.1 – Завантаження малюнку (крок 10)

Далі було виставлено оригінальний розмір картинок (властивість Autosize поставлена у стан true) та встановлено прозорість для картинок (властивість Transparent поставлена у стан true).

Для кнопки «Перевірити» кроку 10 був створений код:

```
// крок 10
void __fastcall TForm11::BitBtn1Click(TObject *Sender)
{
  if ((Edit1->Text=="") || (Edit2->Text=="") ||
      (Edit3->Text=="") || (Edit4->Text==""))
  {
     MessageDlg ("Введіть всі відповіді!", mtWarning,
TMsgDlgButtons() << mbOK, 0);
     Edit1->SetFocus();
```

```
      return;
   }

   if ((Edit1->Text=="1") && (Edit2->Text=="2") &&
       (Edit3->Text=="2") && (Edit4->Text=="1"))
   {
      MessageDlg ("Правильно!", mtInformation,
TMsgDlgButtons() << mbOK, 0);
      Form12->Show();
      Form11->Hide();
   }
   else
   {
      MessageDlg ("Помилка!\nВставляючи n = 2 на кожне з
можливих місць справа наліво в перестановці P1 :
1,\nотримуємо:\nP1 : 1 2\nP2 : 2 1", mtError,
TMsgDlgButtons() << mbOK, 0);
      Edit1->SetFocus();
   }
}
```

Проаналізуємо його.

Якщо в якесь з чотирьох текстових полів Edit не введено інформації, то з’являється повідомлення про це і переривається підпрограма.

Якщо відповіді вірні, то з’являється повідомлення про це, даний крок закривається, а наступний крок відображається на екрані.

Якщо у відповідях є неточності, то з'являється пояснення помилки, і користувач повинен виправити похибки.

Для кнопки «Перевірити» кроку 13 був створений код:

```
// крок 13
void __fastcall TForm14::BitBtn1Click(TObject *Sender)
{
  if ((Edit1->Text=="") || (Edit2->Text=="") ||
      (Edit3->Text=="") || (Edit4->Text=="") ||
      (Edit5->Text=="") || (Edit6->Text=="") ||
      (Edit7->Text=="") || (Edit8->Text=="") ||
      (Edit9->Text==""))
  {
     MessageDlg("Введіть всі відповіді!", mtWarning,
TMsgDlgButtons() << mbOK, 0);
     Edit1->SetFocus();
     return;
  }

  if ((Edit1->Text=="1") && (Edit2->Text=="2") &&
      (Edit3->Text=="3") && (Edit4->Text=="1") &&
      (Edit5->Text=="3") && (Edit6->Text=="2") &&
      (Edit7->Text=="3") && (Edit8->Text=="1") &&
      (Edit9->Text=="2"))
  {
     MessageDlg("Правильно!", mtInformation,
TMsgDlgButtons() << mbOK, 0);
```

```
        Form15->Show();
        Form15->Edit1->SetFocus();
        Form14->Hide();
    }
    else
    {
        MessageDlg("Помилка!\nВставляючи n = 3 на кожне з можливих місць\nсправа наліво в перестановці P1 : 1 2, отримуємо:\nP1 : 1 2 3\nP2 : 1 3 2\nP3 : 3 1 2", mtError, TMsgDlgButtons() << mbOK, 0);
        Edit1->SetFocus();
    }
}
```

Проаналізуємо код.

Якщо хоча б в одне з дев’яти текстових полів Edit не введено інформації, то з’являється повідомлення про це і переривається функція.

Якщо відповіді вірні, то з’являється повідомлення про це, даний крок закривається, а наступний крок відображається на екрані.

Якщо у відповідях є помилки, то з’являється пояснення, в чому помилка, і користувач повинен її/їх виправити.

Програмний код усіх типових кроків представлено у додатку В.

4.2. Інструкція по роботі з програмою

Робота програми показана на рисунках 4.2-4.24, А.1-А.45.

На кожному кроці користувач повинен ввести відповідь. Якщо її немає, але натиснута кнопка «Перевірити», то з’явиться попередження як на рис. 4.4.

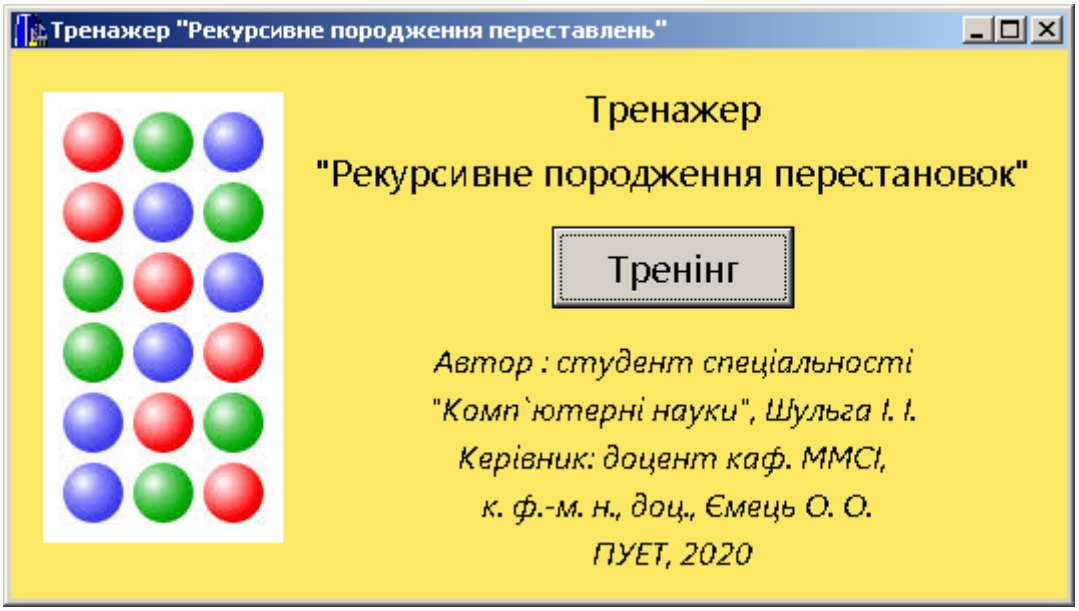


Рисунок 4.2 – Початок

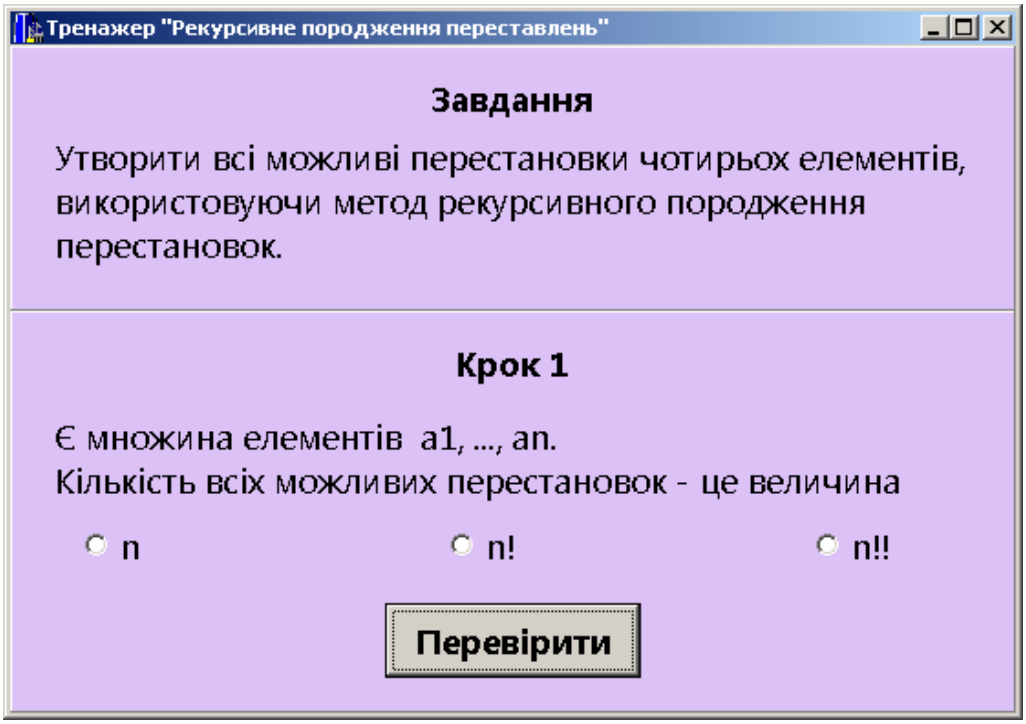


Рисунок 4.3 – Перший крок

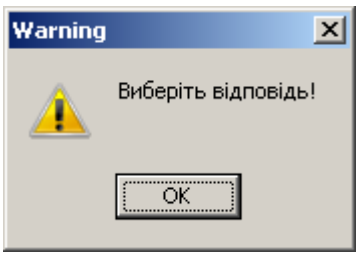


Рисунок 4.4 – Повідомлення при відсутності відповіді

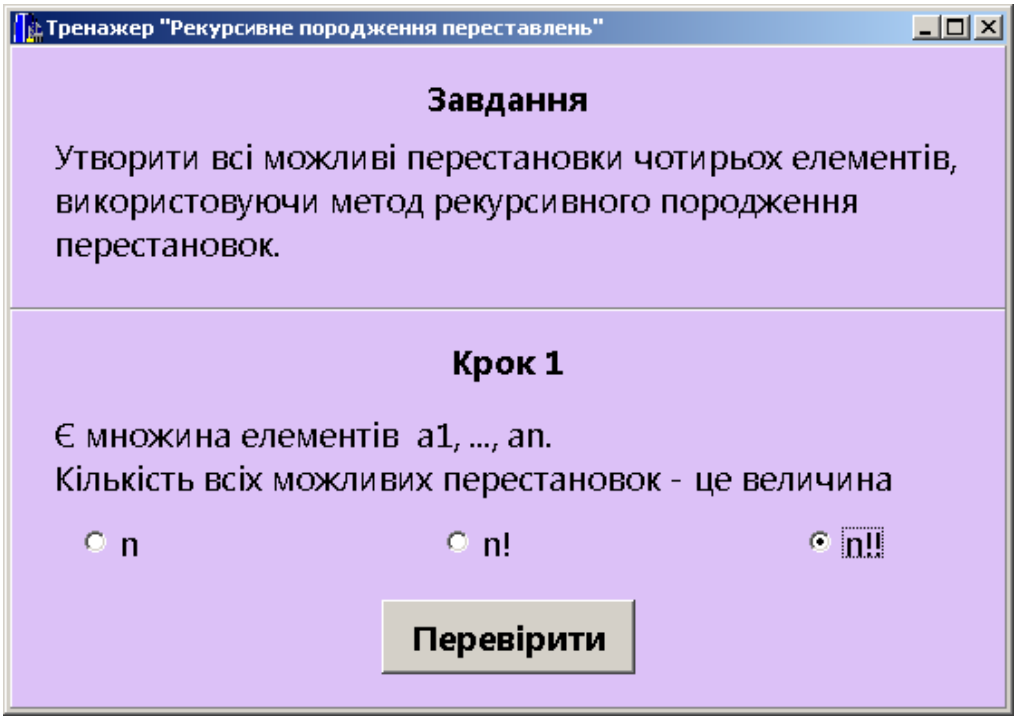


Рисунок 4.5 – Перший крок з помилкою

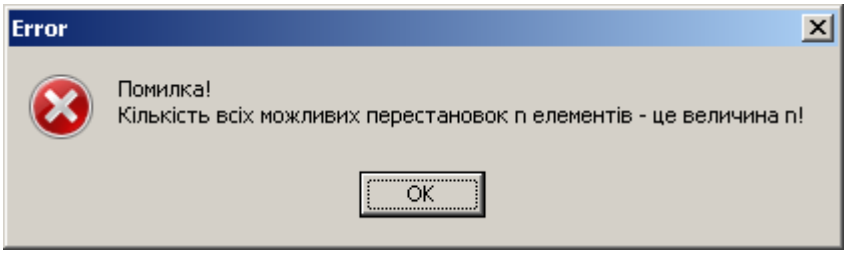


Рисунок 4.6 – Повідомлення при похибці

Якщо надано невірну відповідь (див., наприклад, рис. 4.5), то з’являється пояснення помилки (див. наприклад, рис. 4.6). Якщо надано вірну відповідь (див., наприклад, рис. 4.7), то з’являється підтвердження цього (рис. 4.8) і відбувається перехід до наступного питання тренінгу.

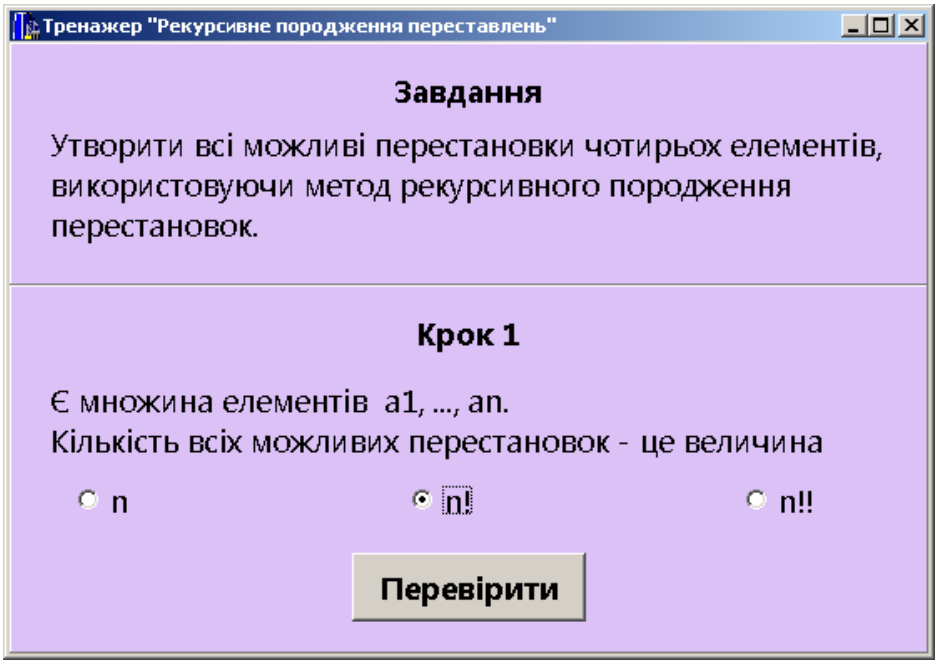


Рисунок 4.7 – Перший крок з вірною відповіддю

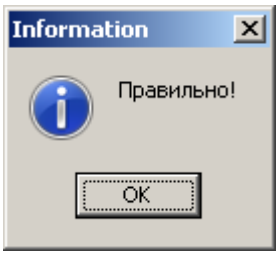


Рисунок 4.8 – Повідомлення при вірній відповіді

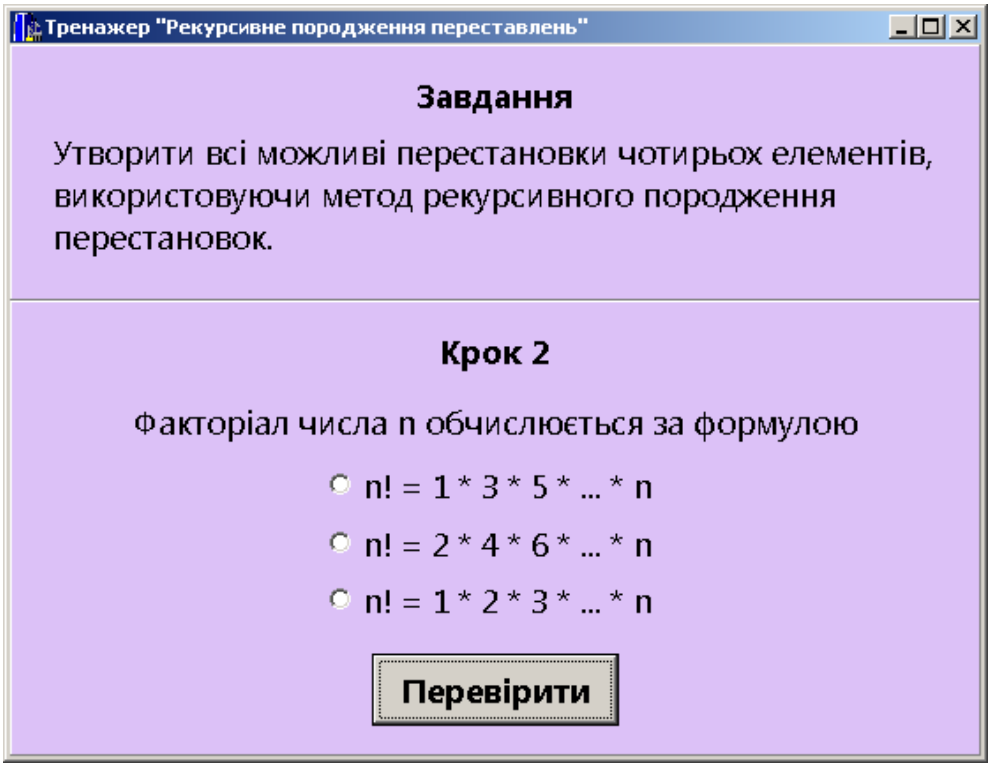


Рисунок 4.9 – Другий крок

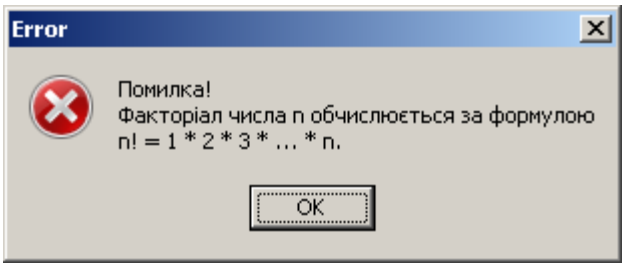


Рисунок 4.10 – Повідомлення при похибці

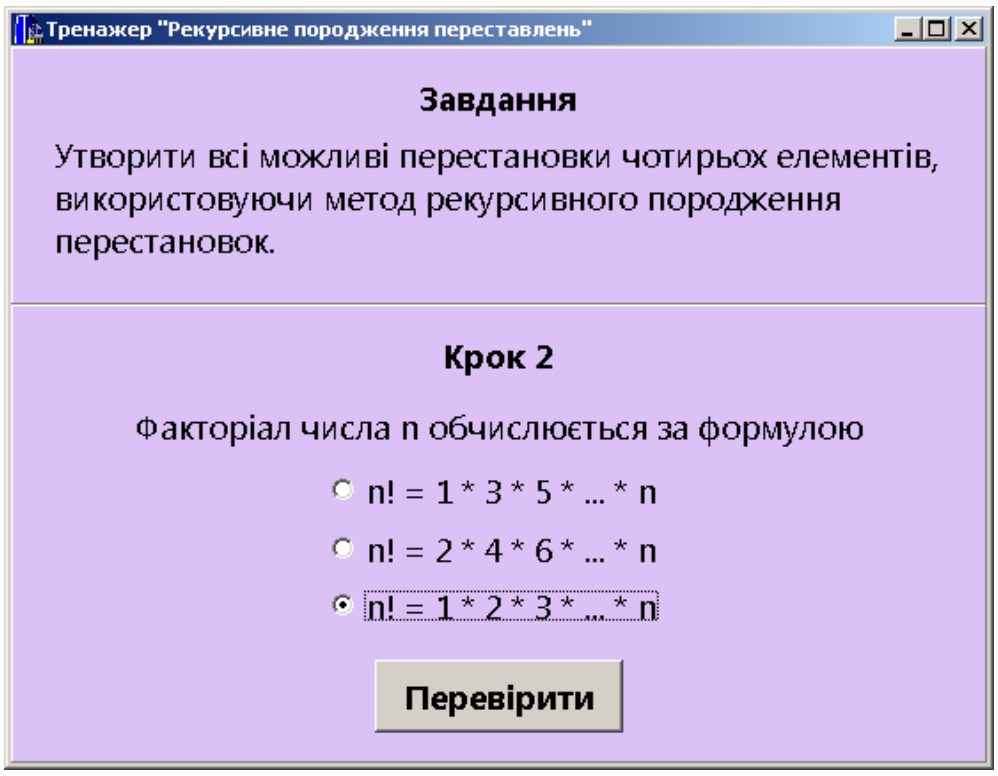


Рисунок 4.11 – Другий крок з вірною відповіддю

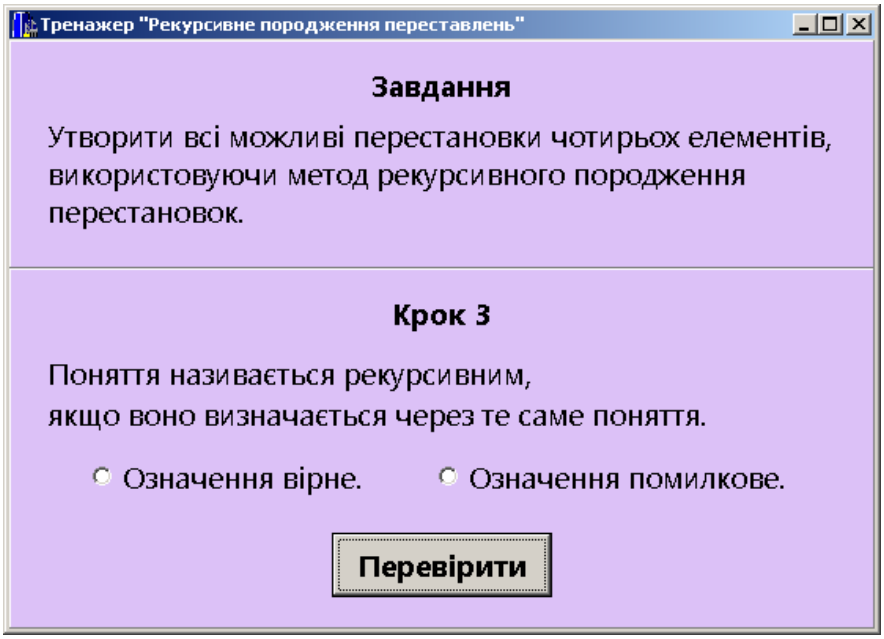


Рисунок 4.12 – Третій крок

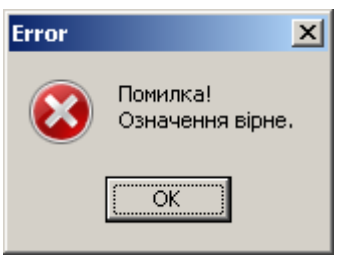


Рисунок 4.13 – Повідомлення при похибці

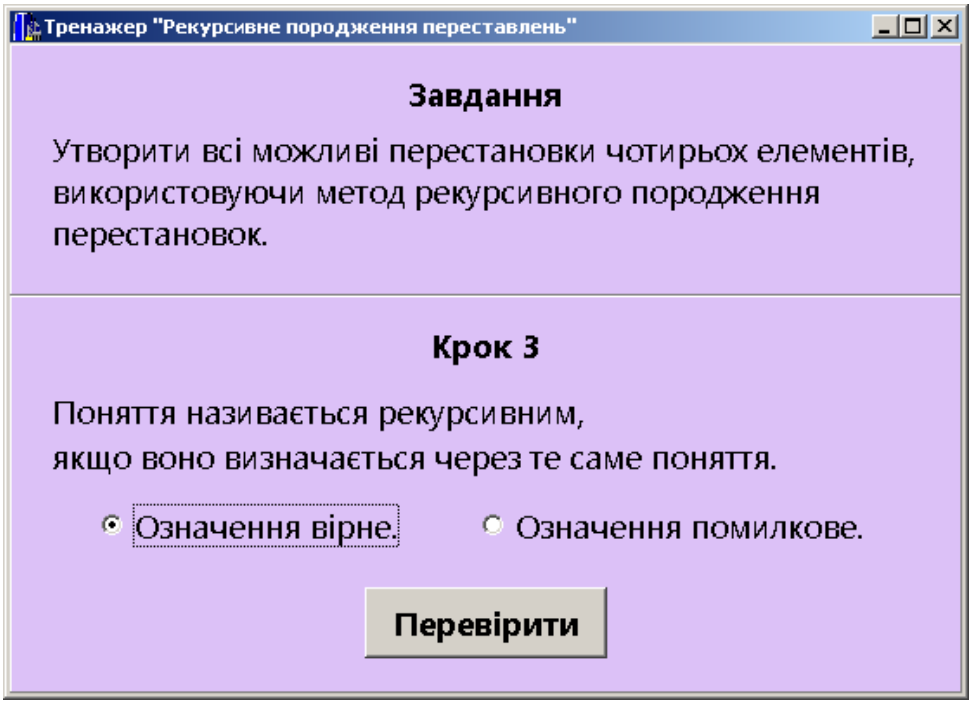


Рисунок 4.14 – Третій крок з вірною відповіддю

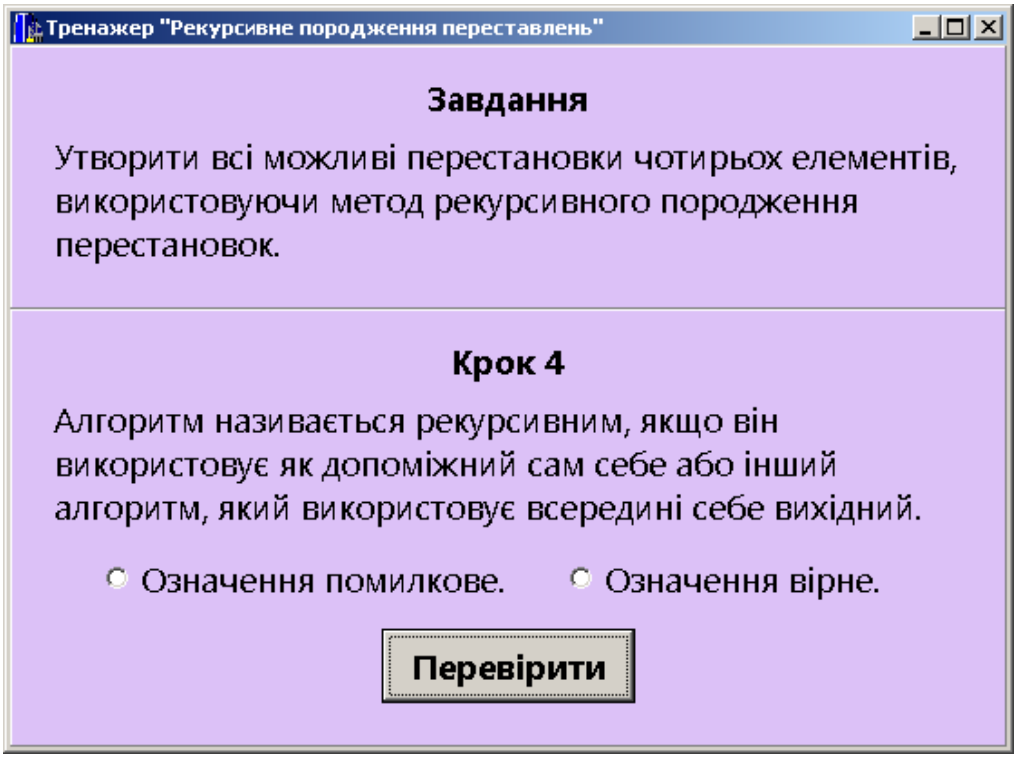


Рисунок 4.15 – Четвертий крок

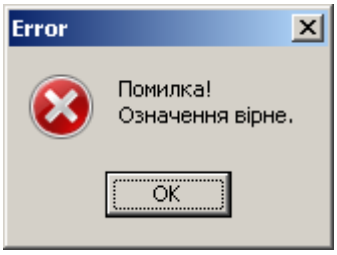


Рисунок 4.16 – Повідомлення при похибці

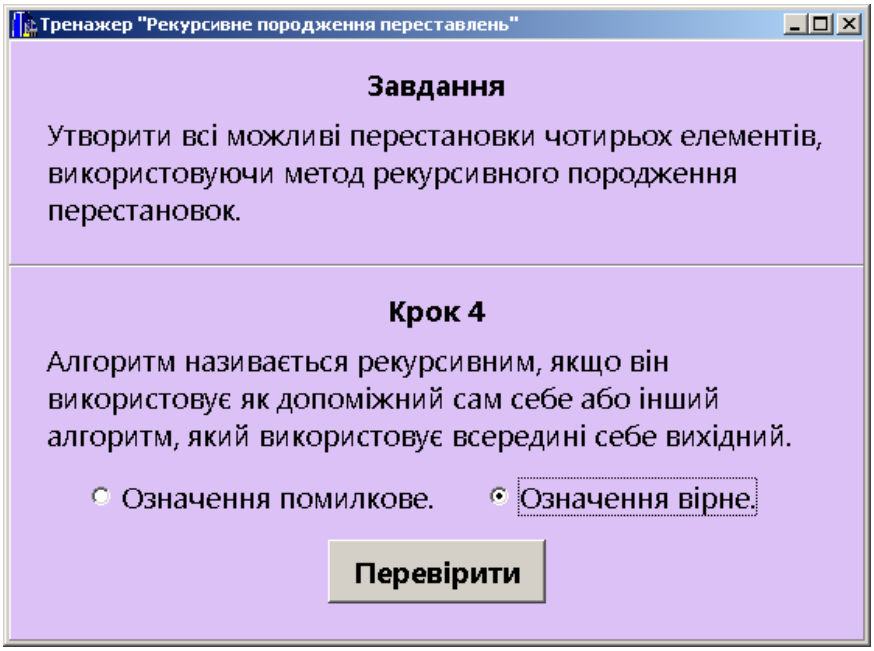


Рисунок 4.17 – Четвертий крок з вірною відповіддю

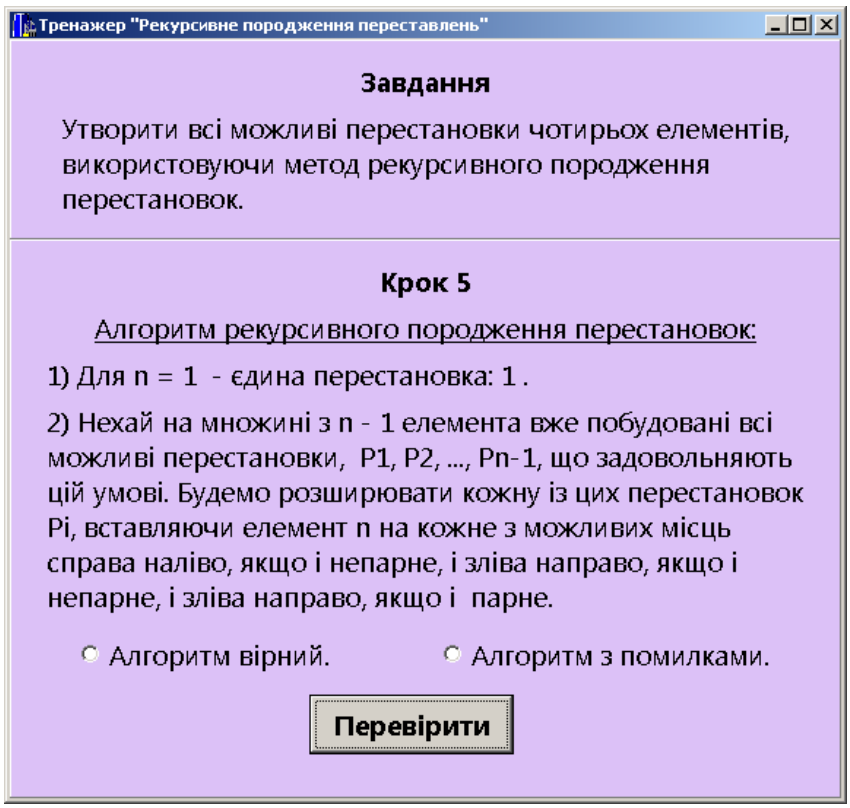


Рисунок 4.18 – П’ятий крок

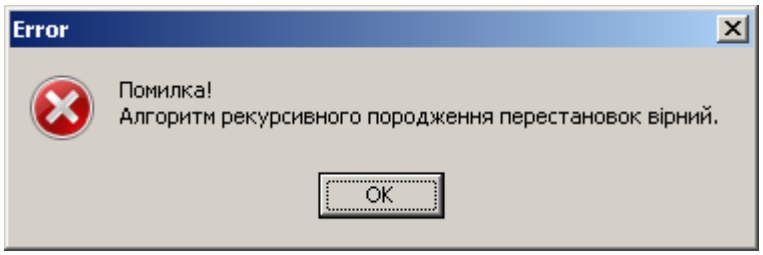


Рисунок 4.19 – Повідомлення при похибці

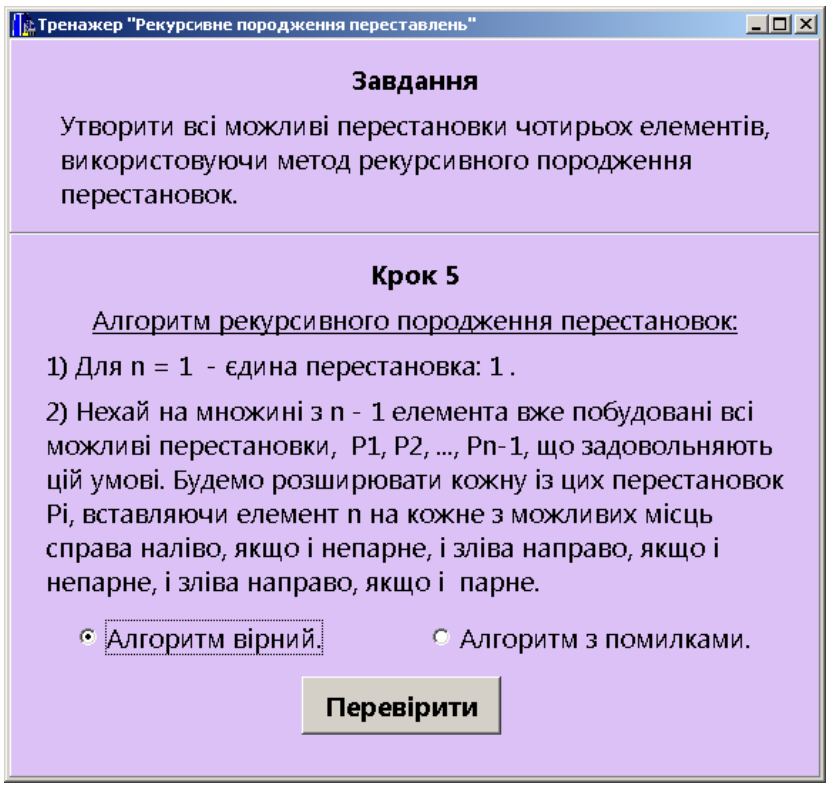


Рисунок 4.20 – П’ятий крок з вірною відповіддю

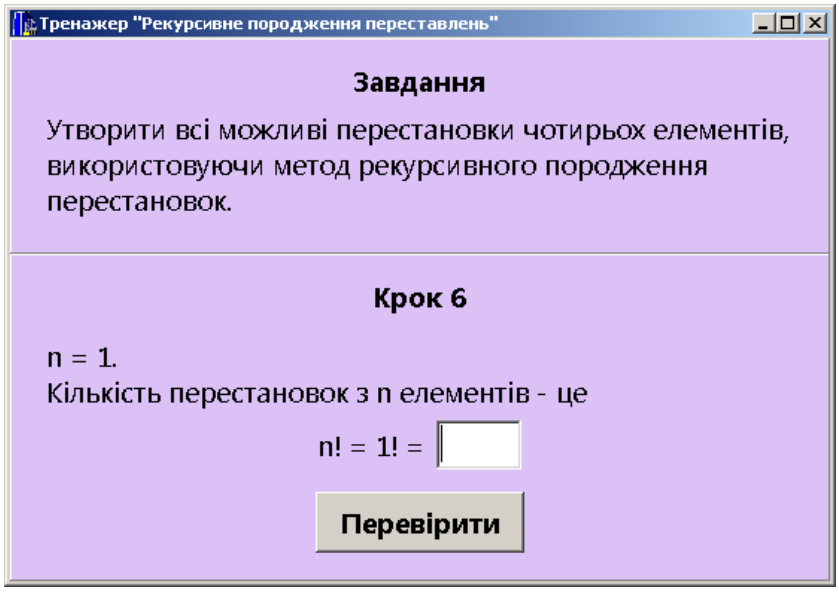


Рисунок 4.21 – Шостий крок

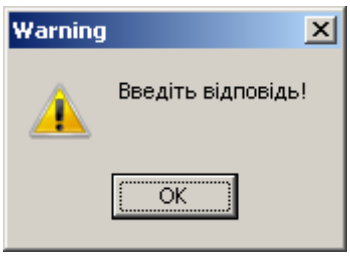


Рисунок 4.22 – Повідомлення при відсутності відповіді

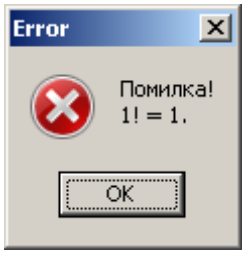


Рисунок 4.23 – Повідомлення при похибці

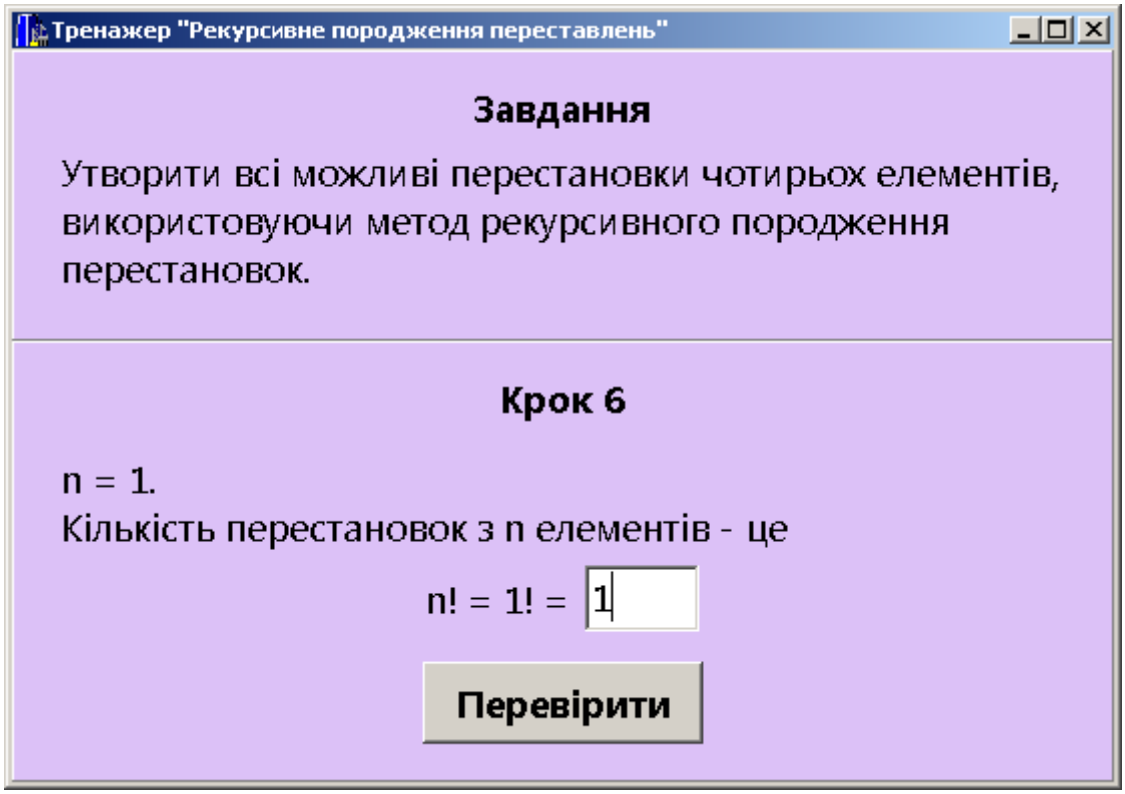


Рисунок 4.24 – Шостий крок з вірною відповіддю

ВИСНОВКИ

При виконанні випускової роботи були виконані наступні задачі:

- ✓ було переглянуто розробки схожої тематики, а саме: вже наявні тренажери з дисципліни «Алгоритми і структури даних»;
- ✓ було виконано аналіз цих розробок;
- ✓ було вивчено теоретичні відомості, як генерувати множину переставлень за методом рекурсивного породження;
- ✓ було розроблено алгоритм тренажеру з теми «Рекурсивне породження перестановок», який складається з 23 кроків (за матеріалами дистанційного курсу «Алгоритми і структури даних» Полтавського університету економіки та торгівлі;
- ✓ була створена блок-схема алгоритму;
- ✓ створений алгоритм був запрограмований мовою C++ у середовищі Borland Builder;
- ✓ програма протестована, усі помилки та похибки усунуті;
- ✓ було здійснено апробацію результатів під час науково-практичного семінару «Комп'ютерні науки та прикладна математика»-2020, який відбувався на базі кафедри математичного моделювання та соціальної інформатики Полтавського університету економіки та торгівлі;
- ✓ були опубліковані тези цього семінару [7];
- ✓ тренажер передано на впровадження у дистанційний курс «Алгоритми і структури даних» Полтавського університету економіки та торгівлі, про що свідчить акт впровадження.

ЛІТЕРАТУРА

1. Ємець О. О. Дистанційний курс ПУЕТ «Алгоритми та структури даних» для студентів спеціальностей «Комп'ютерні науки та інформаційні технології/ О О. Ємець. – [Електронний ресурс].

2. Ємець О. О. Дистанційний курс ПУЕТ «Елементи комбінаторної оптимізації» для студентів спеціальностей «Комп'ютерні науки та інформаційні технології» / О О. Ємець. – [Електронний ресурс].

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

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. Чуб О. І. Тренажер «Рекурсивні алгоритми» / О. І. Чуб, О.О. Ємець // Комп'ютерні науки і прикладна математика (КНіПМ-2019): матеріали наук.-практ. семінару. Випуск 4. / За ред. Ємця О. О. – Полтава: Кафедра ММСІ ПУЕТ, 2019. – С. 16-19. – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/7456.

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

7. Шульга І. І. Тренажер «Рекурсивне породження переставлень» / І. І. Шульга, 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, 2020. – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/8270.