Мал. 4.52. **Сцена** з перешкодою

4.6. ЦИКЛИ З ПЕРЕДУМОВОЮ

1. Що таке цикли?
2. Який вигляд має команда циклу з лічильником?
3. Що таке тіло циклу?

ЦИКЛ З ПЕРЕДУМОВОЮ

Розглянемо таку задачу.

Задача. Є діжка, відро і колодязь з водою. Використовуючи відро, наповнити діжку водою.

Ми вже розглядали аналогічну задачу на заповнення діжки водою. Але там було відомо, що діжка та відро порожні, а також ми знали їх місткість. Тому в тій задачі можна було одразу визначити, що для наповнення діжки команди тіла циклу потрібно виконати 5 разів.

Оскільки в цій задачі невідомо ні місткість діжки, ні місткість відра, чи є вода в діжці, чи діжка порожня, то аналогічний висновок тут зробити не можна.

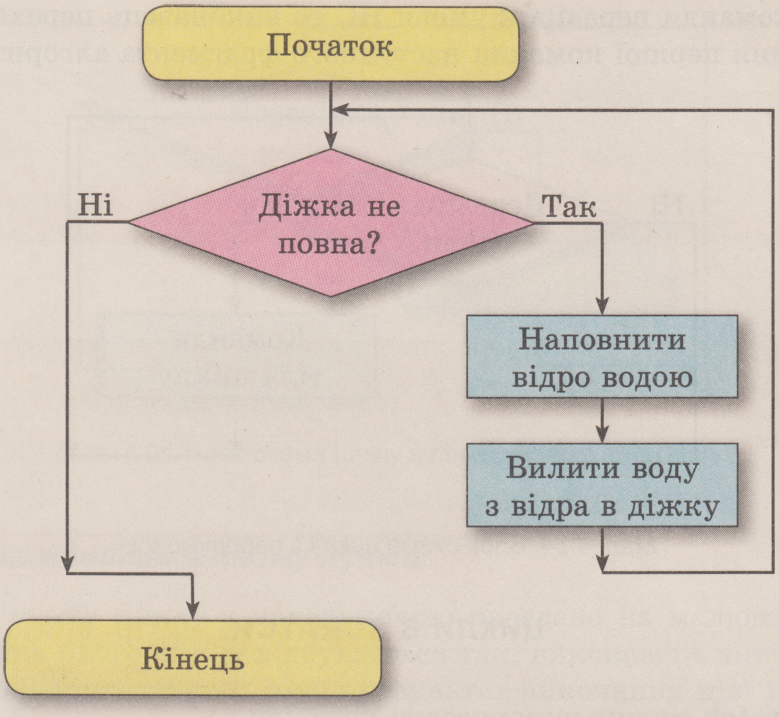
Розглянемо виконавця із системою команд:

1. Наповнити відро водою з колодязя.
2. Вилити воду з відра в діжку.
3. Перевірити умову «Діжка не повна?».

Алгоритм розв'язування цієї задачі для розглянутого виконавця матиме такий вигляд:

1. Перевірити умову «Діжка не повна?».
2. Якщо результат виконання попередньої команди **Так**, виконати команду 3, **інакше** (тобто якщо результат виконання попередньої команди **Ні**), виконати команду 6.
3. Наповнити відро водою з колодязя.
4. Вилити воду з відра в діжку.
5. Виконати команду 1.
6. Закінчити виконання алгоритму.

На малюнку 4.53 показано блок-схему цього алгоритму.



Мал. 4.53. Блок-схема алгоритму розв'язування задачі

У цьому алгоритмі команди 3–5 можуть бути виконані більше ніж один раз і тому утворюють тіло циклу. Чергове виконання чи невиконання цих команд залежить від результату виконання команди перевірки умови «Діжка не повна?» у команді 1. Якщо цей результат **Так**, то команди 3–5 виконуються ще раз, якщо ж **Ні**, то ці команди більше не виконуються.

Звертаємо вашу увагу!

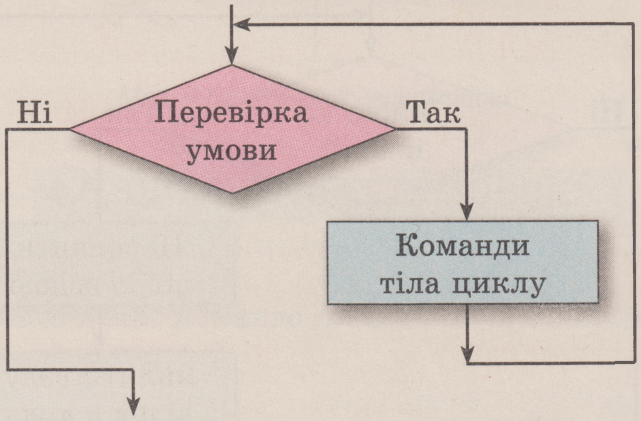
Команди тіла циклу саме «*можуть бути виконані більше ніж один раз*», а не «*обов'язково виконуються більше ніж один раз*». Адже розміри відра і діжки можуть бути такі, що під час першого ж виливання води з відра в діжку вона наповниться і виконання алгоритму закінчиться.

Крім того, діжка може одразу бути повною. У такому разі команди тіла циклу *не виконуватимуться жодного разу*.

Розглянутий вище цикл називають **циклом з передумовою.**

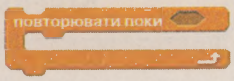
Загальний вигляд циклу з передумовою наведено на малюнку 4.54. Виконання цього циклу відбувається так: виконавець виконує команду перевірки умови; якщо результат виконання цієї команди **Так**, то виконавець виконує **команди тіла циклу**, після чого знову виконує **команду перевірки умови**; якщо ж результат вико-

нання команди перевірки умови **Ні**, то виконавець переходить до виконання першої команди наступного фрагмента алгоритму.



Мал. 4.54. Блок-схема циклу з передумовою

ЦИКЛИ В SCRATCH

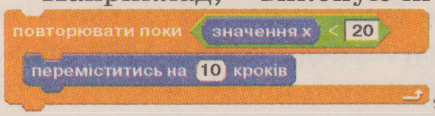
У **Scratch** можна використати команду 

з групи **Керувати** для організації циклу з передумовою.

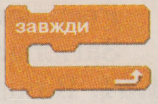
Але, на відміну від традиційного циклу з передумовою, команди тіла циклу з передумовою в **Scratch** виконуватимуться, якщо результат виконання команди перевірки умови буде **Ні**. Якщо ж результат виконання команди перевірки умови буде **Так**, то команди тіла циклу виконуватися не будуть.

У загальному вигляді блок-схема команди циклу з передумовою в **Scratch** наведена на малюнку 4.55.

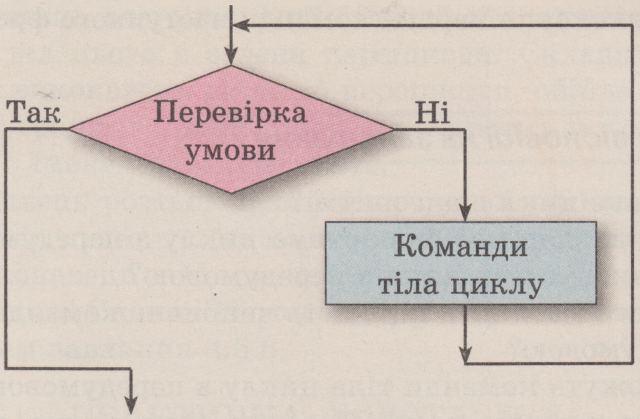
Наприклад, виконуючи команду циклу з передумовою



, виконавець переміщуватиметься на 10 кроків, якщо він **не доторкається до границі**. Тільки-но виконавець **доторкнеться до границі**, виконання команди переміщення не відбудеться, і виконавець зупиниться біля границі.

У **Scratch** можна організувати виконання так званого **безумовного циклу**. Для цього можна використати команду 

. Команди тіла такого циклу будуть виконуватися до моменту натиснення користувачем кнопки  **Зупини все** у правому верхньому куті вікна, яке перериває виконання всього алгоритму.



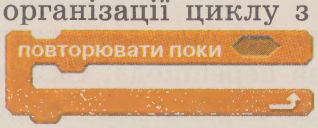
Мал. 4.55. Блок-схема циклу з передумовою в **Scratch**

Найважливіше в цьому пункті

Блок-схему **циклу з передумовою** наведено на малюнку 4.54. Виконання цього циклу відбувається так: виконавець виконує **команду перевірки умови**; якщо результат виконання цієї команди **Так**, то виконавець виконує **команди тіла циклу**, після чого знову виконує **команду перевірки умови**; якщо ж результат виконання команди перевірки умови **Ні**, то виконавець переходить до виконання першої команди наступного фрагмента алгоритму.

Команди тіла циклу з передумовою саме «*можуть бути виконані більше ніж один раз*», а не «*обов'язково виконуються більше ніж один раз*». Адже результат виконання команди перевірки умови перший раз може бути **Так**, а другий раз – **Ні**, і виконання циклу після цього припиняється.

Крім того, команди тіла циклу з передумовою можуть *не виконуватися жодного разу*. Адже результат виконання команди перевірки умови вже й першого разу може бути **Ні**, і виконання циклу одразу припиняється.

У **Scratch** для організації циклу з передумовою можна використати команду 

з групи **Керувати**.

Блок-схему **циклу з передумовою** у **Scratch** наведено на малюнку 4.55. Виконання цього циклу відбувається так: виконавець виконує **команду перевірки умови**; якщо результат виконання цієї команди **Ні**, то виконавець виконує **команди тіла циклу**, після чого знову виконує **команду перевірки умови**; якщо ж результат виконання команди перевірки умови **Так**, то виконавець пере-

ходить до виконання першої команди наступного фрагмента алгоритму.

Дайте відповіді на запитання

1°. Що таке цикл в алгоритмі?
2°. Який вигляд має блок-схема циклу з передумовою?
3°. Як виконується цикл з передумовою?
4°. Від чого залежить кількість виконань команд тіла циклу з передумовою?
5°. Чи можуть команди тіла циклу з передумовою не виконуватися жодного разу? Поясніть свою відповідь, проілюструйте пояснення прикладами.
6*. Чи може виконання циклу з передумовою ніколи не закінчитися? Поясніть свою відповідь, проілюструйте пояснення прикладами.
7*. Що спільного і чим відрізняються цикл з лічильником і цикл з передумовою?
8°. Який вигляд має блок-схема циклу з передумовою в **Scratch**?
9°. Як виконується цикл з передумовою в **Scratch**?
10°. Як організувати в **Scratch** безумовний цикл?

Виконайте завдання

1*. Порівняйте виконання слідування, розгалуження, циклу.
2*. Порівняйте виконання циклу з лічильником і циклу з передумовою.
3°. Петрик збирає їстівні гриби в кошик. Складіть блок-схему наповнення кошика їстівними грибами.
4°. У касі кінотеатру залишилася певна кількість білетів на найближчий сеанс. Складіть блок-схему алгоритму продажу цих білетів.
5°. Виконавець розташований біля лівої границі **Сцени**, праворуч від нього є зелена перешкода. Складіть проект, у якому виконавець дійде до перешкоди і зупиниться. Збережіть проект у вашій папці у файлі з іменем **завдання 4.6.5**.
6°. Виконавець розташований біля правої границі **Сцени**, ліворуч від нього є червона перешкода. Складіть проект, у якому виконавець дійде до перешкоди і зупиниться. Збережіть проект у вашій папці у файлі з іменем **завдання 4.6.6**.

7°. Виконавець розташований біля лівої границі **Сцени**, праворуч від нього є зелена перешкода. Складіть проект, у якому виконавець дійде до перешкоди, обійде її і дійде до правої границі **Сцени**. Збережіть проект у вашій папці у файлі з іменем **завдання 4.6.7**.

8°. Виконавець розташований біля правої границі **Сцени**, ліворуч від нього є синя перешкода. Складіть проект, у якому виконавець дійде до перешкоди, обійде її і дійде до лівої границі **Сцени**. Збережіть проект у вашій папці у файлі з іменем **завдання 4.6.8**.

ПРАКТИЧНА РОБОТА № 6

«Складання і виконання алгоритмів із циклами»

Увага! *Під час роботи з комп'ютером дотримуйтеся правил безпеки та санітарно-гігієнічних норм.*

1. Відкрийте середовище **Scratch**.
2. Відкрийте файл **фон 6_1** для **Сцени**.
3. Розмістіть на **Сцені** виконавця **Рибка**.
4. Складіть проект, у якому виконавець переміщується в горизонтальному напрямку до натиснення клавіші **1**.
5. Збережіть проект у вашій папці у файлі з іменем **практична 6_1**.
6. Відкрийте файл **фон 6_2** для **Сцени**.
7. Розмістіть двох виконавців біля протилежних границь **Сцени**.
8. Складіть проект, у якому виконавці рухаються назустріч один одному до натиснення клавіші **Пропуск**.
9. Збережіть проект у вашій папці у файлі з іменем **практична 6_2**.
10. Закрийте вікно середовища **Scratch**.