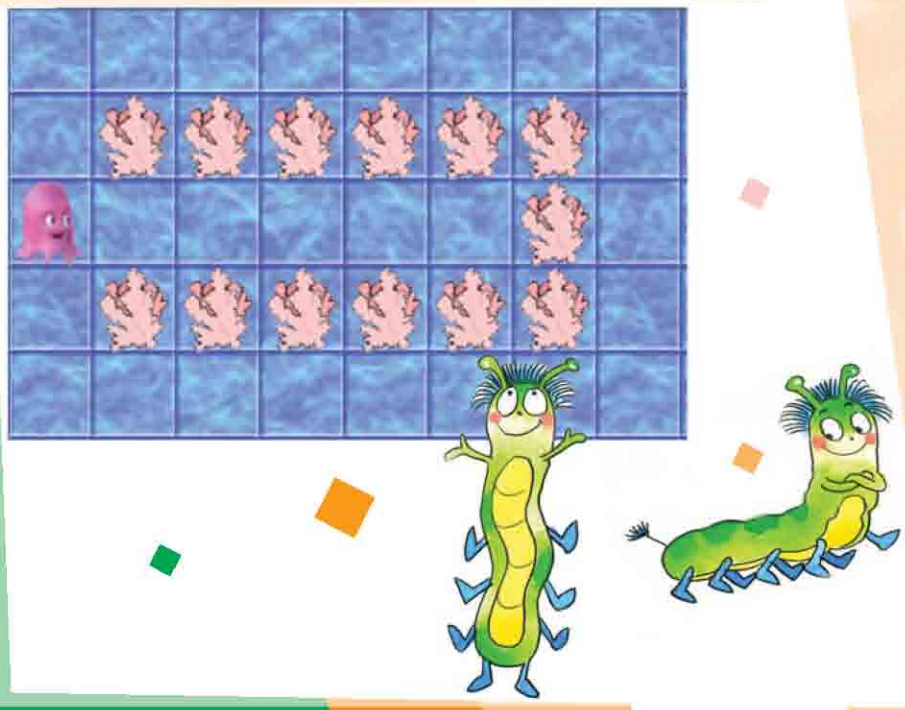
ОСНОВИ АЛГОРИТМІЗАЦІЇ

РОЗДІЛ 3



3.1 Восьминіжка і команда циклу Повтори N разів

1. *Поясніть поняття: команда, виконавець, система команд виконавця, алгоритм.*
2. *Які процеси називаються циклічними? Наведіть приклади циклічних процесів з оточуючого життя.*
3. *Наведіть приклади циклів, з якими ви зустрічалися при вивченні шкільних предметів.*
4. *Наведіть приклад команди циклу для виконавця* **Черепашка**. *Як вона виконувалася?*

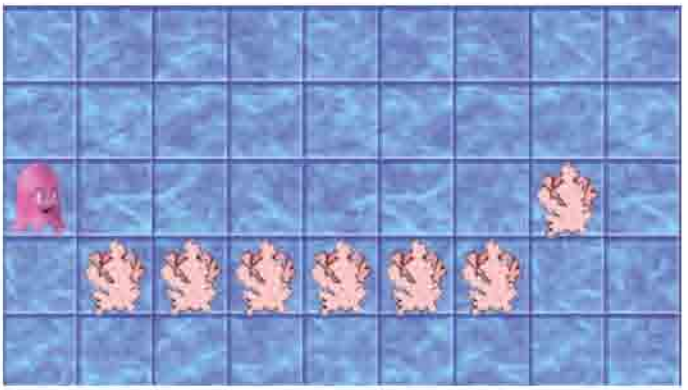


Рис. 74

На цьому уроці ми знову повертаємося до складання алгоритмів для знайомого вам виконавця **Восьминіжка**.

Нагадаємо, що виконавець **Восьминіжка** працює на полі, яке розділене на рівні квадратні клітинки. У деяких клітинках можуть бути перешкоди. Клітинку, у якій на даний момент знаходиться **Восьминіжка**, називають **поточною** (*рис. 74*).

Система команд **Восьминіжки** містить такі команди:

Команда	Результат виконання
Вліво	Переміщується на 1 клітинку вліво
Вправо	Переміщується на 1 клітинку вправо
Вгору	Переміщується на 1 клітинку вгору
Вниз	Переміщується на 1 клітинку вниз
Зафарбуй	Фарбує встановленим кольором поточну клітинку

Восьминіжка також може виконувати команду циклу:

Повтори N разів
<Команди тіла циклу>
Все
(**N** – натуральне число від 1 до 25)

Виконуючи цю команду циклу, **Восьминіжка** виконує вказану послідовність команд зазначене число разів. Наприклад, для того щоб потрапити до зафарбованої клітинки, **Восьминіжка** повинна виконати алгоритм, поданий на *рис. 75*.

Як і **Черепашка**, **Восьминіжка** може виконувати **вкладені команди циклу**. Не важко помітити (*рис. 76*), що шлях **Восьминіжки** до зафарбованої клітинки складається з двох однакових ділянок, кожна з яких складається з чотирьох кроків уліво і трьох кроків униз. Тому при складанні алгоритму, виконавши який **Восьминіжка** потрапить до зафарбованої клітинки, можна використати команду циклу **Повтори 2 рази**, до якого увійдуть дві команди циклу: **Повтори 4 рази Вліво Все** і **Повтори 3 рази Вниз Все**.

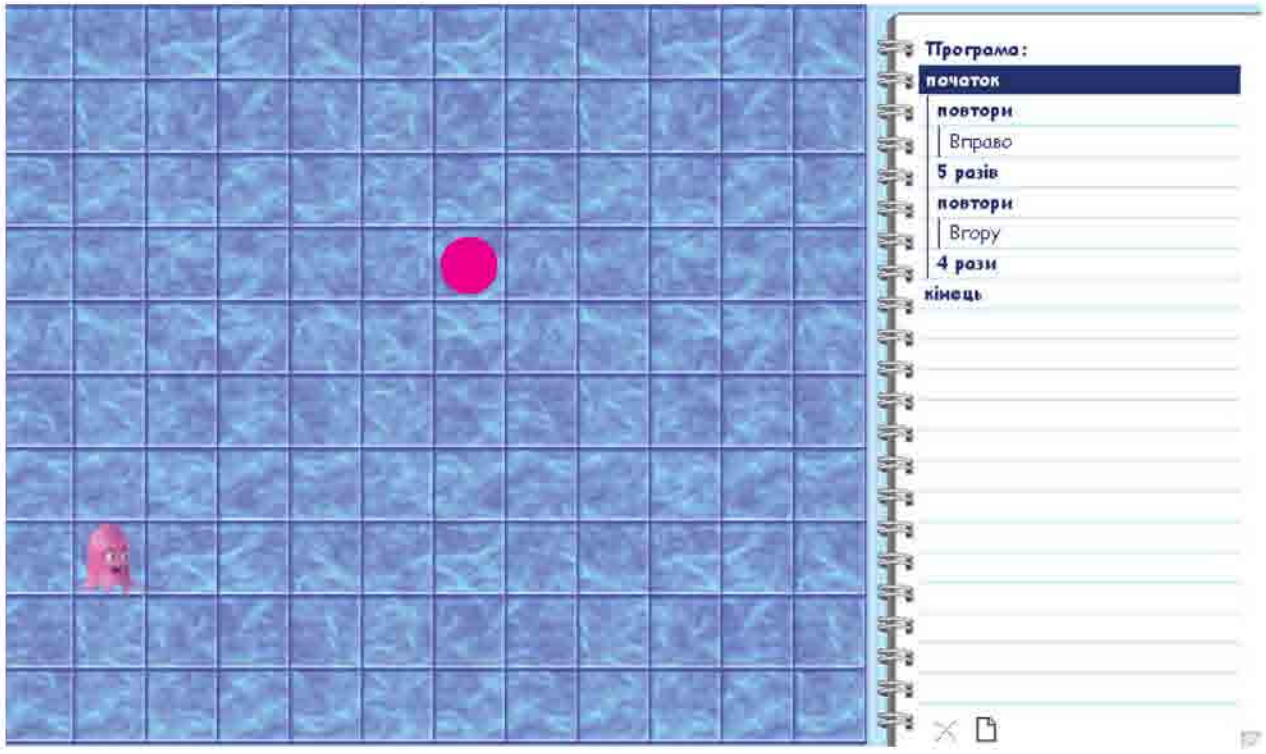

Програма:
початок
повтори
Вправо
5 разів
повтори
Вгору
4 рази
кінець

Рис. 75


Програма:
початок
повтори
повтори
Вліво
4 рази
повтори
Вниз
3 рази
2 рази
кінець

Рис. 76

ЗАПИТАННЯ ТА ЗАВДАННЯ

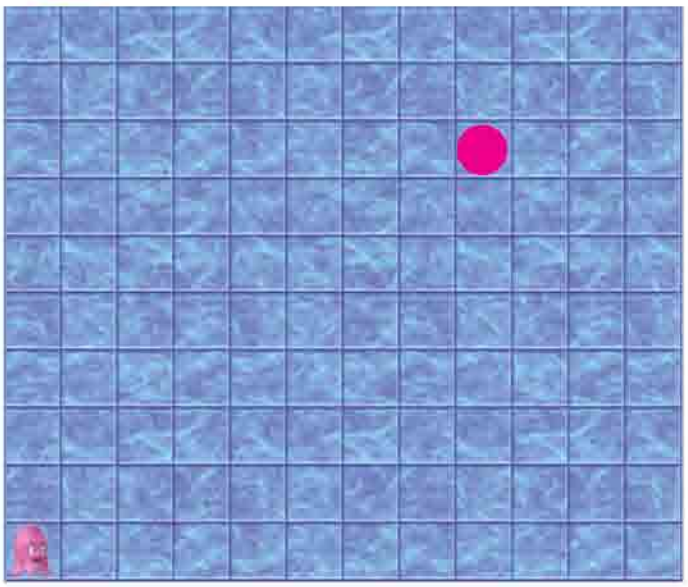


Рис. 77

1. Наведіть приклад команди циклу для виконавця **Восьминіжка**. Поясніть, як вона виконується.
2. Складіть алгоритми для **Восьминіжки**, виконавши які, вона перейде до зафарбованої клітинки (*рис. 77–80*).

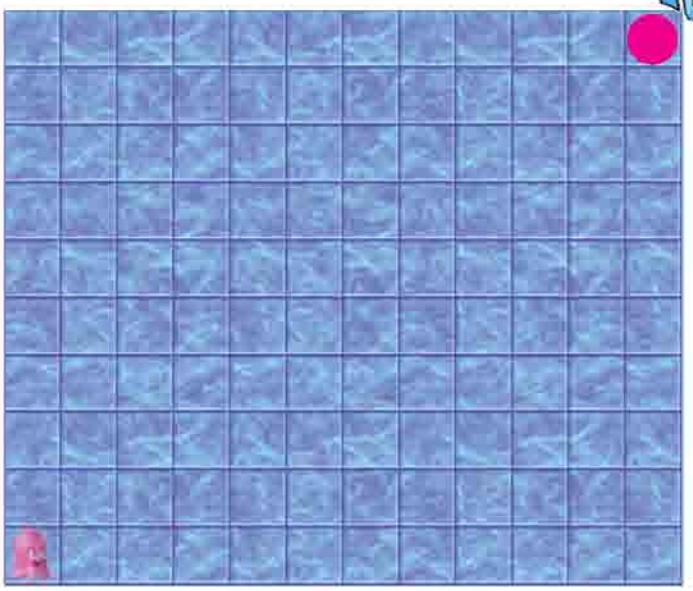


Рис. 78

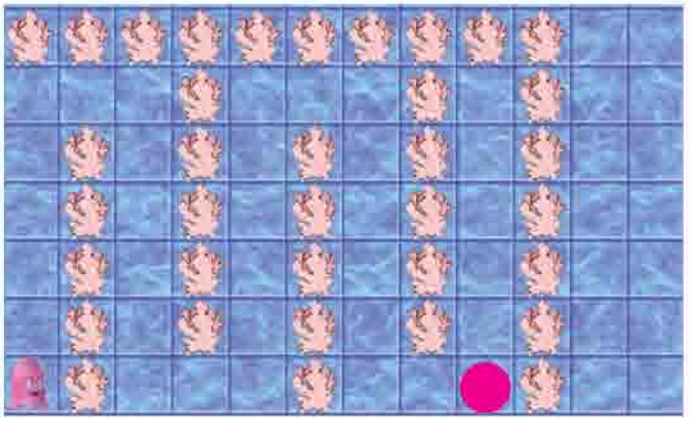


Рис. 79

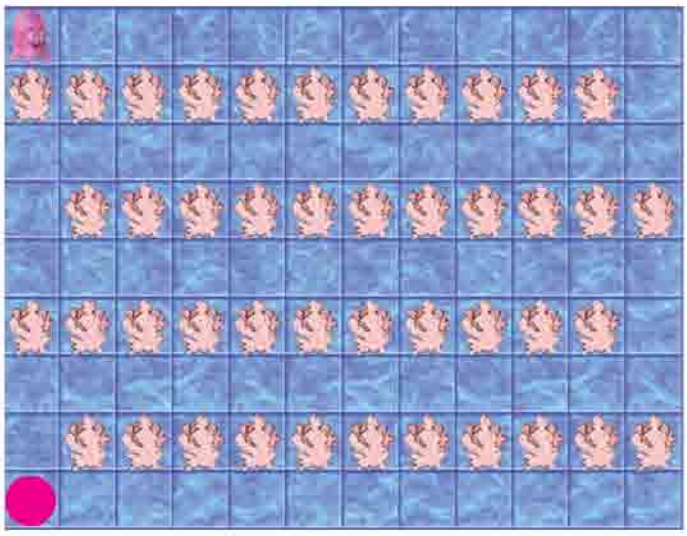


Рис. 80

3. Складіть алгоритми, виконавши які **Восьминіжка** зафарбує вказані клітинки (*рис. 81–86*).

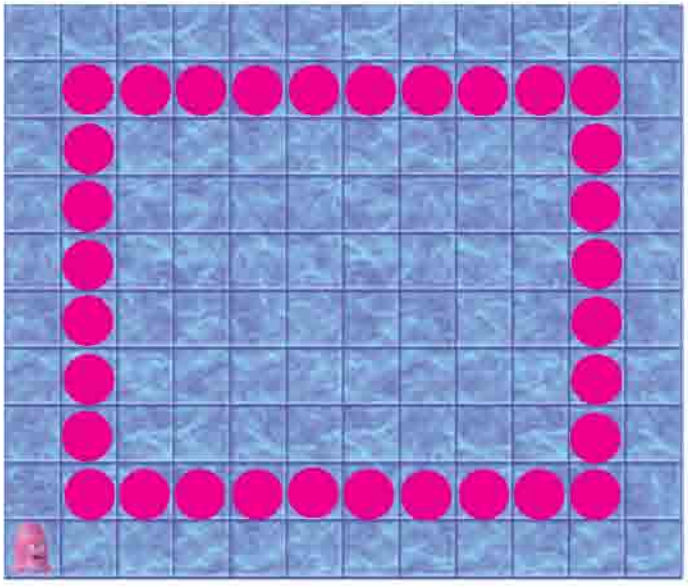


Рис. 81

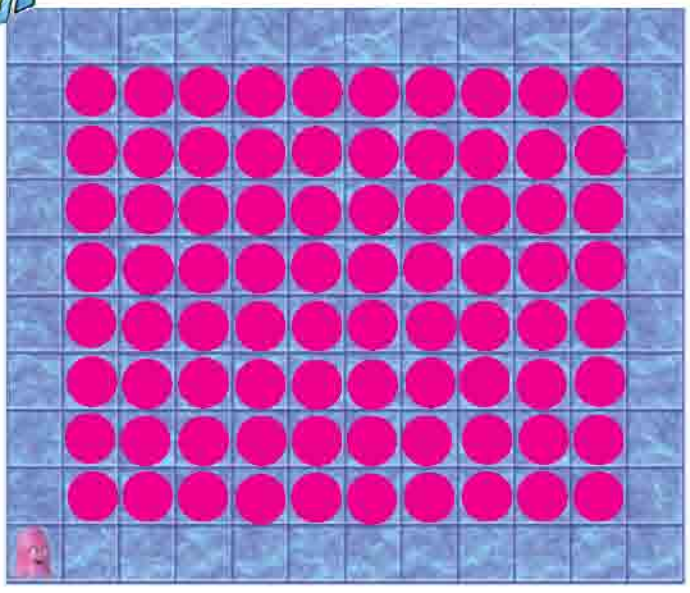
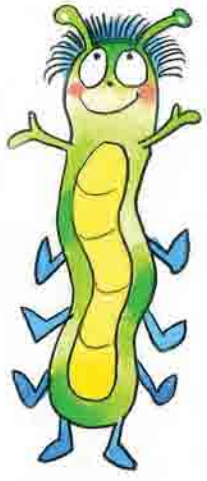


Рис. 82



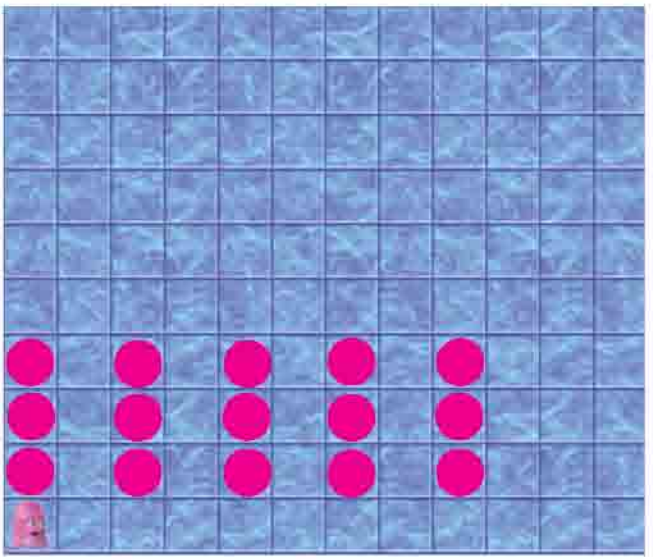
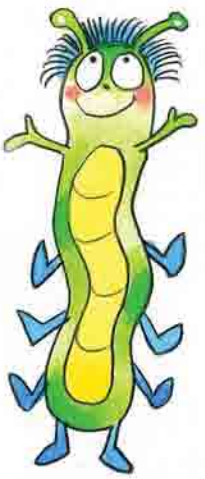


Рис. 83



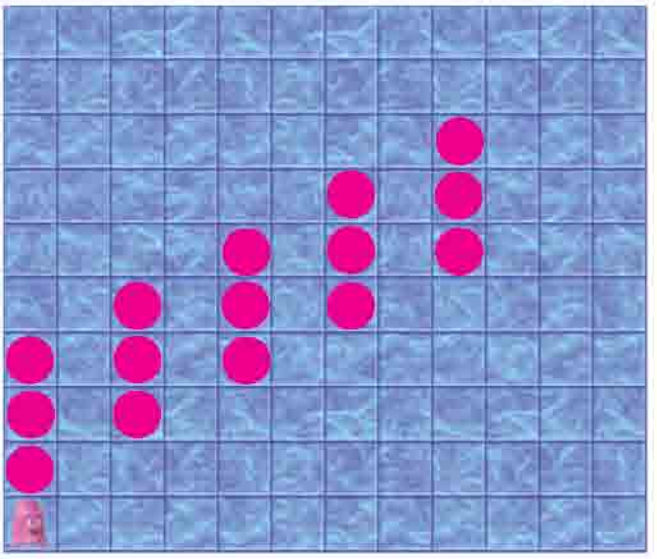


Рис. 84

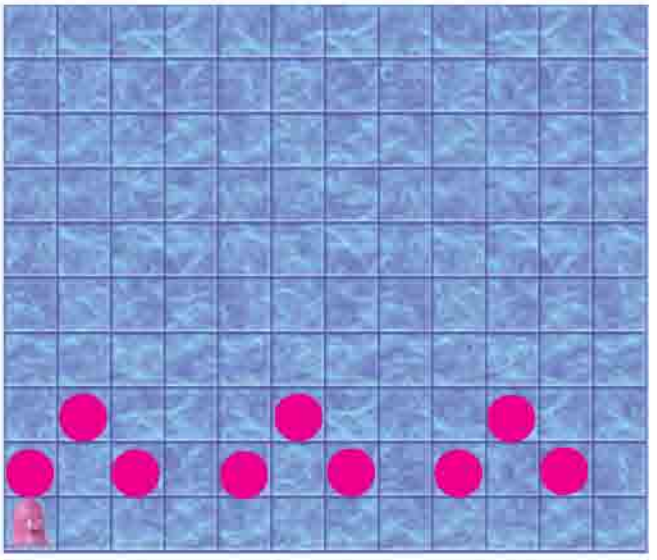


Рис. 85

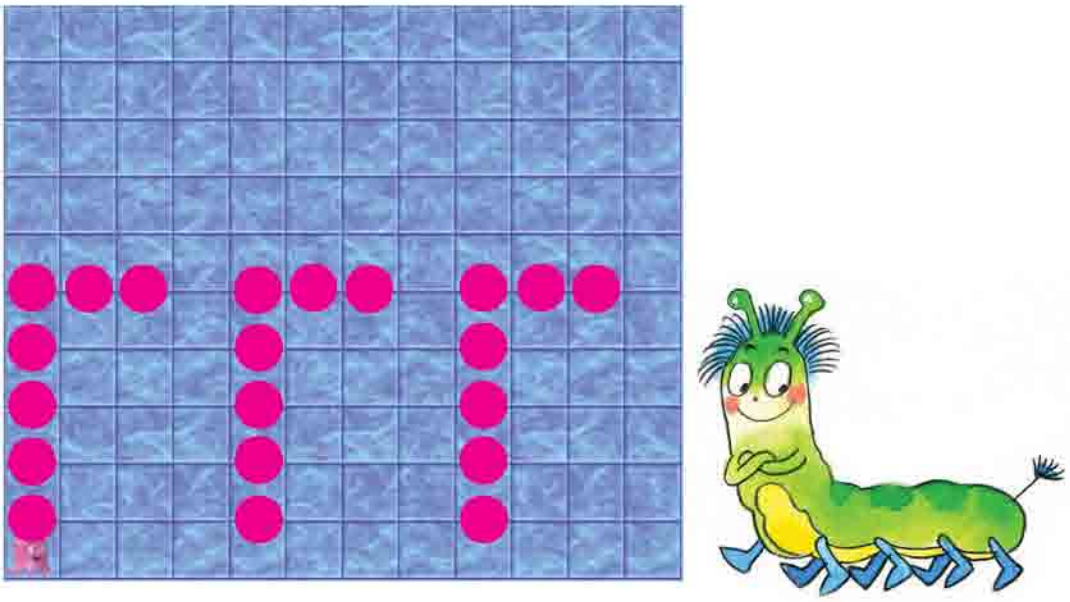


Рис. 86

4. Нарисуйте результат виконання наведених нижче фрагментів алгоритмів (**Восьминіжка** знаходиться в довільній клітинці необмеженого поля без перешкод):

а) **Повтори 5 разів**
Зафарбуй
Вправо
Все

б) **Повтори 3 рази**
Зафарбуй
Вправо
Зафарбуй
Вниз
Все

в) **Повтори 4 рази**
Зафарбуй
Вниз
Вниз
Все
Вправо
Повтори 4 рази
Вверх
Зафарбуй
Вверх
Все

г) **Повтори 5 разів**
Зафарбуй
Вліво
Вниз
Все
Повтори 5 разів
Вправо
Вверх
Все
Вправо
Повтори 5 разів
Зафарбуй
Вправо
Вниз
Все

3.2 Восьминіжка і процедури

1. Що таке процедура?
2. У яких випадках доцільно складати процедури без аргументів, а в яких — з аргументами?
3. З чого складається рядок заголовка процедури?
4. Який вигляд має команда виклику процедури?
5. Як виконуються алгоритми, які містять процедури?

Процедури без аргументів

Як і алгоритми для **Черепашки**, алгоритми для **Восьминіжки** можуть містити процедури. Наприклад, для того щоб **Восьминіжка** зафарбувала вказані клітинки, які утворюють однакові фрагменти (*рис. 87*), можна створити процедуру для фарбування одного фрагмента клітинок і викликати її в основній частині алгоритму.

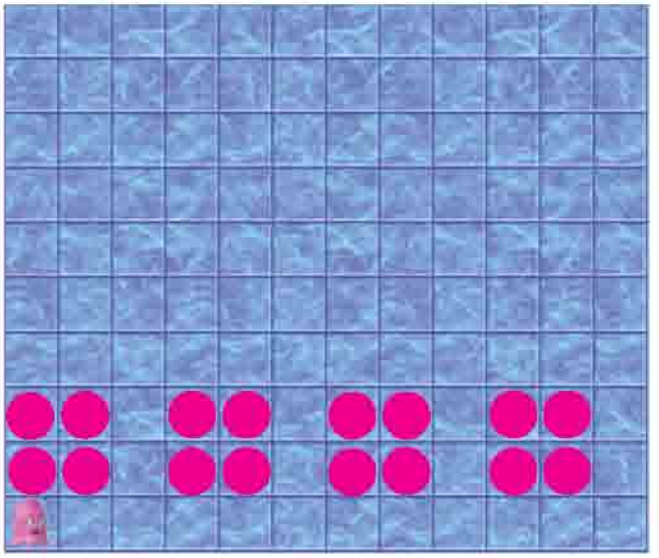


Рис. 87

```
Проц Квадрат
Початок
    Зафарбуй
    Вверх
    Зафарбуй
    Вправо
    Зафарбуй
    Вниз
    Зафарбуй
Кінець

Початок
Вверх
Вверх
Повтори 3 рази
    Квадрат
    Вправо
    Вправо
    Все
    Квадрат
Кінець
```

Поясніть, чому останній раз процедура **Квадрат** викликається поза циклом.

Процедури з аргументами

Нехай **Восьминіжка** повинна зафарбувати клітинки в кількох тупиках різної довжини. Для цього в алгоритмі можна використати дві процедури з аргументами. Виконуючи першу, **Восьминіжка** буде йти в глибину тупика і фарбувати клітинки, а виконуючи другу – переходити на початок наступного тупика (*рис. 88*).

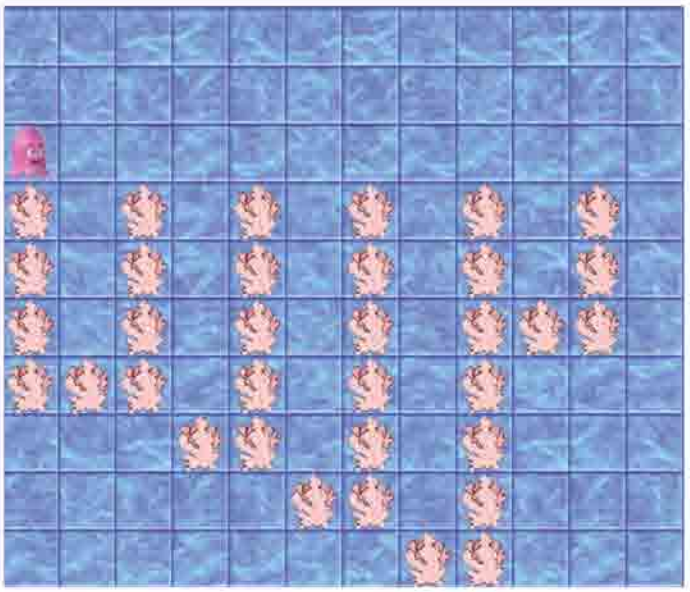


Рис. 88

```
Проц Тупик (х)
Початок
    Повтори х разів
    Вниз
    Зафарбуй
Все
Кінець

Проц Перехід (у)
Початок
Повтори у разів
    Вниз
Все
Повтори 2 рази
    Вправо
Все
Кінець

Початок
    Вправо
    Тупик (3)
    Перехід (3)
    Тупик (4)
    Перехід (4)
    Тупик (5)
    Перехід (5)
    Тупик (6)
    Перехід (6)
    Тупик (2)
Кінець
```

1. *Складіть алгоритми з використанням процедур, виконавши які* **Восьминіжка** *зафарбує вказані клітинки (рис. 89–94).*

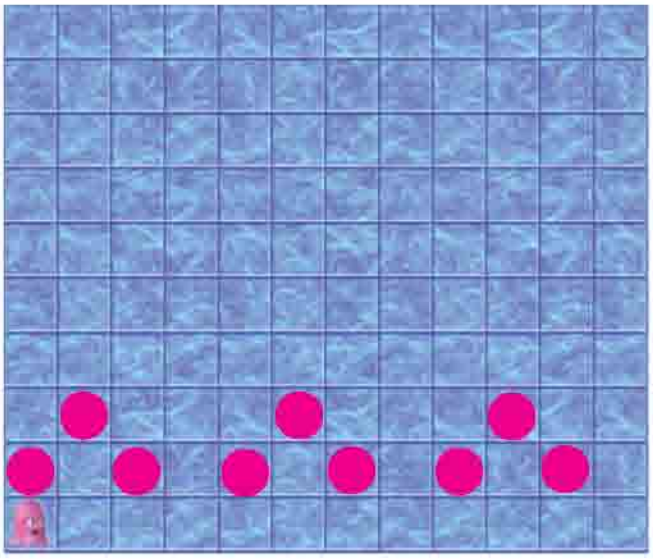


Рис. 89

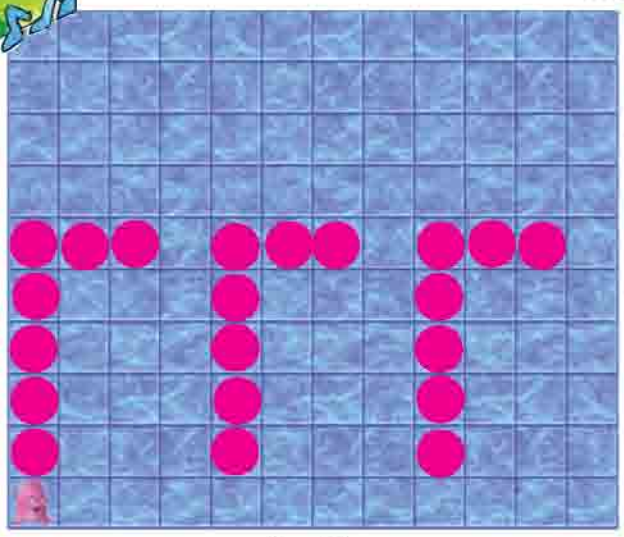


Рис. 90

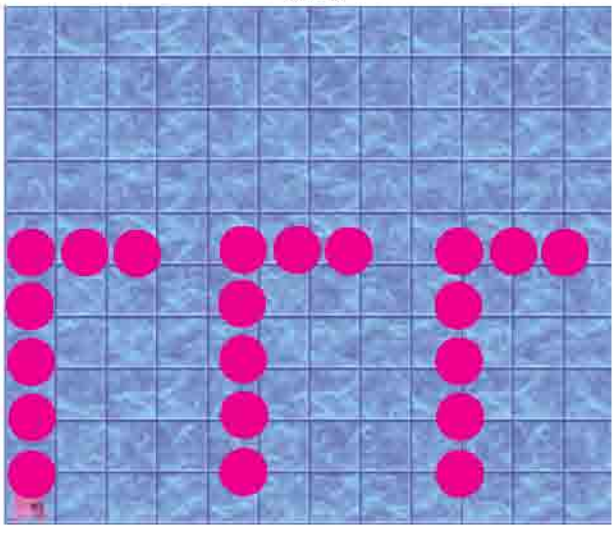
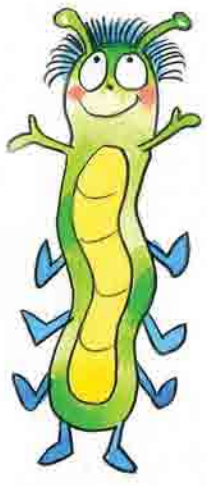


Рис. 91



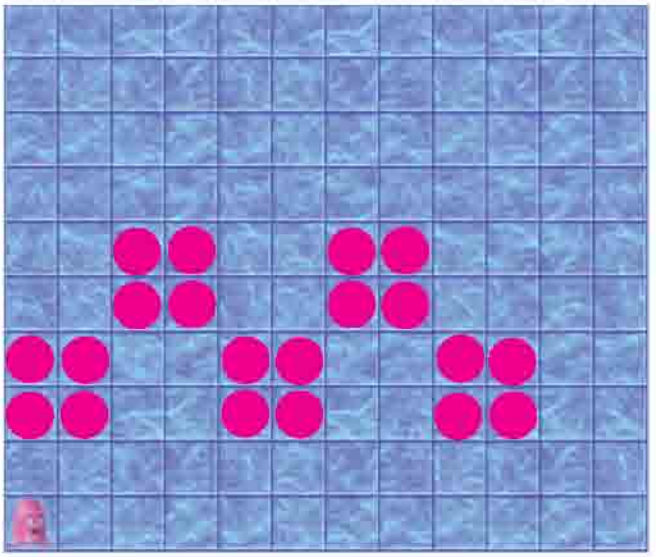


Рис. 92

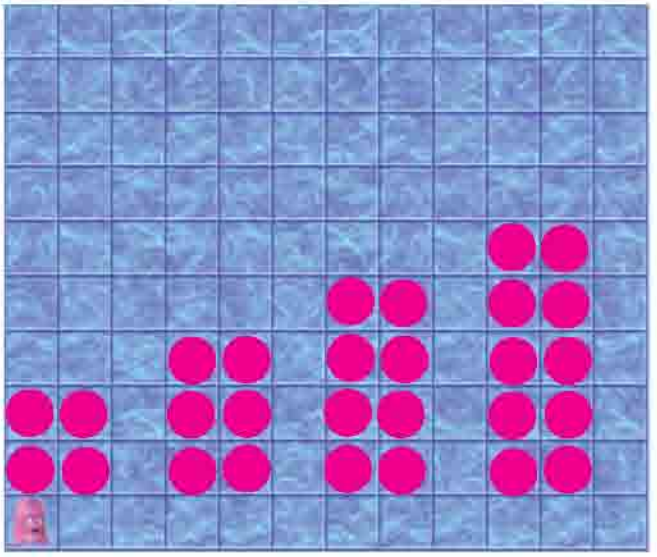


Рис. 93

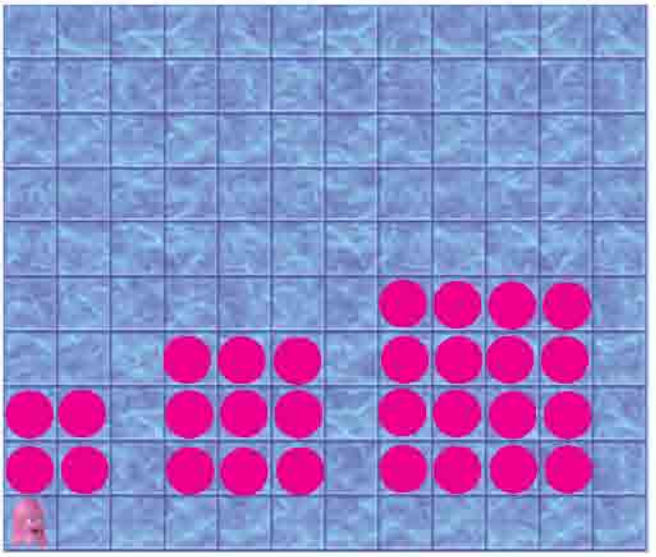


Рис. 94

2. *Складіть алгоритми з використанням процедур, виконавши які* **Восьминіжка** *зафарбує клітинки над кожною горизонтальною стіною з перешкод (рис. 95–96).*
3. *Нарисуйте результат виконання наведених нижче фрагментів алгоритмів з процедурами (***Восьминіжка** *знаходиться в довільній клітинці необмеженого поля без перешкод):*

а) **Проц Лінія**
Початок
Повтори 6 разів
Зафарбуй
Вверх
Все
Повтори 5 разів
Вниз
Все
Вліво
Кінець

Початок
Повтори 5 разів
Лінія
Все
Кінець

б) **Проц Фігура**
Початок
Повтори 3 рази
Зафарбуй
Вверх
Все
Вліво
Вниз
Вниз
Повтори 3 рази
Зафарбуй
Вправо
Все
Вправо
Вправо
Вниз
Кінець

Початок
Фігура
Вверх
Фігура
Фігура
Вниз
Фігура
Кінець

в) **Проц Стовпець (х)**
Початок
Повтори х разів
Зафарбуй
Вниз
Все
Повтори х разів
Вверх
Все
Кінець

Початок
Стовпець (5)
Вправо
Стовпець (2)
Вправо
Стовпець (4)
Вправо
Стовпець (6)
Кінець

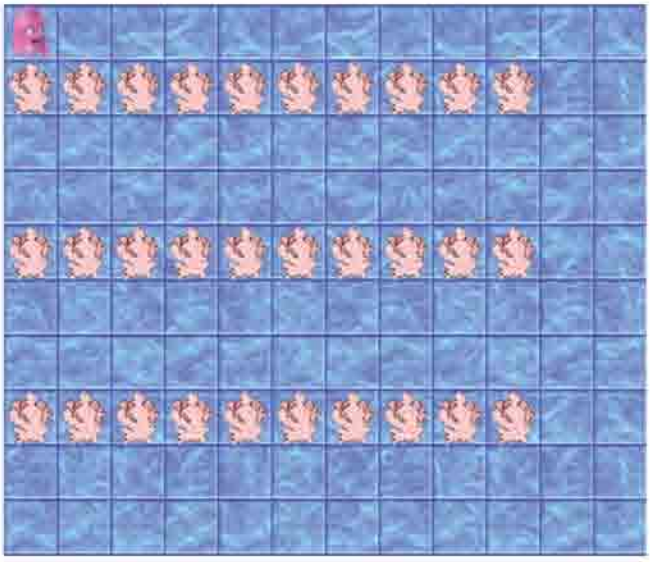


Рис. 95

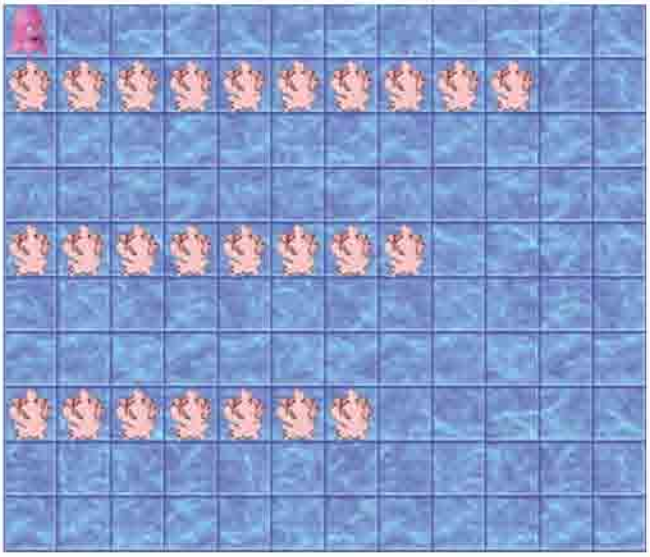


Рис. 96

г) **Проц Рядок (х, у)**
Початок
Повтори х разів
Вправо
Зафарбуй
Все
Повтори х разів
Вліво
Все
Повтори у разів
Вниз
Все
Кінець

Початок
Рядок (6, 2)
Рядок (2, 1)
Рядок (4, 3)
Кінець

3.3 Команди розгалуження та присвоювання

1. *Наведіть приклади ситуацій, у яких виконуються різні дії в залежності від результату перевірки певної умови.*
2. *З яких команд складається і як виконується розгалуження в алгоритмі?*
3. *Чим відрізняється лінійна частина алгоритму від розгалуження?*
4. *Наведіть приклади розгалужень, з якими ви зустрічалися при вивченні шкільних предметів.*
5. *Якщо в алгоритмі використовується процедура* **Малюнок (x, y)**, *то що відбувається по команді* **Малюнок (3, 4)**?

Команди розгалуження

Восьминіжка стоїть перед коридором відомої довжини. У верхній стіні цього коридору в невідомих місцях є одноклітинні виступи. Потрібно зафарбувати клітинки в цих виступах.

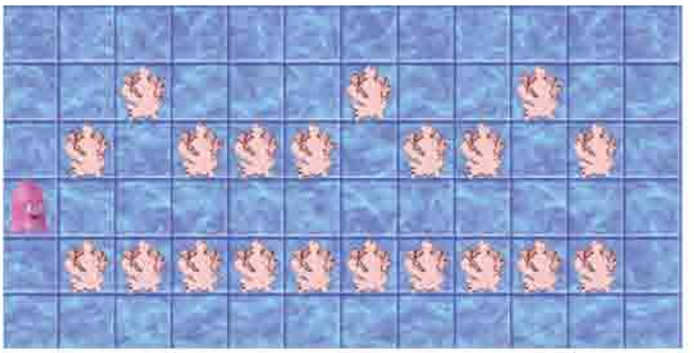


Рис. 97

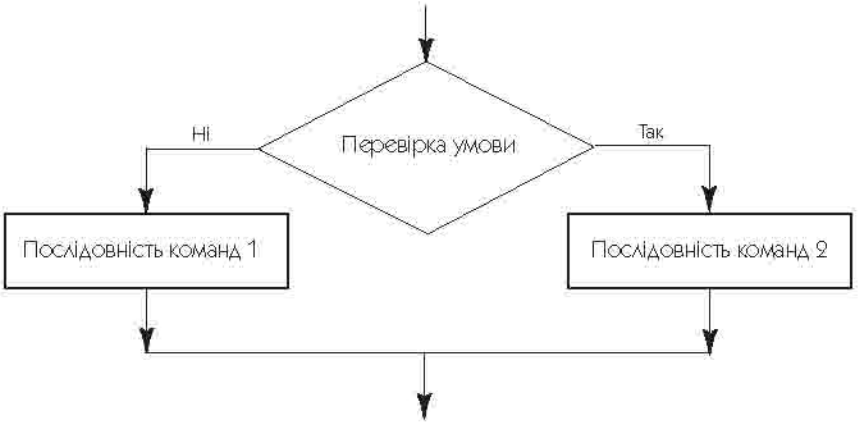
Очевидно, **Восьминіжка** має йти по коридору і перевіряти, чи є зверху перешкода. Якщо перешкоди немає, то зайти у виступ, зафарбувати клітинку, повернутися назад у коридор і рухатися далі. Якщо перешкода є, то просто рухатися далі по коридору.

Отже, маємо ситуацію, коли **Восьминіжка** має виконати різні дії в залежності від результату перевірки умови. Тобто маємо знайоме нам розгалуження в алгоритмі.

Восьминіжка може виконувати таку команду розгалуження:

```
Якщо <умова>
   <Послідовність команд 1>
Інакше
   <Послідовність команд 2>
Все
```

Блок-схемою цю команду можна зобразити так:



Нагадаємо, що виконується така команда розгалуження таким чином: виконується команда перевірки умови; якщо результат виконання цієї команди **Так**, то виконується **<Послідовність команд 1>**, а **<Послідовність команд 2>** не виконується; якщо ж результат виконання команди перевірки умови **Ні**, то виконується **<Послідовність команд 2>**, а **<Послідовність команд 1>** не виконується.

Система команд виконавця **Восьминіжка** містить такі команди перевірки умови:

Команда	Результат виконання
Зліва перешкода	**Так**, якщо в клітинці, зліва від поточної, є перешкода **Ні**, якщо в клітинці, зліва від поточної, немає перешкоди
Справа перешкода	**Так**, якщо в клітинці, справа від поточної, є перешкода **Ні**, якщо в клітинці, справа від поточної, немає перешкоди
Вгорі перешкода	**Так**, якщо в клітинці, зверху від поточної, є перешкода **Ні**, якщо в клітинці, зверху від поточної, немає перешкоди
Внизу перешкода	**Так**, якщо в клітинці, знизу від поточної, є перешкода **Ні**, якщо в клітинці, знизу від поточної, немає перешкоди
Зліва вільно	**Так**, якщо в клітинці, зліва від поточної, немає перешкоди **Ні**, якщо в клітинці, зліва від поточної, є перешкода
Справа вільно	**Так**, якщо в клітинці, справа від поточної, немає перешкоди **Ні**, якщо в клітинці, справа від поточної, є перешкода
Вгорі вільно	**Так**, якщо в клітинці, зверху від поточної, немає перешкоди **Ні**, якщо в клітинці, зверху від поточної, є перешкода
Внизу вільно	**Так**, якщо в клітинці, знизу від поточної, немає перешкоди **Ні**, якщо в клітинці, знизу від поточної, є перешкода
Зафарбовано	**Так**, якщо поточна клітинка зафарбована **Ні**, якщо поточна клітинка незафарбована
Не зафарбовано	**Так**, якщо поточна клітинка незафарбована **Ні**, якщо поточна клітинка зафарбована

Наприклад, якщо **Восьминіжка** знаходиться в такому положенні (*рис. 98*), то результати виконання відповідних команд будуть наступними:

Команда	Результат виконання
Зліва перешкода	Так
Справа перешкода	Ні
Вгорі перешкода	Так
Внизу перешкода	Ні
Зліва вільно	Ні
Справа вільно	Так
Вгорі вільно	Ні
Внизу вільно	Так
Зафарбовано	Ні
Не зафарбовано	Так

Рис. 98

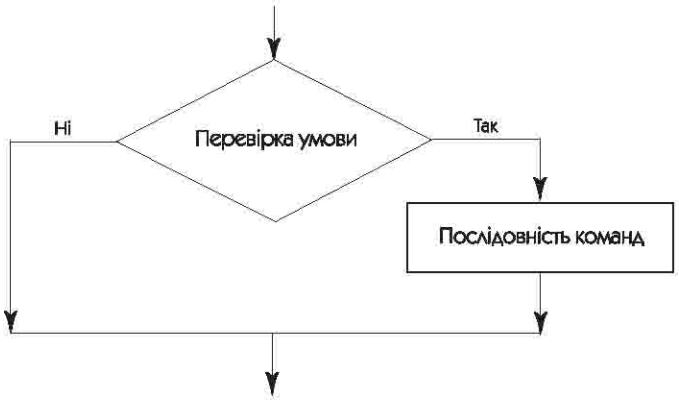
Крім зазначених команд перевірки умови з системи команд виконавця **Восьминіжка**, умовами можуть бути звичайні рівності або нерівності. Наприклад, $15 > 7$, $5 < 0$, **а** >3.

Алгоритм розв'язування поставленої задачі буде такий:

```
Вправо
Повтори 10 разів
  Якщо Зверху вільно
    Вгору
    Зафарбуй
    Вниз
    Вправо
  Інакше
    Вправо
  Все
Все
```

Звертаємо увагу, що й у випадку, коли результат перевірки умови **Зверху вільно – Так**, і в протилежному випадку **Восьминіжка** повинна виконати команду **Вправо**. У таких випадках цю команду можна вилучити з команди розгалуження і поставити її після неї. Але тоді після слова **Інакше** взагалі команди відсутні. Для таких випадків використовується інший вид команди розгалуження:

```
Якщо <умова>
    <Послідовність команд>
Все
```



Блок-схемою цю форму команди розгалуження можна зобразити так:

Виконується ця команда розгалуження так: виконується команда перевірки умови; якщо результат виконання цієї команди **Так**, то виконується **<Послідовність команд>**; якщо ж результат виконання команди перевірки умови **Ні**, то одразу виконується наступна команда алгоритму.

З використанням такої команди розгалуження алгоритм для розв'язування попередньої задачі буде такий:

```
Вправо
Повтори 10 разів
  Якщо Зверху вільно
    Вгору
    Зафарбуй
    Вниз
  Все
  Вправо
Все
```

Перша з розглянутих форм команди розгалуження називається **повною**, а друга – **неповною**.

Команда присвоювання

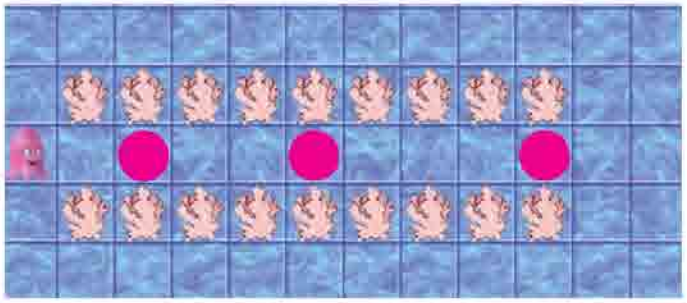


Рис. 99

Раніше ви складали алгоритми для різних виконавців, виконуючи які, вони створювали рисунки, саджали дерева, переходили з однієї позиції до іншої. Тепер настав час розглянути алгоритми, у ході виконання яких виконуються певні обчислення.

Нехай **Восьминіжка** знаходиться перед горизонтальним коридором заданої довжини, деякі клітинки якого зафарбовані (*рис. 99*). Потрібно підрахувати кількість зафарбованих клітинок.

Очевидно, **Восьминіжка** повинна, виконуючи команди циклу, йти по коридору і кожну клітинку перевіряти на зафарбованість. Якщо їй зустрінеться зафарбована клітинка, потрібно збільшити кількість зафарбованих клітинок на 1.

Для підрахунку кількості зафарбованих клітинок використаємо змінну з іменем **кількість**. Перед початком руху **Восьминіжки** по коридору **присвоїмо (надамо)** цій змінній значення 0. Це абсолютно логічно, адже **Восьминіжка** поки що не знайшла в коридорі жодної зафарбованої клітинки. Це можна зробити командою: **кількість := 0**.

Зі змінними ви вже зустрічалися і в 6-му, і в 7-му класах, коли використовували процедури з аргументами. Ці змінні набували своїх значень при кожному виклику такої процедури, і ці значення вказувалися саме в команді виклику.

Взагалі, для того щоб змінній присвоїти (надати) певне значення, використовується спеціальна команда – **команда присвоювання**. Її загальний вигляд такий:

<ім'я змінної> := <число, ім'я змінної або арифметичний вираз>.

Знак ":=" називається знаком присвоювання.

Наведемо приклади таких команд:

x := 45; a := –496; число := a; y := x; x := 4 * 5 + 22;
сума := 2 * c + a; кількість := 4; кількість := кількість + 2.

Звертаємо увагу, що ім'я змінної може містити літери латинського та українського алфавітів, а також цифри. Але цифра не може бути першим символом імені.

Звертаємо увагу, що дія множення, на відміну від математики, позначається значком *, а дія ділення – значком /.

Якщо в правій частині команди присвоювання стоїть число, то в результаті її виконання значення змінної, ім'я якої вказано в лівій частині команди, дорівнюватиме цьому числу. Наприклад, після виконання команди **a := –496** змінна **a** матиме значення – **496**.

Якщо в правій частині команди присвоювання стоїть ім'я змінної, то в результаті її виконання змінна, ім'я якої вказано в лівій частині команди, одержує зна-

чення змінної, ім'я якої стоїть у правій частині команди. Наприклад, після виконання команди **y := x** змінна **y** матиме значення, що дорівнює значенню змінної **x** на момент виконання цієї команди. Зазначимо, що після виконання цієї команди значення змінної **x** не змінюється.

Якщо в правій частині команди присвоювання стоїть арифметичний вираз, то спочатку обчислюється значення цього виразу, а потім змінна, ім'я якої вказано в лівій частині команди, одержує значення, що дорівнює обчисленому. Наприклад, після виконання команди **x := 4 * 5 + 22** змінна **x** буде мати значення **42**. А при виконанні команди **x := 2 * c + a** комп'ютер помножить поточне значення змінної **c** на **2**, до результату додасть поточне значення змінної **a**, після чого одержане значення присвоїть змінній **x**. Так, якщо на момент виконання цієї команди **c = 25** і **a = 55**, то після її виконання змінна **x** матиме значення **105**.

Якщо в результаті виконання команди присвоювання деяка змінна набула певного значення, то вона зберігає його при подальшому виконанні програми до того часу, поки в програмі не виконається інша команда присвоювання, у лівій частині якої буде вказано ім'я цієї змінної.

Звернемо особливу увагу на такі приклади команди присвоювання: **кількість := 4**; **кількість := кількість + 2**. У першій з них змінна **кількість** одержує значення **4**, а в другій до цього значення додається **2**, у результаті чого змінна **кількість** одержує нове значення **6**. Іншими словами, виконання команди **кількість := кількість + 2** збільшує значення змінної **кількість** на **2**.

Аналогічно, виконання команди **c := c − 1** зменшує значення змінної **c** на **1**, а команда **t := t * 4** множить поточне значення змінної **t** на **4**.

Зважаючи на вищесказане, алгоритм для розв'язування поставленої задачі виглядатиме так:

```
Вправо
кількість := 0
Повтори 9 разів
  Якщо Зафарбовано
    кількість := кількість + 1
  Все
  Вправо
Все
Повідомлення ("Кількість зафарбованих клітинок у коридорі: ", кількість)
```

Звертаємо увагу, що останній алгоритм містить нову команду – **команду виведення повідомлення**. За цією командою відкривається спеціальне вікно, у якому й виводиться повідомлення. Текст, який узято в лапки, виводиться повністю, як він і записаний у самій команді. Потім після коми вказано ім'я змінної, значення якої потрібно вивести.

ЗАПИТАННЯ ТА ЗАВДАННЯ

1. *Назвіть команди перевірки умови виконавця* **Восьминіжка**. *Поясніть їх виконання.*
2. *У чому відмінність результатів виконання команд перевірки умови виконавця* **Восьминіжка** *від результатів виконання інших його команд?*
3. *Наведіть приклад команди розгалуження для* **Восьминіжки**. *Поясніть її виконання.*
4. *Поясніть різницю між повною і неповною формою команди розгалуження.*
5. *Наведіть блок-схеми загального виду повної і неповної форм команди розгалуження. Поясніть, як виконується кожна з них.*
6. *Чи може в команді розгалуження після виконання команди перевірки умови більше не виконуватися жодна команда? Якщо так, наведіть приклади.*
7. **Восьминіжка** *стоїть у крайній зліва клітинці горизонтального коридору заданої довжини. Складіть алгоритм, у результаті виконання якого вона зафарбує незафарбовані клітинки коридору.*
8. *На полі* **Восьминіжки** *немає стін. Серед восьми клітинок, які знаходяться праворуч від* **Восьминіжки**, *є зафарбовані. Складіть алгоритм, у результаті виконання якого вона зафарбує клітинки над і під зафарбованими.*
9. **Восьминіжка** *стоїть у лівому нижньому куті прямокутної ділянки заданих розмірів, обмеженої з усіх боків перешкодами. Уздовж перешкод є зафарбовані клітинки. Складіть алгоритм, у результаті виконання якого вона зафарбує незафарбовані клітинки уздовж перешкод.*
10. **Восьминіжка** *стоїть у лівому нижньому куті прямокутної ділянки заданих розмірів, обмеженої з усіх боків перешкодами. Усередині ділянки є зафарбовані клітинки. Складіть алгоритм, у результаті виконання якого вона зафарбує всі незафарбовані клітинки ділянки.*
11. *Визначте значення змінних у результаті виконання кожної з наведених послідовностей команд:*

а)
```
a := 5;
b := a;
a := 12;
a := b;
b := b-a.
```

б)
```
a := 3;
b := 10;
b := a;
a := a-b;
```

в)
```
a := 5;
b := 4;
c := a+b;
c := c/10.
```

г)
```
x := 12;
y := -10;
t := x+y;
x := t;
```

д)
```
c := 3;
c := 2*c;
c := c*c.
```

12. *Змінна* **x** *має значення 10. Чому буде дорівнювати значення цієї змінної після виконання команд:*

а) `x := 2;` б) `x := x + 5;` в) `y := x`?

13. *Яке значення мала змінна* **x** *до виконання наведених команд, якщо після виконання кожної з них її значення стало дорівнювати 1?*

а) `x := x+5;` б) `x := -x;` в) `y := x;` г) `y := 1;` д) `y := x; x := x + y; x := y.`

14. Після виконання яких з наведених послідовностей команд змінні **x** *та* **y** *обміняються своїми значеннями? Якщо вам складно зробити висновок, розгляньте конкретні значення змінних* **x** *та* **y**.

а)
```
x := y;
y := x;
y := t;
```

б)
```
t := x;
x := y;
x := x-y;
```

в)
```
x := x + y;
y := x - y;
x := y;
```

г)
```
t := x;
y := t.
```

15. Порівняйте початкове і кінцеве значення змінної **c** *при виконанні наведених послідовностей команд:*

а)
```
c := c + 3;
k := c - 3;
c := c - k;
```

б)
```
c := c*2;
y := c/2
c := c-y.
```

16. Запишіть відповідні команди присвоювання:
а) збільшити значення змінної **x** *на 1; на 5; на 7; на 12;*
б) зменшити значення змінної **y** *на 2; на 7; на 20;*
в) помножити поточне значення змінної **x** *на 3; на 5; на 10;*
г) поділити поточне значення змінної **t** *на 4; на 2.*

17. Учень складав алгоритм, який підраховує кількість зафарбованих клітинок, і забув включити до нього перед командою циклу команду **кількість := 0**. *Поясніть, що відбуватиметься при цьому.*

18. Учень складав алгоритм, який підраховує кількість зафарбованих клітинок, і включив команду **кількість := 0** *до тіла циклу. Поясніть, що відбуватиметься при цьому.*

19. **Восьминіжка** *стоїть перед горизонтальним коридором вліво заданої довжини. Складіть алгоритм, у якому порівнюються кількості зафарбованих і незафарбованих клітинок у цьому коридорі.*

20. **Восьминіжка** *стоїть між двома горизонтальними коридорами відомої довжини. Складіть алгоритм, у якому порівнюються кількості зафарбованих клітинок у цих коридорах.*

21. **Восьминіжка** *стоїть на відстані 5 клітинок від перешкоди, яка знаходиться праворуч від неї. Деякі клітинки між нею і перешкодою зафарбовані. Складіть алгоритм, у результаті виконання якого вона зафарбує за перешкодою стільки клітинок підряд, скільки зафарбованих клітинок між нею і перешкодою.*

22. **Восьминіжка** *стоїть зліва від горизонтальної стіни перешкод заданої довжини. Складіть алгоритм, у якому порівнюється кількість зафарбованих клітинок над стіною і під стіною.*

23. **Восьминіжка** *стоїть у лівому нижньому куті прямокутної ділянки заданих розмірів, обмеженої з усіх боків перешкодами. Уздовж перешкод є зафарбовані клітинки. Складіть алгоритм, у результаті виконання якого вона визначить кількість зафарбованих клітинок.*

24. **Восьминіжка** *стоїть у лівому нижньому куті прямокутної ділянки заданих розмірів, обмеженої з усіх боків перешкодами. Усередині ділянки є зафарбовані клітинки. Складіть алгоритм, у результаті виконання якого вона визначить кількість зафарбованих клітинок.*

3.4 Команда циклу Повтори поки

1. Як виконується команда циклу **Повтори N разів***?*
2. Чи завжди заздалегідь відома кількість повторень циклу? Наведіть приклади.

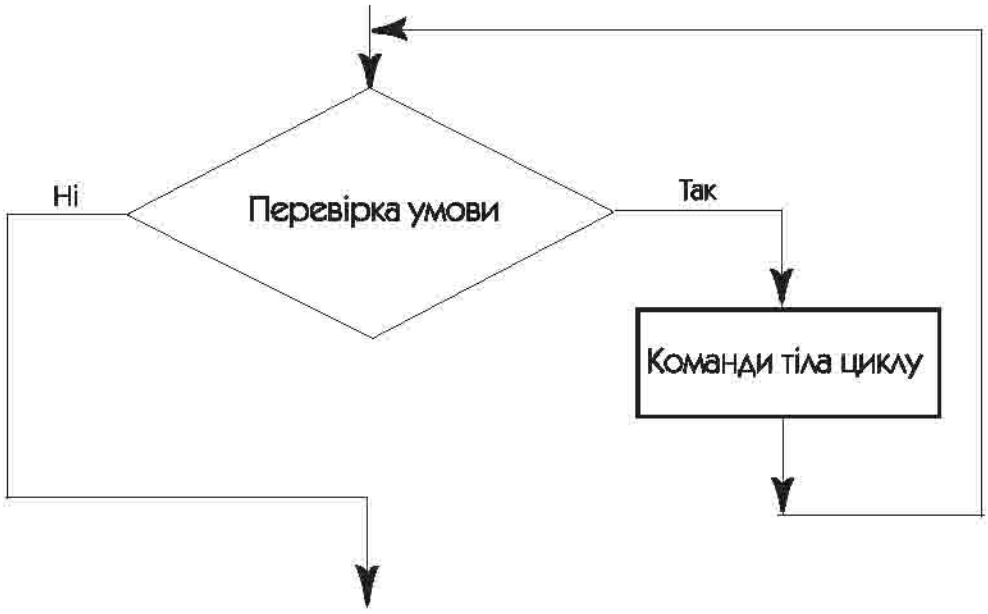
У попередніх задачах ви складали алгоритм для **Восьминіжки** при умові, що розміри ділянки заздалегідь відомі. Тому, якщо змінити ці розміри, наприклад, довжину коридору, обмеженого перешкодами, потрібно вносити зміни до тексту програми, зокрема змінювати кількість повторень циклу.

А чи не можна скласти такий універсальний алгоритм, який би правильно працював при довільній довжині коридору? Виявляється, можна. Але для цього потрібно в алгоритмі використати іншу команду циклу, кількість повторень якого заздалегідь невідома, а залежить від результату перевірки певної умови.

Загальний вигляд цієї команди циклу такий:

Повтори поки <умова>
<Команди тіла циклу>
Все

Блок-схемою цю команду циклу можна зобразити так:



Виконується ця команда циклу так: виконується команда перевірки умови; якщо результат виконання цієї команди **Так**, то виконуються **<Команди тіла циклу>**, після чого знову виконується команда перевірки умови; якщо ж результат виконання команди перевірки умови **Ні**, то виконується наступна команда алгоритму.

Покажемо застосування цієї команди циклу.

Нехай **Восьминіжка** стоїть перед горизонтальним коридором невідомої довжини. Потрібно визначити кількість зафарбованих клітинок у ньому.

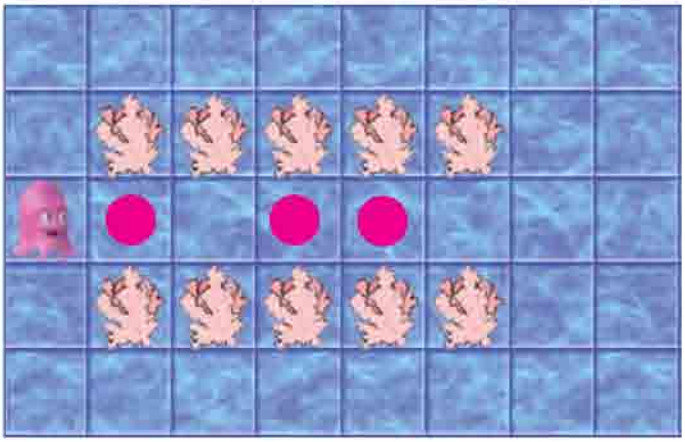


Рис. 100

```
Вправо
кількість := 0
Повтори поки Зверху перешкода
  Якщо Зафарбовано
        кількість := кількість + 1
  Все
  Вправо
Все
Повідомлення ("Кількість зафарбованих клітинок у коридорі: ", кількість)
```

Продемонструємо виконання цього алгоритму для конкретного коридору довжиною 5 клітинок, у якому зафарбовані вказані клітинки (*рис. 100*).

Команда	Результат виконання
Вправо	**Восьминіжка** переходить до першої клітинки коридору
кількість := 0	кількість = 0
Зверху перешкода	**Так**
Зафарбовано	**Так**
кількість := кількість + 1	кількість = 0 + 1 = 1
Вправо	**Восьминіжка** переходить до другої клітинки коридору
Зверху перешкода	**Так**
Зафарбовано	**Ні**
Вправо	**Восьминіжка** переходить до третьої клітинки коридору
Зверху перешкода	**Так**
Зафарбовано	**Так**
кількість := кількість + 1	кількість = 1 + 1 = 2
Вправо	**Восьминіжка** переходить до четвертої клітинки коридору
Зверху перешкода	**Так**
Зафарбовано	**Так**
кількість := кількість + 1	кількість = 2 + 1 = 3
Вправо	**Восьминіжка** переходить до п'ятої клітинки коридору
Зверху перешкода	**Так**
Зафарбовано	**Ні**
Вправо	**Восьминіжка** виходить з коридору
Зверху перешкода	**Ні**
Повідомлення	Кількість зафарбованих клітинок у коридорі: 3

ЗАПИТАННЯ ТА ЗАВДАННЯ

1. *Наведіть приклад команди циклу* **Повтори поки**. *Поясніть її виконання.*
2. *Назвіть загальний вигляд команди циклу* **Повтори поки**. *Поясніть його виконання.*
3. *Чи можуть команди тіла циклу не виконуватися жодного разу? Якщо так, наведіть приклад.*
4. *Чи може виконання команди циклу ніколи не закінчитися? Якщо так, наведіть приклад.*
5. *Порівняйте дві команди циклу:* **Повтори N разів** *і* **Повтори поки**. *Що в них спільного? Чим вони відрізняються одна від одної?*
6. *Чим відрізняється команда циклу* **Повторити поки** *від команди розгалуження?*
7. *Визначте, що є результатом наведених фрагментів алгоритмів для даної обстановки (рис. 101).*

Повтори поки Справа вільно
  Зафарбувати
  Вправо
Все

Повтори поки Справа вільно
  Вправо
  Зафарбувати
Все

Повтори поки Знизу перешкода
  Зафарбувати
  Вправо
Все

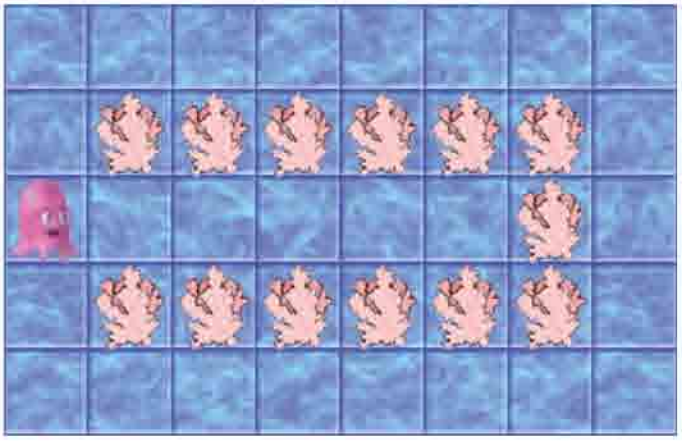


Рис. 101

Виконайте наведені фрагменти програми для даної обстановки.

8. *Визначте, що є результатом наведених фрагментів алгоритмів для даної обстановки (рис. 102).*

Повтори поки Знизу перешкода
  Зафарбувати
  Вправо
Все

Повтори поки Знизу перешкода
  Вправо
  Зафарбувати
Все

Повтори поки Справа вільно
  Зафарбувати
  Вправо
Все

Виконайте наведені фрагменти програми для даної обстановки.

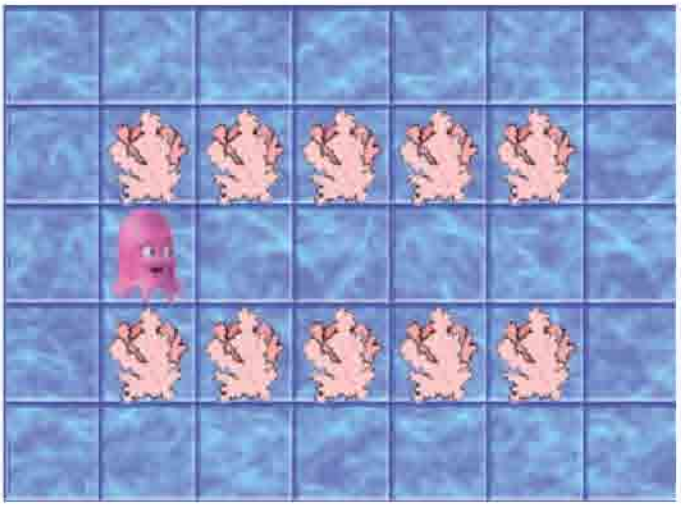


Рис. 102

9. *Зліва від* **Восьминіжки** *знаходиться перешкода. Складіть алгоритм, виконавши який вона зафарбує всі клітинки між початковим положенням і перешкодою.*
10. **Восьминіжка** *стоїть перед горизонтальним коридором з перешкод. Складіть алгоритм, виконавши який вона зафарбує всі клітинки коридору.*
11. **Восьминіжка** *стоїть у першій клітинці горизонтального коридору з перешкод. Складіть алгоритм, виконавши який вона зафарбує всі клітинки коридору.*
12. **Восьминіжка** *стоїть у деякій внутрішній клітинці горизонтального коридору з перешкод. Складіть алгоритм, виконавши який вона зафарбує всі клітинки коридору, крім тієї, у якій вона початково знаходилася.*
13. **Восьминіжка** *стоїть у деякій внутрішній клітинці горизонтального коридору з перешкод. Складіть алгоритм, виконавши який вона зафарбує всі клітинки коридору.*
14. **Восьминіжка** *стоїть у лівому нижньому куті прямокутної ділянки, обмеженої перешкодами. Складіть алгоритм, виконавши який вона зафарбує всі клітинки уздовж стін.*

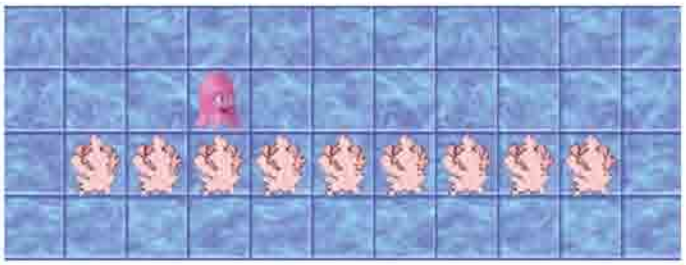


Рис. 103

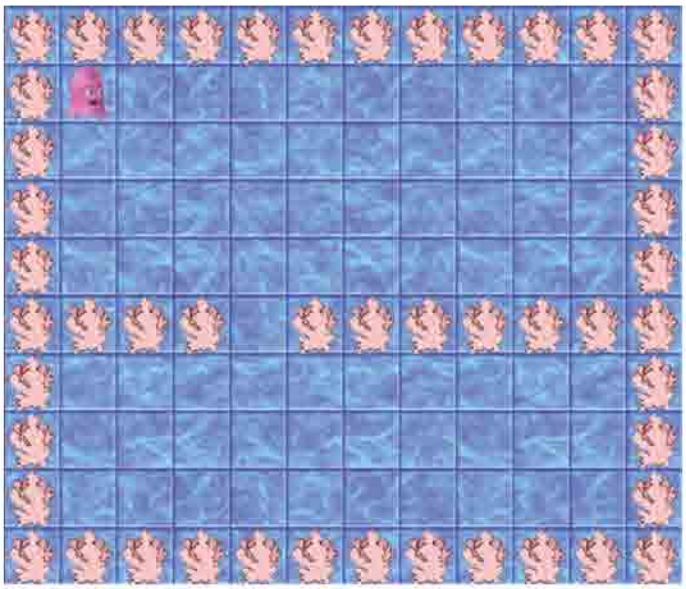


Рис. 104

15. **Восьминіжка** *стоїть у лівому нижньому куті прямокутної ділянки, обмеженої перешкодами. Складіть алгоритм, виконавши який вона зафарбує всі клітинки ділянки.*
16. *Складіть алгоритм, виконавши який* **Восьминіжка** *зафарбує всі клітинки навколо горизонтальної стіни з перешкод (рис. 103).*
17. *Складіть алгоритм, виконуючи який* **Восьминіжка** *перейде до правого нижнього кута ділянки. Положення горизонтальної стіни і місце проходу в ній невідомі (рис. 104).*
18. **Восьминіжка** *стоїть перед горизонтальним коридором вліво невідомої довжини. Складіть алгоритм, у якому порівнюються кількості зафарбованих і незафарбованих клітинок у цьому коридорі.*
19. **Восьминіжка** *стоїть між двома горизонтальними коридорами невідомої довжини. Складіть алгоритм, у якому порівнюються кількості зафарбованих клітинок у цих коридорах.*
20. **Восьминіжка** *стоїть на деякій відстані від перешкоди, яка знаходиться праворуч від неї. Деякі клітинки між нею і перешкодою зафарбовані. Складіть алгоритм, у результаті виконання якого вона*

зафарбує за перешкодою стільки клітинок підряд, скільки зафарбованих клітинок між нею і перешкодою.

21. **Восьминіжка** *стоїть зліва від горизонтальної стіни перешкод невідомої довжини. Складіть алгоритм, у якому порівнюються кількості зафарбованих клітинок над стіною і під нею.*
22. **Восьминіжка** *стоїть у лівому нижньому куті прямокутної ділянки невідомих розмірів, обмеженої з усіх боків перешкодами. Уздовж перешкод є зафарбовані клітинки. Складіть алгоритм, у результаті виконання якого вона визначить кількість зафарбованих клітинок.*
23. **Восьминіжка** *стоїть у лівому нижньому куті прямокутної ділянки невідомих розмірів, обмеженої з усіх боків перешкодами. Усередині ділянки є зафарбовані клітинки. Складіть алгоритм, у результаті виконання якого вона визначить кількість зафарбованих клітинок.*