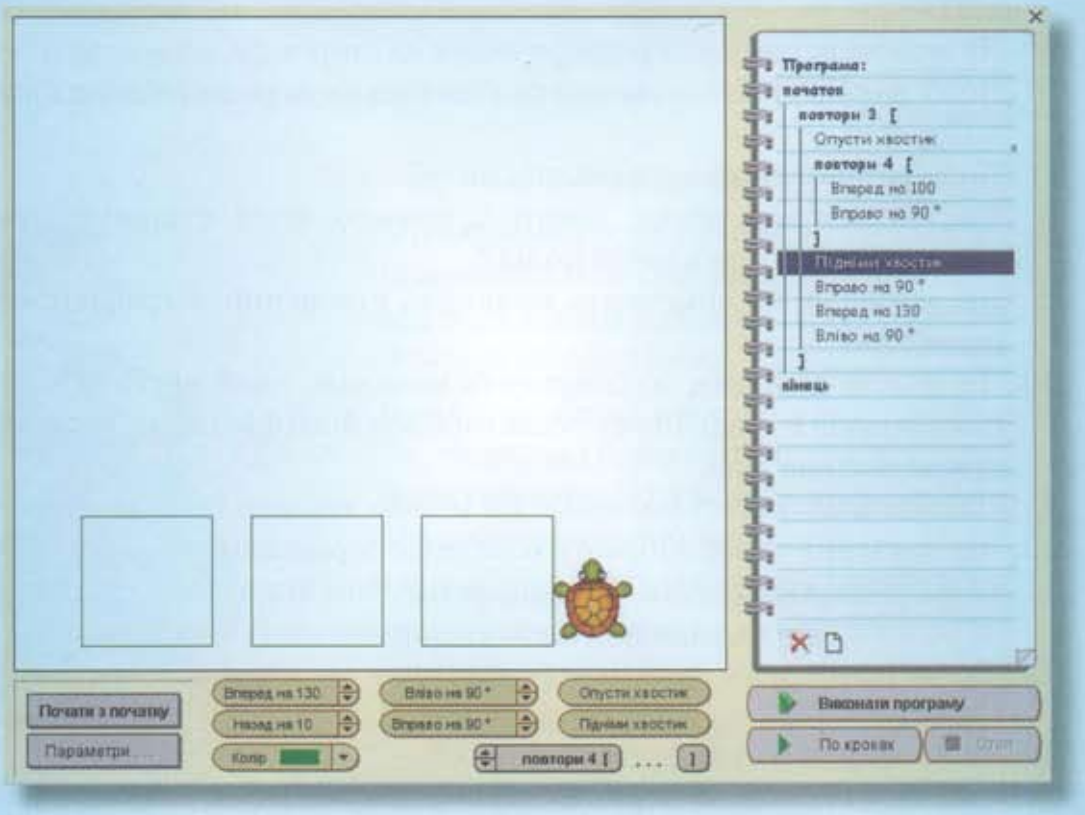
Розділ 3

ОСНОВИ АЛГОРИТМІЗАЦІЇ



3.1

ЧЕРЕПАШКА І ЦИКЛИ

1. *Пригадайте систему команд виконавця* **Черепашка**. *Які дії виконує* **Черепашка** *по кожній з цих команд?*
2. *Що таке алгоритм?*
3. *Які алгоритми називаються лінійними, з розгалуженням, з циклом?*

У п'ятому класі ви познайомилися з виконавцем **Черепашка**. Сподіваємося, вам сподобалося складати алгоритми для цього виконавця. Ви складали для нього лінійні алгоритми і алгоритми з циклами. Продовжимо цю роботу і в шостому класі.

Нагадаємо, що

- циклом називається одна або кілька команд алгоритму, які можуть виконуватися більше одного разу;
- цикли в алгоритмі доцільно використовувати тоді, коли малюнок, який повинна намалювати **Черепашка**, містить кілька однакових фрагментів;
- починається цикл в алгоритмі для **Черепашки** командою **Повтори N разів** (**N** — кількість повторень команд в циклі);
- команди, які повинні виконуватися в циклі, потрібно брати в квадратні дужки.

Подивіться на малюнок.

Щоб скласти правильний і раціональний алгоритм для **Черепашки**, ви повинні з'ясувати:

- на які однакові фрагменти можна розділити цей малюнок;
- скільки разів цей фрагмент повторюється;
- чи містить він й інші фрагменти.

Звичайно ж, ви з'ясували, що малюнок складається з 6 таких фрагментів і ще одного відрізка.

Тому для **Черепашки** потрібно скласти алгоритм, в якому вона спочатку підготується до малювання (повернеться вправо на 90° і опустить хвостик), потім 6 разів намалює фрагмент, після чого домалює відрізок.

Виглядатиме цей алгоритм так:

Вправо на 90
Опусти хвостик
Повтори 6 разів
[
Вперед на 10
Вліво на 90
Вперед на 50
Вправо на 90
Вперед на 20
Вправо на 90
Вперед на 50
Вліво на 90
]
Вперед на 10

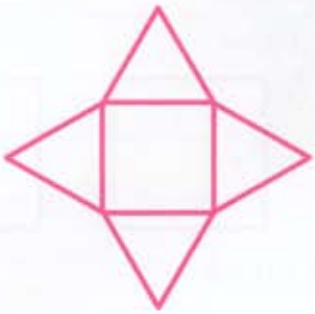
Можливо, хтось із вас, уважно дивлячись на малюнок, розділив його на інші однакові фрагменти. Якщо ні, то спробуйте це зробити зараз. Як виглядатиме алгоритм для **Черепашки** в цьому випадку?

ЗАПИТАННЯ ТА ЗАВДАННЯ

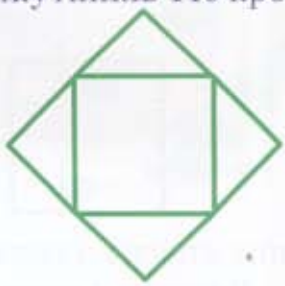
1. Що називається циклом? Наведіть приклади алгоритмів з циклами з оточуючого життя.
2. Складіть алгоритми для **Черепашки**:
   а) Довжина сторін трикутників 60 кроків, відстань між ними 15 кроків.

б) Довжина катетів 40 кроків, відстань між трикутниками 10 кроків (пригадайте формулу для обчислення довжини гіпотенузи рівнобедреного прямокутного трикутника, якщо відомі довжини катетів).

в) Довжина сторін трикутників 100 кроків.



г) Довжина гіпотенуз рівнобедрених прямокутних трикутників 140 кроків.



д) Довжина сторін трикутників 80 кроків.

е) Довжина катетів 60 кроків.

3. Виконайте в зошиті алгоритми

а)
```
Опусти хвостик
Повтори 4 рази
[
 Вправо на 45
 Вперед на 28
 Вліво на 45
 Назад на 20
]
```

б)
```
Вперед на 35
Вправо на 90
Повтори 5 разів
[
 Опусти хвостик
 Вперед на 70
 Вліво на 45
 Назад на 50
 Вліво на 90
 Вперед на 50
 Вправо на 135
 Підніми хвостик
 Вперед на 90
]
```

в)
```
Вправо на 45
Повтори 4 рази
[
 Опусти хвостик
 Вперед на 40
 Вправо на 90
 Вперед на 40
 Вправо на 90
 Вперед на 40
 Вправо на 90
 Вперед на 40
 Вправо на 135
 Підніми хвостик
 Вперед на 70
 Вліво на 45
]
```

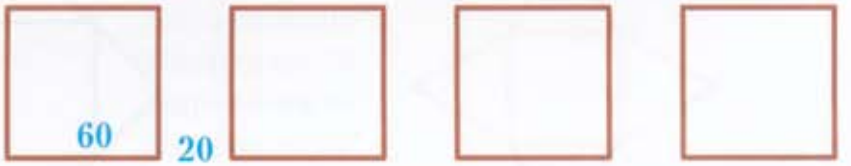
г)
```
Опусти хвостик
Повтори 12 разів
[
 Вперед на 50
 Вправо на 90
 Вперед на 50
 Вправо на 90
 Вперед на 50
 Вправо на 90
 Вперед на 30
 Вправо на 120
]
```

3.2

ЧЕРЕПАШКА І ВКЛАДЕНІ ЦИКЛИ

1. Що таке цикл?
2. Наведіть приклади циклічних процесів у навколишньому світі.

Розгляньте малюнок.



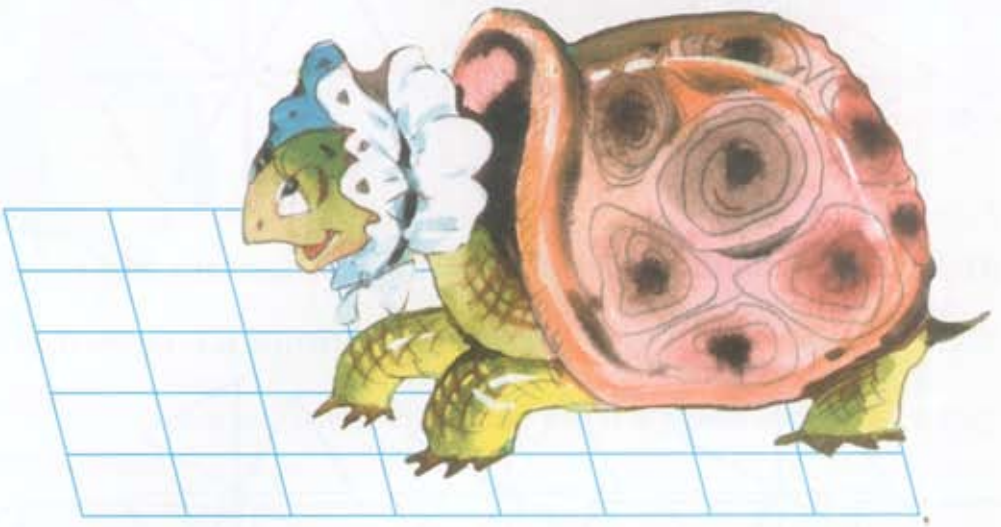
Алгоритм його малювання може бути таким:

Повтори 4 рази
[
Опусти хвостик
Вперед на 60
Вправо на 90
Вперед на 60
Вправо на 90
Вперед на 60
Вправо на 90
Вперед на 60
Підніми хвостик
Назад на 80
Вправо на 90
]

Якщо проаналізувати команди циклу цього алгоритму, то ви побачите серед них послідовність з двох команд **Вперед на 60** і **Вправо на 90**, які повторюються тричі підряд. Це означає, що цю послідовність команд можна записати за допомогою команди циклу. І тоді алгоритм виглядатиме так:

Повтори 4 рази
[
Опусти хвостик
Повтори 3 рази
[
Вперед на 60
Вправо на 90
]
Вперед на 60
Підніми хвостик
Назад на 80
Вправо на 90
]

Запам'ятайте, що цикл, який знаходиться всередині іншого циклу, називається **внутрішнім (вкладеним) циклом,** а цикл, в якому знаходиться інший цикл, називається **зовнішнім циклом.**



ЗАПИТАННЯ ТА ЗАВДАННЯ

1. Які цикли називаються зовнішніми, а які внутрішніми?
2. Наведіть приклади вкладених циклів.
3. Складіть алгоритми для **Черепашки**, використовуючи вкладені цикли:
   а) Довжина сторін трикутників 100 кроків, відстань між ними 20 кроків.

   б) Довжина сторін квадратів 60 кроків, відстань між ними 10 кроків.

в) Довжина сторін трикутників 80 кроків.

г) Довжина сторін трикутників 60 кроків.

д) Довжина сторін трикутників 80 кроків, відстань між квітками 10 кроків.

5. Виконайте в зошиті алгоритми:

а)
```
Повтори 3 рази
[
 Вправо на 45
 Опусти хвостик
 Повтори 4 рази
 [
  Вперед на 14
  Вправо на 90
 ]
 Вліво на 45
 Підніми хвостик
 Назад на 40
]
```

б)
```
Повтори 4 рази
[
 Вправо на 30
 Опусти хвостик
 Повтори 3 рази
 [
  Вперед на 20
  Вправо на 120
 ]
 Вліво на 30
 Підніми хвостик
 Назад на 40
]
```

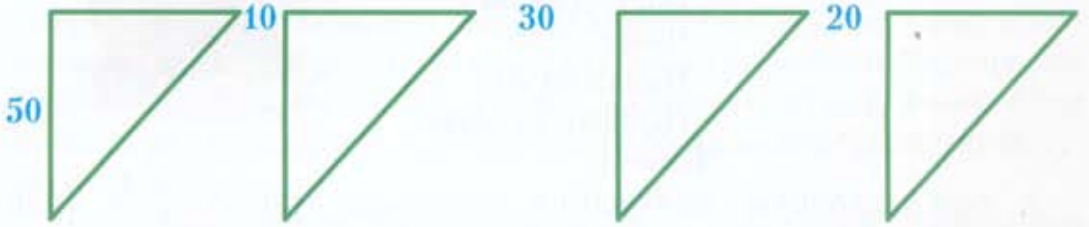
3.3

ПРОЦЕДУРИ

1. *В яких випадках при складанні алгоритму для* Черепашки *можна використовувати цикли?*
2. *Пригадайте випадки, коли ви виконували одну й ту саму послідовність дій, але через різні проміжки часу.*

Алгоритм з процедурою

Розглянемо малюнок. На ньому все ті ж рівнобедрені прямокутні трикутники.



Вам відома послідовність команд для малювання одного такого трикутника:

Опусти хвостик
Вперед на 50
Вправо на 90
Вперед на 50
Вліво на 45
Назад на 70
Підніми хвостик

Оскільки на малюнку чотири рівні між собою трикутники, то в таких випадках ви використовували команду циклу. До цього циклу включалися також і команди переходу від одного трикутника до іншого.

На всіх попередніх малюнках відстані між рівними об'єктами були рівні, тому й команди переходу від одного об'єкта до іншого теж були однакові і могли бути включені до циклу. На цьому ж малюнку ситуація принципово інша. Відстані між трикутниками різні, тому й команди переходу від одного трикутника до іншого теж будуть різні, а значить, вони не можуть бути включені до циклу.

Якщо ж не використовувати команду циклу, то потрібно в алгоритмі чотири рази повторити одні й ті ж самі сім команд малювання трикутника. Це незручно і зробить алгоритм досить громіздким. А якщо таких трикутників буде не 4, а значно більше?

Щоб спростити запис такого алгоритму, команди, що повинні повторюватись, виносять в окрему частину і дають їй ім'я. Таку частину алгоритму називають процедурою.

Щоб викликати на виконання команди процедури, потрібно лише вказати її ім'я. Команда, що складається з імені процедури, називається **командою виклику процедури.**

Ім'я процедури — це довільна послідовність букв і цифр, в якій першою є буква. Великі та малі букви в імені не розрізняються.

Для наведеного малюнка процедура для малювання прямокутного трикутника **ПрямТр** виглядатиме так:

```
Проц ПрямТр
Початок
  Опусти хвостик
  Вперед на 50
  Вправо на 90
  Вперед на 50
  Вліво на 45
  Назад на 70
  Підніми хвостик
Кінець
```

А сам алгоритм малювання чотирьох трикутників буде таким:

```
Проц ПрямТр
Початок
  Опусти хвостик
  Вперед на 50
  Вправо на 90
  Вперед на 50
  Вліво на 45
  Назад на 70
  Підніми хвостик
Кінець

ПрямТр
Вправо на 45
Вперед на 60
Вліво на 90
ПрямТр
Вправо на 45
Вперед на 80
Вліво на 90
ПрямТр
Вправо на 45
Вперед на 70
Вліво на 90
ПрямТр
```

Розглянемо особливості запису і виконання алгоритму, що містить процедуру.

Якщо алгоритм містить процедуру, то його запис розпочинається із запису цієї процедури.

Запис процедури розпочинається з **рядка заголовка**, який має вигляд **Проц *ім'я процедури***. У наступному рядку записується спеціальне слово **Початок**, далі йдуть команди, що входять до процедури, і закінчується запис процедури спеціальним словом **Кінець** Після цього записуються команди **основної частини алгоритму**

Зауважимо, що в наведеному прикладі між процедурою і основною частиною алгоритму пропущено один рядок. Це зроблено для зручності сприйняття алгоритму в цілому. Рекомендуємо вам завжди так робити.

Виконання алгоритму з процедурою

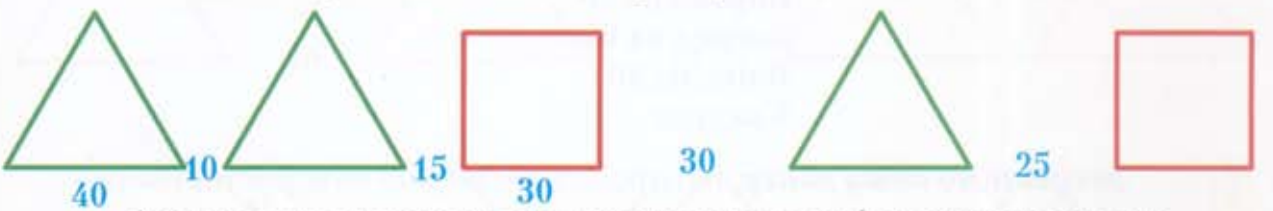
Черепашка розпочинає виконання алгоритму з процедурою з першої команди основної частини алгоритму. У наведеному алгоритмі це команда **ПрямТр**. У системі команд виконавця **Черепашка** команди **ПрямТр** немає. Це означає, що в алгоритмі повинна бути процедура з таким іменем, і **Черепашка** починає послідовно виконувати її команди.

Після закінчення виконання команд процедури **Черепашка** виконує наступну команду основної частини алгоритму **Вправо на 45**. Коли в основній частині алгоритму знову зустрінеться команда **ПрямТр**, **Черепашка** знову виконає команди цієї процедури, після чого виконає наступну команду основної частини алгоритму. І так далі, доки **Черепашка** не виконає всі команди основної частини алгоритму.

Алгоритм з кількома процедурами

Алгоритм може містити не одну, а кілька процедур. Усі вони мають бути записані послідовно одна за одною на початку алгоритму, і кожна повинна мати своє унікальне для цього алгоритму ім'я.

Наведемо приклад такого алгоритму для малюнка.



Малюнок складається з трьох рівних між собою рівносторонніх трикутників і двох рівних між собою квадратів. Відстані між ними різні. Тому доцільно використати в алгоритмі дві процедури: процедуру для малювання рівностороннього трикутника і процедуру для малювання квадрата.

```
Проц РівнТр
Початок
 Опусти хвостик
 Повтори 3 рази
 [
  Вперед на 40
  Вправо на 120
 ]
 Підніми хвостик
Кінець
```

```
Проц Квадрат
Початок
 Опусти хвостик
 Повтори 4 рази
 [
  Вперед на 30
  Вправо на 90
 ]
 Підніми хвостик
Кінець

Вправо на 30
РівнТр
Вправо на 60
Вперед на 50
Вліво на 90
РівнТр
Вправо на 60
Вперед на 55
Вліво на 90
Квадрат
Вправо на 90
Вперед на 60
Вліво на 60
РівнТр
Вправо на 60
Вперед на 65
Вліво на 90
Квадрат
```

Звертаємо вашу увагу, що процедури можна використовувати не тільки тоді, коли малюнок містить кілька однакових об'єктів, а й тоді, коли малюнок складається з різних об'єктів.

Так, наприклад, якщо на малюнку один рівносторонній трикутник, один рівнобедрений прямокутний трикутник і один прямо-

кутник, то можна скласти три процедури для малювання кожного з цих об'єктів, а в основній частині алгоритму викликати їх однією командою і переходити до малювання наступного об'єкта.

Такий підхід до складання алгоритму називається **процедурним**. Алгоритм з процедурами значно легше сприймається, в ньому легше, при потребі, шукати помилки.

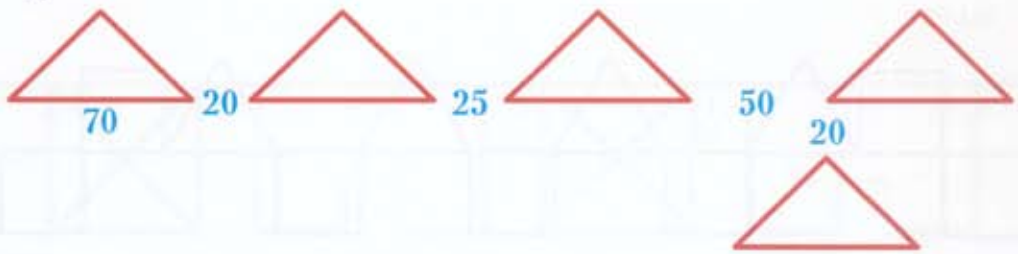
Крім того, процедури одного алгоритму можуть бути включені й до іншого алгоритму, якщо малюнок, для малювання якого він складається, містить точно такі ж об'єкти. Це дає змогу створити **бібліотеку процедур** малювання найбільш поширених об'єктів і використовувати її для складання різноманітних алгоритмів.

ЗАПИТАННЯ ТА ЗАВДАННЯ

1. Що таке процедура?
2. В яких випадках використовують процедури при складанні алгоритмів?
3. Наведіть приклади процедур з оточуючого життя.
4. Як записуються процедури в алгоритмі?
5. Що може бути іменем процедури?
6. Як виглядає команда виклику процедури?
7. Як виконавець виконує алгоритм, який містить процедури?
8. У чому полягає суть процедурного методу при складанні алгоритму?
9. Складіть алгоритми для **Черепашки**, використовуючи процедури:

а)

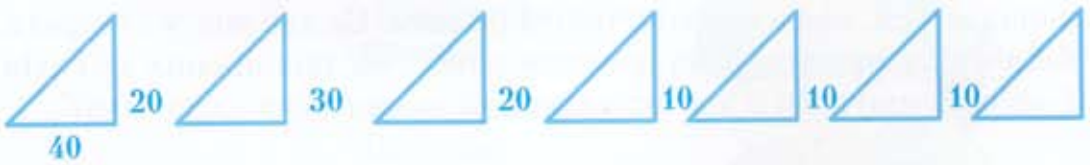
б)



в)


30
10
10
10
30
40

г)


20
30
20
10
10
10
40

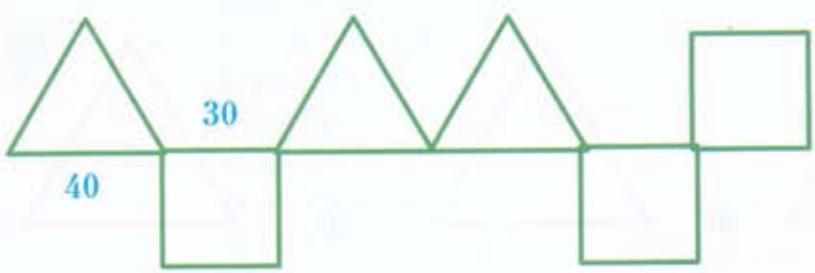
д)


20
15
30
40
30
20
60
25

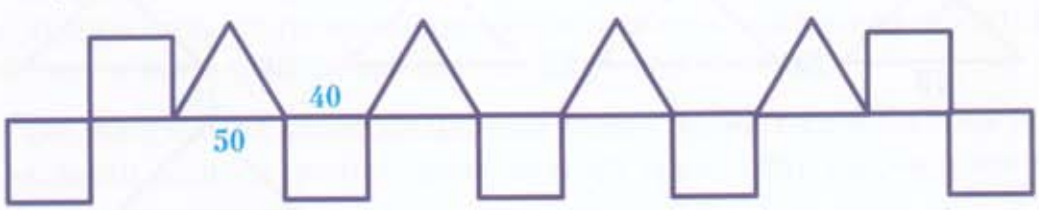
е)


20
20
20
30
20
40
50
30

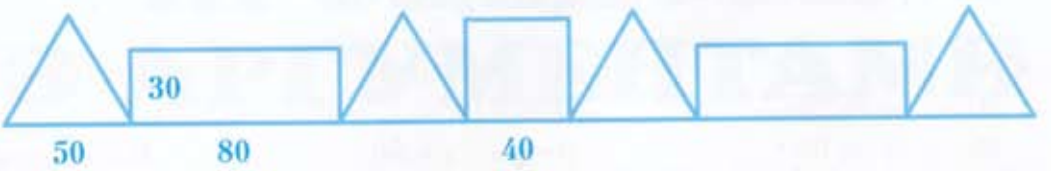
є)


30
40

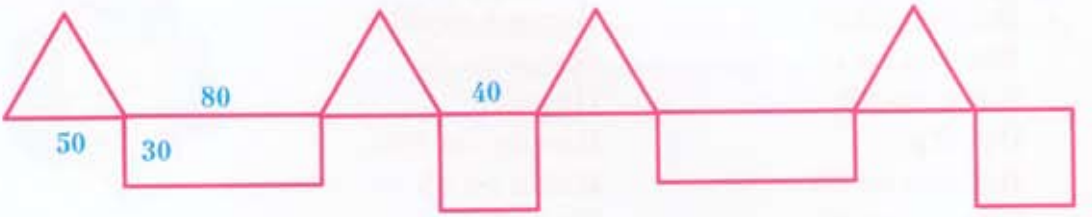
ж)


40
50

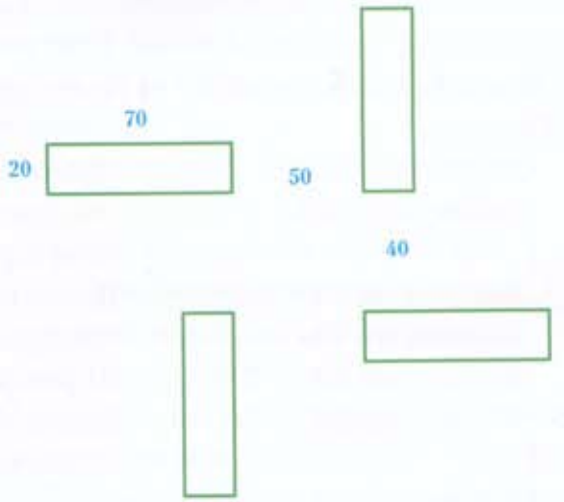
з)



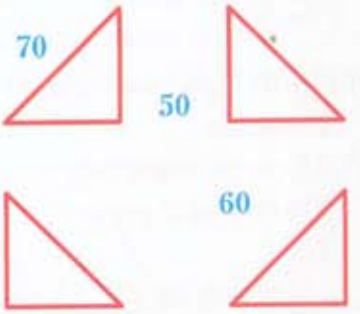
и)



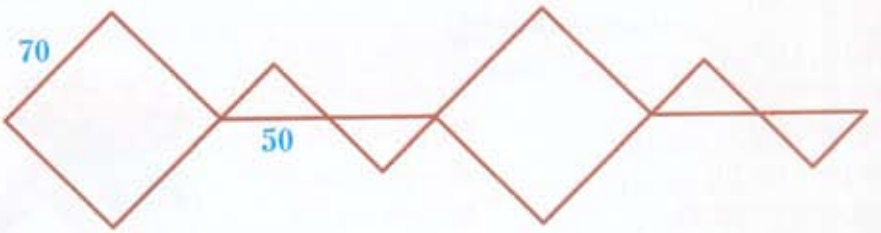
і)



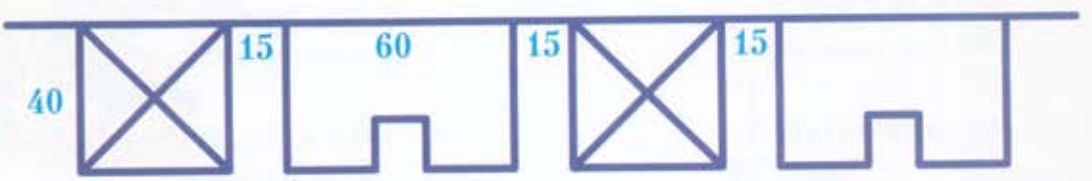
ї)



й)



к)



10. Виконайте в зошиті алгоритм (процедури **РівнТр**, **ПрямТр**, **Квадрат** наведені на сторінках 78 і 80):

а)
```
РівнТр
Вправо на 90
Вперед на 60
Вліво на 90
РівнТр
Вправо на 90
Вперед на 70
Вліво на 90
РівнТр
Вправо на 90
Вперед на 80
Вліво на 90
Назад на 50
РівнТр
```

б)
```
Вправо на 45
ПрямТр
Вперед на 80
Вліво на 45
ПрямТр
Вперед на 90
Вліво на 45
ПрямТр
Вперед на 100
Вліво на 45
ПрямТр
```

в)
```
Повтори 3 рази
[
 Вліво на 45
 ПрямТр
 Назад на 80
]
```

г)
```
Проц Прямокутник1
Початок
 Опусти хвостик
 Повтори 2 рази
 [
  Вперед на 20
  Вправо на 90
  Вперед на 40
  Вправо на 90
 ]
 Підніми хвостик
Кінець

Прямокутник1
Вправо 90
Вперед на 50
Прямокутник1
Вправо 90
Вперед на 60
Прямокутник1
Вправо 90
Вперед на 70
Прямокутник1
```

д)
```
Проц Прямокутник2
Початок
 Опусти хвостик
 Повтори 2 рази
 [
  Вперед на 20
  Вправо на 90
  Вперед на 30
  Вправо на 90
 ]
 Підніми хвостик
Кінець

Повтори 3 рази
[
 Вправо на 45
 Прямокутник2
 Вправо на 45
 Вперед на 50
 Вліво на 90
]
```

е)
```
Повтори 3 рази
[
 ПрямТр
 Вправо на 45
 Вперед на 70
 Вліво на 90
 Квадрат
 Вправо на 90
 Вперед на 40
 Вліво на 90
]
```

3.4

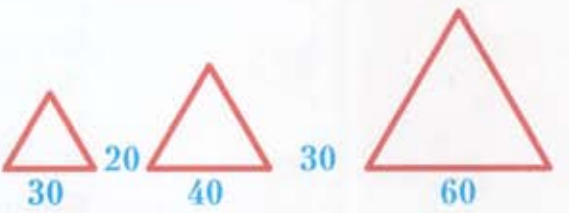
ПРОЦЕДУРИ З АРГУМЕНТАМИ

1. *Що таке процедура? Які правила її запису в алгоритмі?*
2. *Як виконується алгоритм з процедурами?*
3. *У чому полягає суть процедурного методу створення алгоритму?*

Процедури з одним аргументом

У попередньому пункті ви дізналися, як за допомогою процедур спростити запис алгоритму у випадках, коли малюнок містить рівні між собою об'єкти, але вони розташовані на різних відстанях один від одного.

Підемо ще далі. Розглянемо малюнок, на якому є схожі об'єкти, але нерівні між собою. А саме, три нерівні рівносторонні трикутники, які ще й знаходяться на різних відстанях один від одного.



Очевидно, що для малювання кожного з цих трикутників потрібно скласти окрему процедуру. Але це, знову ж таки, не раціонально. Тим більше, що таких нерівних рівносторонніх трикутників на малюнку може бути не три, а значно більше.

Спробуємо скласти одну процедуру, за допомогою якої **Черепашка** могла б намалювати рівносторонній трикутник з будь-якою довжиною сторони. Проаналізуємо, чим відрізняються процедури для малювання першого та другого трикутника:

```
Проц РівнТр1
Початок
 Опусти хвостик
 Повтори 3 рази
 [
  Вперед на 30
  Вправо на 120
 ]
 Підніми хвостик
Кінець
```

```
Проц РівнТр2
Початок
 Опусти хвостик
 Повтори 3 рази
 [
  Вперед на 40
  Вправо на 120
 ]
 Підніми хвостик
Кінець
```

Оскільки самі трикутники відрізняються один від одного тільки довжиною сторони, то й процедури для їх малювання відрізняються одна від одної тільки однією командою. Причому, в цих командах напрям руху **Черепашки** один і той самий (**Вперед**), а відмінність полягає лише в кількості кроків, які **Черепашка** повинна пройти (**Вперед на 30** і **Вперед на 40**). Аналогічну ситуацію маємо і з третім трикутником.

До процедури, за допомогою якої **Черепашка** зможе намалювати довільний рівносторонній трикутник, замість команд **Вперед на 30**, **Вперед на 40**, **Вперед на 60** включимо команду **Вперед на х** (**х** — змінна). Те, що в процедурі будуть використані команди зі змінними, потрібно вказати в заголовку процедури, і він матиме такий вигляд **Проц РівнТр (х)**.

Тобто, процедура для малювання довільного рівностороннього трикутника виглядатиме так:

```
Проц РівнТр (х)
Початок
Опусти хвостик
Повтори 3 рази
[
Вперед на х
Вправо на 120
]
Підніми хвостик
Кінець
```

Виконання процедур з аргументом

Якщо змінна **х** не матиме конкретного значення, то **Черепашка** не зможе виконати команду **Вперед на х**, адже вона не знатиме, скільки саме кроків їй потрібно пройти вперед. Повідомити про це ми можемо в команді виклику цієї процедури.

Для малювання першого трикутника команда виклику процедури виглядатиме так: **РівнТр (30)**. Ви вже знаєте, що команди **РівнТр** немає серед системи команд виконавця **Черепашка**. Тому вона шукатиме в алгоритмі процедуру з цим іменем. Така процедура в алгоритмі є, і в її заголовку в дужках вказана змінна **х**. **Черепашка** надасть цій змінній значення 30, яке вказано в дужках у команді виклику процедури, після чого команда **Вперед на х** набуде вигляду **Вперед на 30** і **Черепашка** зможе виконати її та намалювати перший з даних рівносторонніх трикутників.

Для малювання другого трикутника команда виклику процедури виглядатиме так: **РівнТр (40)**. Тобто, викликається та ж сама процедура, але при виклику змінній **х** надається інше значення: 40. Тому тепер команда **Вперед на х** набуде вигляду **Вперед на 40** і **Черепашка** малюватиме другий з даних рівносторонніх трикутників.

Очевидно ви вже зрозуміли, що для малювання третього з даних рівносторонніх трикутників потрібно викликати цю ж саму процедуру командою **РівнТр (60)**.

Таким чином, алгоритм для малювання наведеного малюнка виглядатиме так:

```
Проц РівнТр (х)
Початок
  Опусти хвостик
  Повтори 3 рази
  [
   Вперед на х
   Вправо на 120
  ]
  Підніми хвостик
Кінець

Вправо на 30
РівнТр (30)
Вправо на 60
Вперед на 50
Вліво на 60
РівнТр (40)
Вправо на 60
Вперед на 70
Вліво на 60
РівнТр (60)
```

Процедура, що містить команди зі змінними, які набувають своїх значень при виклику процедури, називається **процедурою з аргументами**.

Процедури з кількома аргументами

Процедура може містити не тільки один, а й кілька аргументів. Для малювання цього ж самого малюнка ми можемо скласти процедуру, за допомогою якої можна буде не тільки намалювати довільний рівносторонній трикутник, а й перейти до наступного трикут-

ника. Відстані між трикутниками будемо задавати при виклику процедури як значення ще однієї змінної.

Виглядатиме ця процедура і основна частина алгоритму так:

```
Проц РівнТрТаПерехід (х, у)
Початок
  Опусти хвостик
  Повтори 3 рази
  [
   Вперед на х
   Вправо на 120
  ]
  Підніми хвостик
  Вправо на 60
  Вперед на у
  Вліво на 60
Кінець
Вправо на 30
РівнТрТаПерехід (30, 50)
РівнТрТаПерехід (40, 70)
РівнТрТаПерехід (60, 0)
```

При виклику процедури **РівнТрТаПерехід (х, у)** командою **РівнТрТаПерехід (30, 50)** змінній **х** надається значення 30, а змінній **у** — значення 50. Відповідно команда **Вперед на х** набуває вигляду **Вперед на 30**, а команда **Вперед на у** — **Вперед на 50**. Виконуючи команди цього алгоритму, **Черепашка** малює перший з даних рівносторонніх трикутників і переходить до початку малювання другого.

При виклику процедури **РівнТрТаПерехід (х, у)** командою **РівнТрТаПерехід (40, 70)** змінній **х** надається значення 40, а змінній **у** — значення 70. Відповідно команда **Вперед на х** набуває вигляду **Вперед на 40**, а команда **Вперед на у** — **Вперед на 70**. Виконуючи команди цього алгоритму, **Черепашка** малює другий з даних рівносторонніх трикутників і переходить до початку малювання третього.

Аналогічним є й третій виклик цієї процедури для малювання третього трикутника. При цьому змінна **у** набуває значення 0 і команда **Вперед на у** перетворюється на команду **Вперед на 0**. При виконанні цієї команди **Черепашка** залишається на місці. Це цілком відповідає ситуації, коли, намалювавши третій трикутник, **Черепашка** вже не повинна переходити до малювання наступного трикутника.

Процедуру з двома аргументами можна також використовувати, наприклад, для малювання довільного прямокутника. Вона виглядатиме так:

```
Проц Прямокутник (х, у)
Початок
  Опусти хвостик
  Повтори 2 рази
  [
    Вперед на х
    Вправо на 90
    Вперед на у
    Вправо на 90
  ]
  Підніми хвостик
Кінець
```

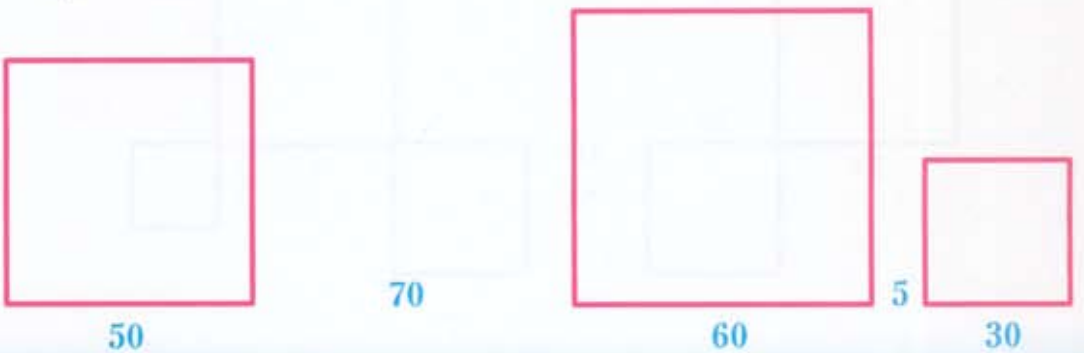
Виклик цієї процедури може відбутися, наприклад, командою **Прямокутник (20, 30)**. Після цього, як і в попередньому прикладі, змінна **х** набуде значення 20, а змінна **у** — значення 30. Отже, команда **Вперед на х** перетвориться на команду **Вперед на 20**, а команда **Вперед на у** — на команду **Вперед на 30**.

Сподіваємося, ви зрозуміли, що аналогічним чином можна використовувати процедури з трьома, чотирма та з іншою кількістю аргументів.

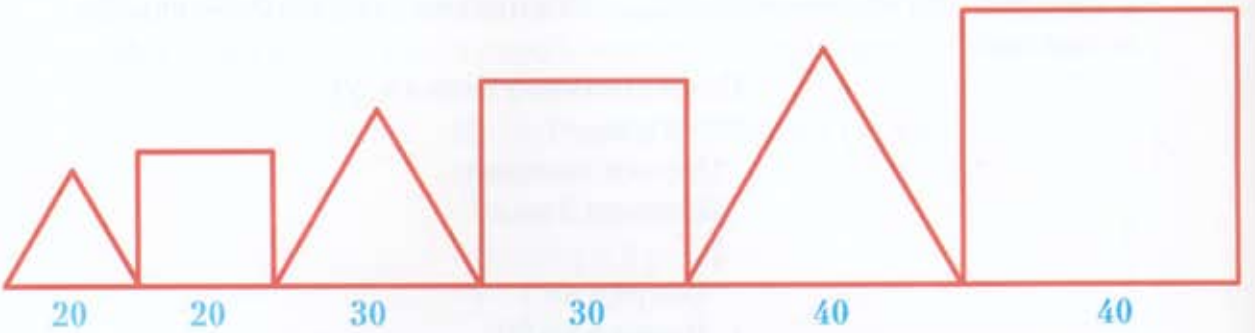
ЗАПИТАННЯ ТА ЗАВДАННЯ

1. Що таке процедура з аргументами? Для чого вони використовуються в алгоритмі?
2. Як записуються процедури з аргументами в алгоритмі?
3. Як викликаються і виконуються процедури з аргументами?
4. Складіть алгоритми для **Черепашки**, використовуючи процедури з аргументами:

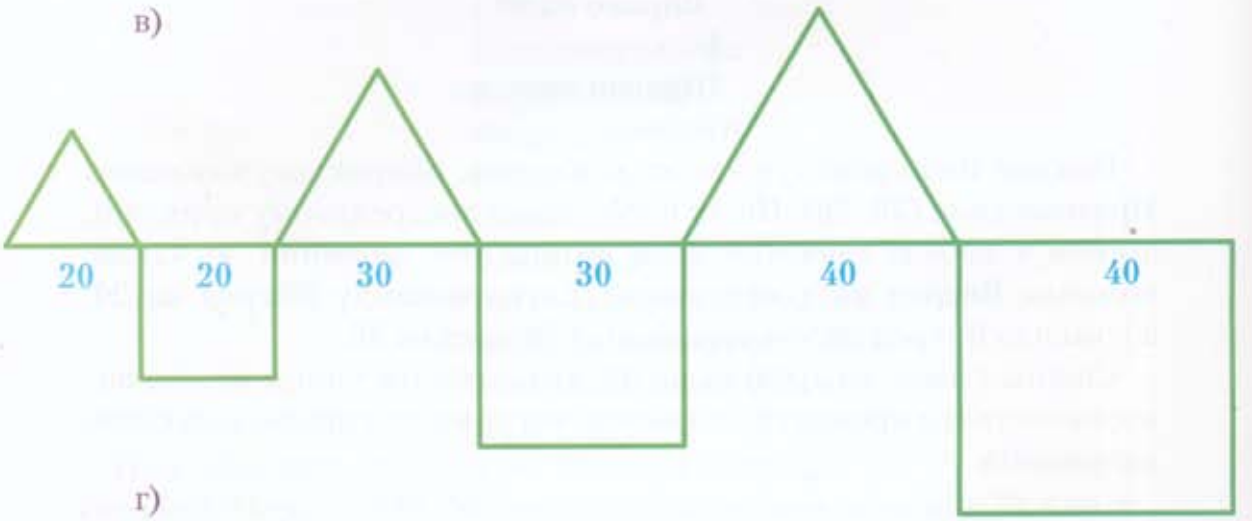
а)



б)

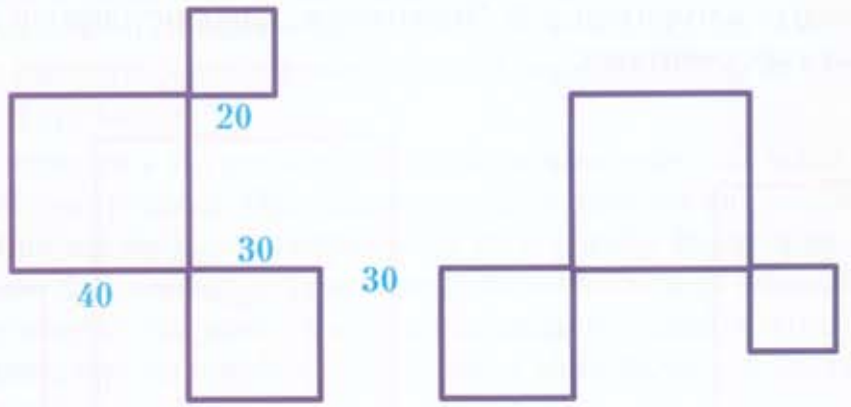


в)



г)

д)



5. Виконайте в зошиті алгоритми, що містять процедури з аргументами

а)
```
Проц Фігура1 (х)
Початок
  Опусти хвостик
  Вперед на х
  Вліво на 90
  Вперед на х
  Вліво на 90
  Вперед на х
  Підніми хвостик
Кінець

Вправо на 90
Фігура1 (20)
Назад на 30
Вправо на 90
Назад на 20
Вправо на 90
Фігура1 (30)
```

б)
```
Проц Фігура2 (х, у)
Початок
  Опусти хвостик
  Вперед на у
  Вправо на 90
  Вперед на х
  Вліво на 90
  Назад на у
  Підніми хвостик
Кінець

Фігура2 (20, 30)
Вправо на 90
Вперед на 20
Вліво на 90
Фігура2 (30, 40)
```

в)
```
Проц Фігура3 (х)
Початок
  Опусти хвостик
  Вперед на х
  Вліво на 120
  Вперед на х
  Підніми хвостик
Кінець

Вправо на 90
Фігура3 (20)
Назад на 20
Вправо на 120
Вперед на 10
Фігура3 (30)
Назад на 30
Вправо на 120
Вперед на 20
Фігура3 (40)
```