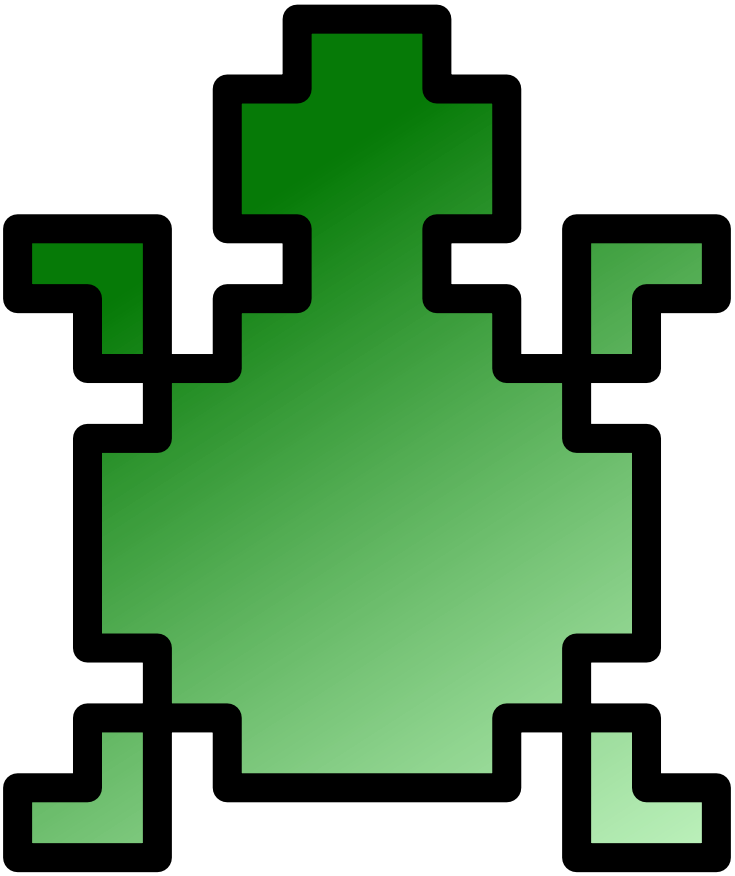
Підручник з KTurtle

Cies Breijs
Anne-Marie Mahfouf
Mauricio Piacentini
Переклад українською: Юрій Чорноіван



Підручник з KTurtle

Зміст

Перелік таблиць

Анотація

KTurtle є навчальним середовищем програмування, що спрямоване на якомога простіший процес навчання програмуванню. Для цього у KTurtle передбачено доступ до всіх інструментів програмування за допомогою графічного інтерфейсу. Мовою програмування середовища є TurtleScript, у цій мові команди може бути перекладено мовою користувача.

Розділ 1

Вступ

KTurtle навчальне середовище для програмування, яке використовує мову програмування TurtleScript, дуже близьку і засновану на Logo. Метою KTurtle є зробити програмування якомога легшим і доступнішим. Це робить KTurtle придатним для навчання дітей математиці, геометрії і... так, програмуванню. Одною з основних особливостей TurtleScript є можливість перекладу команд мовою, якою розмовляє програміст.

KTurtle названо на честь «черепашки (turtle)», яка відіграє головну роль у середовищі для програмування. Користувач, зазвичай, створює перелік наказів черепашці, за допомогою команд мови TurtleScript, з метою намалювати на полотні картинку.

1.1 Звідки взявся TurtleScript?

TurtleScript, мова програмування, яка використовується у KTurtle, засновано на деяких з фундаментальних концепцій сімейства мов програмування Logo. Першу версію Logo було створено Сеймуром Папертом з лабораторії штучного інтелекту Массачусетського Технологічного Університету у 1967 році, як відгалуження мови програмування LISP. З того часу було випущено багато версій Logo. До 1980 року популярність Logo досягла максимуму, існували версії для систем MSX, Commodore, Atari, Apple II і IBM PC. Ці версії мали в основному навчальну мету. LCSI випустила програму Mac®Logo у 1985 році як інструмент для професійних програмістів, але він так і залишився непопулярним. Массачусетський Технологічний Університет все ще підтримує сайт про Logo. На цьому сайті ви знайдете список декількох популярних реалізацій цієї мови.

У TurtleScript передбачено можливість, характерну для інших реалізацій мови Logo: можливість перекласти команди рідною мовою учня. Це полегшує вивчення програмування учнями без знання англійської або з незначними знаннями цієї мови. Окрім цієї особливості, у KTurtle передбачено багато інших можливостей, спрямованих на полегшення для учнів початкових кроків у програмуванні.

1.2 Можливості KTurtle

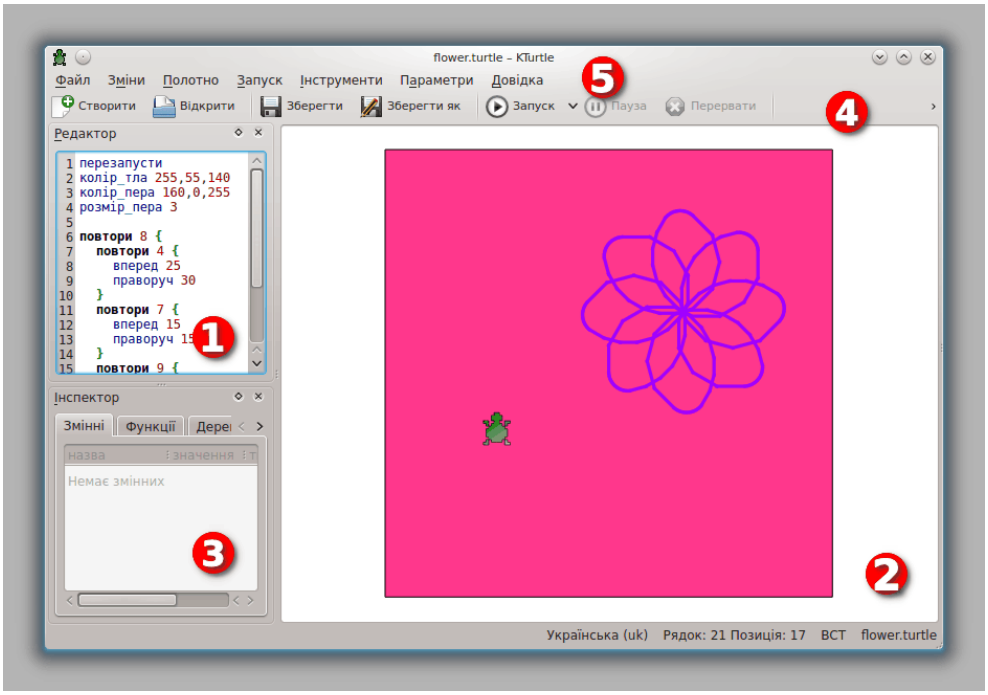
KTurtle має декілька приємних особливостей, які роблять перші кроки у програмуванні фантастично простими. Ось декілька основних можливостей KTurtle:

- Комплексне середовище з інтерпретатором TurtleScript, редактором, полотном та іншими інструментами у одній програмі (без зовнішніх залежностей).
- Можливість перекладу команд TurtleScript за допомогою інструментів перекладу KDE.
- TurtleScript підтримує визначені користувачем функції, рекурсію та динамічне перемикання типів.

- Виконання програми можна сповільнити, призупинити або припинити у будь-який зручний час.
- Потужний редактор з інтуїтивно зрозумілим підсвічуванням синтаксису, нумеруванням рядків, позначенням помилок, візуальним виконанням та додатковими можливостями.
- Полотно, на якому малює черепашка, можна надрукувати або зберегти як зображення (PNG) або креслення (SVG).
- Контекстна довідка: довідка саме там, де вона потрібна. Просто натисніть клавішу F2 (або скористайтеся пунктом меню Довідка → Довідка щодо: ...), щоб отримати довідку з фрагмента коду, який перебуває під курсором.
- Діалогове вікно помилок, у якому повідомлення про помилку пов'язуються з рядками, де ці помилки трапилися у програмі, і позначаються червоним кольором.
- Спрощена термінологія програмування.
- Вбудовані приклади програм зроблять перші кроки надзвичайно простими. Ці приклади можна перекласти за допомогою інструментів перекладу KDE.

Розділ 2

Користування KTurtle



Головне вікно KTurtle поділено на три частини: ліворуч знаходиться редактор (1), в ньому ви пишете команди мовою TurtleScript, праворуч — полотно (2), на ньому, власне, ви і бачите результат виконання команд, і інспектор (3), де показано інформацію про змінні під час виконання програми. Окрім цих елементів ви побачите смужку меню (5), за допомогою якої ви можете отримувати доступ до дій програми, панель інструментів (4), яка надає вам змогу швидко дістатися до найвикористовуваніших дій програми, і смужка стану (уздовж нижнього краю вікна), де буде показано інформацію щодо поточного стану KTurtle.

2.1 Редактор

У полі редактора ви пишете команди мовою TurtleScript. Можливості редактора зосереджено у меню Файл і меню Зміни. Редактор можна прив'язати до будь-якого краю головного вікна або від'єднати і розташувати у будь-якому місці стільниці.

Є декілька шляхів введення коду у редакторі. Найпростішим є використання готового прикладу: виберіть пункт Файл → Приклади з меню «Файл», потім виберіть якийсь із прикладів.

Файл прикладу, який ви обрали, буде відкрито у редакторі. Щоб виконати цей код, ви можете скористатися пунктом меню Запуск → Запуск (скорочення: Alt+F2) або натисканням кнопки Запуск на панелі інструментів.

Файли з програмами на TurtleScript можна відкривати за допомогою пункту меню Файл → Відкрити....

Третім шляхом є напряму набрати ваш код у редакторі, або скопіювати і вставити код.

2.2 Полотно

Полотно — це область, де відбувається візуальне представлення результату виконання команд, місце, де за допомогою команд малюється картинка. Якщо написати якісь команди у редакторі і виконати їх, можуть статися дві речі: код буде належним чином виконано, в результаті чого ви скоріше за все побачите якісь зміни на полотні; або, якщо ви зробили помилку у вашому коді, буде показано вкладку помилки з поясненням щодо вашої помилки.

Збільшувати і зменшувати зображення полотна можна за допомогою коліщатка вашої миші.

2.3 Інспектор

Інспектор інформує вас про змінні, вивчені функції і ієрархічну структуру програми під час її виконання.

Інспектор можна прив'язати до будь-якого краю головного вікна або від'єднати і розташувати у будь-якому місці вашої стільниці.

2.4 Панель інструментів

За допомогою цієї панелі ви можете пришвидшити доступ до найпоширеніших дій. На панелі інструментів розташовано Консоль, за допомогою якої ви зможете швидко вводити команди, що може бути корисним, якщо ви бажаєте перевірити наслідки виконання команди без внесення змін до вмісту редактора.

Доступ до налаштування панелі інструментів з метою зміни її вигляду відповідно до ваших уподобань можна отримати за допомогою пункту меню Параметри → Налаштувати пенали...

2.5 Панель меню

На панелі меню ви знайдете всі дії, які виконує програма KTurtle. Ці дії розділено на такі групи: Файл, Зміни, Полотно, Запуск, Інструменти, Параметри, і Довідка. У цьому розділі ви знайдете докладне пояснення щодо цих груп.

2.5.1 Меню «Файл»

Файл → Створити (Ctrl-N)

Створює новий, порожній файл на TurtleScript.

Файл → Відкрити... (Ctrl-O)

Відкриває файл з програмою на TurtleScript.

Файл → Відкрити недавні

Відкриває файл з програмою на TurtleScript, який вже було нещодавно відкрито.

Файл → Приклади

Відкриває приклади програм на TurtleScript. Приклади написано вибраною вами у підменю Параметри → Мова програм мовою.

Файл → Отримати додаткові приклади...

Відкрити діалогове вікно Отримати нові матеріали для отримання додаткових файлів TurtleScript з інтернету.

Файл → Зберегти (Ctrl-S)

Зберігає поточний файл програми на TurtleScript.

Файл → Зберегти як... (Ctrl-Shift-S)

Зберігає поточний файл програми на TurtleScript у вказаному місці.

Файл → Експортувати в HTML...

Експортує поточний вміст редактора до файла HTML разом з підсвічуванням кольором.

Файл → Друкувати... (Ctrl-P)

Друкує на принтері поточний код у редакторі.

Файл → Вийти (Ctrl-Q)

Завершує роботу KTurtle.

2.5.2 Меню «Зміни»

Зміни → Вернути (Ctrl-Z)

Скасовує останню зміну у коді. Кількість таких скасувань у KTurtle не обмежено.

Зміни → Повторити (Ctrl-Shift-Z)

Повторює виконання останньої скасованої зміни у коді.

Зміни → Вирізати (Ctrl-X)

Вирізає вибраний текст з редактора до буфера.

Зміни → Копіювати (Ctrl-C)

Копіює вибраний текст з редактора до буфера.

Зміни → Вставити (Ctrl-V)

Вставляє текст з буфера до редактора.

Зміни → Вибрати все (Ctrl-A)

Вибирає весь текст у вікні редактора.

Зміни → Пошук... (Ctrl-F)

За допомогою цієї дії ви можете шукати фрази у коді.

Зміни → Знайти далі (F3)

Використовуйте цю дію для пошуку наступного входження фрази, яку ви раніше шукали, до тексту.

Зміни → Знайти позаду (Shift-F3)

Використовуйте цю дію для того, щоб знайти попереднє входження фрази, яку ви раніше шукали, до тексту.

Зміни → Режим перезапису (Ins)

Перемикач між режимами «вставити» та «перезаписати».

2.5.3 Меню «Полотно»

Полотно → Експортувати до зображення (PNG)...

Експортує поточний вміст Полотно до растрового зображення PNG (Portable Network Graphics або машинонезалежна мережева графіка).

Полотно → Експортувати у креслення (SVG)...

Експортує поточний вміст Полотно до векторного креслення типу SVG (Scalable Vector Graphics або масштабована векторна графіка).

Полотно → Надрукувати полотно...

Надсилає на друк поточний вміст Полотна.

2.5.4 Меню «Запуск»

Запуск → Запуск (F5)

Запускає на виконання команди з редактора.

Запуск → Пауза (F6)

Призупиняє виконання. Цю дію можна увімкнути, лише якщо у даний момент виконуються якісь команди.

Запуск → Перервати (F7)

Зупиняє виконання. Цю дію можна увімкнути, лише якщо у даний момент виконуються якісь команди.

Запуск → Швидкість виконання

Показує вам список всіх можливих швидкостей виконання, що складається з таких пунктів: На повну швидкість (без пригальмовування і перевірки), На повну швидкість, Повільно, Повільніше, Найповільніше і Крок-за-кроком. Якщо швидкість виконання встановлено у значення На повну швидкість (типова поведінка), ви навряд чи помітите події, що відбуваються під час виконання програми. Іноді така поведінка — саме те, що потрібно, але іноді важливо прослідкувати за виконанням. У останньому випадку вам слід встановити швидкість виконання у значення Повільно, Повільніше або Найповільніше. Якщо вибрано один з повільних режимів у редакторі коду буде помітно, яка саме команда зараз виконується. У режимі Крок-за-кроком програма виконуватиме по одній команді на кожен запуск.

2.5.5 Меню «Інструменти»

Інструменти → Вибір напрямку...

Цей пункт відкриває діалогове вікно вибору напрямку.

Інструменти → Піпетка...

Цей пункт відкриває діалогове вікно піпетки.

2.5.6 Меню «Параметри»

Параметри → Мова програм

Надає змогу вибрати мову коду.

Параметри → Показати редактор (Ctrl-E)

Наказує програмі показати або сховати вікно редактора.

Параметри → Показати інспектор (Ctrl-I)

Показати або сховати Інспектор.

Параметри → Показати помилки

Показати або сховати вкладку Помилки зі списком помилок, які сталися у результаті спроби виконання коду. Якщо увімкнено цей параметр, для того, щоб побачити черепашку знову, вам слід буде натиснути кнопку Полотно.

Параметри → Показати номери рядків (F11)

Ця дія надасть вам змогу побачити номери рядків програми у редакторі. Це зручно під час пошуку помилок.

Параметри → Показати пенал

Перемкнути видимість головної панелі інструментів.

Параметри → Показати смужку стану

Перемкнути видимість рядка стану.

Параметри → Налаштувати скорочення...

Відкрити стандартне діалогове вікно KDE для налаштування клавіатурних скорочень.

Параметри → Налаштувати пенали...

Відкрити стандартне діалогове вікно KDE для налаштування панелі інструментів.

2.5.7 Меню «Довідка»

У KTurtle передбачено типове меню Довідка, яке описано у підручнику з основ KDE, з одним додатковим пунктом:

Довідка → Довідка щодо: ... (F2)

Цей пункт є дуже корисним: за його допомогою можна отримати довідку щодо інструкцій, у яких перебуває курсор редактора. Отже, наприклад, ви використали у вашій програмі команду print і бажаєте дізнатися, що написано у підручнику щодо цієї команди. Вам достатньо пересунути курсор на цю команду print і натиснути клавішу F2. Буде відкрито вікно довідки з усіма відомостями щодо команди print.

Ця функціональна можливість може знадобитися під час вивчення TurtleScript.

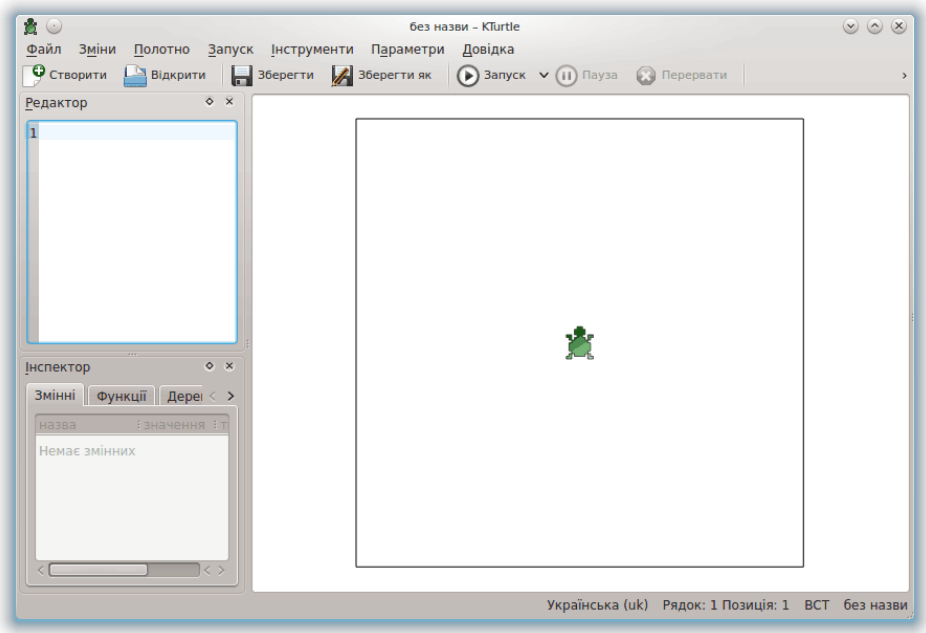
2.6 Смужка стану

У смужці стану ви зможете бачити повідомлення про стан KTurtle. Ліворуч на ній розташовано інформацію про останню дію. Праворуч ви побачите інформацію про поточне розташування курсора (номер рядка і позицію у рядку). Посередині смужки стану показано назву поточної мови, яку програма використовує для команд.

Розділ 3

Початок роботи

Після запуску KTurtle ви побачите щось таке:



У цій інструкції щодо перших кроків у роботі з програмою припускатиметься, що мовою команд TurtleScript є англійська. Ви можете змінити цю мову за допомогою підменю Параметри → Мова скриптів. Майте на увазі, що мову, яку ви там встановите для KTurtle є мовою, якою ви віддаватимете команди на TurtleScript, а не мовою, яку використовує KDE на вашому комп'ютері і якою перекладено інтерфейс і меню KTurtle.

3.1 Перші кроки у TurtleScript: привіт, Черепашко!

Ви, певне, вже побачили черепашку посередині полотна: зараз ми навчимося, як керувати нею за допомогою команд з редактора.

3.1.1 Рухи черепашки

Почнемо з того, як змусити черепашку рухатися. Наша черепашка може виконувати три типи рухів 3: (1) вона може пересуватися вперед або назад, (2) вона може повертати ліворуч і праворуч і, нарешті, (3) вона може йти напряму (перестрибувати) до точки на екрані. Спробуйте ввести такий приклад:

```
вперед 100
ліворуч 90
```

Щоб побачити результат, наберіть або скопіюйте цей код у редактор коду і виконайте його (для цього скористайтесь пунктом меню Запуск → Виконати).

Коли ви набираєте і виконуєте команди на зразок наведених вище у редакторі, ви можете помітити такі речі:

1. Після запуску команд на виконання черепашка починає рухатися вгору, малюючи лінію, а потім повертається на чверть оберту ліворуч. Причиною цього є використання команд вперед і ліворуч.
2. Колір коду змінюється під час набирання: це називається інтуїтивним підсвічуванням — команди різного типу підсвічуються різними кольорами. Таким чином спрощується читання великих блоків коду.
3. Черепашка малює товсту чорну лінію.
4. Можливо, ви побачите повідомлення про помилку. Причиною цього можуть стати дві речі: ви зробили помилку під час копіювання команд або ви ще не встановили відповідної мови команд TurtleScript (ви можете це зробити за допомогою підменю Параметри → Мова скриптів).

Скоріше за все, ви зрозуміли, що команда вперед 100 відповідає наказові черепашці рухатися вперед, малюючи лінію, а команда ліворуч 90 — наказові повернутися на 90 градусів ліворуч.

Будь ласка, прочитайте статті за наведеними нижче довідковими посиланнями для повного роз'яснення суті нових команд: вперед, назад, ліворуч і праворуч.

3.1.2 Додаткові приклади

Перший приклад був дуже простим, перейдемо до складніших!

```
перезапусти

розмір_полотна 200,200
колір_полотна 0,0,0
колір_пера 255,0,0
розмір_пера 5

перейди 20,20
напрямок 135

вперед 200
ліворуч 135
вперед 100
ліворуч 135
вперед 141
ліворуч 135
вперед 100
ліворуч 45

перейди 40,100
```

Щоб побачити результати, ви знову ж таки можете набрати або скопіювати код до редактора або відкрити приклад arrow з меню Приклади і виконати його (для цього ви скористайтесь пунктом підменю Запуск → Виконати). У наступних прикладах ви зможете отримати глибші знання про програмування.

Можливо, ви вже помітили, що у другому прикладі використано набагато більше коду. Також там є декілька нових команд. Ось коротеньке пояснення всіх нових команд:

Після команди перезапусти зображення стає точно таким самим, яким воно було, коли ви тільки-но запустили програму KTurtle.

розмір_полотна 200,200 встановлює ширину і висоту полотна у значення 200 пікселів. Оскільки висота дорівнює ширині, полотно буде квадратним.

колір_полотна 0,0,0 робить колір полотна чорним. 0,0,0 — це RGB-комбінація, у якій всі значення компонентів встановлено у 0, що дає чорний колір.

колір_пера 255,0,0 встановлює червоний колір пера. 255,0,0 — це RGB-комбінація, у якій лише значення червоного компонента встановлено у 255 (на повну силу), тоді як інші компоненти (зелена і синя) мають значення 0 (їх вимкнено). У результаті маємо яскравий червоний колір.

Якщо ви не розумієтеся на значеннях компонент кольорів, уважніше прочитайте статтю глосарія про RGB-комбінації.

розмір_пера 5 встановлює товщину (розмір) пера у значення 5 пікселів. З моменту встановлення цього значення всі лінії, які малюватиме черепашка, будуть мати товщину 5, до того часу, доки ми не змінимо значення розмір_пера.

перейди 20,20 наказує черепашці рухатися у певну точку на полотні. У цьому випадку відповідною точкою буде точка, розташована відносно верхнього лівого кута полотна у 20 пікселях від лівого краю полотна, і у 20 пікселях вниз від верхнього краю полотна. Зауважте, що під час виконання команди перейди черепашка не малює лінії.

напрямок 135 вказує черепашці напрям руху. Команди ліворуч і праворуч змінюють кут руху черепашки відносно поточного напрямку. Команда ж напрямок змінює кут руху черепашки від нульового напрямку, а, отже, її результат не залежить від попереднього напрямку руху черепашки.

За командою напрямок йде велика кількість команд вперед і ліворуч. Ці команди, власне, і призначені для малювання.

Наприкінці йде іще одна команда перейди, яку використано з метою відвести черепашку вбік від малюнка.

Спробуйте перейти за посиланнями на довідку з команд. У цій довідці кожну з команд докладно пояснено.

Розділ 4

Довідка з програмування на TurtleScript

Тут наведено довідник з мови KTurtle, TurtleScript. У першому розділі цієї глави ми поглянемо на деякі з аспектів граматики програм TurtleScript. Другий розділ присвячено математичним операторам, булевим (логічним) операторам і операторам порівняння. Третій розділ є величезним списком всіх команд з почерговим їх обговоренням. У четвертому розділі наведено пояснення щодо надання значень змінним. Нарешті, ми пояснимо як впорядкувати виконання команд за допомогою інструкцій регулювання виконання у п'ятому розділі і як створювати власні команди за допомогою команди вивчи у розділі під номером шість.

4.1 Граматика TurtleScript

Як і у будь-якій мові, у TurtleScript існують різні типи слів і символів. В українській ми відрізняємо дієслова (наприклад «йти» або «співати») та іменники (наприклад «сестра» або «будинок»). Ці слова використовують з різною метою. TurtleScript — це мова програмування, за її посередництвом ви можете повідомити KTurtle про потрібні вам дії.

У цьому розділі ми коротко зупинимося на деяких з різних типів слів та символів TurtleScript. Ми надамо пояснення щодо коментарів, команд та трьох інших типів буквених виразів: чисел, рядків і булевих (так/ні) значень.

4.1.1 Коментарі

Програма складається з інструкцій, які виконуються після запуску команди, і так званих коментарів. Коментарі не виконуються, KTurtle просто ігнорує їх під час виконання програми. Коментарі призначено для програмістів: вони допомагають їм краще розуміти програму. Все, що буде вказано після символу # у TurtleScript вважається коментарем. Ось приклад невеличкої програми, яка не виконує жодних дій:

```
# ця невеличка програма нічого не робить, вона складається лише з комента ↩
    ря!
```

Програма, хоч і зовсім непотрібна, але добре пояснює суть коментарів.

Коментарі є дуже корисними, якщо програма стає трохи складнішою. Коментарі є «порадником» для інших програмістів. У наведеній нижче програмі коментарі використано разом з командою напиши.

```
# Цю програму було створено Cies Breijs.
напиши "цей текст буде написано на полотні"
# попередній рядок не є коментарем, а наступний – коментар:
# напиши "цей текст не буде написано!"
```

У першому рядку наведено опис програми. Другий рядок буде виконано KTurtle: черепашка намалює напис цей текст буде написано на полотні на полотні. Третій рядок є коментарем. У четвертому рядку наведено коментар з фрагментом коду TurtleScript, якщо вилучити символ # з четвертого рядка, KTurtle виконає інструкцію з друку. Програмісти кажуть: інструкцію «напиши» у четвертому рядку було «закоментовано».

Закоментовані рядки підсвічуються світло-сірим кольором у редакторі коду.

4.1.2 Команди

За допомогою команди ви можете повідомити черепашці або KTurtle про те, що слід виконати певну дію. Для деяких команд потрібні вхідні дані, деякі команди самі виводять дані.

```
# для команди вперед потрібні вхідні дані, у нашому випадку ними є число ↩
    100:
вперед 100
```

Перший рядок програми є коментарем. У другому рядку міститься команда вперед і число 100. Число не є частиною команди, воно є «вхідними даними» команди.

Для роботи деяких команд, наприклад перейди, потрібно декілька вхідних значень. Ці декілька значень слід відокремлювати між собою символом , (комою).

Щоб прочитати детальний огляд всіх команд, які підтримує KTurtle, перейдіть за цим посиланням. Вбудовані команди буде підсвічено темно-синім.

4.1.3 Числа

Скоріше за все, ви вже знаєте дещо про числа. Спосіб, у який числа використовуються у KTurtle, не дуже відрізняється від розмовної мови або математики.

Існують так звані натуральні числа: 0, 1, 2, 3, 4, 5 тощо. Від'ємні числа: -1, -2, -3 тощо. Крім того, існують десяткові дроби або числа з крапкою: 0.1, 3.14, 33.3333, -5.05, -1.0. Символ . (крапка) є символом, що використовується для відокремлення дробової частини.

Числа можна використовувати у математичних операторах і операторах порівняння. Їх також можна зберігати у змінних. Числа буде підсвічено темно-червоним кольором.

4.1.4 Рядки

Спочатку наведемо приклад:

```
напиши "Привіт, я – рядок."
```

У цьому прикладі напиши — команда, а ˝Привіт, я — рядок.˝ — рядок. Рядки починаються і закінчуються позначкою лапок, ˝, за цими позначками і розпізнає рядки програма KTurtle.

Рядки можна зберігати у змінних, подібно до чисел. Але, на відміну від чисел, рядки не можна використовувати у математичних діях або діях з порівняння. Рядки буде підсвічено червоним кольором.

4.1.5 Булеві (так/ні) значення

Існує лише два булевих значення: так і ні. У цих значень є і інші назви: увімкнути і вимкнути, одиниця і нуль. Але у TurtleScript ми завжди називатимемо їх так і ні. Погляньте на цей фрагмент коду TurtleScript:

```
$a = так
```

Якщо ви зазирнете до вікна інспектора ви побачите, що значенням змінної $a є так, і що ця змінна належить до булевого типу.

Часто булеві значення є результатом обчислення оператора порівняння, наприклад, у такому фрагменті коду TurtleScript:

```
$відповідь = 10 > 3
```

Значенням змінної $відповідь буде так, оскільки 10 більше за 3.

Булеві значення, так і ні, буде підсвічено темно-червоним.

4.2 Математичні, булеві оператори та оператори порівняння

Заголовок цього розділу наче натякає на щось складне, але все не так складно, як здається.

4.2.1 Математичні оператори

Ось основні математичні символи: додавання (+), віднімання (-), множення (*), ділення (/) і піднесення до степеня (^).

Ось невеличкий приклад математичних операторів, якими ви можете скористатися у TurtleScript:

```
$add      = 1 + 1
$subtract = 20 - 5
$multiply = 15 * 2
$divide   = 30 / 30
$power    = 2 ^ 2
```

Результат виконання математичних операцій слід призначати різноманітним змінним. За допомогою інспектора ви зможете переглянути ці змінні.

Якщо вам потрібно виконати прості обчислення, ви можете зробити щось на зразок цього:

```
напиши 2010-12
```

А тепер приклад з дужками:

```
напиши ( ( 20 - 5 ) * 2 / 30 ) + 1
```

Спочатку буде обчислено вираз у дужках. У нашому випадку, буде обчислено 20-5, потім це значення буде помножено на 2, поділено на 30, і додано до нього 1 (у результаті отримаємо 2). Дужками можна скористатися і у інших випадках.

У KTurtle передбачено і додаткові математичні можливості у формі команд. Нижче наведено список команд, але без означення способу, у який вони працюють: округли, випадкове, корінь, пі, sin, cos, tg, arcsin, arccos, arctg.

4.2.2 Булеві (так/ні) оператори

Тоді як математичні оператори в основному призначено для чисел, булеві оператори призначено для булевих значень (так і ні). Передбачено лише три булеві оператори, а саме: та, або і ні. У наведеному нижче фрагменті TurtleScript показано використання всіх цих операторів:

```
$та_1_1 = так та так    # -> так
$та_1_0 = так та ні   # -> ні
$та_0_1 = ні та так   # -> ні
$та_0_0 = ні та ні  # -> ні

$або_1_1 = так або так    # -> так
$або_1_0 = так або ні   # -> так
$або_0_1 = ні або так   # -> так
$або_0_0 = ні або ні  # -> ні

$не_1 = не так   # -> ні
$не_0 = не ні  # -> так
```

За допомогою інспектора ви зможете переглянути значення, крім того, ми навели результати виконання дій у коментарях. Результат застосування та дорівнює так, лише якщо з обох боків від нього стоять значення так. Результатом застосування або буде так, якщо з одного з боків від оператора стоїть так. Оператор не перетворює так на ні, а ні на так.

Булеві оператори буде підсвічено рожевим.

4.2.2.1 Декілька додаткових прикладів

Розглянемо наведений нижче приклад з та:

```
$a = 1
$b = 5
якщо (($a < 10) та ($b == 5)) та ($a < $b) {
  напиши "привіт"
}
```

У цьому фрагменті TurtleScript результат застосування трьох операторів порівняння буде об'єднано за допомогою операторів та. Це означає, що для того, щоб черепашка написала «привіт», потрібно, щоб всі дії з порівняння давали результат «так».

Приклад з або:

```
$n = 1
якщо ($n < 10) або ($n == 2) {
  напиши "привіт"
}
```

У цьому фрагменті коду TurtleScript частина, розташована ліворуч від або, дає «так», а частина, розташована праворуч, — «ні». Оскільки одна з двох частин виразу у операторі або дорівнює «так», результатом обчислення або буде «так». Це означає, що черепашка напише «привіт».

І нарешті, приклад з не, у якому ми змінними «так» на «ні», а «ні» — на «так». Дивіться:

```
$n = 1
якщо не ($n == 3) {
  напиши "привіт"
} інакше {
  напиши "не привіт ;-)"
}
```

4.2.3 Оператори порівняння

Розглянемо таке елементарне порівняння:

```
$answer = 10 > 3
```

Тут ми порівнюємо 10 з 3 за допомогою оператора «більше, ніж». Результатом цього порівняння буде булеве значення так, яке буде збережено у змінній $answer.

Всі числа і змінні (які містять числа) можна порівнювати за допомогою операторів порівняння.

Ось список можливих операторів порівняння:

$A == $B	дорівнює	відповіддю буде «так», якщо $A рівне $B
$A != $B	не дорівнює	відповіддю буде «так», якщо $A не дорівнює $B
$A > $B	більше, ніж	відповіддю буде «так», якщо $A більше, ніж $B
$A < $B	менше, ніж	відповіддю буде «так», якщо $A менше, ніж $B
$A >= $B	більше або дорівнює	відповіддю буде «так», якщо $A більше або дорівнює $B
$A <= $B	менше або дорівнює	відповіддю буде «так», якщо $A менше або дорівнює $B

Табл. 4.1: Типи питань

Будь ласка, зауважте, що $A і $B мають бути числами або змінними, які містять числа.

4.3 Команди

За допомогою команд ви наказуєте черепашці або програмі KTurtle виконати якусь дію. Деяким з команд потрібні вхідні дані, деякі — самі виводять дані. У цьому розділі буде пояснено всі вбудовані команди, які можна використовувати у KTurtle. Крім того, ви можете скористатися командою вивчи, щоб створити власні команди. Вбудовані команди, які ми тут обговорюємо, підсвічуються темно-синім кольором.

4.3.1 Пересування черепашки

Існує декілька команд для пересування черепашки екраном.

вперед (вп)

```
вперед X
```

вперед наказує черепашці просунутися вперед на X пікселів. Якщо перо опущено, черепашка залишатиме по собі слід у вигляді лінії. Команду вперед можна скоротити до вп.

назад (нд)

```
назад X
```

Команда назад наказує черепашці просунутися назад на X пікселів. Якщо перо опущено, черепашка залишатиме по собі слід у вигляді лінії. Команду назад можна скоротити до нд

ліворуч (лв)

```
ліворуч X
```

Команда ліворуч наказує черепашці повернутися на X градусів ліворуч. Команду ліворуч можна скоротити до лв.

праворуч (пр)

```
праворуч X
```

Команда праворуч наказує черепашці повернутися на X градусів праворуч. Команду праворуч можна скоротити до пв.

напрямок (нпр)

```
напрямок X
```

Команда напрямок спрямовує черепашку у напрямку X градусів, відрахованих від нульового напрямку, безвідносно від попереднього напрямку, куди її було спрямовано. Команду напрямок можна скоротити до нпр.

візьми_напрям

```
візьми_напрям
```

Команда візьми_напрям повертає напрямок черепашки у форматі градусів, відрахованих від нульового напрямку, де нулеві відповідає напрямок вгору.

центр

```
центр
```

Команда центр пересуває черепашку у центр полотна.

перейди

```
перейди X,Y
```

Команда перейди наказує черепашці перейти до вказаного місця на полотні. Це місце знаходиться у X пікселях від лівого краю полотна, і у Y пікселях від верхнього краю полотна.

перейди_x

```
перейди_x X
```

Команда перейди_x наказує черепашці перейти до точки, розміщеної за X пікселів від лівого краю полотна, залишаючись на тій самій висоті. Команду перейди_x можна скоротити до пх.

перейди_y

```
перейди_y Y
```

Команда перейди_y наказує черепашці перейти до точки, розміщеної за Y пікселів від верхнього краю полотна, залишаючись на тій же відстані від його лівого краю. Команду перейди_y можна скоротити до пу.

> **Примітка**
> Під час виконання команд перейди, перейди_x, перейди_y і центр черепашка не малюватиме лінії, незалежно від того, піднято чи опущено перо.

4.3.2 Де знаходиться черепашка?

Існує дві команди, які повертають позицію черепашки на екрані.

візьми_x

візьми_x повертає кількість пікселів від лівого краю полотна до поточної позиції черепашки.

візьми_y

візьми_y повертає кількість пікселів від верхнього краю полотна до поточної позиції черепашки.

4.3.3 У черепашки є перо

У черепашки є перо, яке малює лінію під час руху черепашки. Існує декілька команд, які керують цим пером. У цьому розділі пояснено зміст цих команд.

підніми_перо (пп)

```
підніми_перо
```

Команда підніми_перо піднімає перо над полотном. Якщо перо «піднято» під час руху черепашки лінії не малюватиметься. Прочитайте також інформацію щодо команди опусти_перо. Команду підніми_перо можна скоротити до ри.

опусти_перо (оп)

```
опусти_перо
```

Команда опусти_перо опускає перо на полотно. Якщо перо «опущено» на полотно, під час рухів, черепашка лишатиме по собі слід. Прочитайте також інформацію щодо команди підніми_перо. Команду опусти_перо можна скоротити до оп.

розмір_пера (рп)

```
розмір_пера X
```

Команда розмір_пера встановлює товщину лінії, яку лишає перо, у значення X пікселів. Команду розмір_пера можна скоротити до рп.

колір_пера (кп)

```
колір_пера R,G,B
```

Команда колір_пера встановлює колір лінії, яку лишає по собі перо. Команда колір_пера отримує дані для кольору у вигляді RGB-комбінації. Команду колір_пера можна скоротити до кп.

4.3.4 Команди керування полотном

Існує декілька команд, які надають можливість керувати полотном.

розмір_полотна (рпл)

```
розмір_полотна X,Y
```

За допомогою команди розмір_полотна ви можете встановити розмір полотна. Ця команда отримує вхідні значення X і Y, де X — нова ширина полотна у пікселях, а Y — нова висота полотна у пікселях. Команду розмір_полотна можна скоротити до рпл.

колір_полотна (кпл)

```
колір_полотна R,G,B
```

Команда колір_полотна встановлює колір тла полотна. Команда колір_полотна отримує вхідні дані у вигляді RGB-комбінації. Команду колір_полотна можна скоротити до кпл.

4.3.5 Команди для очищення

Існує дві команди для очищення полотна, у разі якщо ви зробили помилку.

зітри (зтр)

```
зітри
```

За допомогою команди зітри ви можете витерти всі малюнки з полотна. При цьому всі інші параметри, позиція і напрямок черепашки, колір тла полотна, видимість черепашки і розмір полотна, залишаться незмінними:

перезапусти

```
перезапусти
```

Команда перезапусти очищує все набагато ґрунтовніше за команду зітри. Після виконання команди перезапусти все виглядатиме так, неначе ви тільки-но запустили KTurtle. Черепашку буде розташовано по центру екрана, колір тла стане білим, черепашка малюватиме чорну лінію на полотні, а саме полотно матиме розмір 400 на 400 пікселів.

4.3.6 Черепашка — це спрайт

Спочатку, коротеньке пояснення того, що таке спрайти: спрайти — це маленькі зображення, які можна пересувати екраном, які ми часто бачимо у комп'ютерних іграх. Наша черепашка — також спрайт. Докладнішу інформацію ви зможете отримати з розділу глосарія, присвяченого спрайтам.

Далі, ви зможете ознайомитися з повним списком всіх команд для роботи зі спрайтами.

> **Примітка**
> Поточна версія KTurtle ще не підтримує використання інших спрайтів, окрім черепашки. У майбутніх версіях ви зможете змінити черепашку на щось інше, що вам сподобається.

покажи_черепашку (пч)

```
покажи_черепашку
```

Команда покажи_черепашку робить черепашку знову видимою після того, як її було сховано. Команду покажи_черепашку можна скоротити до пч.

сховай_черепашку (сч)

```
сховай_черепашку
```

Команда сховай_черепашку ховає черепашку. Цією командою можна скористатися, якщо черепашка не вписується у ваш малюнок. Команду сховай_черепашку можна скоротити до сч.

4.3.7 Чи може черепашка писати?

Відповідь: «так». Черепашка може писати: вона напише для вас все, що ви їй накажете написати.

напиши

```
напиши X
```

Команда напиши використовується для того, щоб наказати черепашці написати щось на полотні. Команда напиши отримує числа і рядки як вхідні дані. Ви можете писати комбінації з чисел і рядків за допомогою символу «+». Ось подивіться на цей приклад:

```
$рік = 2003
$автор = "Сіес"
напиши $автор + " почав роботу над проєктом KTurtle у " + $year + "ро ↩
    ці, і все ще отримує задоволення від роботи над ним!"
```

розмір_літер

```
розмір_літер X
```

Команда розмір_літер встановлює розмір шрифту, який використовуватиме команда напиши. Команда розмір_літер отримує одне вхідне значення, ним має бути число. Розмір встановлюється у пікселях.

4.3.8 Математичні команди

Наведені нижче команди є додатковими математичними командами KTurtle.

округли

```
округли(x)
```

округлює задане число до найближчого цілого.

```
напиши округли(10.8)
вперед 20
напиши округли(10.3)
```

Виконуючи цей код, черепашка напише числа 11 і 10.

випадкове (вип)

```
випадкове X,Y
```

Команда випадкове отримує вхідні дані і виводить вихідні. Як вхідні дані їй потрібні два числа, перше (X) вказує на мінімальне значення числа, яке можна вивести, а друге (Y) — встановлює максимальне значення цього числа. У результаті дії команди буде виведено випадково вибране число, яке буде більше або рівне за мінімальне значення і менше або рівне за максимальне значення. Ось невеликий приклад:

```
повтори 500 {
  $x = випадкове 1,20
  вперед $x
  ліворуч 10 - $x
}
```

За допомогою команди випадкове ви можете ввести елемент випадковості у вашу програму.

mod

```
mod X,Y
```

Команда mod повертає залишок від ділення першого числа на друге.

корінь

```
корінь X
```

Команду корінь призначено для обчислення квадратного кореня з числа, X.

пі

```
пі
```

Ця команда повертає сталу π, 3.14159.

sin, cos, tg

```
sin X
cos X
tg X
```

За допомогою цих команд задаються три відомі тригонометричні функції sin, cos і tg. Вхідним параметром для всіх цих команд, X, є число.

arcsin, arccos, arctg

```
arcsin X
arccos X
arctg X
```

Ці команди є оберненими функціями sin, cos і tg. Вхідним параметром для всіх цих команд, X, є число.

4.3.9 Введення і отримання інформації за допомогою діалогових вікон

Діалогове вікно — це невеличке контекстне вікно, у якому розміщено деяку інформацію від програми, або у яке слід ввести деяку інформацію, потрібну програмі. У KTurtle є дві команди для виклику діалогових вікон, а саме: повідом і спитай

повідом

```
повідом X
```

Команда повідом отримує як вхідні дані рядок. У результаті виконання команди буде показано контекстне вікно з текстом, який визначає вхідний рядок.

```
повідом "Сієс почав роботу над проєктом KTurtle у 2003 році, і все ще ↩
    отримує задоволення від роботи над ним!"
```

спитай

```
спитай X
```

Команда спитай отримує вхідні дані у вигляді рядка. У результаті виконання команди буде показано діалогове вікно, у якому міститиметься текст з рядка, точно так само як і у команді повідом. Але, окрім цього, у вікні буде і поле для введенні інформації. У це поле введення користувач може ввести число або рядок, який можна зберегти у змінній або передано, як параметр, команді. Приклад:

```
$питання = спитай "Скільки вам років?"
$роки = 2003 - $in
напиши "У 2003 році вам було " + $роки + " років."
```

Якщо користувач закриває діалогове вікно введення або нічого не вводить у поле для введення, змінній надається порожнє значення.

4.4 Присвоювання змінним значень

Спочатку ми розглянемо змінні, а потім поговоримо про те, як надати цим змінним значень.

Змінними є слова, що починаються на «$», у редакторі ці слова підсвічуються пурпуровим кольором.

Змінні можуть містити будь-які числа, рядки або булеві значення. За допомогою оператора присвоювання, =, ви можете надати змінній її значення. Змінна зберігатиме свій вміст до завершення виконання програми або до зміни значення змінної на якесь інше.

Після надання значень ви можете використовувати змінні так, неначебто вони збігаються з власним вмістом. Ось приклад з фрагментом коду TurtleScript:

```
$x = 10
$x = $x / 3
напиши $x
```

Спочатку змінній $x надано значення 10. Потім $x надано значення цієї ж змінної, поділеної на 3 — тобто $x надано значення частки 10 / 3. Нарешті, $x буде намальовано черепашкою. Ви можете бачити, що у рядках два і три $x використано замість вмісту змінної.

Для того, щоб змінними можна було користуватися, цим змінним слід надати значення. Приклад:

```
напиши $n
```

Результатом виконання буде повідомлення про помилку.

Будь ласка, погляньте на цей приклад TurtleScript:

```
$a = 2004
$b = 25

# наступна команда намалює рядок "2029"
напиши $a + $b
назад 30
# а ця команда надрукує "2004 плюс 25 дорівнює 2029":
напиши $a + " плюс " + $b + " дорівнює " + ($a + $b)
```

У перших двох рядках встановлено значення змінних $a і $b 2004 і 25. Далі йдуть дві команди напиши з командою назад 30 між ними. Коментарі перед командами напиши пояснюють призначення цих команд. Команду назад 30 додано для того, щоб забезпечити вивід кожної з порцій даних у окремому рядку. Як бачите змінними можна користуватися як замінниками їх вмісту, ви можете використовувати змінні з будь-якими операторами або передавати їх як вхідні дані під час викликукоманд.

Ще один приклад:

```
$ім'я = спитай "Як вас звати?"
напиши "Привіт, " + $ім'я + "! Успіхів у вивченні мистецтва програмування ↩
    ..."
```

Досить зрозуміло. Знову ж таки, змінну $ім'я використано як рядок.

Інспектор буде дуже корисним під час роботи зі змінними. Цей інструмент покаже вам значення всіх використаних у програмі змінних.

4.5 Контроль над виконанням

Регулятори виконання надають вам змогу — як це і випливає з їх назви — керувати виконанням програми.

Команди керування виконанням підсвічуються темно-зеленим кольором і виокремлюються жирним шрифтом. З регуляторами виконання використовують і фігурні дужки, які підсвічуються чорним кольором і виокремлюються чорним шрифтом.

4.5.1 Черепашко, зачекай!

Якщо ви вже повправлялися у програмуванні у KTurtle, ви, мабуть зауважили, що черепашка малює дуже швидко. Ця команда примушує черепашку почекати деякий проміжок часу.

чекай

```
чекай X
```

Команда чекай наказує черепашці чекати X секунд.

```
повтори 36 {
  вперед 5
  праворуч 10
  чекай 0.5
}
```

За допомогою цього коду малюється коло, але черепашка чекатиме по пів секунди перш ніж перейти до наступного кроку. Це створює враження черепашки, що рухається повільно.

4.5.2 Виконання «якщо»

якщо

```
якщо булеве значення { ... }
```

Код між двома фігурними дужками буде виконано, лише якщо (якщо) результат обчислення булевого значення дорівнюватиме «так».

```
$x = 6
якщо $x > 5 {
  напиши "$x більше за п'ять!"
}
```

У першому рядку змінній $x надається значення 6. У другому рядку використано оператор порівняння для знаходження значення $x > 5$. Оскільки відповіддю на це питання є «так» регулятор виконання якщо дозволить виконання коду між фігурними дужками.

4.5.3 Якщо «ні», значить «інакше»

інакше

```
якщо булеве значення { ... } інакше { ... }
```

Команду інакше можна використовувати як додаток до регулятора виконання якщо. код між фігурними дужками після інакше виконуватиметься, лише якщо булеве значення дорівнюватиме «ні».

```
перезапусти
$x = 4
якщо $x > 5 {
  напиши "$x більше за п'ять!"
} інакше {
  напиши "$x менше за шість!"
}
```

Оператор порівняння обчислює значення $x > 5. Оскільки 4 не перевищує 5 значенням буде «ні». Це означає, що буде виконано код між фігурними дужками після команди інакше.

4.5.4 Цикл «поки»

поки

```
поки булеве значення { ... }
```

Регулятор виконання поки багато у чому схожий на якщо. Різниця між ними полягає у тому, що поки продовжує виконання коду між фігурними дужками (цикл), аж доки булевий вираз не дорівнюватиме «ні».

```
$x = 1
поки $x < 5 {
  вперед 10
  чекай 1
  $x = $x + 1
}
```

У першому рядку змінній $x надано значення 1. У другому рядку виконано порівняння $x < 5. Оскільки відповіддю на це питання буде «так» регулятор виконання поки починатиме виконання коду між фігурними дужками з початку, до того часу, коли значенням $x < 5 не стане «ні». У нашому випадку, код між фігурними дужками буде виконано чотири рази, оскільки кожного разу, коли виконується п'ятий рядок $x збільшується на 1.

4.5.5 Цикл «повтори»

повтори

```
повтори число { ... }
```

Регулятор виконання повтори дуже схожий на поки. Різниця полягає у тому, що повтори повторює виконання команд (цикл) задану кількість разів.

4.5.6 Цикл з лічильником, «для»

для

```
для змінна = число до число { ... }
```

Цикл для — це «цикл з лічильником», тобто він лічить повтори за вас. Перше число вказує значення змінної під час першого виконання циклу. Під час кожного виконання це число збільшується, аж доки не досягне другого числа.

```
для $x = 1 до 10 {
  напиши $x * 7
  вперед 15
}
```

Кожне виконання коду між фігурними дужками збільшує значення $x на 1, до того часу, коли $x досягне значення 10. Код між фігурними дужками пише на полотні значення $x помноженого на 7. Після виконання програми ви побачите на полотні таблицю множення на 7.

Типовим розміром кроку циклу є 1, але ви можете використовувати інше значення за допомогою команди на зразок

```
для змінна = число до число крок число { ... }
```

4.5.7 Полишити цикл

перерви

```
перерви
```

Негайно припиняє виконання програми у рамках циклу і передає керування команді, яку розташовано одразу за поточним циклом.

4.5.8 Перервати виконання вашої програми

зупини

```
зупини
```

Завершує виконання вашої програми.

4.5.9 Перевірка умови під час виконання програми

перевір

```
перевір булеве значення
```

Може бути використано для перевірки правильності роботи програми або введених даних.

```
$введення = ask "У якому році ви народилися?"
# рік має бути вказано як додатне значення
перевір $введення
> 0
```

4.6 Створення власних команд за допомогою «вивчи»

вивчи є дуже особливою командою, оскільки цю команду використовують для створення нових команд. Команда, яку ви створюєте за її допомогою може отримувати вхідні дані і повертати вихідні дані. Давайте поглянемо як створюється нова команда:

```
вивчи circle $x {
  повтори 36 {
    вперед $x
    ліворуч 10
  }
}
```

Нова команда називається circle. Команда circle отримує одне вхідне значення, параметр, який задає розмір кола. circle не повертає ніяких вихідних даних. Тепер у решті коду можна використовувати команду circle як звичайну команду. Ось приклад:

```
вивчи коло $X {
  повтори 36 {
    вперед $X
    ліворуч 10
  }
}

перейди 200,200
коло 20

перейди 300,200
коло 40
```

У наступному прикладі буде створено команду, яка повертає значення.

```
вивчи факторіал $x {
  $r = 1
  для $i = 1 до $x {
    $r = $r * $i
  }
  поверни $r
}

напиши факторіал 5
```

У цьому прикладі створено нову команду з назвою faculty. Якщо вхідними даними цієї команди буде число 5, буде повернуто 5*4*3*2*1. З допомогою команди поверни можна вказати вихідне значення і повернути його програмі.

Команди можуть приймати і по декілька вхідних значень. У наступному прикладі буде створено команду, яка малює прямокутник.

```
вивчи прямокутник $x, $y {
  вперед $y
  праворуч 90
  вперед $x
  праворуч 90
  вперед $y
  праворуч 90
  вперед $x
  праворуч 90
}
```

Тепер ви можете виконати команду прямокутник 50, 100, і черепашка намалює на полотні прямокутник.

Розділ 5

Глосарій

У цьому розділі ви знайдете пояснення більшості «незрозумілих» слів, які використовуються у цьому довіднику.

градуси

Градуси — це одиниці виміру кутів або степеня обертання. Повне обертання містить 360 градусів, половина обертання — 180 градусів, а чвертина обертання — 90 градусів. Для команд ліворуч, праворуч і напрямок потрібно ввести кут у градусах.

вхідні і вихідні дані команд

Деякі з команд отримують вхідні дані, деякі — виводять вихідні дані, деякі команди отримують вхідні дані і виводять вихідні, а деякі команди не отримують ніяких даних і нічого не виводять.

Ось приклади команд, які лише отримують дані:

```
вперед 50
колір_пера 255,0,0
напиши "Привіт!"
```

Команда вперед отримує вхідні дані 50. Команді вперед ці дані потрібні для визначення кількості пікселів, які черепашці слід пройти вперед. pencolor отримує вхідні дані про колір, а напиши отримує рядок (шматочок тексту). Будь ласка, зауважте, що вхідні дані можуть також бути контейнером. Цю можливість показано у наступному прикладі:

```
$x = 50
напиши $x
вперед 50
$рядок = "привіт!"
напиши $рядок
```

Тепер декілька прикладів команд, які виводять дані:

```
$x = спитай "Будь ласка, наберіть щось і натисніть кнопку Гаразд... Д ↩
    якуємо!"
$r = випадкове 1,100
```

Команда спитай отримує рядок як вхідні дані і виводить число або рядок, який було введено. Очевидно, у цьому випадку вихідні дані команди спитай збережено у контейнері x. Команда випадкове також виводить дані. У нашому випадку вона виводить випадкове число від 1 до 100. Виведені дані, знову ж таки, збережено у контейнері з назвою r. Зауважте, що контейнери x та r не використано у коді попереднього прикладу.

Існують і команди, які нічого не отримують і нічого не виводять. Ось декілька прикладів:

```
зітри
підніми_перо
```

інтуїтивне підсвічування

Це можливість у KTurtle, як ще більше спрощує процес програмування. Інтуїтивне підсвічування надає кодові, який ви пишете, кольорів, що відповідають типу цього коду. У наведеному нижче списку ви знайдете типи коду і кольори, які відповідають цим типам у вікні редактора.

звичайні команди	темно-синій	Звичайні команди описано тут.
команди-регулятори виконання	чорний (жирний шрифт)	Особливі команди, які регулюють виконання, докладніше дізнатися про ці команди можна тут.
коментарі	сірий	Рядки з коментарями починаються з символу коментування (#). Під час виконання коду ці рядки ігноруються. Коментарі використовуються програмістами для пояснення суті коду або для тимчасової заборони на виконання певного коду.
дужки {, }	темно-зелений (жирний шрифт)	Дужки використовують для групування частин коду. Дужки часто використовують разом з регуляторами виконання.
команда вивчи	світло-зелений (жирний шрифт)	Команда вивчи використовується для створення нових команд.
рядки	червоний	Про рядки (текст) можна сказати небагато, за винятком того, що їх завжди слід починати і завершувати подвійними лапками (″).
числа	темно-червоний	Числа, ну що тут ще сказати.
булеві значення	темно-червоний	Існує точно два булевих значення, а саме: так і ні.
змінні	пурпуровий	Починаються з «$» і можуть містити числа, рядки або булеві значення.

математичні оператори	сірий	Приклади математичних операторів: +, -, *, / і ^.
оператори порівняння	світло-синій (жирний шрифт)	Приклади операторів порівняння: ==, !=, <, >, <= і >=.
булеві оператори	рожевий (жирний шрифт)	Приклади булевих операторів: та, або і не.
звичайний текст	чорний	

Табл. 5.1: Типи кодових рядків і кольори їх підсвічування

пікселі

Піксель — це точка на екрані. Якщо ви зблизька подивитеся на зображення на екрані вашого монітора, ви побачите, що зображення складається з точок-пікселів. З цих пікселів складаються і всі зображення на екрані. Загалом кажучи, піксель — це найменша частинка, яку можна зобразити на екрані.

Велика кількість команд потребують вхідних даних у вигляді кількості пікселів. Цими командами є: вперед, назад, перейди, перейди_x, перейди_y, розмір_полотна і розмір_пера.

У ранніх версіях KTurtle полотно було растровим зображенням, у сучасних версіях програми полотно — це векторний малюнок. Це означає, що полотно можна збільшувати або зменшувати, оскільки один піксель не обов'язково відповідає точно одній точці на екрані.

RGB-комбінації (коди кольорів)

Для опису кольорів використовують RGB-комбінації. Літера «R» позначає «червоний (red)», літера «G» — «зелений (green)», а «B» — «синій (blue)». Прикладом RGB-комбінації може слугувати 255,0,0: перше значення («червоного») рівне 255, а інші рівні 0, отже, ця комбінація відповідає яскравому червоному кольору. Кожна величина RGB-комбінації повинна бути цілим числом у межах між 0 і 255. Ось невеликий перелік деяких розповсюджених кольорів:

0,0,0	чорний
255,255,255	білий
255,0,0	червоний
150,0,0	темно-червоний
0,255,0	зелений
0,0,255	синій
0,255,255	блакитний
255,0,255	рожевий
255,255,0	жовтий

Табл. 5.2: Розповсюджені RGB-комбінації

Вхідні дані у форматі RGB-комбінації використовують дві команди: canvascolor і pencolor.

спрайт

Спрайт — це маленьке зображення, яке можна пересувати екраном. Наприклад, наша улюбленица-черепашка — спрайт.

> **Примітка**
> У цій версії KTurtle неможливо змінити спрайт черепашки на щось інше. У майбутніх версіях KTurtle ви матимете цю можливість.

Розділ 6

Інструкція для перекладача KTurtle

Ймовірно, вам вже відомо, що мову програмування KTurtle, TurtleScript, теж можна перекласти. Таким чином можна усунути мовний бар'єр для деяких, особливо молодших учнів, у їх спробах оволодіти мистецтвом програмування.

Якщо ви перекладатимете KTurtle новою мовою, ви виявите, що окрім рядків графічного інтерфейсу, до стандартного файла .pot, який використовується для перекладу KDE, включено команди програмування, приклади і повідомлення про помилки. Все це можна перекласти за допомогою звичайного способу перекладу, який ви використовуєте у KDE, але ми дуже радимо вам ознайомитися з принципами перекладу програми (а також з коментарями для перекладачів).

Будь ласка, прочитайте інформацію, розміщену за адресою https://edu.kde.org/kturtle/translation.php, щоб дізнатися більше про процес перекладу. Велике вам спасибі за вашу роботу! Просування KTurtle залежить від ваших перекладів.

Розділ 7

Подяки і ліцензія

KTurtle

Авторські права на програмне забезпечення належать Cies Breijs cies AT kde DOT nl, 2003–2007

Авторські права на документацію (2004, 2007, 2009)

- Cies Breijs cies AT kde DOT nl
- Anne-Marie Mahfouf annma@kde.org
- Деякі зміни коректурного характеру — Philip Rodrigues phil@kde.org
- Оновлення довідника з перекладу і деякі коректурні виправлення — Andrew Coles andrew_coles AT yahoo DOT co DOT uk

Переклад українською: Юрій Чорноіван yurchor@ukr.net

Цей документ поширюється за умов дотримання GNU Free Documentation License.

Ця програма поширюється за умов дотримання GNU General Public License.

Розділ 8

Предметний покажчик