www.wioowszkole.org

Karol Sieńkowski

KTurtle

Wstęp do programowania w języku TurtleScript

Publikacja powstała w ramach projektu
Wolne i Otwarte Oprogramowanie w Szkole

KTurtle

Wstęp do programowania
w języku TurtleScript

Spis treści

Wstęp

Program Kturtle to jeden z wielu programów KDE dla edukacji. Działa on w oparciu o KurtleScript, który wywodzi się z rodziny języków programowania Logo. Pierwsza wersja Logo została stworzona w 1967. Od tamtej chwili powstało bardzo wiele odmian Logo. W Polsce Logo dosyć mocno zakorzeniło się w edukacji na poziomie szkoły podstawowej i gimnazjum.

W większości szkół używane jest oprogramowanie komercyjne. Są to przede wszystkim Logo Komeniusz i Logomocja Imagine. Nie wszystkie szkoły stać na zakup licencji. Czy musi to oznaczać rezygnację z nauki programowania w Logo? Nie. KTurtle jest dostępny za darmo (na licencji GPL 2), więc każdy może go legalnie używać w domu. Potrzebna jest do tego jedynie dystrybucja systemu Linux. System można zainstalować na dysku, lub użyć jakiejś edukacyjnej dystrybucji LiveCD z zainstalowanym programem Kturtle. Na pewno jeszcze rok temu program ten był dostępny z dystrybucją Opensuse Education-Li-v-e. Jednak czy nadal jest on dostępny w trybie Live-cd w tej dystrybucji- nie wiem.

W programie KTurtle stawia się na wygodę użytkowania programu z pełną świadomością tego, co użytkownik w danej chwili robi. Program uczy dobrych nawyków przy pisaniu komend, co procentuje w późniejszym czasie podczas nauki programowania w innych językach. Ci, którzy chcą ćwiczyć język angielski mogą próbować programować ruch żółwia w tym właśnie języku. Jeśli język angielski jest zbyt dużą przeszkodą, to język komend można zmienić na polski. Ale uwaga! Polecenia w języku polskim są nieco inne niż w popularnym w polskich szkołach Logo Komeniusz, czy Logomocji.

Cechy KTurtle

KTurtle posiada kilka cech, które sprawiają, że doskonale nadaje się do rozpoczęcia przygody z programowaniem:

- Zintegrowane środowisko graficzne z językiem TurtleScript.
- Polecenia TurtleScript dostępne są w różnych językach.
- TurtleScript wspiera zdefiniowane przez użytkownika funkcje i rekurencję.
- Program może być spowolniony, wstrzymany lub zatrzymany w dowolnym momencie.
- Edytor wyposażony jest w intuicyjne kolorowaniem składni, numerowanie wierszy itp.
- Płótno, na którym żółw rysuje, można wydrukować lub zapisać jako obraz (PNG) lub rysunek (SVG).
- Dostępna jest pomoc do programu (niestety na razie tylko w języku angielskim).
- Program posiada okno błędów, które łączy komunikaty błędów z błędami w programie i zaznacza je na czerwono.
- Do programu załączone są przykładowe pliki z kodem, które pomagają rozpocząć przygodę z KTurtle

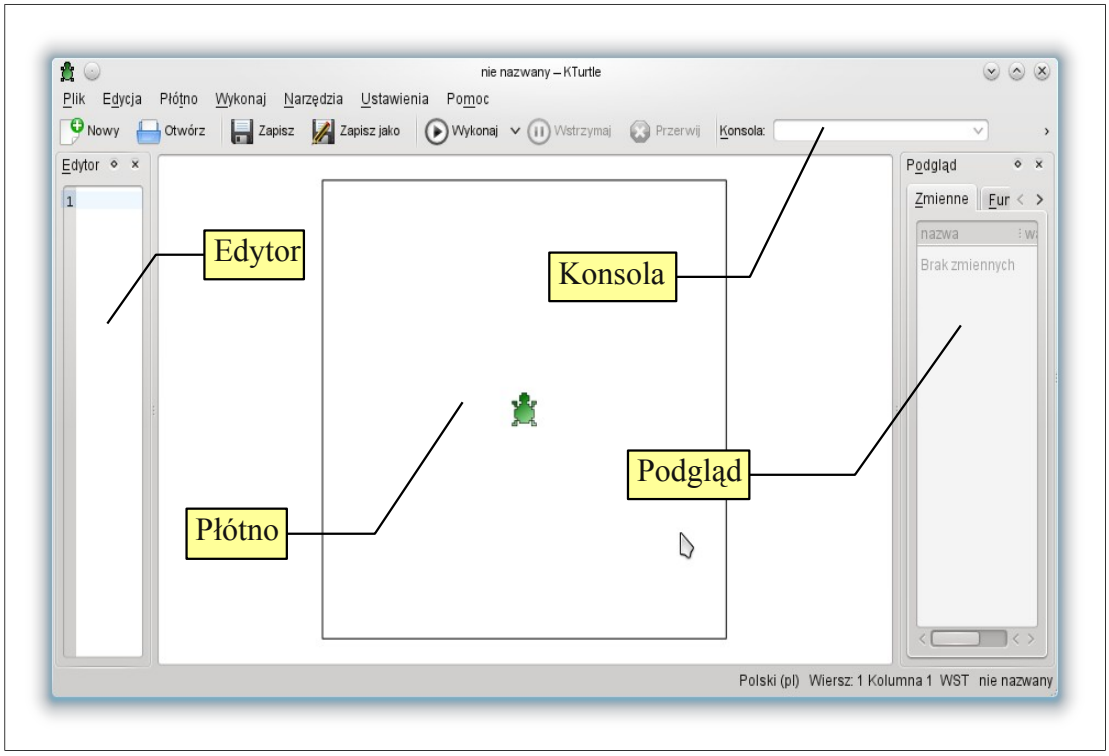
Układ KTurtle

Okno KTurtle składa się z trzech głównych części:

- **edytora** – po lewej, gdzie wpisuje się polecenia TurtleScript, wczytuje je z pliku, albo wkleja wcześniej skopiowane,
- **płótna** – po środku, gdzie rysuje żółw. Płótno może być powiększane i pomniejszane ruchem kółka w myszce,
- **podglądu –** wyświetlającego informację w czasie działania programu. W tym miejscu można podejrzeć zdefiniowane funkcje i zmienne, zobaczyć drzewo programu.

Oprócz nich program posiada:

- **pasek menu**, skąd mogą być wybrane wszystkie akcje,
- **pasek narzędzi**, który pozwala na szybki wybór najczęściej używanych akcji,
- **pasek stanu** (wzdłuż dolnej krawędzi okna), gdzie wyświetlany jest aktualny stan działania KTurtle.



Ilustracja 1: Okno programu KTurtle i jego elementy

Konsola

Na pasku narzędzi znajduje się Konsola (*Ilustracja 1*). W tym miejscu można testować jednowierszowe plecenie. Jest to bardzo użyteczna opcja. Można jej użyć, gdy nie jest się pewnym efektu działania jakiejś funkcji. Polecenie wpisane w Konsoli zatwierdza się klawiszem **Enter** lub klikając przycisk **Wykonaj** znajdujący się kolo niej.

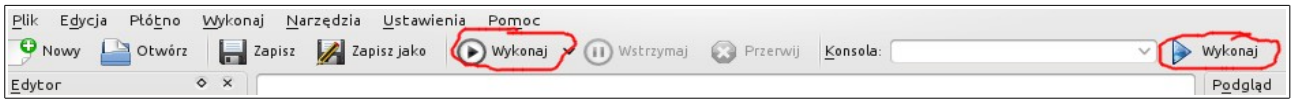
Edytor

W edytorze wprowadza się polecenia TurtleScript. Większość funkcji edytora można znaleźć w menu ***Plik*** i ***Edycja***. Do edytora można wczytać zapisany wcześniej plik, wkopiować tekst programu np. ze strony internetowej, w końcu samodzielnie wpisać polecenia. Edytor może być dowolnie przesuwany, przyczepiany do innego miejsca na oknie głównym, a także może być umieszczony w dowolnym miejscu na pulpicie. Pozwala to dobrze i wygodnie zaplanować pracę.

Kod napisany w edytorze nie jest wykonywany dopóki nie klikniesz przycisku **Wykonaj**, lub nie naciśniesz klawisza funkcyjnego **F5**.

Uwaga!

Na pasku narzędzi znajdują się dwa przyciski **Wykonaj**. Jeden służy do uruchamiania kodu wpisanego w edytorze, a drugi do wykonania komendy wpisanej w **Konsoli** (*Ilustracja 2*).



Ilustracja 2: Pasek narzędzi zawiera dwa przyciski Wykonaj. Każdy służy do czego innego.

Nim zaczniesz

Aby praca przebiegała komfortowo, warto wykonać na początku kilka czynności:

- Ustal tryb pełnoekranowy programu **KTurtle**. W tym celu dokonaj maksymalizacji okna programu.
- Zanim zaczniesz pisać swój pierwszy program musisz nauczyć żółwia rozumienia polskich komend. W tym celu wybierz menu **Ustawienia->Język->Polski(pl)**.
- Warto również zwiększyć szerokość okna **Edytora** do 5-6 cm. Zapewni to większy komfort przy pisaniu kodu.
- Przy bardziej skomplikowanych programach warto jest używać komentarzy, które pomogą w orientacji, co dana część kodu robi. Komentarze to tekst nie brany przez żółwia pod uwagę. Każdy komentarz poprzedza znak # (szczegóły w przykładach zawartych poniżej)

Pierwsze spotkanie z żółwiem

Wpisz w edytorze poniższy kod:

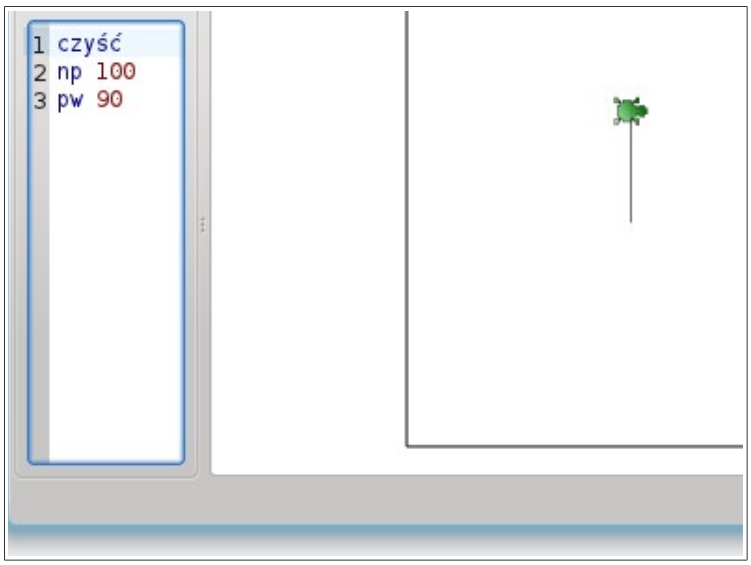
```
czyść
np 100
pw 90
```

Przykład 1: Kod na początek

Po wpisaniu tego kodu w edytorze zauważyłeś, że kolor teksu całego wpisu nie jest jednakowy. Inny kolor ma polecenie **czyść**, inny **pw**, a inny liczby *100* i *90*. W edytorze różne typy komend zaznaczane są innym kolorem. To sprawia, że kod łatwiej jest przeczytać.

Oto co może wydarzyć się po wpisaniu kodu z *Przykładu 1* i naciśnięciu przycisku **Wykonaj** (lub klawisza **F5**):

- Wyświetlono komunikat o błędzie. To może oznaczać dwie rzeczy: zrobiłeś błąd podczas wpisania poleceń, albo nie zmieniłeś języka na język polski.
- żółw przesuwa się w górę, rysuje linię, a następnie wykonuje ćwierć obrotu w lewo (*Ilustracja 3*).



Ilustracja 3: Efekt wykonania kodu z Przykładu 1. Przy okazji widać kolorowanie składni w Edytorze

Pewnie domyślasz się dlaczego tak postąpił żółw. Żółw trzyma pisak i poruszając się zostawia ślad. Wpisane komendy mają swoje znaczenie.

Spróbujmy je rozszyfrować:

czyść – wyczyszczenie płótna

np *100* – żółw poszedł 100 kroków do przodu

lw *90* – żółw obrócił się w lewo o 90 stopni

Teraz coś bardziej skomplikowanego:

```
czyść
ustalRozmiar 200,200
ustalKolTła 0,0,0
ustalKolPis 255,0,0
ustalGrubość 5

idź 50,70
ustalKierunek 135

naprzód 100
lewo 135
naprzód 100
lewo 135
naprzód 100
lewo 135
naprzód 100
lewo 135
naprzód 100
lewo 135
naprzód 100
lewo 135
naprzód 100
lewo 135
naprzód 100

idź 100,100
ustalKierunek 0
```

Przykład 2: Coś trudniejszego

Wyjaśnijmy te wszystkie polecenia:

czyść – czyści płótno

UstalRozmiar *200,200* – ustawia szerokość i wysokość płótna w pikselach. W tym przypadku oba parametry są jednakowe, więc płótno będzie miało kształt kwadratowy

ustalKolTła *0,0,0* – ustala kolor tła w systemie RGB. Tło będzie czarne

ustalKolPis *255,0,0* – ustala kolor pisaka w systemie RGB. Kolor będzie czerwony

ustalGrubość *5* – ustala grubość pisaka w pikselach

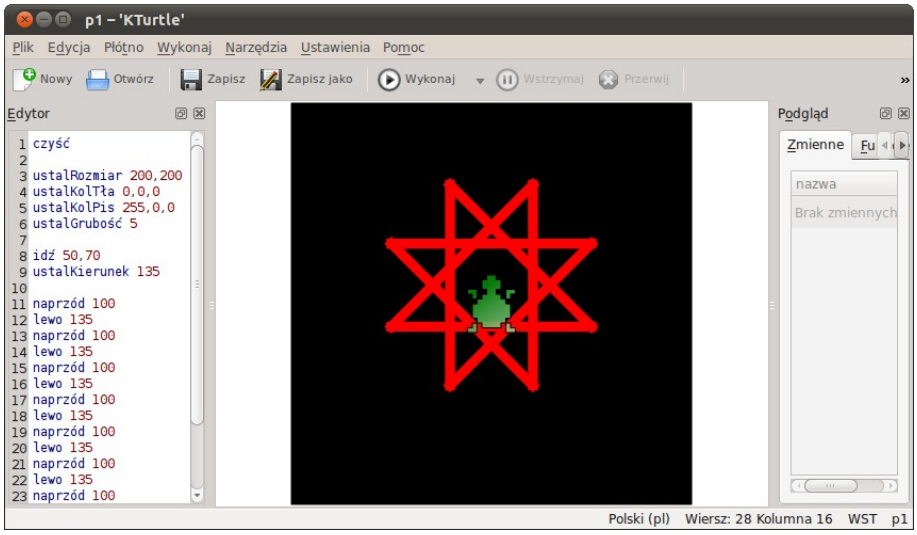
idź *20,20* – przenosi żółwia w określone miejsce na płótnie liczone od lewego górnego rogu. To polecenie nie rysuje linii

ustalKierunek *135* – ustalenie kierunku żółwia o dany kąt względem położenia początkowego

naprzód *200*
lewo *135*
naprzód *100*
lewo *135*
naprzód *141*
lewo *135*
naprzód *100*
lewo *45*

Polecenia sterujące ruchem żółwia

idź *40,100* – to już zostało wyjaśnione wcześniej



Ilustracja 4: Efekt wykonania poleceń z Przykładu 2

Ważne jest by na początku podjąć próbę szczegółowej analizy tego powyższego przykładu krok po korku. Nie jest on może zbyt prosty, ale zawiera najważniejsze elementy na początek zabawy żółwiową grafiką. Już po pobieżnym przyjrzeniu się poleceniom wpisanym w Edytorze w oczy rzuca się powtarzalność komend **naprzód** *100* i **lewo** *135*. Jest to dość uciążliwe, ale na początku niezbędne. W późniejszym etapie ich wielokrotne wpisywanie można zastąpić instrukcją pętli.

Jeśli po tym krótkim wstępie spodobało Ci się rysowanie za pomocą żółwia, to możesz przejść do kolejnego rozdziału. Poznacz polecenia, które pozwolą Ci tworzyć jeszcze bardziej zaawansowane kształty, a nawet zmusić żółwia do wykonania obliczeń i do rozmowy z Tobą.

Polecenia i składnia Języka TurtleScript

Nazwa poleceń TurtleScript w języku polskim jest dość intuicyjna. Ich opanowanie nie sprawi trudności nawet młodszym dzieciom. Trzeba tu jednak zaznaczyć, że polecenia te (choć nie wszystkie) mają inne brzmienie, niż w Logo Komeniusz czy Logomocja Imagine. Niektóre z poleceń można pisać całym słowem, choć w praktyce szybsze i wygodniejsze okazuje się używanie skrótów. Zdecydowana większość komend wymaga podania jednego lub kilku parametrów. W przypadku koloru (który zapisywany jest w schemacie **RGB)** lub kąta obrotu żółwia, w początkowym etapie zabawy z KTurtle mogą pojawić się problemy. Pomóc może sięgnięcie do menu **Narzędzia**, które podpowiada zarówno kod wybranego koloru jak i kąt obrotu.

Polecenia dotyczące płótna

ustalRozmiar *x, y* – ustala rozmiar tablicy (przykład **ustalRozmiar** *200, 400*, gdzie 200– szerokość, 400–wysokość)

ustalKolTła *r, g, b* – ustala kolor tła w formacie r,g,b (przykład **ustalKolTła** *255,50,40*)

czyść – przywraca stan początkowy tablicy (przywraca kolor, rozmiar, czyści tablicę, po której rysuje żółw i ustawia żółwia w położeniu początkowym

zmaż – wymazanie zawartości tablicy, żółw pozostaje na aktualnym miejscu

Polecenia sterujące zachowaniem żółwia

Polecenia z tej grupy są dwojakiego rodzaju. Jedne nie wymagają podania żadnego parametru, a innym typie poleceń należy podać jeden lub dwa parametr, gdyż w przeciwnym razie otrzymamy błąd składni. Polecenia te brzmią następująco:

pokaż – pokazuje żółwia

ukryj – ukrywa żółwia

środek – ustawia żółwia na środku płótna

czekaj *czas* **–** żółw czeka w tym miejscu zadany czas w sekundach (przykład **czekaj** *4*)

np *krok* (lub **naprzód**) – ruch naprzód (przykład **np** *50*)

ws *krok* (lub **wstecz**) – ruch wstecz (przykład **ws** *30*)

pw *kąt* (lub **prawo**) – obrót żółwia w prawo o podany kąt (przykład **pw** *70*)

lw *kąt* (lub **lewo**) – obrót żółwia w w lewo o podany kąt (przykład **lw** *90*)

ustalKierunek *kąt* – ustala kierunek zwrotu żółwia względem położenia początkowego (przykład **stalKierunek** *90* – żółw będzie zwrócony w prawą stronę)

Czasem zachodzi konieczność przeniesienia żółwia w inne miejsce na płótnie. Można tu oczywiście zastosować polecenie podnoszące pisak i dalej kombinacją poleceń obracających i przesuwających żółwia umieścić go tam, gdzie potrzeba. Jest to jednak pracochłonne i nie zawsze precyzyjne. Dlatego warto stosować polecenia przemieszczające żółwia w konkretne miejsce na tablicy. Określa się je odległością od górnej i lewej jej krawędzi płótna w pikselach. Polecenia przemieszczające żółwia bez zostawiania śladu na płótnie mają następujące brzmienie:

idź *x, y* – żółw idzie do współrzędnych określonych przez dwie liczby x i y. Punkt (0,0) znajduje się w prawym lewym rogu (przykład **idź *120, 40)***

idźx x – żółw idzie do podanej współrzędnej x . Druga współrzędna **y** nie zmienia się

idźy y – żółw idzie do podanej współrzędnej y. Druga współrzędna ***x*** nie zmienia się

Polecenia zwracające pozycję żółwia

W niektórych sytuacjach trzeba ustalić dokładną pozycję, w jakiej znajduje się żółw. Przydaje się to zwłaszcza przy tworzeniu bardziej skomplikowanych rysunków. Istnieją dwa polecenia, które pozwalają odczytać współrzędne żółwia na płótnie.

pozx – zwraca pozycję żółwia od lewej krawędzi płótna

pozy – zwraca pozycję żółwia do górnej krawędzi płótna

Polecenia związane z pisakiem żółwia

Można powiedzieć, że żółw przemieszczający się po płótnie trzyma w ręku pisak. Pisak ten ma możliwość zostawiania śladu za poruszającym się żółwiem. Ślad zostawiany na płótnie może mieć różny wygląd. W ograniczony sposób można na niego wpływać za pomocą kilku poleceń. Pozwolą one zdefiniować kolor piska, jego grubość oraz ustalić, czy żółw ma rysować ślad.

ustalKolPis *r, g, b* – ustala kolor pisaka w formacie r,g,b (przykład **ustalKolPis** *255,40,120*)

ustalGrubość *grubość* – ustala grubość pisaka w pikselach (przykład **ustalGrubość** *3*)

pod (lub **podnieś)** – podniesienie pisaka

opu (lub **opuść)** – opuszczenie pisaka

Polecenia wstrzymujące wykonanie programu

W niektórych przypadkach istnieje konieczność wymuszenia zakończenia wykonywanego przez żółwia kodu. Ma to zastosowanie zwłaszcza w przypadku tzw. pętli rekurencyjnych. Może się zdarzyć sytuacja, że pętla może się wykonywać w nieskończoność. Aby takich sytuacji uniknąć można posłużyć się poniższymi poleceniami.

przerwij – przerywa wykonywanie pętli i przechodzi do kolejnej instrukcji poza nią

wyjdź – przerywa wykonywanie programu

Wypisanie informacji na tablicy

Czasem zachodzi potrzeba wypisania na tablicy wyniku działania matematycznego, albo innego tekstu. Może to być np. aktualna pozycja żółwia, czy też odpowiedź na pytanie. TurtleScript posiada dwie komendy służące do tego celu.

rozmiarTekstu *x* – ustala rozmiar czcionki tekstu w pikselach

pisz *"tekst"* **–** wypisuje na tablicy tekst

Przykład:

pisz *((20 - 5) * 2 / 30) + 1* **–** wypisuje na tablicy wynik wyrażenia matematycznego

$i=3

pisz *"Ala ma "+$i+ " lata"* – wypisze na tablicy tekst *„Ala ma 3 lata”*

Definiowanie zmiennych

W programie TurtleScript mogą występować zmienne 3 typów: liczbowe, znakowe i logiczne. Zmienne poprzedzamy znakiem **$ (**przykład: **$x**)

W przypadku zmiennych liczbowych za zmienną mogą być wstawione liczby. Gdy zmienna jest typu znakowego, to może nią być np. wyraz, lub ciąg znaków. Ciąg znaków ujmuje się w znaki " ", np. *"krowa"*. Gdy zmienna jest typu logicznego, to przyjmuje ona jedną z dwóch wartość **prawda** lub **fałsz**.

Przykład:

$x –deklaracja zmiennej **x**

$wynik *10>3*) **–** wykonanie tej komendy przypisze zmiennej **wynik** wartość **prawda**, bo prawdą jest, że 10>3

$imię = *"Ala"* **–** wpisanie tej komendy spowoduje podstawienie w miejsce zmiennej **imię** ciągu znaków *„Ala”*

Liczby, operatory matematyczne, porównania ...

W Kturtle można używać liczb naturalnych (*0, 1, 2, 3...*), całkowitych (*-1, -2, -3...*) oraz ułamków dziesiętnych pisanych z kropką (*0.2, 0.34, 0.5, ...*)

Dozwolone i rozpoznawane operatory logiczne to: **i** oraz **lub** (przykład **jeśli** *(a==1)* **lub** *(b==2)*)

Dozwolone operatory matematyczne to : **+ , – , / , * , < , > , <= , >=**

Inne polecenia związane z liczbami:

losowa *x, y* – generuje liczbę losową z przedziału liczbowego ograniczonego przez podane minimum i maksimum (przykład: **losowa** *3, 8* wygeneruje liczbę losową z zakresu od *3* do *8*, przy czym liczby graniczne przedziału, czyli *3* i *8* też mogą być wylosowane)

zaokrąglij *liczba* – zaokrągli liczbę do jedności (przykład: użycie polecenia **zaokrąglij** *10.3* da w wyniku liczbę *10*)

sqrt *liczba* – zwraca pierwiastek z *liczba*

pi – zwraca wartość liczby *pi*

mod *x, y* – zwraca resztę z dzielenia liczby *x* przez liczbę *y*.

sin *x,* **cos** *x,* **tg** *x* , **arcsin** *x,* **arccos** *x,* **arctg** *x* – zwracają wartości funkcji trygonometrycznych z liczby *x*

Porównywanie liczb, zmiennych i wartości (przykłady):

$a==$b – sprawdzenie, czy zmienne **a** i **b** są sobie równe
$a!=$b – sprawdzenie, czy zmienne **a** i **b** są różne
$a>$b – sprawdzenie, czy zmienna **a** jest większa od **b**
$a<=$b – sprawdzenie, czy zmienna **a** jest mniejsza lub równa zmiennej **b**

Instrukcje sterujące

Instrukcje sterujące pozwalają stawiać żółwiowi warunki i od nich uzależniać jego działanie. Najprostszą instrukcją tego rodzaju jest instrukcja warunkowa. Nieco bardziej złożone są tzw. pętle, które pozwalają wykonać grupę instrukcji kilka razy. Instrukcje sterujące stanowią ważny element przy planowaniu ruchów i zachowania żółwia. Jeśli ma być spełnionych kilka warunków instrukcji warunkowej, to każdy z nich ujmuje się w nawiasy ( ) np. **jeśli** (**$a**= =*3*) i (**$b**= =*5*) {**pisz** *"Dobrze"*}

.

Instrukcja warunkowa

Instrukcja warunkowa to jedna z najważniejszych instrukcji stosowanych w programowaniu. Ważne jest, by ją właściwie rozumieć. W naszym życiu bardzo często stosujemy instrukcję warunkową nie zdając sobie nawet z tego sprawy.

Przykład:

Adam wybiera się do szkoły odległej o 4 km od jego miejsca zamieszkania. W związku z tym co rano używa instrukcji warunkowej, która ma wpływ na jego dalsze działania. Jego wybór może wyglądać następująco:

Jeśli mam pieniądze to

- kupię bilet
- wsiądę do autobusu
- pojadę do szkoły
- pójdę na lekcje

Jeśli nie mam pieniędzy to

- pójdę piechota do szkoły
- pójdę na lekcje”

Ten prosty, banalny przykład pokazuje jak instrukcje warunkowe kształtują nasze codzienne życie. W języku zrozumiałym dla komputera zapisalibyśmy rozumowanie Adama w sposób:

jeśli kwota_pieniędzy>=2zł
{ kupię bilet, wsiądę do autobusu
pojadę do szkoły, pójdę na lekcje }
jeśliNie
{ pójdę piechota do szkoły, pójdę na lekcje }

W języku TurtleScript instrukcję warunkową zapisuje się w polskim brzmieniu w następujący sposób:

jeśli *warunek* {istrukcje1} **jeśliNie** {istrukcje2} – jeśli warunek jest prawdziwy, to wykonają się **instrukcje1**, jeśli nie jest prawdziwy, to wykonają się **instrukcje2**

Przykład:

jeśli *$a==3* { **pisz** *"Ala ma kota”* }

jeśli *$x<0* { **pisz** *"Liczba jest ujemna”* } **jeśliNIe** { **pisz** *„Liczba jest dodatnia”* }

Pętla powtórz

Ta pętla pozwala wykonać instrukcje z góry zadaną ilość razy. Dobrze nadaje się ona np. do rysowania wielokątów foremnych. Składnia pętli jest następująca:

powtórz *liczba* {instrukcje} – to polecenie pozwala wykonać daną instrukcję lub kilka z nich *liczba* razy

Przykład:

powtórz *4* **{np** *30* **pw** *90* **}** – efektem wykonania tej pętli będzie kwadrat o boku równym 30 pikseli

Pętla dla

Pętla tego rodzaju jest tzw. pętlą zliczającą. Wśród instrukcji pętli zazwyczaj znajduje się licznik, który zwiększa zmienną do określonej wartości. Po jej osiągnięciu pętla przestaje się wykonywać. Pętlę zapisujemy w ten sposób:

dla *zmienna=liczba1* **do** *liczba2* **co** *krok* {instrukcje} – wykonanie pętli zależy od zmiennej, która przyjmuje wartości od *liczba1* do *liczba2*

Przykład:

Zapiszmy w edytorze KTurtle poniższy kod.

```
dla $x = 1 do 10
{
pisz $x * 7
 np 15
}
```

W efekcie zostaną wypisane obok siebie liczby *7, 14, 21, 28, 35, 42, 49, 56, 63, 70*

Pętla dopóki

Instrukcje tej pętli wykonują się aż do momentu, gdy warunek przestanie być prawdziwy. Wynika stąd, że wśród tych instrukcji musi być polecenie, które zmienia warunek podczas każdego wykonania pętli. Składnia pętli jest następująca:

dopóki *warunek* {instrukcje} – pętla jest wykonywana dopóki warunek jest prawdziwy

Przykład:

```
$x = 1
dopóki $x < 5
{
naprzód 10
czekaj 1
$x = $x + 1
}
```

W efekcie żółw narysuje w czterech etapach poprzedzonych sekundową przerwą odcinek o długości 40 pikseli

Nauka komend

Czasem może się zdarzyć sytuacja, że tworząc pracę należy za pomocą żółwia narywać wiele takich samych elementów (np. stworzenie na płótnie rysunku domu wymaga narysowania kilku prostokątów o różnych wymiarach: okna, drzwi, ściana przednia, komin itp.). Jest to nieco uciążliwe, ale można sobie trochę pomóc. Żółwia można nauczyć wykonywania pewnych komend. A więc w przypadku rysunku domu możemy nauczyć żółwia rysowania prostokąta. W tym celu musimy utworzyć nową funkcję. Używamy do tego polecenia

poznaj nazwa *zmienna1, zmienna2*. . . .

Dla przykładu jeśli chcemy go nauczyć rysować kwadrat, to możemy to zdefiniować funkcję jego rysowania na dwa sposoby:

Sposób 1
Jest to sposób łatwiejszy

```
Poznaj kwadrat $bok
{
np $bok pw 90
np $bok pw 90
np $bok pw 90
np $bok pw 90
}
```

Sposób 2
Jest to sposób trudniejszy, ale bardziej efektywny

```
poznaj kwadrat $bok
{
  powtórz 4 { np. $bok pw 90 }
}
```

Narysowanie kwadratu osiągamy w obu przypadkach wydając polecenie np. **kwadrat** *100*. Żółw narysuje wówczas kwadrat o boku 100 pikseli.

Komunikacja z użytkownikiem

Żółw może się z nami w ograniczony sposób komunikować, np. zadając nam pytanie, na które udzielamy mu odpowiedzi. Dzięki temu może np. zwracać się do nas po imieniu, ocenić naszą wiedzę zadając nam pytania itp. W zasadzie da się to osiągnąć za pomocą dwóch poleceń.

spytaj *"pytanie"* – powoduje wyświetlenie okna z pytaniem, na które udzielamy. Ta komenda może służyć np. do zadawania pytań w teście
wiadomość *"tekst"* – powoduje wyświetlenie okna z wiadomością *tekst*. Ta komenda może służyć np. do wyświetlenia wyniku testu.

Przykład:

$imie = **spyta**j *"Jak się nazywasz?"* – spowoduje to przypisanie zmiennej **$imię** wprowadzonego z klawiatury imienia. Dzięki temu możliwe jest np. wyświetlenie okna z napisem „Witaj Ania”
wiadomość "Witaj "+**$imie** – jako kontynuacja powyższego przykładu spowoduje wyświetlenie okna z informacją „Witaj Ania”

Dodatek

A. Podstawowe kolory RGB

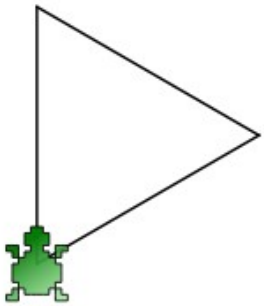
W wyborze koloru jak również kąta obrotu żółwia mogą pomóc odpowiednie opcje z menu **Narzędzia**.

Kod koloru RGB	Nazwa koloru	Kolor
0,0,0	czarny	
255,255,255	biały	
255,0,0	czerwony	
150,0,0	brązowy	
0,255,0	zielony	
0,0,255	niebieski	
0,255,255	jasnoniebieski	
255,0,255	różowy	
255,255,0	żółty	

B. Przykłady

1. Kwadrat

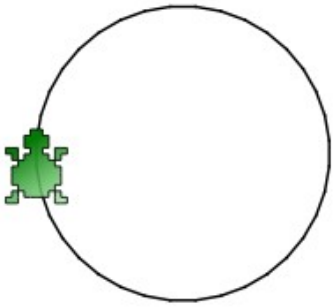
```
# funkcja rysująca trójkąt równoboczny
# $bok – długość boku kwadratu

poznaj trójkąt $bok
  {
   powtórz 3{ np $bok pw 120}
  }
```



Przykład wywołania funkcji: **trójkąt** *100*

2. Koło

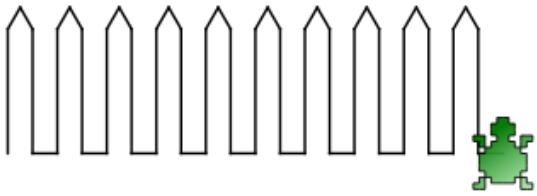
```
# rysowanie 36-kąta o boku  długości $bok
# który przypomina koło

poznaj koło $bok
  {
  powtórz 36 {np $bok pw 10}
 }
```



Przykład wywołania funkcji **koło** *10*

3. Płot

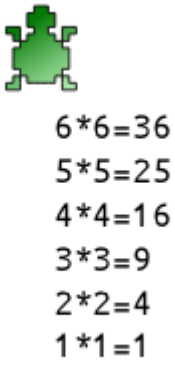
```
# funkcja rysująca płot
# $bok - długość boku
# $ile – liczba szczebelków
poznaj płot  $bok, $ile
 {
  powtórz $ile
   {
     np $bok pw 30 np 10
     pw 120 np 10 pw 30 np $bok
     lw 90 np 10 lw 90
   }
}
```



Przykład wywołania funkcji: **płot** *50,10*

4. Wypisanie kwadratów liczb

```
# wypisanie kwadratów n liczb

poznaj kwadraty $n
{ # ustalenie początkowej pozycji żółwia
 czyść
 pod
 ws 150

 # wypisanie kwadratów liczb
 dla $i=1 do $n
  {
   pisz $i+"*"+$i+"="+ $i^2
   np 15
  }
}
```



Przykład wywołania funkcji: **kwadraty 6**

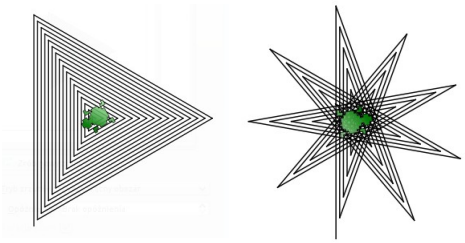
5. Prosty test

```
#sprawdzenie sprawności obliczeń

czyść
$x= spytaj "Ile to 3*4?"
jeśli $x==12
  {wiadomość "Bardzo dobrze!"}
jeśliNie
  {pisz "Odpowiedziałeś źle!!!"}

#ukrycie żółwia, by był widoczny sam napis
ukryj

#odczekanie 2 sekund
czekaj 2

#wyczyszczenie ekranu i pokazanie żółwia
czyść
pokaż
```

6. Rekurencyjne obrazki

Rekurencja polega na wywołaniu funkcji w niej samej. W ten sposób można osiągnąć skomplikowane i ciekawe kształty geometryczne.

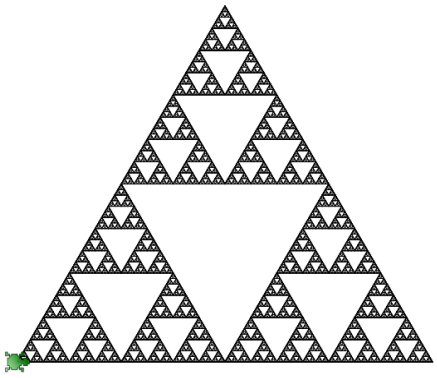
```
poznaj spirala $bok, $kąt
{
jeśli $bok < 2 {wyjdź}
np $bok pw $kąt
spirala $bok – 4, $kąt
}
```



Przykłady wywołania funkcji : **spirala** *200, 120,* **spirala** *200, 160.* Dla każdego wywołania otrzymamy inny obrazek

7. Trójkąt Sierpińskiego

```
#funkcja rysująca dywan Sierpińskiego
poznaj sierpiński $bok
{
 jeśli $l > 2
  {
   powtórz 3
    {
     sierpiński $l/2
     np $l
    lw 120
    }
  }
}

# przygotowanie płótna
czyść
ustalRozmiar 600, 600
idź 50, 500
pw 90
```



Przykład wywołania funkcji: **sierpiński** *500*