Wstęp

Program komputerowy to ciąg zrozumiałych dla komputera poleceń, każdy program jest napisany w jakimś języku programowania (np. C#, C++, Batch :), Java, Basic, PHP i wiele, wiele innych). Każdy z tych języków zawiera specyficzną dla siebie składnie
np. krótki programik w c++ :

```
#include <iostream>
#include <conio.h>
using namespace std;

int main()
{
 cout << "Hello world!";
 return 0;
}
```

(Szczerze to ten program to nie robi zbyt wiele bo tylko wywświetla napis: Hello world!)
Ten i inne programy programiści piszą w kompilatorze no ale własnie co to jest kompilator ?
kompilator jest to tzw. translator (tłumacz).
Kompilator wczytuje cały kod programu, dokonuje jego optymalizacji i tłumaczy na kod wynikowy.
W rezultacie otrzymujemy program wynikowy gotowy w każdej chwili do wykonania.
Wynik kompilacji jest zapamiętywany w postaci pliku wykonywalnego skompilowanie programu umożliwia jego wykonanie w dowolnym momencie na dowolnym komputerze.
Kompilator jest nam potrzebny tylko raz (podczas kompilacji)(np. pliki *.cpp na *.exe).

Drugim rodzajem translatora jest:
interpretator - owy rodzaj translatora tłumaczy kod programu i go od razu wykonuje nie zapisując go do pliku *.exe. Czyli musimy go używać do każdego wykonania programu.

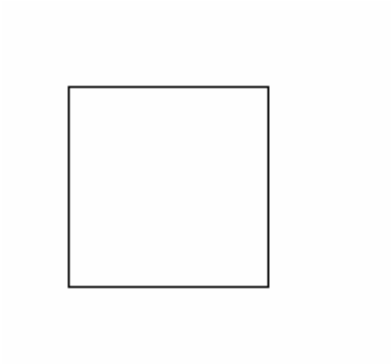
Do programowania w logo będziemy używać Logo Komeniusza.
Logo Komeniusz jest interpretatorem i świetnie nadaje się do pracy w logo.
Zaczniemy od prostych poleceń później przejdziemy do procedu.

Opis programu :

Podstawy

Podstawowe komendy:
np X - naprzód o X pikseli
pw X - obrót o X stopni w prawo
lw X - obrót o X stopni w lewo
pod- podnosi żółwia, dzięki czemu możemy się przesuwać po ekranie nie pozostawiając za sobą śladu.
opu - opuszcza za sobą żółwia, który już pozostawia ślad.
ścier - żółw porusza się ścierając swoje ślady.
ustalgrubośćpisaka
powtórz X [xxxxx] - powtarza x razy jakieś komendy

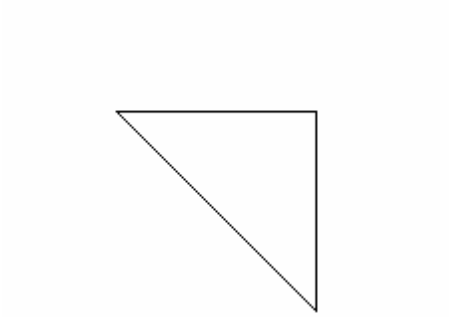
W Logo Komeniuszu kierujemy "Żółwiem" wydając mu polecenia np. :
? np 100 lw 90
? np 100 lw 90
? np 100 lw 90
? np 100 lw 90
co żółw z tym fantem zrobi ?
pójdzie 100 pikseli na przód potem obróci się o 90 ` w lewo itd.
czyli wykona kwadrat :



No ale to trochę męczące jest kiedy musimy pisać tyle poleceń, więc pokarze jak to trochę streścić zamiast 4 linii komend możemy napisać jedną:

? powtórz 4 [np 100 lw 90]

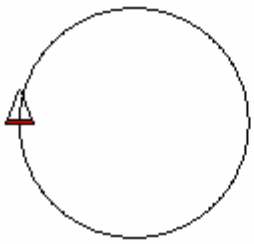
powtórz 4 - żółwiowi wydajemy polecenie żeby powtórzył 4 razy wartości w nawiasie kwadratowym. Mianowicie posunie się o 100 pikseli na przód 90` w lewo itd.
Uwaga! trzeba pamiętać o nawiasie i o liczbie przed inaczej nic nie wyjdzie z polecenia.
Czyli aby rysować figury w języku logo trzeba po prostu znać podstawy Matematyki
znaczy ,musimy wiedzieć że każdy bok w kwadracie jest taki sam a suma jego kątów wewnętrznych wynosi 360` i że każdy kąt jest kątem prostym (90`)
Trójkąt prostokątny :
? pw 90
? np 100
? pw 90
? np 100
? pw 135
? np 140



Skąd wiem o ile stopni się obracać, a no wiem że suma kątów wewnętrznych trójkąta wynosi 180 stopni i jeżeli jest to trójkąt prostokątny to ma 1 kąt prosty (90`),
więc z prostego odejmowania i dzieleni a będę wiedział ile mają pozostałem kąty (180` - 90`=90`; 90`:2= 45`)
Czyli pozostałe kąty mają po 45 `. Jak obliczyć ile ma mieć najdłuższy bok trójkąta prostokątnego ?Myślę że do tego trza było by się posłużyć twierdzeniem Pitagorasa a
to jest w klasie 2 gim. Więc nie będę tłumaczył całego twierdzenia. Na początek polecam posuwanie się o długość przyprostokątnych (krótszych jego boków)
np. kiedy mamy krótsze boki po 100 pikseli to na początek dajemy te 100 a potem co 10 pikseli posuwamy się do przodu.

Koło

No to już mamy większy problem ponieważ w Logo raczej nie ma okrągłych figur, ale wiadomo są one nie zbędne więc żółwia trzeba oszukać.
Wiemy ,ze koło ma 360` więc musimy to wykorzystać, mianowicie kazać żółwiowi posuwać się o 1pixel na przód i obracać o 1 stopień w lewo lub prawo, tylko jak
w takim razie będzie wyglądać polecenie, o na przykład tak:
? powtórz 360 [np 1 lw 1]
czego efektem będzie :



Procedury

Pokazałem wam jak tworzyć figury i jak skracać sobie czas na wypisywaniu komend (powtórz X [X X X X]) ale można jeszcze bardziej umilić sobie życie, ucząc żółwia jak rysować poszczególne figury na przykład kiedy mu wydamy polecenia
? powtórz 4 [np 200 lw 90]
wykona figurę zwana kwadratem. My o tym wiemy, więc teoretycznie moglibyśmy wpisać w wierszu poleceń nazwę dowolnej figury
może niech to będzie po prostu kwadrat. No niestety wyskoczył nam błąd, żółw nie wie co to jest kwadrat, prostokąt, trójkąt, koło itd. Ale możemy żółwia tego wszystkiego nauczyć pisząc procedury. Procedura może wywoływać funkcje, zmienne i inne procedury, ale za to, funkcje i zmienne nie mogą wywoływać procedury. A więc PROCEDURA to zbiór funkcji wywoływanych przez 1 polecenie.

Pisanie procedury możemy podzielić na 3 części
a) Wskazanie żółwiowi słowa na które ma reagować (odtworzyć zawarte w procedurze polecenia), figury obrazy itp.
Takie słowo wskazuje się takim poleceniem:
oto *kwadrat*
a potem piszemy słowo na które ma reagować żółw
b) W drugiej części, wpisujemy komendy które ma wykonać żółw po wskazaniu przez nas słowa. np. :
? powtórz 4 [np 100 lw 90]
c) Ostatnie jest najłatwiejsze ponieważ zawsze zawiera jedno słowo które brzmi : już
i obwieszcza żółwiowi koniec nauki.

Czyli procedura może wyglądać tak :
? oto kwadrat
? powtórz 4 [np 100 pw 90]
? już
Czyli streszczając to nadajemy nazwę procedurze używają polecenia "oto" (kwadrat:), wpisujemy komendy które ma wykonać żółw po "usłyszeniu" słowa kluczowego :)
i kończymy nauczanie żółwia słowem: "już". Teraz możemy się spodziewać że żółw po " usłyszeniu" słowa kwadrat narysuje nam go.

Aby stworzyć procedurę kwadratu, z możliwością szybkiej zmiany długości boków piszemy coś takiego :

Oto kwadrat :długość_boków
Powtórz 4 [np :długość_boków pw 90]
Już

Gdzie dwukropek to zmienna która będzie wstawiać wszędzie liczbę którą wpiszemy do okienka które się wyświetli po wywołaniu procedury, ‘’długość_boków ‘’ to nazwa która się wyświetli obok miejsca do wpisania. Zmiennymi mogą być długości wybranych boków, kąty, wysokości itd.
Zmienne można stosować wtedy gdy mamy bardzo złożone figury, ale głównie w tedy gdy efektem pracy żółwia ma nie być tylko figura o jednym rozmiarze, ale figury o rozmaitych rozmiarach.

Przykładowe okno zmiennej :

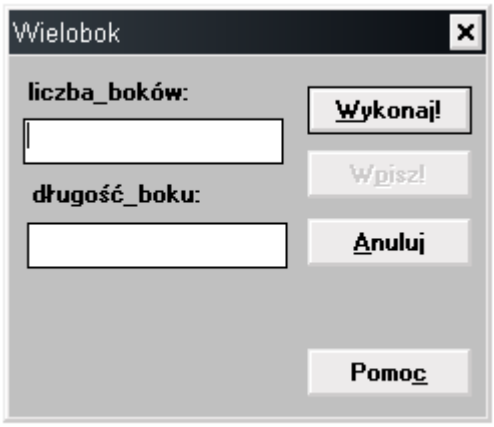

```
? Oto kwadrat :długość_boków
> Powtórz 4 [ np :długość_boków pw 90 ]
> Już
kwadrat zdefiniowano
? kwadrat
```

Możemy pominąć wyświetlanie okna zmiennej od razu podając długości nawiązując do powyższej procedury można ją wywołać w taki oto sposób :
?kwadrat 150
Efekt będzie taki sam jak przy oknie zmiennej.
A co jeśli mamy więcej zmiennych ?

Świetnym przykładem jest ta procedura :

oto wielobok :liczba_boków :długość_boku
powtórz :liczba_boków [naprzód :długość_boku lewo 360 / :liczba_boków]
już

Można wywołać tą procedurę na 2 sposoby z oknem lub bez okna zmiennej
Z oknem zmiennych :
? wielobok



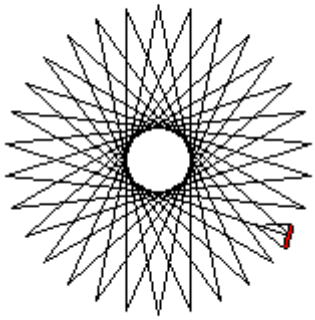
lub :
? wielobok 10 60
czego efekt będzie taki sam.

Uwaga !
Przy dodawaniu zmiennej (:długość_boków) NIE WOLNO robić SAPCJI ! Ponieważ żółw zinterpretuje to jako 2 inne zmienne ale ich wykorzystanie nie jest możliwe !! Znak "_" stosuje się dla utrzymania estetyki i zachowanie "ciągu" znaków

Inne procedury:

Procedura "gwiazda"

oto gwiazda :bok :kąt
np :bok pw :kąt
gwiazda :bok :kąt
już



Lecz jak można zauważyć efekty tej procedury mogą być różne w zależności od podanego kąta i długości boku.

Procedura „gwiazda" jest rekurencją , czyli opisaliśmy wykonanie procedury w której nie ma końca.
W logo Komeniuszu zakończenie rekurencji jest możliwe przez pewny przycisk :

który zatrzymuje wykonywanie wszystkich procedur w programie logo komeniusz
Zastanówmy się jak ja wywołałem rekurencję.
Otóż przy definiowaniu procedury w przed ostatniej linijce znów wywołałem procedurę „gwiazda" .

oto gwiazda :bok :kąt
np :bok pw :kąt
gwiazda :bok :kąt
już

Zastanówcie się co by się stało gdyby w przedostatnim wersie :P zmienić nazwy zmiennych ?

Ciąg dalszy nastąpi :)