Кузнецова Ирина Николаевна

ЛОГО МИРЫ 2.0. РАЗВЕТВЛЕНИЕ. ЦИКЛЫ. ПРОЦЕССЫ

О СТРУКТУРНОМ ПРОГРАММИРОВАНИИ

> Нахождение глубинной простоты в запутанном клубке сущностей – это и есть творчество в программировании
>
> *Х.Д.Милс*

Среда Лого ориентирована на объектный подход в программировании. Однако такой подход не исключает просто обработку данных. Следовательно, разработчик сталкивается с необходимостью организовать в алгоритме выбор действия, повторение действий, в зависимости от условий.

Еще в 1973 году появилась книга Х.Д. Милса (H.D.Mills) «Новая дисциплина программирования» («New Discipline Wins Programmer Approval»), в которой излагался общий подход к программированию, выражены идеи и эффективные методы, систематическое применение которых должно было сделать программирование похожим на науку. Так и произошло. Профессор Э.В.Дейкстра (E.W. Dijkstra) доказал, что любую задачу обработки данных можно решить с использованием только линейной, циклической и разветвляющей алгоритмической конструкции. Теория алгоритмов и программирование стали областями математики.

Программирование вообще включает проектирование, кодирование (запись на формальном языке) и тестирование алгоритма достижения поставленной цели. «Подобно поэту, который сочиняет поэму, так или иначе соблюдая размер и форму, программист создает программу из базовых логических структур [1]».

Сегодня достаточно литературы, в которой описываются элементарные алгоритмические структуры. Из языков программирования давно удален пресловутый оператор безусловного перехода. Нынешние начинающие программисты воспринимают эти принципы как азбуку, не особенно задумываясь, откуда она взялась.

А мы все-таки подчеркнем еще один аспект. Программа на языке высокого уровня есть запись алгоритма с математической точностью. По тексту программы можно провести доказательство правильности алгоритма и найти ошибки без ее выполнения. Программу мы пишем не только для компьютера, но и для человека, который будет ее читать, анализировать и модифицировать. Поэтому текст программы должен быть записан так, чтобы его можно было достаточно легко прочесть и понять. Отсюда правила оформления текста программы:

1. Правило одной страницы – весь текст программы должен помещаться на одной странице экрана (рабочего поля). Если это требование не выполняется, необходимо включить вспомогательные алгоритмы.

2. Правило структурной записи – в тексте программы алгоритмические конструкции выделяются с помощью записи их с отступом от левого края.

3. Текст программы должен сопровождаться внешним и внутренним комментариями. Внешний комментарий содержит постановку задачи, неформальное описание данных, способ (пример) запуска и тестовые данные. Внутренние комментарии поясняют действия алгоритма.

Сегодня мы покажем, как на языке Лого описываются элементарные алгоритмические конструкции.

ПРОЦЕДУРА – ВСПОМОГАТЕЛЬНЫЙ АЛГОРИТМ

Вообще говоря, любая программа на Logo – процедура, так как может быть вызвана другой программой.

```
to proc
<действия >
end
```

proc – новое слово языка – команда или датчик (функция).

РАЗВЕТВЛЕНИЕ

Эта алгоритмическая конструкция обеспечивает выбор одного из двух возможных списков действий, в зависимости от значения условия. Условие задается логическим выражением и может быть истинным (значение «да») или ложным (значение «нет»).

Полное разветвление

```
ifelse <условие> [<список действий 1>] [<список действий 2>]
```

Неполное разветвление

Отсутствует список действий для ложного значения условия.

```
if <условие> [<список действий 1>]
```

Выбор из нескольких вариантов

Команду **if** можно использовать для организации выбора одного списка действий из нескольких возможных – так называемой конструкции **case**. Если все условия ложны, выполнение процедуры может заканчиваться, даже если этот выбор включен в цикл. Останов текущей процедуры – команда **stop**.

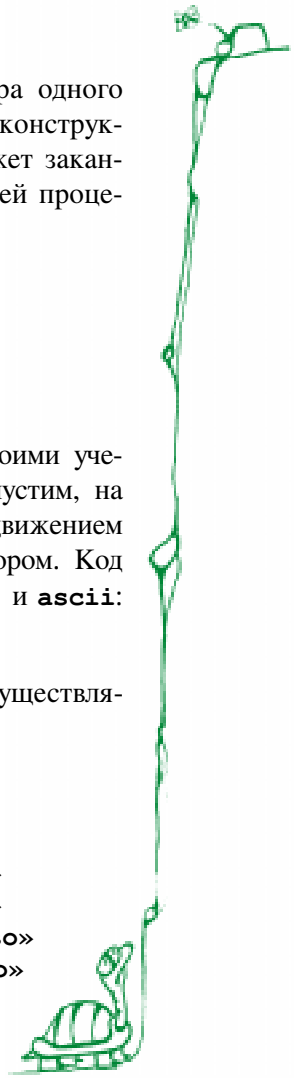
```
if <условие> [<список действий 1>]
if <условие> [<список действий 2>]
if <условие> [<список действий 3>]
...
if <условие> [stop]
```

В качестве примера позвольте привести любимую всеми моими учениками идею управления объектом с помощью клавиатуры. Допустим, на Рабочем поле находится объект – черепашка под именем **t1**, движением которой управляет пользователь через клавиши управления курсором. Код нажатой клавиши считывается и вычисляется датчиками **readchar** и **ascii**:

```
make "cur ascii readchar
```

Выбор действия, соответствующего нажатой клавише, осуществляется по следующему алгоритму:

```
to actcase
make "cur ascii readchar
t1,
if :cur = 38 [seth 0 fd 10]  ;нажата клавиша «вверх»
if :cur = 40 [seth 180 fd 10] ;нажата клавиша «вниз»
if :cur = 39 [seth 90 fd 10 ] ;нажата клавиша «вправо»
if :cur = 37 [seth 270 fd 10 ] ;нажата клавиша «влево»
if :cur = 27 [stop ] ;нажата клавиша «ESC»
end
```



Заметим, что датчик readchar начинает работать после щелчка мышью в любом месте Рабочего поля.

ЦИКЛ

Известны три вида циклов – перечисляемый (с заранее известным числом повторений), с предусловием, с постусловием.

Перечисляемый цикл

В Logo такой цикл можно организовать разными способами:

1. Команда **repeat** обеспечивает повторение указанное число раз указанный список команд.

```
repeat <число повторений> [<повторяемые действия>]
```

2. Команда **dolist** организует повторение указанных действий для каждого указанного значения заданной переменной.

```
dolist  [<имя переменной> [<список значений этой переменной>]]
[<список повторяемых действий>]
```

Предположим, надо нарисовать 5 квадратов, каждый из которых имеет определенный цвет. Цвета квадратов не связаны никакой зависимостью. Команда **dolist** будет содержать список значений этих цветов.

Вспомогательный алгоритм – цветной квадрат:

```
to sqc a c
pd setc :c
repeat 4[fd :a rt 90]
rt 45 pu fd :a / 2 pd fill bk :a / 2 lt 45
end
```

Ряд из квадратов:

```
to row a
dolist [col [15 45 65 75 95]][sqc :a :col fd :a]
end
```

3. Команда **dotimes** обеспечивает повторение указанных действий для каждого значения из диапазона указанной числовой переменной (минимальное значение этой переменной – 0).

```
dotimes  [<имя переменной > <верхнее значение диапазона>]
[<список повторяемых действий >]
```

Если запустить программу **clock**, вы увидите на Рабочем поле секундную стрелку:

```
to clock
seth 0
dotimes [k 360] [setc 9 arrow wait 1 setc 0 arrow   seth  :k]
setc 9 arrow st
end
```

Вспомогательный алгоритм рисования стрелки:

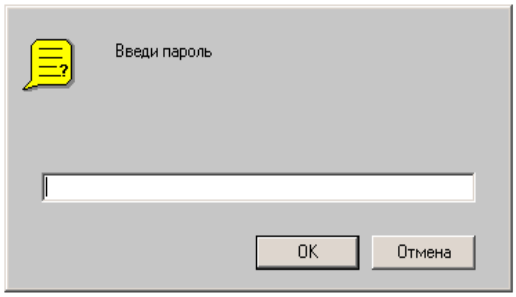
```
to arrow
ht
fd 100
lt 45 bk 20 fd 20 rt 45
rt 45 bk 20 fd 20 lt 45
bk 100
end
```

Цикл с постусловием

```
repeat <max число повторений> [ddd1 if <условие окончания цикла>
[stop]]
to ddd1
<повторяемые действия>
end
```

Рассмотрим пример.

Как установить защиту своего проекта? Можно защитить его паролем. Напишем программу, которая вызывается сразу после загрузки проекта и запрашивает пароль. Дается три попытки для ввода пароля. Если все-таки он оказался неверным, программа запрещает работу с проектом.

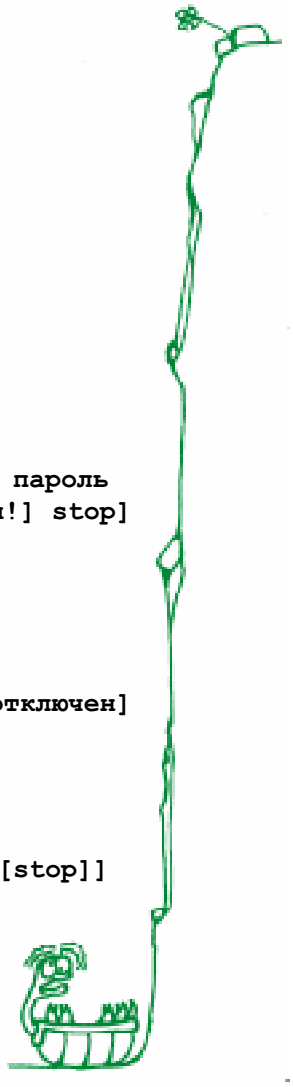



```
to startup
make "parol "1234567 ;устанавливаем значение пароля
repeat 3 [question [Введи пароль]
                make "ans answer   ;запрашиваем и вводим пароль
               if :parol = :ans [announce [Привет, хозяин!] stop]
          ]
;если пароль неверный, экран чернеет,
появляется сообщение о взломе и звучит колокол
if not :parol = :ans
                     [presentationmode setbg 9
                      announce [Это взломщик!!!  Проект отключен]
                       forever [note 60 3]]
end
```

Цикл с предусловием

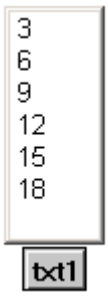
```
repeat <max число повторений> [ifelse  <условие>   [ddd][stop]]
to ddd
<повторяемые действия>
end
```

Рассмотрим в качестве примера классическую задачку поиска чисел с заданным признаком в заданном интервале.



Нужно вывести в текстовое окно N чисел натурального ряда, которые делятся на 3.

```
to startnum
;подготовка данных в переменных
carefully[txt1, ct]
[newtext "txt1 [-100 100]
[50  100]]
local [N k i]
make "k 1 ;текущее число
make "i 0 ;счетчик чисел
question [введи число N]
make "N answer  ;количество чисел
repeat  1000 [ifelse :i < :N
              [if (remainder :k 3) = 0 [pr :k make "i :i + 1]
               make "k :k + 1]
               [stop]
             ]
                 end
```



ОБЪЕКТЫ И ПРОЦЕССЫ

Мы рассмотрели задачи обработки конечных данных. Однако, как было упомянуто ранее, в среде Logo существуют встроенные классы объектов, и есть возможность создавать новые классы. Деятельность объектов часто представляет собой процесс – последовательность сменяющихся состояний. Для управления процессами предназначены команды **forever**, **launch**, **when**, **stopme**, **cancel**, **waituntil**.

В качестве первого примера организации процессов и взаимодействия объектов предложим задачу олимпиады по программированию ДООИ 2001 [2].

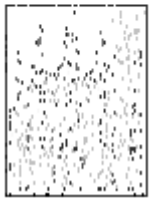
Задача. Проявление фотографии.

Кто занимался фотографией, видел, как проявляется фото. Постепенно и как бы сразу везде. Обратите внимание, что

а) Процесс прорисовки точек случайный. Нет такого момента времени, когда можно было бы увидеть выделенное направление прорисовки точек.

б) Процесс идет равномерно, в том смысле, что плотность точек примерно одинакова на любом участке в любой момент времени.

в) Темп прорисовки всегда один. Вы можете поставить большую задержку и увидеть, что скорость появления новых точек в начале проявления такая же, как и в конце.



cancel.

Описание данных.

Два графических объекта – черепашки t1 и t2 работают в равных геометрически прямоугольниках, которые расположены с известным смещением относительно друг друга.

x1, y1 – координаты левой нижней вершины прямоугольника P1 с проявляемой фотографией.

w – ширина прямоугольника, h – высота прямоугольника, dx – смещение по оси X, dy – смещение по оси У для прямоугольника P2 с данной фотографией.

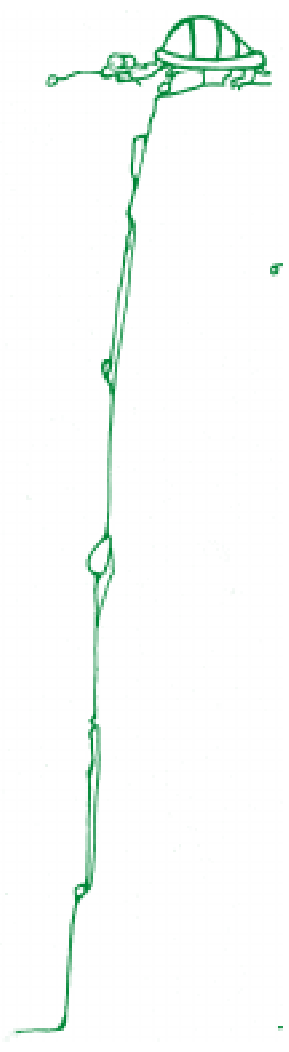
post1 – координаты положения черепашки в прямоугольнике.

cundert2 – цвет пикселя сканируемой фотографии.

Описание процессов.

Процесс **scanphoto** – объект t2 перемещается в точку P2, соответствующую значению переменной post1 (текущему положению черепашки t1 в прямоугольнике P1), измеряет цвет этого пикселя и записывает его значение в переменную cundert2.

Процесс **showphoto** – объект t1 закрашивает пиксель, на котором стоит, цветом, который содержится в переменной cundert2 (то есть цветом того пикселя, на котором стоит черепашка t2). Затем объект t1 случайным образом перемещается внутри прямоугольника P1 и записывает свои координаты в переменную post1.



```
Процедура рисования прямоугольника
to rectangl  w h
repeat 2[fd :h rt 90 fd :w rt 90]
end

Процедура описания деятельности каждого объекта
to process
t2, scanphoto  t1, showphoto
end

Проявление фотографии в прямоугольнике P1
to showphoto
setc :cundert2 pd fd 1 bk 1 pu
; случайное смещение по X и по У
make "dxt  random :w
make "dyt random :h
;Новое положение для t1 в P1
make "xt :x1 +  :dxt
make "yt :y1 + :dyt
make "post1 list :xt :yt
setpos :post1
end

Сканирование фотографии в прямоугольнике P2
to scanphoto
make "xt2 :x2 + :dxt make "yt2 :y2 + :dyt
setpos  list :xt2 :yt2
make "cundert2 colorunder
end

Запускающая процедура
to startup
make "w 70 make "h 95
make  "dx 0 make "dy :h + 20
make "x1 -100 make "y1 -50  make "x2 :x1 + :dx make "y2 :y1 + :dy

;создадим объекты – черепашек t1 и t2 , нарисуем прямоугольники и поместим
в P2 исходный рисунок
cg
```

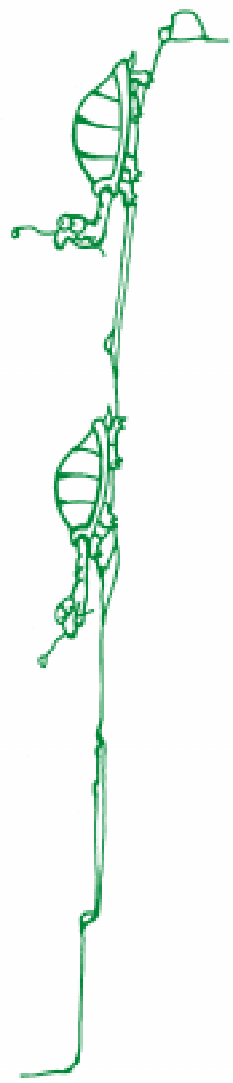
```
carefully  [t1, pu][newturtle "t1]  setpos list :x1 :y1 pd rectangl :w :h
pu
carefully  [t2, pu][newturtle "t2] setpos list :x2 :y2 pd rectangl :w :h
pu
setpos list :x2 +  2 + :w / 2  :y2 + :h / 2
setsh 27 setsize 40 pd stamp pu setsh 0 ht
t1, ht

;начальное смещение для позиции объекта t1:
make "dxt 0 make "dyt 0
;начальный цвет под объектом t2:
make "cundert2 0

;Для окончания процесса - нажать на кнопку на рабочем поле
carefully[let [temp get "cancel.]][newbutton  "cancel. [-50 -100] [cancel.]]

make "finish 0
forever [process]  ;запуск процессов
waituntil [:finish = 1] cancel [process]
;ожидание признака завершения процессов
end

Процедура завершения процесса
to cancel.
make "finish 1
end
```



Описание новых слов LOGO

question <текст> – открывает диалоговое окно, в котором написан **<текст>**, и предлагает ввести ответ. Можно устанавливать положение окна на Рабочем поле.

answer – датчик, который сообщает введенный с клавиатуры ответ на вопрос, заданный командой **question.**

announce <текст> – выводит **<текст>** в сигнальном окне.

local <имена переменных> – переменные или одна переменная становятся локальными для процедуры, в которой они объявлены. При завершении этой процедуры указанные в **<имена переменных>** переменные удаляются.

let <имена и значения переменных> – создает в процедуре локальные переменные и задает им значения.

note <нота> <длительность> – озвучивает указанную ноту в течение указанного времени на выбранном ранее музыкальном инструменте.

newtext <имя> <координаты> <длина и ширина> – создает текстовое окно с именем **<имя>**, левая верхняя вершина которого в точке с указанными координатами, а размеры указаны в **<длина и ширина>**.

newbutton <имя> <координаты> <команды> – создает в указанном месте Рабочего поля кнопку с указанным именем. При нажатии кнопки будут однократно выполняться **<команды>**.

get <имя> <параметр> – датчик, который сообщает значение указанного параметра указанного объекта.

В следующем выпуске более подробно рассмотрим задачи, связанные с созданием объектов и управлением ими.

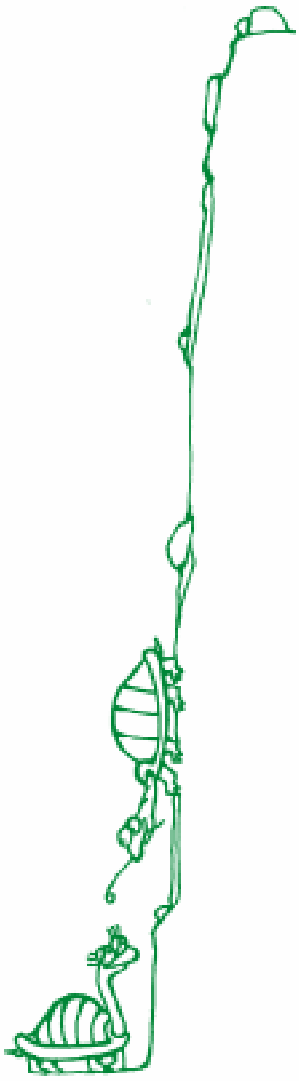
Плейер среды Лого Миры 2.01 можно найти на сервере www.school.edu.ru/int/logo или на сервере Петербургской Интернет-школы www.ipo.spb.ru/internet-school/ch5.htm.

Запустите на выполнение файл mw2player.exe, и вы получите инструмент для просмотра проектов.

Уже 7 лет группа энтузиастов Лого организует командные олимпиады в среде Лого. Некоторые задачи олимпиад были представлены в прошлом номере журнала.

На сервере Петербургской Интернет-школы вы сможете найти также файл olimpo1.mw2. Он содержит задачи командной олимпиады в среде ЛОГО, которая проводилась в 2001 году. В проекте семь листов, на каждом – одна задача. Установочные программы и по одному варианту решения каждой задачи расположены на листе программ.

Более двух лет автор использует систему Лого Миры 2.01 в обучении информатике с 3 по 11 класс в 640 школе Санкт-Петербурга. Среда Лого применяется как иллюстрация понятий объектов, а также как среда программирования.В этом году команда из школы 640 приняла участие в дистанционной олимпиаде по программированию ДООИ. Опыт показал, что среда Лого ничем не хуже любой другой и позволяет решать все задачи.



Литература.

1. Дж. Хьюз, Дж. Мичтом, Структурный подход к программированию. М.: «Мир», 1980.

2. Материалы олимпиады по программированию ДООИ 2001 (http://dooi2001.narod.ru и http://attend.to/dooi, координатор – Потопахин Виталий Валерьевич).

Кузнецова Ирина Николаевна,
учитель высшей категории,
директор НОУ ДО
«Папертовский центр»,
Санкт-Петербург.