МОУ «ЛИЦЕЙ №57»

Программирование на языке Лого

Прокопьевск, 2011

Составитель:

Краснова Наталья Владимировна,

учитель информатики МОУ “Лицей №57”

Программирование на языке Лого

Методическое пособие предназначено для педагогов, а также для учащихся 5-6 классов средней школы, изучающих среды ЛогоМиры или LogoWriter. Содержит теоретический материал и задания для практических работ.

МОУ “Лицей №57”, 653045, Кемеровская обл.,
г. Прокопьевск, ул. Институтская, 41
Телефон: (3846)62-31-20
E-mail: lyceum57@mail.ru

Содержание:

ВВЕДЕНИЕ

Язык программирования, используемый при работе в системе ЛогоМиры, называется Лого. “Лого” происходит от греческого «логос», означающего «слово», «мысль».

Язык Лого был создан в 1968 году и к настоящему времени разделился на множество программных средств, представляющих среду обучения Лого. Программная среда Лого была разработана и реализована под руководством американского психолога Сеймура Пейперта в 1989 г. в Массачусетсом техно-логическом институте.

В Лого реализован движущийся по экрану объект — маленькое изображение черепашки, чьим движением мы можем управлять. Эта черепашка при своем перемещении оставляет на экране след, подобно живой черепахе, движущейся по песку.

Необычное название — “черепашка” связано с одной из работ американского нейрофизиолога Г. Уолтера, проводившего эксперименты с маленькими роботами, которых он называл “черепашками”. Позднее так был назван и первый робот, работавший под управлением компьютера и передвигавшийся по полу по командам вперед, назад, напра-

Лого - универсальный язык простой в изучении, т.к. команды, используемые в нем, имеют не абстрактный для учащихся вид (например, вперед, назад, направо, налево, повтори и др.).

Лого - прекрасный инструмент для создания компьютерной графики одновременно с освоением элементарных навыков структурного программирования.

Элементы Лого-графики сейчас включаются во многие языки программирования, такие как: Паскаль, Си и др.

Среду Лого можно назвать интегрированной, т.к. она включает в себя графический, текстовый, музыкальный редакторы, среду программирования.

Пособие содержит:

- *краткие теоретические описания по каждой теме;*
- *примеры;*
- *вопросы и задания по темам.*

В пособии использованы условные обозначения:

- рассмотри пример самостоятельно.

- запиши и выучи.

Понятие алгоритма, его свойства

Алгоритм — это предписание, набор точных инструкций (команд), записанных в строгом порядке и направленных на решение поставленной задачи.

Того, кто (или То, что) выполняет алгоритм, мы называем ***исполнителем алгоритма***.

Команды, которые может выполнить конкретный исполнитель, образуют *систему команд исполнителя (СКИ).*

Рассмотрим примеры алгоритмов и Исполнителей

В дом привезли новый шкаф... То есть, шкафа как такового еще нет, на полу разложены створки, полки, шурупы и прочие детали будущего вместилища одежды и белья. Вы с отцом, следуя подробной инструкции, приступаете к сборке. Здесь инструкция выступает в роли алгоритма, а вы с отцом — его исполнителей.

На уроках математики вы выполняете разные вычисления — умножаете и делите столбиком, складываете простые дроби. В этих случаях вы являетесь исполнителями соответствующих алгоритмов.

Но исполнителем может быть не только человек. Разнообразные устройства, в том числе и компьютер, также могут выполнять заданные им алгоритмы. Например, "Луноход" — самоходный автоматический аппарат, доставленный на Луну в 1970 году, выполнял сложнейшие алгоритмы, перемещаясь по лунной поверхности и собирая необходимую людям информацию. Промышленные роботы заменяют людей на производстве, в быту на помощь хозяйкам также приходят устройства, способные действовать по заданным алгоритмам.

Для того, чтобы последовательность действий считалась алгоритмом, необходимо чтобы она отвечала следующим ***свойствам***:

Понятность: алгорит м сост авляет ся т олько из команд, входящих в СКИ исполнителя, т.е. понятных исполнителю.

Точность: каж дая команда определяет однозначное действие.

Результативность: выполнение алгорит ма долж но обязательно привести к результату за конечное число шагов.

Формальный и неформальный исполнитель

В отличие от устройств, человек является неформальным исполнителем алгоритма. Что это означает?

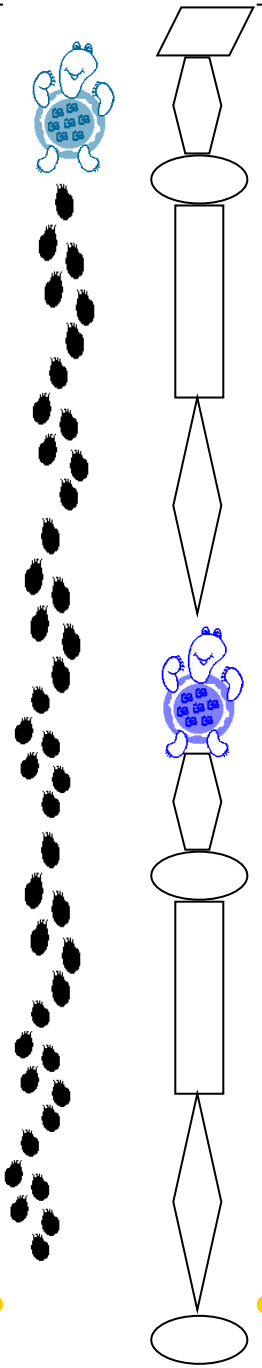
Во-первых, человек не выполняет алгоритм бездумно и формально, соображения здравого смысла часто оерут верх, какими бы строгими ни были инструкции. Если ему скажут, идите прямо до второго перекрестка и никуда не сворачивайте, то, вряд ли он станет наступать на встречных прохожих и опрокидывать попадающиеся на пути предметы.

Во-вторых, человек, даже если он еще мал, обладает определенным жизненным опытом и чем больше этот опыт, тем менее подробным может быть алгоритм. Взрослому человеку достаточно сказать, свари кофе, не объясняя до мельчайших подробностей, как это следует делать.

В-третьих, неформальному исполнителю, как правило, известна конечная цель и он к ней стремится, порою уточняя и дополняя алгоритм.

Формальный исполнитель не обладает ни жизненным опытом, ни здравым смыслом, он не стремится к конечному результату и все инструкции алгоритма выполняет буквально, так, как они записаны в алгоритме.

Поэтому такого исполнителя мы будем называть ***бездумным исполнителем*** или БИ.



И если случится так, что наш БИ будет вести себя неразумно, выполнять что-то, не отвечающее нашим намерениям, то винить все-таки придется нам самих себя и искать неточности и ошибки в алгоритме, ведь БИ повинуется не нашим намерениям, а нашим инструкциям.

Для формального исполнителя язык общения не может быть многозначным, для таких исполнителей разрабатывают и используют специальные искусственные языки, где отдельные слова и выражения не допускают различных толкований.

Программа. Программирование

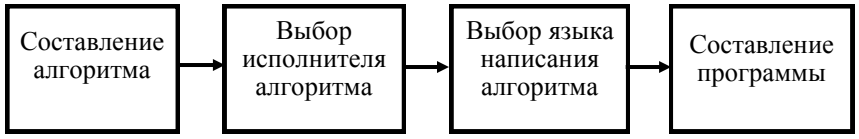
Алгоритм, описанный на языке исполнителя, называется ***программой***.

Программирование — это процесс перевода алгоритма на язык конкретного исполнителя.

Так как разные исполнители владеют разными языками, а один и тот же алгоритм могут выполнять разные исполнители, то на основе одного алгоритма программы могут быть написаны на разных языках.

Чтобы научиться писать программы на том или другом языке, нужно изучить алфавит, словарь и грамматические правила, по которым строятся предложения в этом языке, при этом не допускаются никакие отклонения от правил написания слов и предложений, иначе исполнитель просто откажется выполнять ваши инструкции и не станет недоумевать и переживать за ошибки.

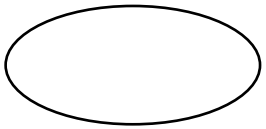
Этапы написания программы:



Блок-схема. Понятие. Назначение

Блок-схема – это графическое представление алгоритма, состоящее из геометрических фигур, соединенных между собой стрелками..

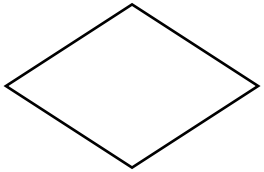
Основные блоки, используемые при составлении блок-схемы:



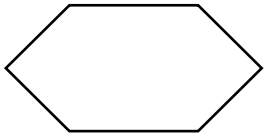
Блок начала (конца) алгоритма

Блок ввода (вывода) значений

Блок выполнения действия



Блок проверки условия



Блок повтора действия (цикла)

Вопросы

1. Что такое алгоритм, исполнитель алгоритма?
2. Что такое система команд исполнителя?
3. В чем различие между формальным и неформаль- ным исполнителями алгоритма?
4. Почему для общения с формальными исполнителям разрабатывают специальные искусственные языки?
5. Какие блоки, используются при графическом описании алгоритма.

Задания

1. Опишите алгоритм умножения столбиком двух двухзначных чисел.
2. Опишите алгоритм перехода улицы, если переход не регулируется светофором.
3. Имеются три емкости для жидкости — А, Б и В, вместимостью 8, 5 и 3 литра соответственно, и автомат, который может переливать жидкость из одной емкости в другую. Автомат регулируется нажатием кнопок: А→Б, А→В, Б→В, Б→А, В→Б и В→А. Емкость 8 литров полностью заполнена жидкостью, остальные две пусты. Автоматом управляет абсолютно формальный и совершенно бездумный исполнитель, понимающий только обозначения кнопок. Составьте алгоритм для этого исполнителя, с помощью которой через конечное число шагов в емкости А остается ровно 7 литров жидкости. Проливать жидкость или добавлять ее из внешнего источника не разрешается.

Решение оформите в виде таблицы:

№ команды	Команда	Количество литров		
		А	Б	В
Начало		8	0	0
...	...	...	...	...
...	...	7	...	...
Конец				

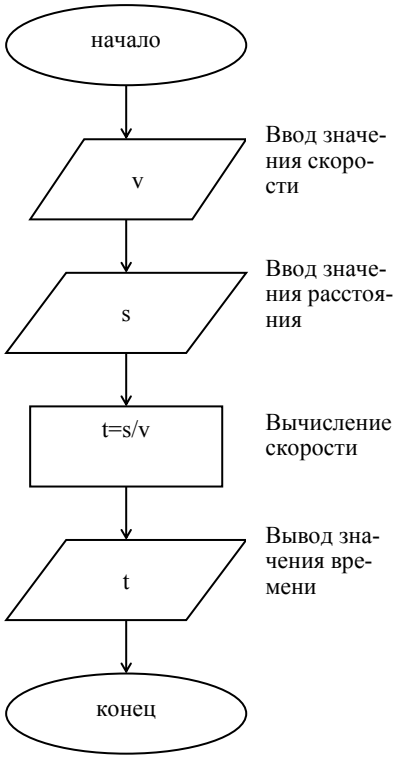
Линейный алгоритм

Алгоритм, в котором все действия выполняются одно за другим (по очереди) называется ***линейным алгоритмом***.

Существует два типа задач, реализующих линейный алгоритм (расчётные и бытовые). Рассмотрим примеры задач каждого типа.

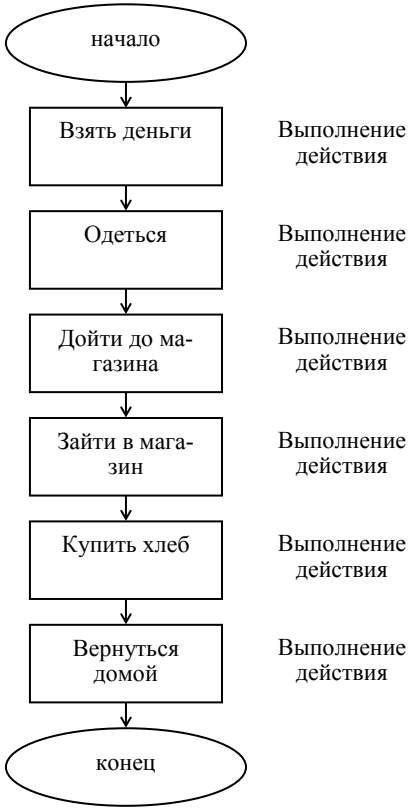
Задача расчётного типа:

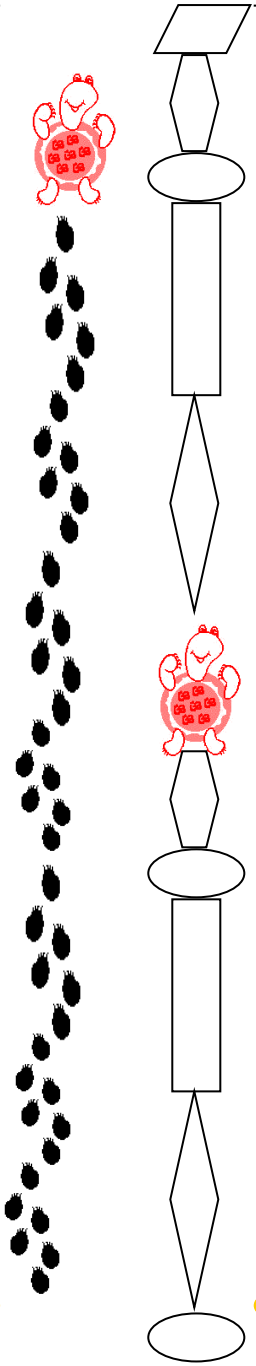
Пешеход идёт со скоростью 5 км/ч и проходит путь в 10 км. Сколько времени затрачивает пешеход?



Задача бытового типа:

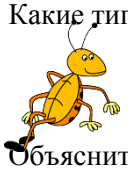
Осуществить покупку хлеба в магазине.





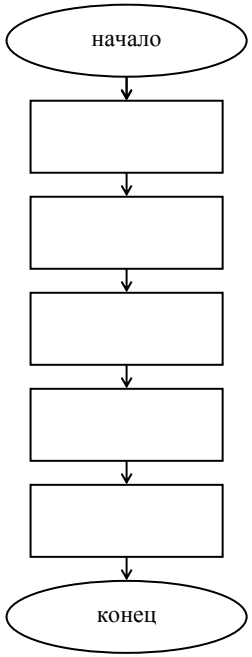
Вопросы

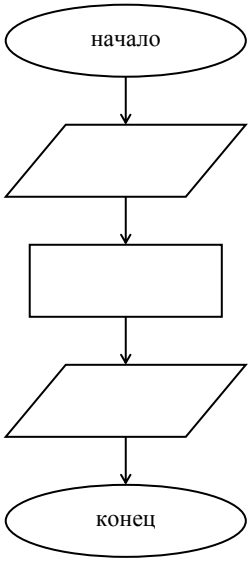
1. Какой алгоритм называется линейным?
2. Какие типы задач реализуются линейным алгоритмом?



Задания

1. Объясните отличие блок-схем расчётных и бытовых задач.
2. Составьте блок-схему решения бытовой задачи: выполнение домашнего задания по информатике.
3. Составьте блок-схему решения расчётной задачи: вычислите, сколько орехов наберёт белка за 6 часов, если за 1 час она собирает 20 орехов.
4. По данным блок-схемам определить типы задач и придумать к ним условия:





Понятие переменной. Имя и значение переменной

Существует несколько определений такого понятия, как переменная.

Переменная – это именованный участок памяти компьютера, предназначенный для хранения какого-либо значения.

Переменная – это величина, которая может менять своё значение в процессе выполнения программы.

При работе с переменной нужно знать и отличать два основных понятия: ***имя переменной*** **и** ***значение переменной***.

Для того, чтобы компьютер различал имя и значения переменной, существуют **правила** их **записи**:

“имя переменной – *имя переменной;*
:имя переменной – *значение переменной.*

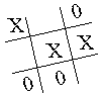
Правило: после кавычек и двоет очия пробел не ст авится.

На данном примере рассмотрим, что будет являться именем, а что значением переменной:

Скорость движения машины 80 км/ч, а расстояние, которое она проехала – 200 км.

Из уроков математики вы знаете, как обозначается скорость и расстояние.

В данном примере **v** и **t** – это **имена** переменных,
80 и **200** – **значения** переменных.



Перевод выражений с языка математики на язык Лого
Арифметические операции, используемые при составлении выражения:

\+ - сложение
\- - вычитание
* - умножение
/ - деление

Правила перевода выражений:

1. Перед знаками арифметических операций и после них ставятся пробелы:

   a+b-c → a_+_b_-_c

2. Если в числители или знаменатели дроби есть операции сложения или вычитания, то числитель и знаменатель дроби заключают в скобки:

$$\frac{a+b}{c-d} \rightarrow (a\_+\_b)\_/\_(c\_-\_d)$$

$$\frac{2b}{a-4} \rightarrow 2\_*\_b\_/\_(a\_-\_4)$$

3. Нижние индексы записываются обычными цифрами, причем между именем и цифрой пробел не ставится:

4. Количество открывающихся скобок должно равняться количеству закрывающихся. $v_1 \rightarrow v1$
5. Лишние, но правильно поставленные скобки не влияют на результат выражения.

Рассмотрим примеры перевода выражений с языка математики на язык Лого:

$$\frac{2a-4b+c}{c-3d}+2b-4 \rightarrow$$

$$(2\_*\_a\_-\_4\_*\_b\_+\_c)\_/\_(c\_-\_3\_*\_d)\_+\_2\_*\_b\_-\_4$$

$$a+\frac{3-b}{8c} \rightarrow a\_+\_(3\_-\_b)\_/\_8\_*\_c$$

Решение задач

При помощи компьютера можно не только рисовать, создавать мультфильмы и записывать музыку, но и решать задачи. Но для этого нужно познакомиться с командами, которые позволяют это сделать:

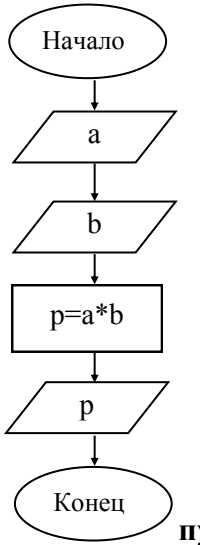
пусть_“имя переменной_значение переменной – ввод данных
покажи_:имя переменной – вывод данных

Пример:

Найти площадь прямоугольника, если известно, что его длина 5 см., а ширина – 3 см.

Введём обозначения: a – длина, b – ширина, p – площадь.

Составим блок-схему решения задачи:



Составим программу:

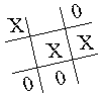
пусть_”a_5
пусть_”b_3

пусть_”p_:a_*_:b

покажи_:p

x+2

x+y=0



a-4+b

***Правило*:** для выполнения команд компьютером необходимо нажать клавишу ENTER после каждой их них!

Найти значение сложного выражения:

$$2a+c-\frac{4-\frac{a+b}{d-7b}}{abc-3}-f+5$$ если a=1, b=4, c=8, d=2, f=3.

пусть_”a_1_пусть_ “b_4_пусть_”c_8_ пусть_”d_2_ пусть_”f_3
пусть_”k_2_*_:a_+_:c_-_(4_-_(:a_+_:b)_/_(:d_-_7_*_:b))_/_
(:a_*_:b_*_:c_-_:3)_-_:f_+_5
покажи_:k

Вопросы

1. Что такое переменная?
2. Как компьютер отличает имя переменной от ее значения?
3. Какие существуют правила записи арифметических выражений?
4. Какая команда служит для ввода данных в компьютер?
5. Какая команда служит для вывода данных?

Задания

1. Запишите на языке Логго следующие выражения:

   а) $\frac{a+b}{d}$ б) $(a+b)(c+d)$

2. Запишите выражения на языке математики:
   а) a_+_g_*_c_*_d_/_e_-_4_*_f_*_g
   б) (a_-_4_*_b)_/_8_-_d
3. Составьте программы вычисления значения выражений:

а) , при a=4, b=1, c=0, d=8.

б) , при a=4, b=1,d=5.

$$\frac{ab-4cd}{8+2a}$$

$$2b-\frac{4a}{5d}$$

Команды исполнителя

Задавая черепашке команды, ею можно управлять: перемещать, поворачивать и даже изменять её свойства (например, цвет).

Черепашка на экране подобна человеку с завязанными глазами, который передвигается по комнате, слепо и беспрекословно выполняя чьи-то команды.

Для изменения местоположения черепашки в языке Лого используются следующие команды:

вп_количество шагов – движение черепашки вперёд
нд_количество шагов – движение черепашки назад

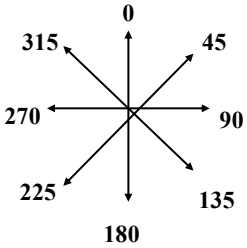
Для изменения направления движения используются команды с указанием угла поворота черепашки относительно исходного направления:

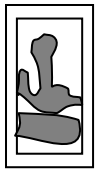
пр_количество градусов – поворот черепашки направо
лв_количество градусов – поворот черепашки налево

У черепашки есть свой компас, который помогает безошибочно выбирать курс. Градусы отсчитываются не от текущего положения черепашки, а от фиксированного начального (нулевого) направления. Это направление соответствует северу на географических картах.

Направление стрелки указывает направление головы черепашки.

нк_количество градусов – новый курс



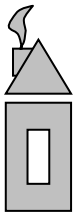


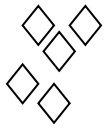
Для возвращения черепашки в исходное положение (центр экрана головой вверх) используется команда **домой**.

Черепашка может рисовать пером, которое располагается в центре панциря.

Пером управляют следующие команды:

по - перо опустить (черепашка прислоняет перо к листу, чтобы оставлять за собой след)
пп – перо поднять (черепашка отрывает перо от листа, чтобы не оставлять следа)
сг – сотри графику (экран очищается, а черепашка возвращается домой)
сотри – экран очищается, а черепашка остаётся на прежнем месте
сч – спрячь черепашку (убирает черепашку с рабочего поля)
пч – покажи черепашку (возвращает черепашку на рабочее поле)
нц_номер цвета– новый цвет (изменяет цвет черепашки)
крась – закрашивает замкнутую область, в которой находится черепашка цветом её пера
нрз_число – новый размер (меняет размер черепашки от 5 до 160)
нрп_число – новый размер пера (изменяет размер пера активной черепашки от 1 до 100)





Нетрудно заметить, что некоторые команды, например, вп, нд, пр, лв, нк, требуют уточнения данных после названия. Это числа, указывающие количество шагов или количество градусов.

Такие числа называются входными параметрами команды или **входными данными**.

Если в команде, требующей входного параметра не указать его, то черепашка не поймёт, что надо делать и выдаст в Поле команд сообщение об ошибке:

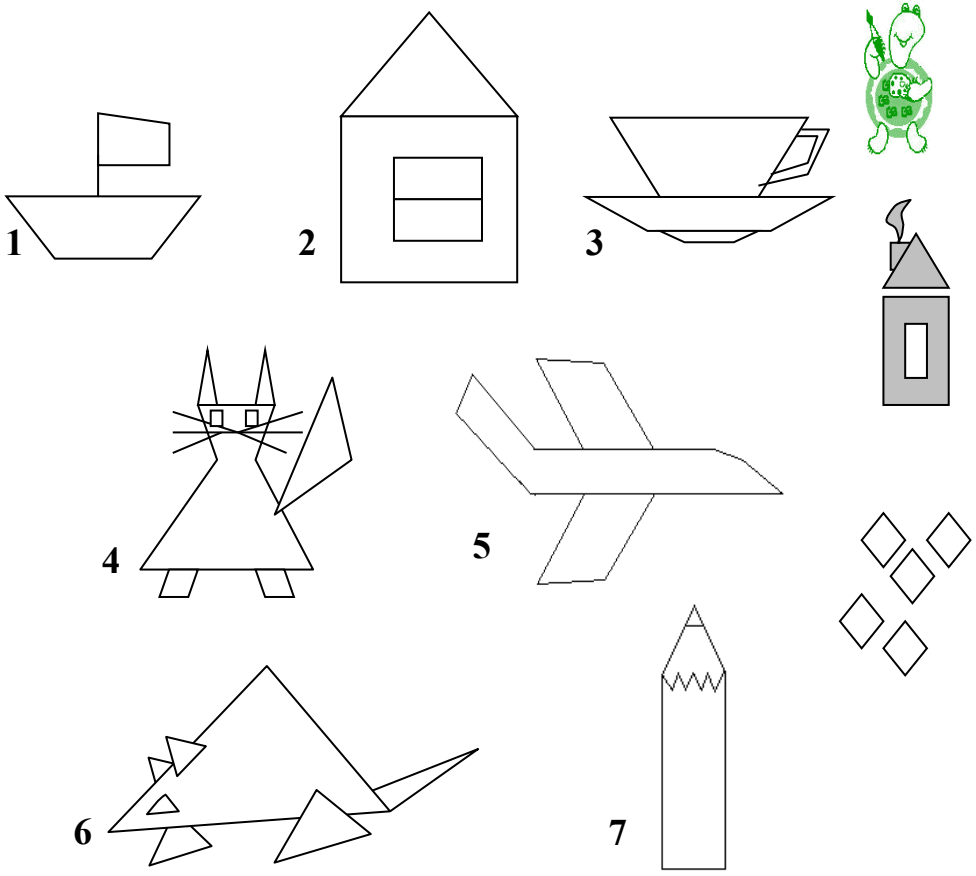
Не хватает входных данных.

Вопросы

1. Какие команды используются для перемещения черепашки?
2. Какие команды используются для изменения направления черепашки?
3. Какие команды используются при создании рисунка?

Задания

Используя известные команды, создайте следующие рисунки:



Циклы

Цикл – это повторяющаяся последовательность команд.

Вложенные циклы – это расположение одного цикла внутри другого.

Циклический алгоритм используется в тех случаях, когда необходимо повторить какие-либо команды или действия несколько раз.

Бытовым примером цикла, например, является смена времён года, движение трамвая по определённому маршруту и т.д.

Где же мы можем использовать цикл, и как записывается оператор цикла?

Пример:

Рассмотрим команды построения квадрата с длиной стороны 50. Причём, после рисования квадрата, черепашка должна вернуться в то же самое место и повернуться в том же самом направлении.

```
по вп 50 пр 90  вп 50  пр 90 вп 50 пр 90  вп 50  пр 90
```

Нетрудно заметить, что в данной программе происходит повторение двух команд: вп 50 и пр 90.

Для того, чтобы не записывать одни и те же команды несколько раз существует оператор цикла, который записывается следующим образом:

повтори_n_[набор команд], где n – число повторов.

Используя оператор цикла, наша программа примет следующий вид:

```
по повтори 4 [вп 50 пр 90]
```

Пример:

Используя оператор цикла, составим программу рисования последовательности:

```
нк 90 повтори 3 [по вп 50 пп вп 20 по]
```

Правильные многоугольники

С использованием оператора цикла можно рисовать и правильные фигуры.

Фигура называет ся ***правильной***, если у неё все ст ороны и углы равны. Например, квадрат, правильный треугольник, шестиугольник.

Для построения правильной фигуры используется команда:

повтори n [вп x пр 360 / n], где n – количество углов (сторон),
x – длина стороны.

Вместо команд **вп** и **пр** могут быть использованы команды **нд** и **лв**.

Составим таблицу для часто используемых многоугольников:

Фигура	*Число вершин*	*Угол поворота*	*Рисунок*
Треугольник	**3**	**360/3=120**	
Квадрат	**4**	**360/4=90**	
Пятиугольник	**5**	**360/5=72**	
Шестиугольник	**6**	**360/6=60**	

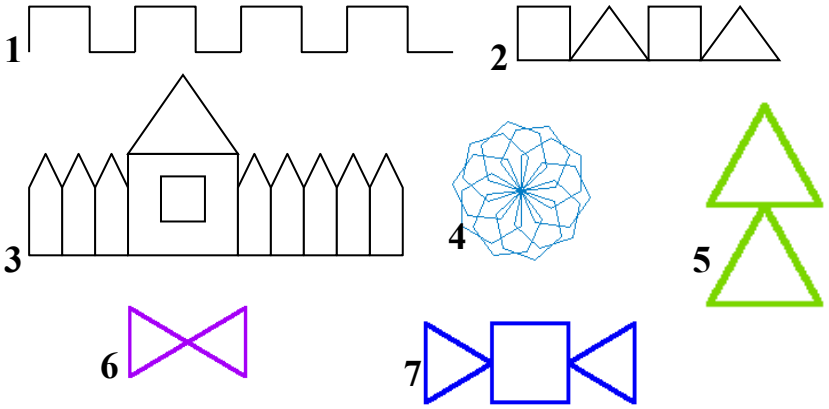
Вопросы

1. Что такое цикл?
2. Что такое вложенный цикл?
3. Общий вид команды записи оператора цикла.
4. Какой многоугольник называется правильным?
5. Общий вид команды построения правильного многоугольника.

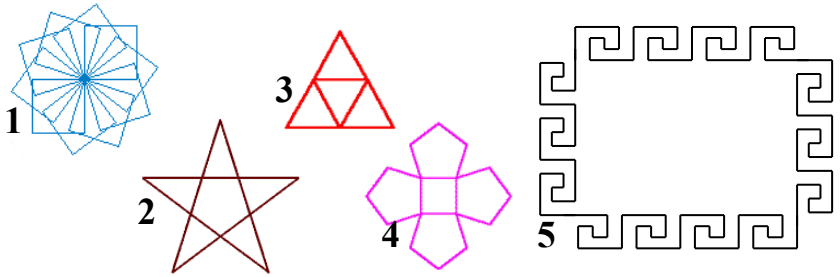
Задания

Используя операторы цикла и правильного многоугольника, составьте команды для построения следующих рисунков:

Обязательные задания по теме:



Дополнительные задания по теме:



Окружность

Окружность – это кривая, каждая точка которой равноудалена от её центра.

Если рассматривать окружность, то можно сказать, что это правильный многоугольник с большим количеством сторон.

повтори 360 [вп 1 пр 1]

Чтобы ***увеличить радиус*** окруж ност и нуж но ***увеличить число шагов***.

Чтобы ***уменьшить радиус*** окруж ност и нуж но ***увеличить число градусов.***

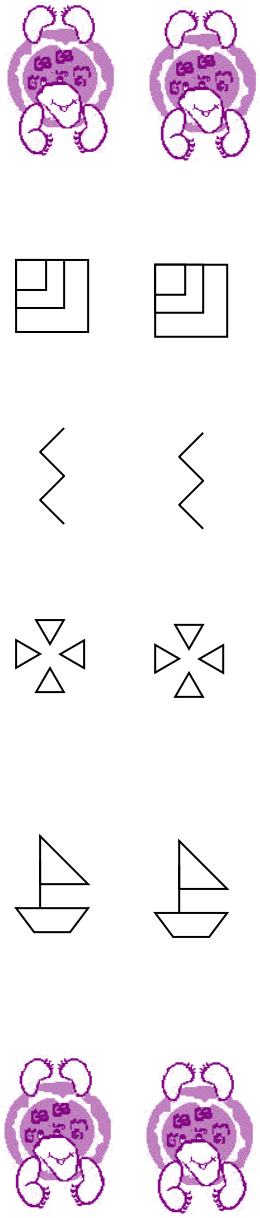
Правило: Число повт орений, умнож енное на число градусов, должно равняться 360.

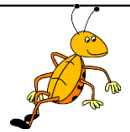
Для рисования полуокружности использует ся команда:

повтори 180 [вп 1 пр  **1]**

Вопросы

1. Что такое окружность?
2. Команда рисования окружности.
3. Что нужно сделать, чтобы увеличить радиус окружности (полуокружности)?
4. Что нужно сделать, чтобы уменьшить радиус окружности (полуокружности)?

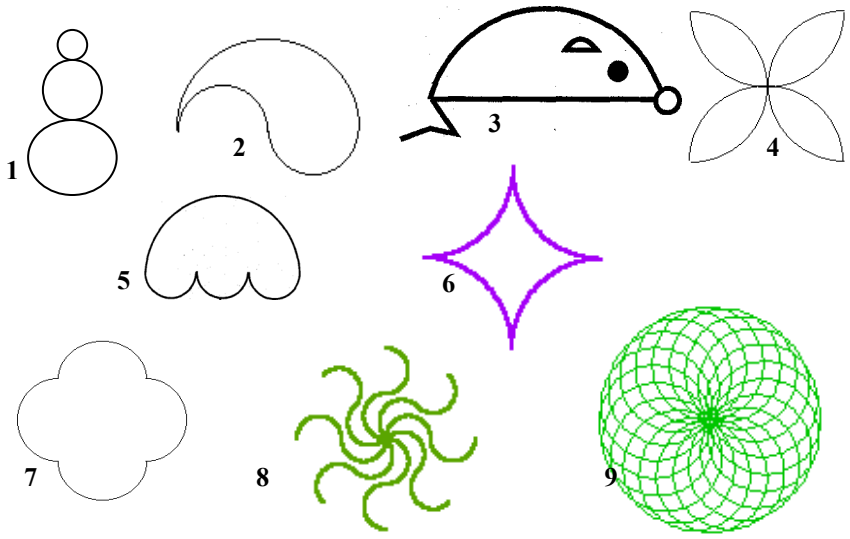




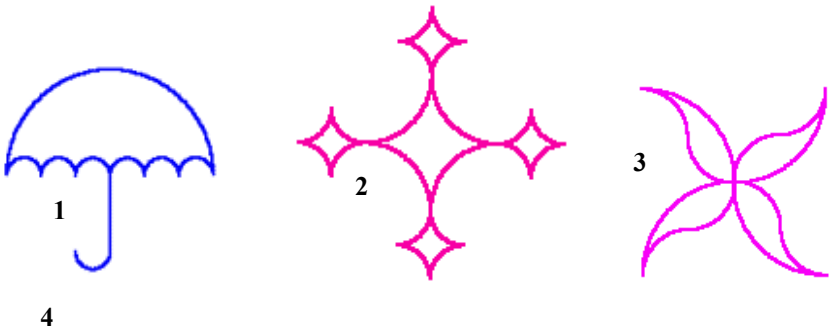
Задания

Используя команды построения окружности и полуокружности, составьте команды для построения следующих изображений:

Обязательные задания по теме:



Дополнительные задания по теме:



Процедуры

Процедура - это самостоятельная часть программы, которая записывается один раз, но может вызываться несколько раз.

Общий вид процедуры:

это_имя процедуры - *заголовок*
набор команд - *тело процедуры*
конец - *завершение процедуры*

Правила:

1. Придумывая имя процедуре, следует придерживаться правила: имя должно говорить о содержании.
2. Имя процедуры, как и имя команды — это одно слово, в имени процедуры пробел недопустим.
3. Кроме того, имя процедуры не должно совпадать со словами из словаря Черепашки — нельзя называть процедуру, например, вперед.
4. В строках заголовка и завершения процедуры не должны располагаться команды.
5. Процедуры записываются в Листе подпрограмм.
6. При вызове процедуры в Поле команд записывается только её имя.

Нередко, особенно на первых шагах, может показаться, что Черепашка живет своей жизнью — вы ей задаете одну программу действий, а она действует по другой. Но не надо забывать, что Черепашка — исполнитель бездумный и выполняет она не наши намерения, а наши инструкции.

При выполнении программы могут встретиться разного рода ошибки.

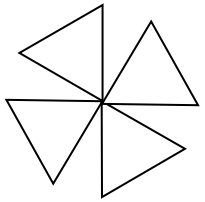
Если на экране появилось сообщение об ошибке, надо его внимательно прочитать и постараться понять. Во многих случаях оно подскажет, какие исправления надо сделать.

Сложнее другая ситуация, когда сообщения об ошибке нет, но результат работы программы не тот, который мы ожидали. В таком случае надо определить место в рисунке и программе, где допущен «сбой», поставить себя на место Черепашки и выполнить на бумаге вместо нее предписанные программой инструкции.

В процессе составления программы можно использовать вложение одной или нескольких процедур в одну.

Пример:

Например, рассмотрим построение следующей фигуры:

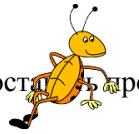


```
это треугольник
по
повтори  3  [вп 50  лв 120]
конец
это  цветок
повтори 4 [треугольник пр 90]
конец
```

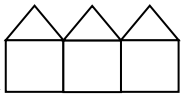
Вопросы

1. Что такое процедура?
2. Общий вид записи процедуры.
3. Правила, применяемые при работе с процедурами.

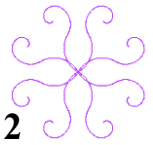
Задания



Составить процедуры построения следующих рисунков:

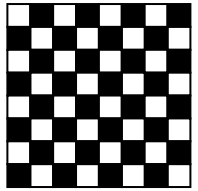


1



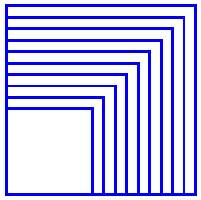
2

3



4

Процедуры с параметром



Посмотрев на фигуру справа, на первый взгляд кажется, нарисовать ее в Лого легко — мы уже умеем писать команды для любых правильных многоугольников, знаем команду **“повтори”**.

Но команды, которые нам знакомы, описывают один конкретный объект — например, квадрат с длиной стороны 60 шагов или окружность радиуса 2. В этих же фигурах повторяются не совсем одинаковые элементы. Это повторяющиеся элементы — квадраты, но они различаются длиной стороны.

Можем ли мы одной процедурой описать множество объектов?

Действительно, для перемещения Черепашки на разные расстояния мы используем **одну** команду вп, но меняем значения ее параметра. Мы можем дать команду вп 100 или вп 40. Также для поворота на **разные углы** Черепашки мы используем команды пр, лв или нк, меняя значения параметров.

Описывая какую-либо процедуру, мы тем самым расширяем словарь Черепашки, которым она владеет от рождения, имя процедуры становится новой командой, известной Черепашке. Точно также эта новая команда может иметь параметр. В качестве параметра выбирают то, что может меняться, оставляя объект тем, что он есть. Для квадрата — это длина стороны. Число вершин или угол при вершине менять нельзя, иначе квадрат перестанет быть квадратом. Если же в качестве объекта рассматривать правильный многоугольник с длиной стороны, например, 20, то изменяемой характеристикой является число вершин (сторон). У окружностей — это радиус.

Какую характеристику объекта выбрать в качестве параметра, зависит от конкретной решаемой задачи.

Объект может иметь несколько изменяемых характеристик и, соответственно, процедура — несколько параметров.

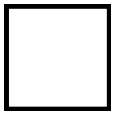
Как и для любой процедуры, работа с процедурой с параметром проходит два этапа — **описание** процедуры и ее **вызов** на выполнение.

При описании процедуры с параметром параметр обязательно указывается в строке заголовка после имени процедуры. (**Не забывайте про пробел!**) Параметр можно обозначать любым набором символов, перед которым ставится “**:**” **(без пробела!)**. При вызове процедуры с параметром надо обязательно указать конкретное значение параметра. Если же при вызове процедуры параметр не будет указан, то появится сообщение об ошибке.

Общий вид процедуры с параметром:

это имя процедуры :список формальных параметров *- заголовок*
набор команд *- тело процедуры*
конец *- завершение процедуры*

Рассмотрим, как работает процедура с параметром на примере квадрата:

Команда	**Вызов процедуры**	**Результат**
это квадрат :сторона по повтори 4 [вп :сторона пр 90] конец	квадрат 20	
	квадрат 20	
	квадрат 20	

Пример:

Вернемся к рисунку, рассмотренному в начале данной темы. Процедура для рисования данных квадратов выглядит следующим образом:

```
это квадраты :сторона
повтори 10 [повтори 4 [вп :сторона пр 90] пусть
“сторона :сторона + 10]
конец
```

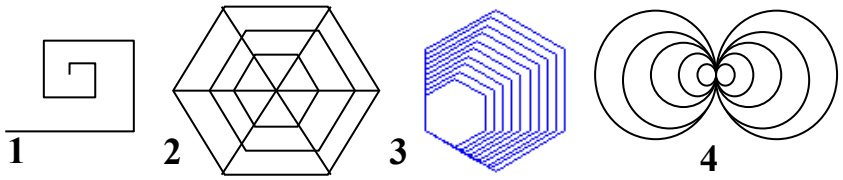
Вопросы

1. Для чего используется процедура с параметром?
2. Общий вид записи процедуры с параметром.
3. Правила, применяемые при работе с процедурами.
4. Сколько параметров может быть использовано в процедуре с параметром, от чего это зависит?

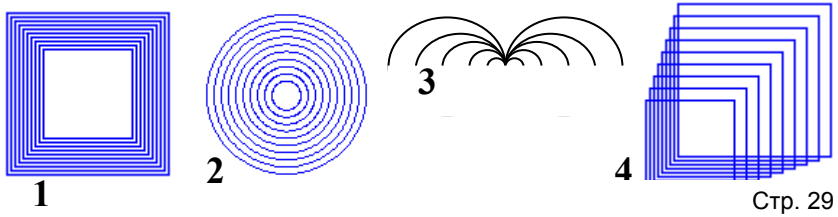
Задания

Составить процедуры построения следующих рисунков:

Обязательные задания по теме:



Дополнительные задания по теме:



А!!!
АБВ&
ГДЕ?Ё
ЖЗИЙ –
КЛМ!
НО: П
РСТУ
ФХ! ЦЧ
Ш? ЩЬ
Ы; Ъ< Э?
Ю!
Я!!!

Разветвляющийся алгоритм

Рассмотрим ещё один вид алгоритма, в котором выполнение какого-либо действия зависит от выполнения или невыполнения конкретного условия. Такой алгоритм называется ***разветвляющимся***, а оператор, описывающий этот алгоритм, называется ***условным***.

Рассмотрим часто используемые команды:

если логическое значение список инструкций

Выполняет список инструкций только в том случае, если условие (первый входной параметр) сообщает значение *да.*

Пример:

Черепашка разворачивается на 180^0 как только встретится с красным цветом:

```
если цвет_поля = 15 пр 180
```

если_иначе логическое значение [инструкции1] [инструкций2]

Выполняет список инструкций1, если первый входной параметр равен *да.* Выполняет список инструкций2, если первый входной параметр равен *нет*

Пример:

Черепашка идёт вперёд на 50 шагов, если цвет поля = 15 и назад на 50 шагов в противном случае:

```
если_иначе цп = 15 [вп 50]  [нд 50]
```

всегда [список-инструкций]

Бесконечно повторяет входной список инструкций. Для остановки процесса используйте команду отмени.

Пример:

Черепашка всегда движется на 1 шаг:

```
всегда [вп [1]]
```

жди_пока [логический датчик]

Компьютер будет ждать, пока входной параметр не сообщит да, и только после этого перейдет к выполнению следующей инструкции. Входной параметр - список инструкций, результатом выполнения которого является да или нет.

Пример:

Черепашка движется по направлению к красному пятну на листе.

```
это препятствие
всегда [вп 1] жди_пока [цп = 15] пр 180
конец
```

останов

Останавливает все работающие процедуры и процессы, включая черепашек и кнопки. *Останов* может использоваться при программировании кнопки, при записи процедуры или вызываться из Поля команд.

Если останов вызывается из процедуры, то останавливает не только ту процедуру, в которой записана команда останов, но и все прочие работающие процедуры.

```
если цп = 15 [останов]
```

стоп

Останавливает работу процедуры, в которой выполнена эта команда. Может использоваться только в процедурах.

Пример:

```
это подсчет :число
если :число > 100 [стоп]
пиши :число
подсчет :число + 5
конец
```

автостоп

Останавливает процесс, при выполнении которого встретилась эта команда.

Пример:

Процесс остановится, когда черепашка с именем ч1 удалится от черепашки с именем ч2 более чем на 100 шагов:

```
всегда [бег]
это бег
ч1, вп 1
если (путь "ч2) > 100 [автостоп]
конец
```

Вопросы

1. Какой алгоритм называется разветвляющимся?
2. Что такое условный оператор?
3. Какие команды используются при работе с разветвляющимся алгоритмом?

Задания

1. Используя описанные в данной теме команды, составьте следующие процедуры:
2. Черепашка движется прямолинейно. Как только она встречает преграду синего цвета, то рисует полуокружность и продолжает движение дальше.
3. Черепашка движется прямолинейно. Как только она встречает преграду красного цвета, то превращается в летящую птицу.
4. Черепашка движется прямолинейно. Как только она встречает преграду зелёного цвета, то превращается в бегущую собаку.

Работа с несколькими исполнителями

Рассматриваемая нами версия программы позволяет работать одновременно с несколькими Черепашками. Активизируются те или иные исполнители (Черепашки) следующими командами:

каждая [список инструкций]

Все черепашки на данном листе выполняют, одна за другой, указанный список инструкций. До тех пор, пока очередная черепашка не выполнит все инструкции, следующая Черепашка ничего делать не будет.

Пример:

а) Все черепашки наденут форму 12
каждая [нф 12]

б) Каждая черепашка нарисует квадрат
каждая [повтори 4 [вп 50 пр 90]]

в) Все черепашки поползут вперед
каждая [всегда [вп 5]]

для [черепашка или список черепашек]

Делает активной черепашку. Входным параметром может быть список черепашек. Команды выполняются одновременно всеми указанными Черепашками.

Еще один способ сделать черепашку активной - написать ее (его) имя с запятой.

Пример:

Первая и вторая черепашки движутся вперёд на 50 шагов.
для [ч1 ч2]
вп 50

скажи [список черепашек] [список инструкций]

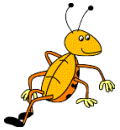
По данной команде Черепашки с указанными номерами временно активизируются только для выполнения тех инструкций, что указаны в команде.

Пример:

На листе есть три черепашки с именами ч1, ч2 и ч3:
скажи [ч1 ч2 ч3] [вп 50 пр 90 вп 50]

Вопросы

1. Какие команды используются для работы с несколькими исполнителями?
2. Для чего необходимо использовать несколько исполнителей?

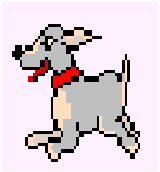


Задания

Используя описанные в данной теме команды выполните следующее задание:

Расположите в углах рабочего листа следующие объекты: собака, девочка, мальчик. Задайте для них следующие команды:

- Мальчик рисует окружность;
- Собака бежит;
- Девочка рисует два вложенных квадрата;



Процедуры обработка слов

Слово – это любой набор символов (букв, цифр, специальных знаков), перед которым ставятся кавычки.

Рассмотрим команды, используемые при обработки слов:

последний “набор символов или **псл** “набор символов
Сообщает последний элемент слова или списка.

Пример: Показат ь последнюю букву слова
покажи псл "хелло
о

первый “набор символов или **прв** “набор символов
Сообщает последний элемент слова или списка.

Пример: Показат ь первую букву слова
покажи прв "хелло
х

кроме первого “набор символов или **кпрв** “набор символов

Пример: Показат ь слово кроме первой буквы
покажи кпрв "свист
вист

кроме последнего “набор символов или **кпсл** “набор символов

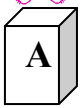
Пример: Показат ь слово кроме последней буквы
покажи кпсл "дом
до

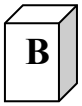
в конец списка “набор символов [список] или **вксп** “набор символов [список]

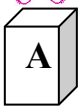
Пример: Вст авляет символ в конец списка символов

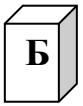
покажи вксп "ы [стол]
столы

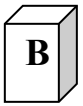
в начало списка “набор символов [список] или **внсп** “набор символов [список]

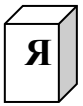












входит? “набор символов1 [набор символов2]
Сообщает да, если первый входной параметр является элементом второго.

Пример:
покажи входит? "а [а b c]
да
покажи входит? "Петя [Вася]
нет

равны? “набор символов1 “набор символов2
Сообщает да, если набор символов1 равен набору символов2. Входные параметры могут быть словами, числами или списками. Прописные и строчные буквы считаются равными

Пример:
покажи равны? "я "Я
да

сколько “набор символов
Сообщает количество элементов в наборе символов.

Пример:
покажи сколько "хелло
5
покажи сколько [0 1 2 3]
4

слово “набор символов1 “набор символов2
Склеивает входные параметры в одно слово и сообщает это слово. Слово допускает более двух входных параметров в этом случае слово вместе с параметрами следует заключить в круглые скобки.

Пример:
покажи слово "лили "пут
лилипут

элемент число “набор символов.

Сообщает элемент набора символов с номером числа. Первый входной параметр должен быть не меньше единицы и не больше количества элементов в слове (списке).

Пример:

```
покажи элемент 2 "хелло
е
покажи элемент 3 [0 1 2 3]
2
```

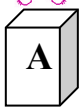
Вопросы

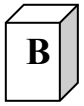
1. Что такое слово?
2. Какие команды используются для обработки слов?

Задания

1. Составьте процедуру, выясняющую, совпадают ли первая и последняя буквы произвольного слова.
2. Составить процедуру, которая подсчитывает, сколько раз заданное число встречается в заданном списке чисел.
3. Составьте процедуру для подсчета количества нулей в заданном списке чисел.
4. Составьте процедуру для постепенного укорачивания заданного слова. Например:

карта		карта
карт		арта
кар	или	рта
ка		та
к		а





Приложение. Команды исполнителя

вперед_число - движение вперед

вп_число - движение вперед

назад_число - движение назад

нд_число - движение назад

направо_число - поворот направо

пр_число - поворот направо

налево_число - поворот налево

лв_число - поворот налево

нк_число - новый курс

по - перо опустить

пп – перо поднять

сг – сотри графику

сотри – экран очищается, а черепашка остаётся на прежнем месте

сч – спрячь черепашку

пч – покажи черепашку

нц_номер цвета– новый цвет

крась – закрашивает замкнутую область, в которой находится черепашка цветом её пера

нрз_число – новый размер (меняет размер черепашки от 5 до 160)

нрп_число – новый размер пера (изменяет размер пера активной черепашки от 1 до 100)

повтори_n_[набор команд] - общий вид команды цикла

повтори n [вп x пр 360 / n] - постороение правильного многоугольника

если логическое значение список инструкций

Выполняет список инструкций только в том случае, если условие (первый входной параметр) сообщает значение *да.*

если_иначе логическое значение[инструкции1] [инструкций2]

Выполняет список инструкций1, если первый входной параметр равен *да.* Выполняет список инструкций2, если первый входной параметр равен *нет*

всегда [список-инструкций]

Бесконечно повторяет входной список инструкций. Для остановки процесса используйте команду отмени.

жди_пока [логический датчик]

Компьютер будет ждать, пока входной параметр не сообщит да, и только после этого перейдет к выполнению следующей инструкции.

останов

Останавливает все работающие процедуры и процессы, включая черепашек и кнопки.

стоп

Останавливает работу процедуры, в которой выполнена эта команда.

каждая [список инструкций]

Все черепашки на данном листе выполняют, одна за другой, указанный список инструкций.

для [черепашка или список черепашек]

Делает активной черепашку.

скажи [список черепашек] [список инструкций]

По данной команде Черепашки с указанными номерами временно активизируются только для выполнения тех инструкций, что указаны в команде.

последний “набор символов

Сообщает последний элемент слова или списка.

первый “набор символов

Сообщает последний элемент слова или списка.

кроме первого “набор символов

Выводит все символы кроме первого

кроме последнего “набор символов

Выводит все символы кроме последнего

в конец списка “набор символов [список]

Вставляет набор символов в конец списка

в начало списка “набор символов [список]

Вставляет набор символов в начало списка

входит? “набор символов1 [набор символов2]

Сообщает да, если первый входной параметр является элементом второго.

равны? “набор символов1 “набор символов2

Сообщает да, если набор символов1 равен набору символов2.

сколько “набор символов

Сообщает количество элементов в наборе символов.

слово “набор символов1 “набор символов2

Склеивает входные параметры в одно слово и сообщает это слово.

элемент число “набор символов

Сообщает элемент набора символов с номером числа

Список литературы:

1. ЛогоМиры. Справочное пособие.-М.: ИНТ, 1995.
2. ЛогоМиры. С чего начать.-М.: ИНТ, 1995.
3. Добудько Т.В. Информатика 7.-Самара.: Федоров, 1997.
4. Юдина А.Г. Практикум по информатике в среде Logo Writer. -М.: Мнемозина, 1999.
5. Юдина А.Г. Методическое пособие к прктикуму по информатике в среде LogoWriter. -М.: Мнемозина, 1999.
6. Информатика: основы компьютерной грамоты. Начальный курс / Под ред. Н.В.Макаровой. - СПб.: Питер, 2000

МОУ «ЛИЦЕЙ №57»

г. Прокопьевск, ул. Институтская, 41

Телефон: (3846)62-31-20
Эл. почта: lyceum57@mail.ru